# quaderni di management

luglio.agosto 2004 - n° 10

Convegno ADACI.
L'evoluzione del buyer: da esecutore a stratega
*Elisabetta Colombo*

La tracciabilità nel settore agro-alimentare:
una risposta o un problema per la competitività delle aziende italiane?
*Andy Cavallini e Duilio Luca Molinari*

Gli algoritmi genetici: uno strumento di risoluzione ai problemi di schedulazione della produzione
*Alberto F. De Toni e Fabio Nonino*

Gestione per competenze. Il modello Festo Competence
*Matteo Cervone, Monica Pirali e Luigi Roveda*

Il cliente lavora in comunità
*Simone Testa*

LETTURE STRANIERE
L'amplificazione nelle supply chain dei servizi:
un case study operativo
*Henk Akkermans e Bart Vos*

COSA CI ASPETTA NEL FUTURO:
Intervista a Alessandro Perego
Professore associato di Gestione dei Sistemi Logistici e Produttivi e Logistica Integrata

E.G.V. Edizioni

Poste Italiane Spa - Spedizione in A.P. - D.L. 353/2003 (conv. In L. 27/02/2004 n. 46) art. 1, comma 1, DCB Milano

# quaderni di management

N.10 - Luglio/Agosto 2004

*www.quaderni-di-management.it*
*qdmredazione@libero.it*

**Direttore Responsabile**
Giancarlo Oriani

**Redazione**
Elena Santambrogio

**Relazioni esterne**
Riccardo Garavaglia - Socio FERPI

**E.G.V. Edizioni srl**
Direzione, Redazione:
Via Trabattoni 15 - 20038 Seregno - MI
tel.: 0362-222615

Amministrazione:
Via Fratelli Cairoli 17 - 20035 Lissone - MI

**Amministratore Unico**
Vittorio Galimberti

Registrazione presso il Tribunale di Monza n.1648
Registrazione presso R.O.C. richiesta il 7 Febbraio 2003

**Stampa:** Bozzi Multimedia Srl, Milano

**Periodicità:** bimestrale

**Abbonamenti:**
Annuale 80 euro
Annuale estero 120 euro
**Modalità di abbonamento:**
Effettuare bonifico su c/c n. 50105 - ABI 3104 CAB 01609
CIN U - IBAN IT61 U031 0401 6090 0000 0050 105
Deutsche Bank Agenzia I Milano intestato a E.G.V.
Edizioni srl ed inviare richiesta tramite email

Editoriale ..... 3

Note sugli autori ..... 7

Consigli per la lettura ..... 8

Convegno ADACI. L'evoluzione del buyer: da esecutore a stratega
*di Elisabetta Colombo* ..... 10

La tracciabilità nel settore agro-alimentare: una risposta o un problema per la competitività delle aziende italiane?
*di Andy Cavallini e Duilio Luca Molinari* ..... 18

Gli algoritmi genetici: uno strumento di risoluzione ai problemi di schedulazione della produzione *di Alberto F. De Toni e Fabio Nonino* ..... 28

Gestione per competenze. Il modello Festo Competence
*di Matteo Cervone, Monica Pirali e Luigi Roveda* ..... 52

Il cliente lavora in comunità *di Simone Testa* ..... 57

LETTURE STRANIERE
L'amplificazione nelle supply chain dei servizi: un case study operativo
*di Henk Akkermans e Bart Vos* ..... 67

COSA CI ASPETTA NEL FUTURO:
Intervista a Alessandro Perego
Professore associato di Gestione dei Sistemi Logistici e Produttivi e Logistica Integrata ..... 93

**Comitato di Direzione**

*Danilo Bonato* — Direttore Generale, E-TREE
*Ivan Ciceri* — Membro System Dynamics Society
*Paolo Citterio* — Presidente GIDP
*Mario D'Ambrosio* — Presidente AIDP
*Matteo De Angelis* — Direttore Organizzazione e Sistemi, Fischer Italia
*Franco Favaro* — Direttore Risorse Umane, SanBenedetto
*Enrico Frizzera* — Amministratore Delegato, Gruppo Manni
*Alberto Gandolfi* — AFG Management Consulting
*Cristiano Liuzzo* — Amministratore Delegato, Impresa! Progetti Risorse Umane
*Giuseppe Marengon* — E-business manager, Hewlett Packard
*Luca Molinari* — Dir.Consulenza Industria e Retail, Atos Origin Executive Partner
*Ersindo Nuzzo* — Direttore Sviluppo Risorse Umane, S.I.P.A. Bindi
*Roberto Piccinelli* — Direttore del Personale, Roche
*Valerio Ricciardelli* — Direttore Generale, Festo CTE
*Flavio Sciuccati* — Partner Ambrosetti

**Comitato Scientifico**

*Emilio Bartezzaghi* — Gestione Aziendale, Politecnico di Milano
*Giovanni Bernardi* — Economia e Organizzazione Aziendale, Università di Padova
*Carmine Bianchi* — Economia Aziendale, Università di Palermo
*Fabio Corno* — Metodologie Quantitative d'Impresa, Università LIUC Castellanza
*Maurizio Decastri* — Organizzazione Aziendale, Università di Roma II – Tor Vergata
*Marco De Marco* — Organizzazione Sistemi Informativi Aziendali, Università Cattolica del Sacro Cuore Milano
*Alberto De Toni* — Strategia e Gestione della Produzione, Università di Udine
*Susi Dulli* — Business Intelligence, Università di Padova
*Roberto Panizzolo* — Economia e Organizzazione Aziendale, Università di Padova

**Sintesi**
*I due autori affrontano il tema della schedulazione della produzione realizzata tramite l'impiego degli Algoritmi Genetici (AG). Dopo una classificazione introduttiva dei metodi utilizzabili nella programmazione operativa, l'attenzione si focalizza sugli algoritmi evolutivi ed in particolare su quelli genetici di cui viene fornita una ampia descrizione ed un modello applicativo utile per risolvere i problemi di schedulazione. L'articolo presenta inoltre un caso aziendale riguardante la pianificazione a breve termine delle operazioni di assemblaggio di un'importante azienda motociclistica Italiana; la soluzione al caso richiede un particolare modello di* scheduling *della produzione. L'approccio di schedulazione proposto si articola in due passaggi, un* macrostep *ed un* microstep. *In quest'ultimo è presente l'applicazione degli Algoritmi Genetici (AG) tramite l'adattamento del modello applicativo proposto. Vengono forniti i diagrammi di flusso utilizzati nell'implementazione del prototipo software di schedulazione progettato come soluzione informatica al caso proposto. I tempi ridotti di elaborazione e la bontà dei piani prossima all'ottimo riscontrati in fase di test confermano la funzionalità della scelta degli Algoritmi Genetici. Inoltre grazie alla generalizzazione degli assunti presi in fase di progettazione, il prodotto software ottenuto diviene facilmente* general purpose *e "customizzabile" a realtà differenti da quella presentata nel caso studio.*

Alberto F. De Toni
Fabio Nonino

# Gli algoritmi genetici: uno strumento di risoluzione ai problemi di schedulazione della produzione

## 1. Introduzione

Nel presente articolo si analizza il caso della programmazione operativa delle linee di assemblaggio *mixed model*, in cui sono realizzati differenti prodotti attraverso tempi di attrezzaggio della linea molto bassi o addirittura assenti. Questo problema di schedulazione è spesso denominato *mixed-model scheduling*. In un tale sistema produttivo i manager desiderano idealmente sequenziare i differenti prodotti ottenendo un livello di servizio alto, senza incorrere in un eccessivo numero di setup della linea per cambio prodotto che potrebbero causare perdite di tempo e disallineamento nei tempi di consegna: la difficoltà della programmazione operativa consiste dunque nell'affrontare il *trade-off* fra livello di servizio offerto (mix di prodotto) e perdite di tempo nell'attrezzaggio rispettando allo stesso tempo i vincoli

di capacità e di tempo di consegna. Siamo quindi di fronte a quello che tecnicamente viene chiamato un "problema di ottimizzazione combinatorio multiobiettivo", un problema cioè in cui si deve trovare la soluzione ottima in presenza di molti obiettivi e di variabili correlate tra loro.
La ricerca della soluzione ottima ai grandi problemi combinatori non è un'opzione praticabile perché è richiesta una rilevante quantità di tempo, non compatibile con le esigenze operative di applicativi software per realtà aziendali. Si ricorre allora alle cosiddette tecniche euristiche che forniscono soluzioni soddisfacenti, anche se non ottimizzate, con tempi e consumi di risorse accettabili.
Il focus del presente lavoro è l'applicazione di una di queste euristiche [gli Algoritmi Genetici (AG)] come tecnica per la risoluzione di problemi di tale complessità Nella prima parte dell'articolo gli AG vengono inquadrati all'interno di una classificazione dei metodi utili alla programmazione operativa; successivamente sono descritti i principi e le fondamentali caratteristiche che hanno portato al successo questa tecnica e si forniscono le nozioni di base ed un modello applicativo per il loro impiego nei problemi di ottimizzazione combinatoria. La seconda parte dell'articolo presenta un caso aziendale riguardante la pianificazione a breve termine delle operazioni di assemblaggio che richiede un particolare modello di *scheduling* della produzione. L'approccio di schedulazione proposto si articola su due passaggi, un *macrostep* ed un *microstep*, che operano rispettivamente una macroschedulazione e microschedulazione degli ordini in ingresso; in particolare all'interno del *microstep* è presente l'applicazione degli Algoritmi Genetici (AG). Lo schedulatore progettato in tal modo diviene un utile strumento di risoluzione al caso proposto, grazie ai tempi ridotti di elaborazione ed alla bontà dei piani generati prossima all'ottimo. Inoltre grazie alla generalizzazione degli assunti presi in fase di progettazione il prototipo software ottenuto diviene agevolmente *general purpose* e "customizzabile" a realtà differenti da quella presentata nel caso studio.

## 2. Algoritmi metaeuristici evolutivi

Gli algoritmi metaeuristici (metaeuristiche) sono algoritmi non deterministici approssimati: infatti non garantiscono una soluzione ottima in un tempo definito. Vengono abitualmente applicati ai problemi di ottimizzazione combinatoria o di vincolo e non sono legati ad uno specifico problema come le tecniche euristiche. Un'euristica, infatti, per essere efficace deve sfruttare le caratteristiche strutturali del problema che deve risolvere; per questo motivo non esiste una euristica generale. Esistono però degli approcci generali a cui si può fare riferimento per sviluppare euristiche specifiche. Tali approcci vengono detti metaeuristiche. In figura 1 sono riassunti i metodi utilizzati per la programmazione operativa; una prima distinzione può essere effettuata fra i metodi di ottimizzazione, che garantiscono la soluzione ottima per un dato problema in rapporto ai vincoli ed agli obiettivi prefissati e le euristiche che come detto garantiscono una soluzione "buona" in un tempo ragionevole.
La scelta cade dunque sulle tecniche (meta)euristiche quando il problema ha grandi dimensioni e si cerca una soluzione vicina all'ottimo in tempi brevi.
L'obiettivo delle tecniche metaeuristiche è una efficace ed efficiente esplorazione dello spazio delle soluzioni

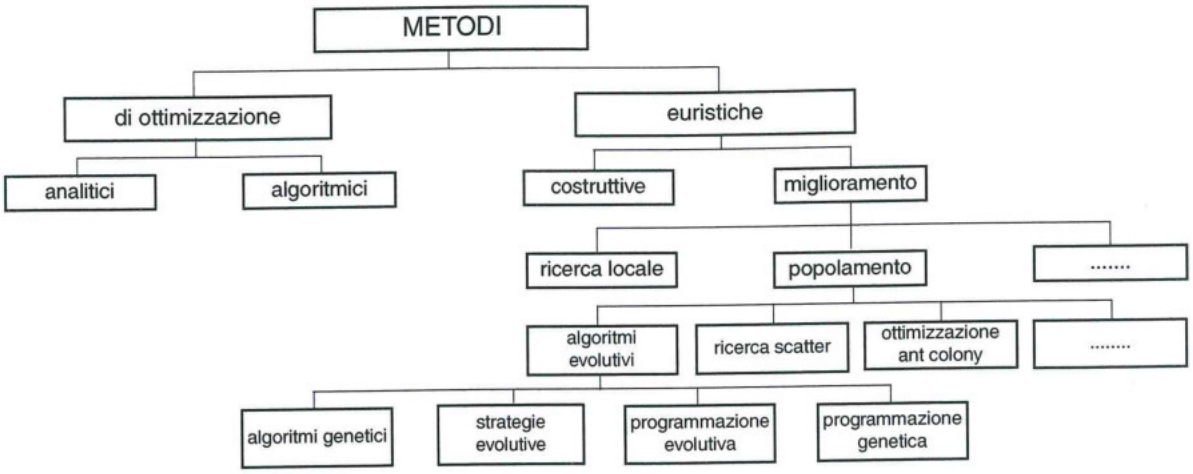



*Figura 1 - Classificazione dei metodi utilizzabili per la programmazione operativa*

utilizzando (Roli, 2003):

- ricerca storica: mantengono in memoria le esperienze passate;
- adattività: non sono *problem specific* ma si adattano a nuove condizioni;
- strategie generali per bilanciare intensificazione (valorizzazione dell'esperienza di ricerca accumulata) e diversificazione (ampliamento dello spazio di ricerca).

Le tecniche metaeuristiche possono essere classificate (Pesenti, 2002) in:

- euristiche costruttive (*Random Search*, adattive,...) che costruiscono passo a passo una soluzione del problema;
- euristiche di miglioramento che iterativamente, a partire da una soluzione del problema, cercano di determinarne una migliore nell'intorno di quella data; tra queste possiamo comprendere:

- metodi di ricerca locale (*Tabù Search, Simulated Annealing, Iterated local Search,...*) che a partire da una soluzione del problema cercano iterativamente di determinarne una migliore nell'intorno di quella data;
- metodi di popolamento (Algoritmi Evolutivi, Ricerca Scatter[1], ottimizzazione *Ant Colony*[2],...) che creano una popolazione di soluzioni e la fanno muovere verso buone regioni dello spazio di ricerca.

I metodi di popolamento, in particolare gli Algoritmi Evolutivi, sono fra quelli presi più in considerazione negli ultimi decenni per risolvere problemi di ottimizzazione. A partire dagli anni Cinquanta molti informatici studiarono i sistemi evolutivi intuendo che i meccanismi tipici dell'evoluzione e dell'ereditarietà teorizzati dall'antropologo Darwin (1859) potessero essere utilizzati come strumenti di ottimizzazione sviluppando i cosiddetti Algoritmi Evolutivi (EA - *Evolutionary Algorithms*). Questi metodi di ricerca vengono classificati (Eiben e Schippers, 1998) in quattro gruppi aventi lo stesso nucleo algoritmico di riferimento (figura 2):

***1. Algoritmi Genetici*** (GA - *Genetic Algorithms*): inventati da John Holland (1975), utilizzati nei problemi di ottimizzazione combinatoria e nei problemi di schedulazione utilizzano gli operatori

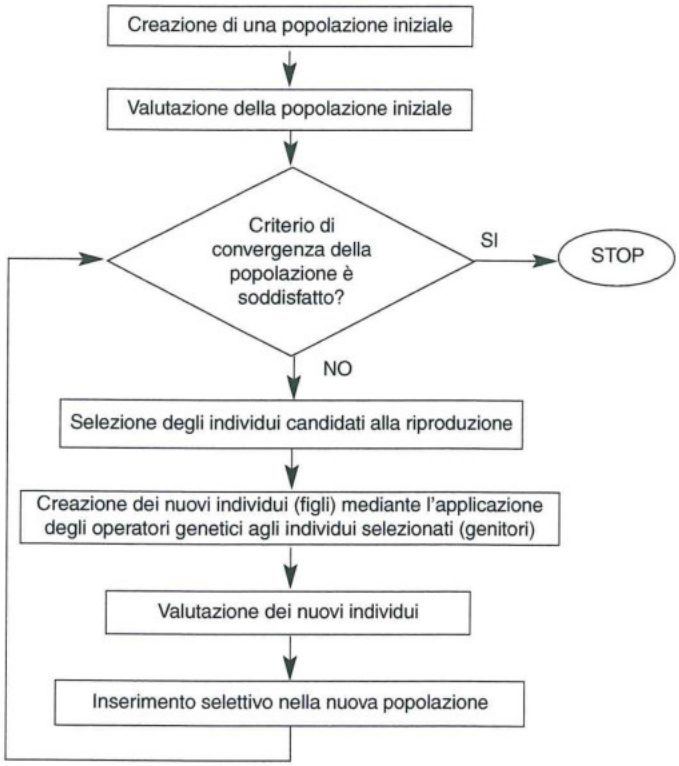



*Figura 2 - Nucleo algoritmico di riferimento degli Algoritmi Evolutivi*

genetici di selezione naturale, incrocio (*crossover*) e mutazione (*mutation*) (vedi paragrafo 3.2).

***2. Strategie Evolutive*** (ES - *Evolutionary Strategies*): sviluppate negli anni Sessanta da Rechenberg, (Rechenberg 1973, Schwefel 1981) furono utilizzate originariamente per problemi di ingegneria civile e strutturale come l'ottimizzazione dei parametri a valori reali per strutture aerodinamiche; sono simili alla classe degli AG, ma diversamente favoriscono la mutazione come operatore genetico principale piuttosto che l'incrocio.

***3. Programmazione Evolutiva*** (EP - *Evolutionary Programming*): fu introdotta da Fogel, Owens e Walsh (1966) ed è anch'essa simile agli AG ma basa tutta l'evoluzione della popolazione sull'operatore di mutazione senza utilizzare l'incrocio; la probabilità di mutazione decresce man mano che ci si avvicina all'ottimo.

***4. Programmazione Genetica*** (GP - *Genetic Programming*): fu introdotta da Koza (1992) ed è un'applicazione informatica degli AG utilizzata per far evolvere programmi software; i programmi vengono codificati con una struttura ad albero: il *crossover* è effettuato scambiando porzioni di albero fra due individui e la mutazione non è utilizzata.

I quattro gruppi differiscono quindi per la rappresentazione degli individui, il tipo di selezione ma soprattutto per l'utilizzo degli operatori genetici (*crossover* e *mutation* descritti nel paragrafo 3.2) e le applicazioni per cui nascono e sono principalmente impiegati, come riassunto nella tabella 1.

Zitler (2002) suggerisce l'utilizzo degli Algoritmi Evolutivi (AE) per tre motivi:

- Flessibilità (Flexibility): la formulazione del problema può essere facilmente modificata/estesa (minimo requisito).
- Obiettivo multiplo (Multiple Objectives): lo spazio di soluzione può essere esplorato con singolo percorso di ottimizzazione.
- Fattibilità (Feasibility): gli AE sono applicabili in spazi di ricerca enormi e complessi.

## 3. Algoritmi genetici

L'opera fondamentale sugli algoritmi genetici è *Adaptation in Natural and Artificial Systems* di J. Holland (1975), in cui si presenta l'algoritmo genetico come un'astrazione dell'evoluzione biologica e si

| | | Indice di utilizzo degli operatori genetici | |
|---|---|---|---|
| ALGORITMI EVOLUTIVI | Principali applicazioni | *Crossover* | *Mutation* |
| Algoritmi Genetici | Ottimizzazione combinatoria, problemi di schedulazione,.... | Alto (0.6 - 0.8) | Basso (0.1 - 0.2) |
| Strategie Evolutive | Ottimizzazione di parametri a valori reali per strutture aerodinamiche, problemi di ingegneria civile | Basso (0.2 - 0.3) | Alto (0.6 - 0.8) |
| Programmazione Evolutiva | Ottimizzazione combinatoria, problemi di schedulazione,.... | | Alto (0.9 e decresce) |
| Programmazione Genetica | Evoluzione di strutture software | Alto (0.6 -0.8) | |

*Tabella 1 - Principali applicazioni degli algoritmi evolutivi*

fornisce una struttura teorica per la nozione di adattamento. Gli AG nacquero negli anni Sessanta con l'obiettivo di studiare formalmente il fenomeno dell'adattamento così come avviene in natura e di inventare tecniche per trasferire all'interno dei sistemi informatici i meccanismi di questo tipo d'adattamento (fig. 3); essi combinano la sopravvivenza del più forte con lo scambio di conoscenza strutturata per formare un algoritmo di ricerca innovativo.

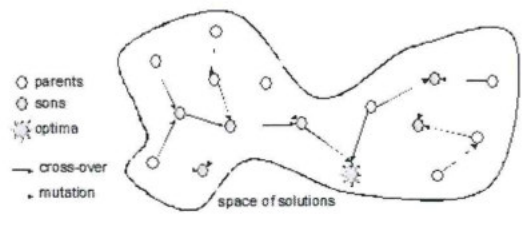



*Figura 3 - Algoritmi "meta-euristici" evolutivi di tipo genetico*

La peculiarità di questa tecnica deriva dal fatto che s'ispira all'evoluzione naturale ed è quindi fondata sui principi darwinisti della selezione e dell'adattamento, e, naturalmente, su meccanismi di riproduzione e di mutazione genetica.

L'approccio degli Algoritmi Genetici è differente da alcune tipologie di AE come le strategie evolutive ed la programmazione evolutiva, che intendono progettare algoritmi per risolvere problemi specifici piuttosto che "customizzabili" rispetto al problema in analisi. Gli AG non fanno assunti circa la funzione che deve essere ottimizzata, l'unica cosa che richiedono è una misura delle performance, una rappresentazione del problema e gli operatori che generano i nuovi membri della popolazione. Levine (1997) dice: "Quando paragonati ad altri metodi di ottimizzazione, un vantaggio degli AG che spicca è l'ampio raggio dei problemi ai quali possono essere applicati: ottimizzazione combinatoria, ..., problemi di schedulazione".

Possiamo evidenziare tre grandi vantaggi degli AG:
1) Sono un metodo di ottimizzazione globale: molti problemi hanno un numero (talvolta grande) di ottimi locali in cui alcuni metodi di ricerca tradizionali possono rimanere intrappolati.
2) Forniscono flessibilità nella gestione delle variazioni di un problema di ottimizzazione cosa in cui i metodi più specifici possono avere difficoltà.
3) Ad ogni interazione, un algoritmo genetico contiene una popolazione di possibili soluzioni: in pratica è più desiderabile analizzare una famiglia di soluzioni piuttosto che una singola. Arabeyre (1969) dice: "La conoscenza di una famiglia di buone soluzioni è molto più importante che ottenere un ottimo isolato".
In un Algoritmo Genetico dal funzionamento efficace devono essere definiti i seguenti 4 aspetti:

1. Una rappresentazione genetica ovvero una codifica delle potenziali soluzioni e la funzione di valutazione che svolge il ruolo dell'ambiente stimando le soluzioni in termini d'idoneità (*fitness*).
2. Gli operatori genetici che alterano i cromosomi dei discendenti per passare da una popolazione ad una nuova compresa la loro probabilità di applicazione; gli operatori genetici sono:
   - operatore di selezione (*natural selection*);
   - incrocio (*crossover*);
   - mutazione (*mutation*).
3. Un modo per creare una popolazione iniziale di potenziali soluzioni.
4. Le condizioni di stop dell'algoritmo.

### *3.1 Codifica del problema: i cromosomi*

Gli organismi sono composti da cellule, costituite a loro volta dai cromosomi rappresentanti il "progetto" dell'organismo ovvero il genoma. Il cromosoma è diviso in geni, ciascuno dei quali codifica una caratteristica dell'organismo, come ad esempio il colore dei capelli o degli occhi. Ogni gene occupa particolare posizione all'interno del cromosoma. L'idoneità (*o fitness*) di un organismo è definita tipicamente come la probabilità che l'organismo viva abbastanza da riprodursi o come la funzione del numero di discendenti che esso genera. Negli algoritmi genetici una soluzione è un genoma e può essere composto da più cromosomi (individui) che rappresentano la popolazione; un cromosoma (fig. 4) è composto dai geni ed il gene rappresenta una particolarità od una proprietà del problema considerato.

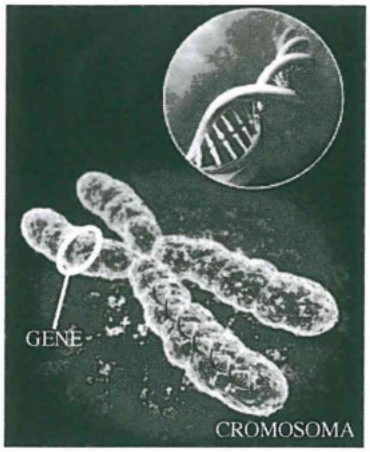



*Figura 4 - Cromosoma e geni*

Un processo evolutivo utilizza una popolazione di cromosomi per ricercare possibili soluzioni, sfruttando le soluzioni potenzialmente migliori, ed esplorando lo spazio di ricerca. Negli AG di Holland la popolazione subisce un'evoluzione simulata: ad ogni generazione i rappresentanti "migliori" della popolazione si riproducono, mentre quelli meno buoni muoiono. Ogni individuo rappresenta una potenziale soluzione al problema ed ogni soluzione viene valutata mediante l'assegnazione di un valore di idoneità (*fitness*) tramite una funzione obiettivo. Una funzione obiettivo misura

la performance di una sistema come una funzione di una o più variabili decisionali (Hall, 1991).

Nel caso di un *flow shop*, oggetto dell'articolo, si può utilizzare l'ordinamento dei job come cromosomi non modellizzando però l'ordine di esecuzione delle operazioni, ma solo quello dei job. L'obiettivo della codifica è di modellizzare al meglio una soluzione; per i problemi di ordinamento troviamo principalmente due tipi di codifica:

1. Codifica diretta: la descrizione di una soluzione è data direttamente a partire da un cromosoma. Per esempio un cromosoma è una sequenza dove gli elementi sono gli indici dei job e l'ordine degli elementi di questo vettore rappresenta le posizioni dei job sulle macchine.
2. Codifica indiretta : il cromosoma non fornisce una soluzione diretta ma rappresenta le caratteristiche che permettono di ottenerne una; ad esempio descrive l'ordine di priorità dei job.

Gli AG utilizzano diversi operatori genetici per passare da una popolazione di "cromosomi" (rappresentati mediante permutazioni di stringhe binarie o con codifica simbolica) ad una nuova popolazione.

### *3.2 Gli operatori genetici*

*Selezione degli individui*

La selezione delle permutazioni ovvero del genoma degli individui rappresenta la base degli AE ed allo stesso modo degli Algoritmi Genetici, perché si basa sul principio Darwiniano che l'individuo che meglio si adatta resisterà alla selezione naturale. Si selezionano, all'interno della popolazione, alcuni cromosomi destinati alla riproduzione ed, in media, i più adatti produrranno più discendenti idonei alla sopravvivenza rispetto a quelli meno adatti.

La selezione interviene prima degli altri operatori genetici determinando la dimensione della popolazione con l'obiettivo di formare le coppie di genitori. Si possono classificare i metodi differenti di selezionare i genitori in due classi: deterministica o mista (aleatoria e deterministica).

La selezione deterministica può essere:

- selezione *ranking*: sono scelti i migliori individui seguendo i valori della funzione obiettivo;
- selezione di troncatura: sono scelti i migliori individui entro una "troncatura" ovvero una frazione predeterminata dell'intera popolazione;
- sostituzione della popolazione: si sostituiscono i genitori con i figli ottenuti dal loro incrocio e dalla mutazione.

La selezione mista (aleatoria e deterministica):

- selezione *tournament*: sono scelti casualmente un insieme di cromosomi e mantenuto il migliore all'interno di questo insieme per la riproduzione;
- selezione *fitness proportional*: conosciuta anche col nome di *roulette wheel* consiste nell'associare a ciascun individuo una probabilità di essere selezionato proporzionale al valore della funzione obiettivo;
- selezione stocastica semplice: sono scelti gli individui proporzionalmente alla loro probabilità ed in seguito è applicato il metodo *ranking* per la selezione.

*Crossover*

L'operatore *crossover* combina alcuni segmenti dei cromosomi di due "genitori" per formare i discendenti con probabilità pari all'indice di *crossover* (Pc); il suo significato è lo scambio d'informazioni tra le due

potenziali soluzioni imitando la ricombinazione biologica fra due organismi dotati di un solo cromosoma. L'operatore *crossover* ha valore solo se capace di acquisire le caratteristiche migliori di ambedue i genitori.

Esistono numerose tipologie di *crossover*, estremamente utili a patto che mantengano le caratteristiche del problema in analisi; ne proponiamo alcuni esempi:

• *Crossover* semplice a 2 punti per le permutazioni - KX.

Sono selezionati casualmente 2 punti sui due genitori (stringhe di codifica simbolica) dove avverrà un taglio (fig. 5) con l'obiettivo di generare 4 figli. Il figlio mantiene le parti "esterne" del primo (secondo) genitore; la parte centrale è completata prendendo i valori del secondo (primo) genitore non già selezionati e seguendo l'ordine dei geni da sinistra a destra. Per il terzo ed il quarto figlio si mantiene la parte centrale e si procede con la stessa logica.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Parent 2 | 4 | 8 | 5 | 6 | 2 | 1 | 7 | 3 |
| Child 1 | 1 | 2 | 4 | 5 | 3 | 6 | 7 | 8 |
| Child 2 | 4 | 8 | 2 | 5 | 6 | 1 | 7 | 3 |
| Child 3 | 8 | 6 | 3 | 4 | 5 | 2 | 1 | 7 |
| Child 4 | 1 | 3 | 5 | 6 | 2 | 4 | 7 | 8 |

*Figura 5 - Crossover KX*

• *Crossover* LOX.

Questo operatore (fig 6) deriva dal precedente ed usa la stessa logica ma genera solo 2 figli in quanto si vuole mantenere dei "blocchi" delle stringhe genitrici .

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Parent 2 | 4 | 1 | 5 | 6 | 2 | 8 | 7 | 3 |
| Child 1 | 1 | 6 | 3 | 4 | 5 | 2 | 8 | 7 |
| Child 2 | 1 | 3 | 5 | 6 | 2 | 4 | 7 | 8 |

*Figura 6 - Crossover LOX*

• *Crossover* base di posizione

Questo operatore parte con una scelta casuale di alcune posizioni (quattro in figura 7); tutte le posizioni selezionate su un genitore sono ricopiate sul figlio rispettivo. Per completarlo si scelgono i valori dell'altro genitore, non già selezionati, seguendo l'ordine dei geni da sinistra a destra. 7

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Parent 2 | 4 | 1 | 5 | 6 | 2 | 8 | 7 | 3 |
| Child 1 | 1 | 6 | 2 | 4 | 5 | 8 | 7 | 3 |
| Child 2 | 4 | 1 | 3 | 6 | 2 | 5 | 7 | 8 |

*Figura 7 - Crossover base di posizione*

• *Crossover* single-point (1X).

Questo operatore genera 4 figli con la stessa logica del KX, ma taglia ciascuno dei due genitori selezionati in un solo punto scelto casualmente.

• *Crossover* di ciclo.

Questo operatore genera 2 figli operando un particolare ciclo di ricerca delle parti da ricopiare dei due genitori.

• *Crossover* aleatorio

Il principio di tale operatore di *crossover* si basa sulla generazione casuale ottenendo così dei figli che non hanno connessioni con i genitori. L'obiettivo di questo particolare operatore è di test per gli altri operatori: se un *crossover* è peggiore di quello aleatorio significa che la soluzione sta degradando invece di migliorare.

*Mutation*

La mutazione altera arbitrariamente uno o più geni di un cromosoma con probabilità pari all'indice di *mutation* (Pm): il suo significato è l'introduzione di una variabilità extra nella popolazione imitando le

mutazioni biologiche dipendenti da fattori ambientali e culturali diversi dal patrimonio genetico; opera attraverso uno scambio casuale di posizione di uno (fig.8) o più geni all'interno di un cromosoma modificando le caratteristiche del genoma senza degenerarlo.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Child | 1 | 2 | 8 | 4 | 5 | 6 | 7 | 3 |

*Figura 8 - Mutazione di due elemento*

Nella mutazione di 2N elementi (figura 9) lo scambio avviene tra gruppi di geni.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Parent | 4 | 5 | 3 | 2 | 1 | 6 | 7 | 8 |
| Child | 4 | 1 | 6 | 2 | 5 | 3 | 7 | 3 |

*Figura 9 - Mutazione di 2N elementi*

### *3.3 Popolazione iniziale*

La popolazione iniziale rappresenta un insieme di soluzioni ammissibili. Può essere determinata tramite:

- metodo aleatorio: la popolazione è generata in modo casuale;
- metodi euristici: la popolazione è generata attraverso soluzioni provenienti da euristiche, l'inconveniente è che questo metodo può influire sull'algoritmo facendolo convergere troppo rapidamente verso un minimo locale;
- combinazione dei due metodi: parte della popolazione proviene da un'euristica e la rimanente parte da soluzioni generate in modo casuale;
- duplicazione ed evoluzione: se un problema è fortemente vincolato non è facile trovare un insieme di soluzioni ammissibili, per cui è sufficiente trovarne una ed applicarvi gli operatori genetici ottenendo una popolazione iniziale della dimensione definita.

Non esiste una metodologia particolarmente consigliata per la definizione della popolazione iniziale; Reeves (1995) considera la popolazione iniziale sia in modo casuale sia "seminando" la popolazione iniziale con una buona soluzione generata da un'euristica costruttiva. Sperimentalmente l'algoritmo con popolazione seminata arrivava alla popolazione finale più velocemente senza diminuzione della qualità della soluzione. Levine (1997) consiglia al contrario di generarla casualmente al fine di esplorare il più possibile le diverse zone dello spazio delle soluzioni.

### *3.4 Condizioni di stop*

Il criterio di arresto di un algoritmo genetico deve essere scelto in fase di progettazione dell'algoritmo rispettando le necessità legate al tempo di computazione del software che si intende implementare o alla qualità della soluzione che si intende ottenere; in particolare la condizione di stop può essere definita sulla base di un:

- numero massimo di generazioni;
- tempo massimo d'elaborazione;
- numero massimo di generazioni senza migliorare la soluzione trovata.

## 4. Diagramma di flusso degli algoritmi genetici

Proponiamo lo schema generale del *Simple Genetic Algorithm* (Goldberg, 1989) che si sviluppa in sei passaggi:

1. genera una popolazione iniziale di N cromosomi.

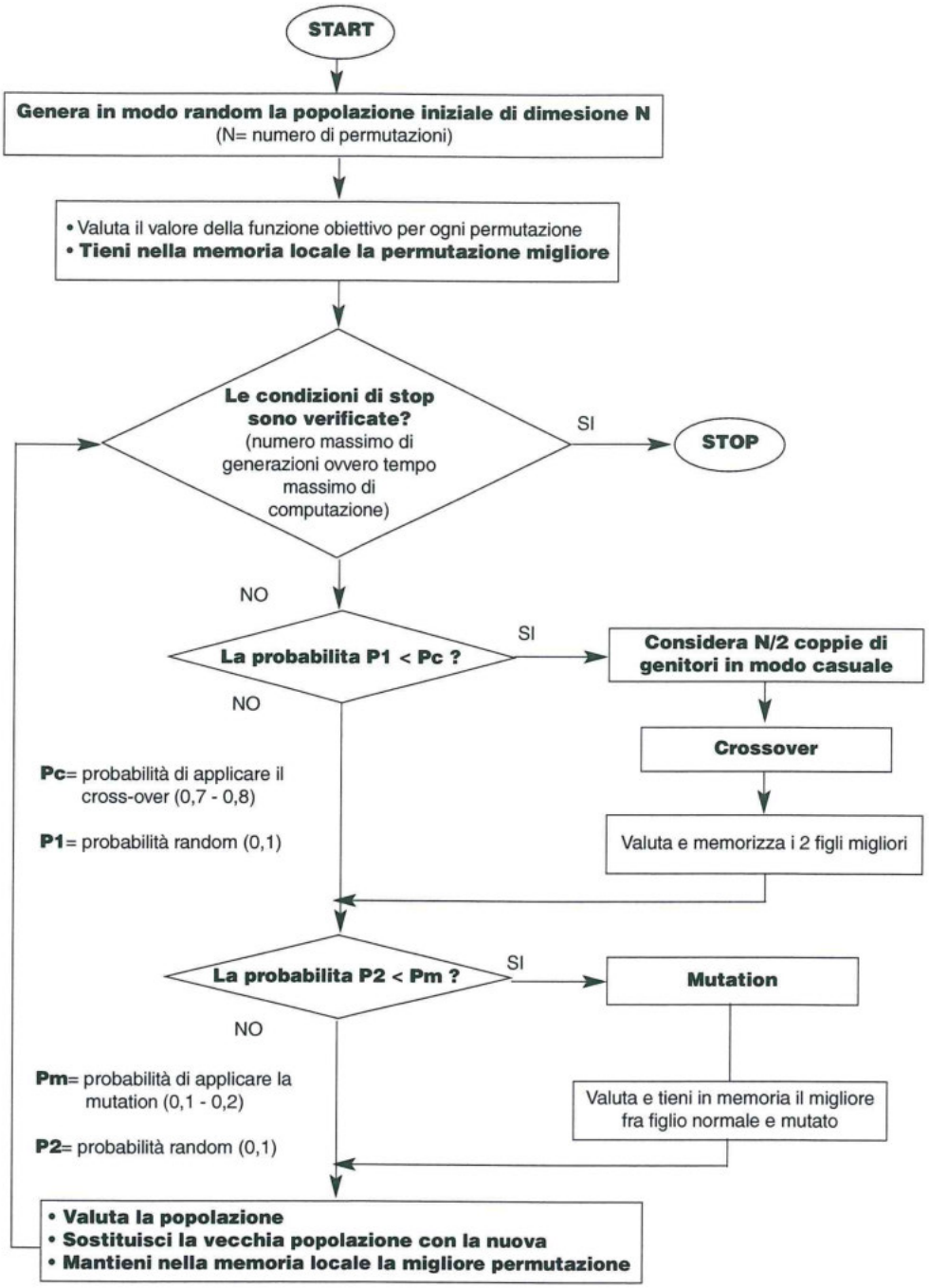



*Figura 10 - Diagramma di flusso degli algoritmi genetic*

2. Seleziona N coppie di genitori (parent 1, parent 2) dalla vecchia popolazione.
3. Crea una nuova popolazione con l'operatore *crossover* sulle coppie selezionate con probabilità Pc, pari all'indice di *crossover*, o mantenendo i genitori con probabilità 1-Pc.
4. Con una probabilità molto piccola Pm, pari all'indice di *mutation*, ogni cromosoma può essere modificato dall'operatore mutazione.
5. Valuta il nuovo cromosoma e rimpiazza la vecchia popolazione con una nuova.
6. Se la condizione di stop è verificata fermati e ritorna la soluzione migliore altrimenti torna a 2.

L'applicazione degli algoritmi genetici rappresenta il cuore del software di schedulazione implementato per risolvere il problema del caso studio descritto nel paragrafo 5. Vengono applicati sulla popolazione iniziale gli operatori genetici *crossover* e *mutation*; il primo al fine di generare i 4 figli delle N/2 coppie scelte casualmente, il secondo per modificarne alcuni geni all'interno dei cromosomi. Per una migliore comprensione degli stessi viene proposto in figura 10 il diagramma di flusso adattato dallo schema generale di Golberg (1989) che procede con i seguenti passaggi:

- l'algoritmo inizia generando casualmente N cromosomi genitori di dimensione N; ad ogni gene è associato un simbolo ad esempio un numero compreso fra 1 e N.
- Vengono valutati tutti gli N genitori e mantenuto il migliore fra questi che rappresenterà il minimo relativo della funzione obiettivo.
- L'operazione successiva consiste nel verificare le condizioni di stop.
- Il software procede applicando una scelta basata su una probabilità Pc (indice di *crossover*) dell'80%; se si rientra all'interno di questa vengono scelte N/2 coppie di genitori e generati 4 figli con l'operatore *crossover*.
- I figli vengono valutati dal modulo valutatore e vengono mantenuti i 2 migliori figli che sostituiranno nella popolazione successiva i propri genitori.
- Si prosegue applicando una probabilità Pm (indice di *mutation*) molto più bassa, pari infatti al 20%; se si rientra all'interno di questa allora si modificano i figli con l'operatore *mutation*.
- Si valuta il figlio con il cromosoma "mutato" e viene mantenuto il migliore fra questo ultimo e l'originale.
- La vecchia popolazione è sostituita con la nuova generazione composta dai migliori figli procedendo di nuovo con *crossover* e *mutation* fino al raggiungimento delle condizioni di stop. Ad ogni step si mantiene sempre l'individuo migliore che alla fine sarà l'ottimo.

## 5. Il caso studio: complessità di gestione dei flussi produttivi di un'azienda motociclistica

La capacità produttiva dell'impianto di un'importante Azienda motociclistica Italiana è di circa 1.800 unità al giorno, utilizzando undici linee di montaggio: le prime quattro linee sono dedicate ai motocicli mentre le sette restanti sono per gli scooters.

La direzione dell'Azienda ha assegnato un numero limitato di modelli ad ogni linea, ottenendo con questa politica linee "multi-modello" caratterizzate da tempi trascurabili di attrezzaggio (*setup*) grazie all'affinità dei modelli montati sulla medesima linea. Questa scelta genera grandi vantaggi in quanto facilita molto la configurazione della linea ed il suo arresto ma, d'altra parte specialmente durante la bassa stagione (figura 11), porta notevoli vincoli alla massimizzazione della

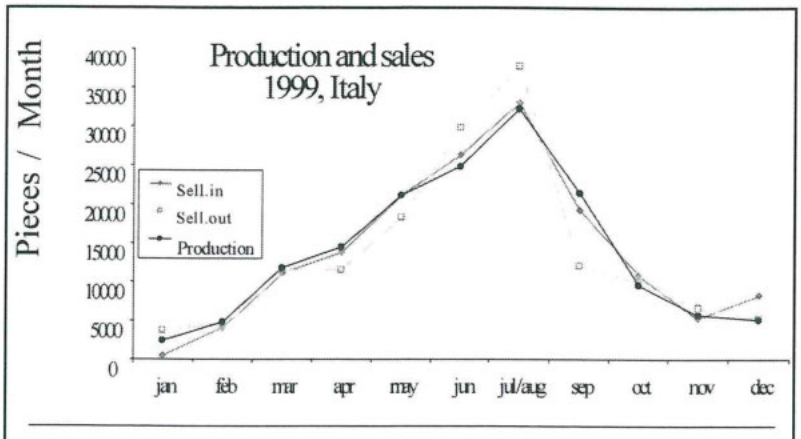



*Figura 11 - Stagionalità nella produzione di motocicli*

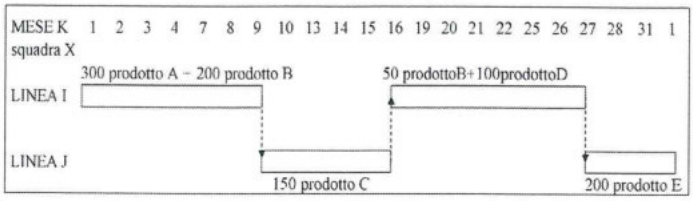



*Figura 12 - Esempio spostamento squadra su due linee*

| STATO DELLA LINEA LASCIATA | STATO DELLA LINEA RAGGIUNTA: PIENO | STATO DELLA LINEA RAGGIUNTA: VUOTO |
|---|---|---|
| PIENO | SPOSTAMENTO PIENO - PIENO | SPOSTAMENTO PIENO - VUOTO |
| VUOTO | SPOSTAMENTO VUOTO - PIENO | SPOSTAMENTO VUOTO - VUOTO |

*Figura 13 - Scelte operative nel chessboard*

saturazione delle linee stesse. Quando la richiesta del mercato non è elevata, e conseguentemente non tutte le linee sono attive, le squadre di lavoro (*workteam*) si dimezzano e sono obbligate a spostarsi a scacchiera (*chessboard*) fra differenti linee. Se ad esempio (figura 12) la squadra X opera sulle linee I e J su cui sono assemblati modelli differenti, si sposta periodicamente fra le due per garantire la produzione di tutte le tipologie di prodotto.

Vi sono quattro possibilità di scelta operativa nel *chessboard* fra le linee (figura 13) come conseguenza di due differenti politiche di "abbandono" della linea. Lasciando la linea "piena" dopo il suo arresto vi rimangono modelli semiassemblati: questo metodo è utile quando si prevede che la squadra tornerà su quella linea e sullo stesso modello entro pochi giorni. Lasciando la linea "vuota" il *workteam* finisce di assemblare tutti i modelli previsti: questa politica è conveniente quando si prevede che in pochi giorni la squadra assemblerà un modello differente su quella linea. Tramite una simulazione software è stato calcolato il tempo perso dagli operatori di una squadra nel *chessboard* nelle quattro situazioni considerando i dati in tabella 2.

| Linea lasciata | Linea raggiunta |
|---|---|
| Tempo ciclo = 54 minuti | Tempo ciclo = 90 minuti |
| Numero operatori = 9 | Numero operatori = 9 |
| Durata ciascuna operazione = 6 minuti | Durata ciascuna operazione = 10 minuti |

*Tabella 2 - Dati utilizzati nella simulazione*

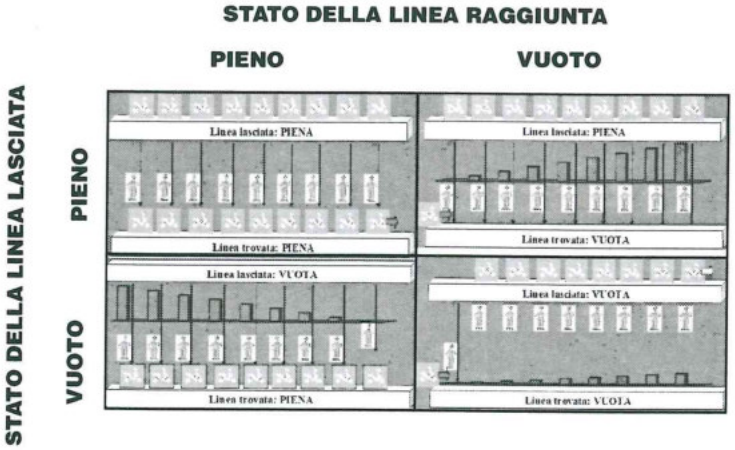



*Figura 14 - Tempo perso nei quattro possibili spostamenti*

**STATO DELLA LINEA RAGGIUNTA**

| STATO DELLA LINEA LASCIATA | PIENO | VUOTO |
|---|---|---|
| PIENO | **NESSUNO** | **TEMPO PERSO** = $TC_{raggiunta} \cdot (N_{raggiunta}-1)/2$ |
| VUOTO | **TEMPO PERSO** = $TC_{lasciata} \cdot (N_{lasciata}-1)/2$ | **TEMPO PERSO** = $\Delta TC \cdot (N-1)/2$ |

*Tabella 3 - Calcolo del tempo perso nei quattro possibili spostamenti*

Come illustrato in figura 14, la circostanza migliore è quella in cui un *workteam* si muove da una linea piena verso un'altra piena, perché non comporta alcuna perdita di tempo. Una situazione comunque abbastanza buona è quella in cui la squadra si muove da una linea vuota verso un'altra linea vuota, procurando solo piccole perdite di tempo per riprendere la produzione.

Le circostanze peggiori si hanno quando la squadra si muove da una linea vuota verso una linea piena e viceversa: in questo caso il primo (ultimo) operaio del *workteam* è obbligato ad aspettare l'ultimo (primo), prima di cominciare a montare il prodotto sulla nuova linea.

Inoltre sono state sviluppate le formule che esprimono il tempo perso con un'ottima approssimazione verificata proprio grazie alla simulazione esposto in tabella 3, ove:

- TC = tempo ciclo;
- ΔTC = differenza, positiva, tra i tempi ciclo delle due linee;
- N = numero di operatori presenti quel giorno sulla linea.

Le perdite di tempo nel cambio lotto (*setup* e *chessboard*) possono essere calcolate sulla base della tabella 4 che riassume la situazione reale dell'azienda.

Nell'Azienda il software di

**PERDITE DI TEMPO**

| | Fisse | Variabili (dipendenti dai tempi di ciclo dei modelli) |
|---|---|---|
| **Spostamento a scacchiera** | tempo necessario per lo spostamento della squadra | Politica di spostamento tra linee (vuoto-vuoto, pieno-pieno, ecc.) |
| **Cambio di lotto su stessa linea** | tempo di spostamento di due morse libere | Adattamento al modello più lento presente |
| **In entrambi i casi** | Motociclo di esempio | |

*Tabella 4 - Perdite di tempo nel cambio lotto*

schedulazione tradizionale utilizzato non valuta la convenienza a lasciare la linea "piena" o "vuota", cioè tenendo conto dei tempi persi nel *chessboard*, in quanto pianifica l'assemblaggio dei modelli schedulando le linee di produzione e non team i *workteam*. Questa situazione crea grosse difficoltà alle persone incaricate del processo di programmazione e spiega il perché dell'interesse allo sviluppo di un nuovo schedulatore con oggetto i *workteam* e che utilizzi in particolare gli Algoritmi Genetici.

### *5.1 Gestione del chessboard e delle curve di efficienza dei workteam*

La gestione dello spostamento a scacchiera delle squadre di lavoro tra le linee in periodo di bassa stagione è un elemento di notevole complessità come sopra descritto.

In bassa stagione, le squadre operanti sulle linee si dimezzano, dunque ogni squadra si trova a lavorare al minimo su due linee ed al massimo su tre. Nessun modello viene mai assemblato su più linee dalla stessa squadra o su una stessa linea da squadre diverse. Nel caso dunque in cui la squadra X si trovi a lavorare sui modelli prodotti sulle linee I e J (figura 12), ad essa afferiranno due tabelle di ordini relative alle due linee; ovviamente due ordini appartenenti a due linee differenti non potranno avere la stessa data di inizio se la squadra che lavora su quelle due linee sarà la stessa.

Per razionalizzare questa situazione sono stati quindi considerati modelli attribuibili a ciascuna squadra in modo da ottenere delle tabelle di ordini con i soli modelli della singola squadra considerata, anche se assemblati su linee diverse; in questo modo si schedula la produzione non più gestendo la programmazione per linee ma per *workteam*.

Un problema derivato dalle caratteristiche di produzione stagionale è inoltre la valutazione della curva di efficienza per le squadre che, in bassa stagione, sono composte da operai con elevata esperienza ed in alta stagione sono composte anche da operai stagionali inesperti. Ad ogni avvio di produzione il personale esperto viene affiancato a squadre di operatori stagionali inesperti e deve fermare la catena di montaggio frequentemente per dar modo ai nuovi operatori di apprendere il proprio lavoro. L'efficienza della squadra dunque parte da valori bassi intorno al 40% per poi salire all'aumentare dei volumi prodotti secondo il principio del *learning by doing*.

Da studi empirici si è quindi giunti a definire una tabella di efficienza (tab. 5) a seconda della composizione del *workteam* e del numero di moto assemblate da una squadra, per ogni modello sul quale questa squadra lavora. Quest'ultima variabile si azzera ad ogni avvio di produzione inteso come passaggio all'assemblaggio di un nuovo modello. L'efficienza diviene dunque un valore parametrico funzione di due variabili (composizione del team e numero di moto).

| | | numero di moto | | | |
|---|---|---|---|---|---|
| | | fino a 500 | 501-2000 | 2001-6000 | >6000 |
| | **personale stagionale** | | | | |
| A | nuovi stag+ciclo new+cad.<4 min | *0.39* | *0.61* | *0.69* | *0.88* |
| B | nuovi stag+ciclo new | *0.46* | *0.67* | *0.80* | *0.91* |
| C | nuovi stag+cad<4min | *0.47* | *0.70* | *0.80* | *0.91* |
| D | nuovi stag | *0.48* | *0.73* | *0.86* | *0.93* |
| | **personale esperto** | | | | |
| E | ciclo new+cad.< 4 min | *0.48* | *0.74* | *0.86* | *0.95* |
| F | ciclo new | *0.49* | *0.77* | *0.87* | *0.95* |
| G | cad.< 4min | *0.56* | *0.79* | *0.90* | *0.97* |
| H | ottimale | *0.67* | *0.91* | *0.96* | *0.97* |

*Tabella 5 - Efficienza dei workteam*

Riassumendo, le due principali criticità evidenziate dall'Azienda sono:

- la gestione dello spostamento a scacchiera dei team fra le linee di produzione (*chessboard*) nei periodi di bassa stagione;
- la realizzazione di piani di produzione fattibili perché non aderenti alla capacità delle linee d'assemblaggio misurata secondo la reale efficienza dei *workteam*.

Per soddisfare queste esigenze si è pensato di:

- schedulare la produzione non più gestendo la programmazione per linee ma per *workteam;*
- controllare la fattibilità dei piani di produzione tenendo conto di vincoli interni (efficienza dei *workteam*) ed esterni (capacità dei fornitori).

## 6. L'approccio proposto per la progettazione dello schedulatore

Al fine di gestire al meglio le complessità evidenziate sopra, si è deciso di modellizzare e sviluppare uno schedulatore e valutatore dei piani schedulati che tenga maggiormente conto delle leve e dei vincoli del modello di fabbrica reale e della *Supply Chain,* ottenendo in tal modo dei piani sicuramente fattibili. Il problema identificato è denominato *permutation flow shop sequencing problem* in cui n job (i lotti nel caso delle linee di produzione) devono essere processati (nello stesso ordine) su m macchine. L'obiettivo è di trovare la permutazione dei job che minimizzerà la durata del lavoro, tipicamente il tempo entro cui l'ultimo job è completato sulla macchina m (makespan). Considerata la complessità del problema ed il tempo di computazione necessario a trovare una soluzione ottima sono stati scelti gli AG per risolvere il problema di schedulazione delle linee di assemblaggio

### *6.1 Dati d'ingresso, principali vincoli e leve*

Il dato di ingresso principale al motore di schedulazione è la tabella degli ordini MPS *(Master Production Schedule)* a cui corrispondono i lotti da produrre pianificati nel medio termine (tab. 6):

| | Linea | Composizione squadra | N° Ordine | Codice prodotto | Descrizione | Quantità | Due date | Start | End | Sequenza |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | L07 | A | MPS.1756 | 64021LL02 | Mtocicdo 50 i.e. | 25 | 03/04/02 | | | 1600 |
| 2 | L07 | A | MPS.1757 | 64021GHB | Mtocicdo 50 i.e. | 32 | 06/04/02 | | | 1600 |
| 3 | L07 | A | MPS.1758 | 64021LL04 | Mtocicdo 50 i.e. | 12 | 12/04/02 | | | 1600 |
| 4 | L08 | A | MPS.1759 | 64140LL04 | Mtocicdo 100-4t | 26 | 12/04/02 | | | 1600 |
| 5 | L07 | A | MPS.1760 | 64021SFOB | Mtocicdo 50 i.e. | 89 | 15/04/02 | | | 1600 |
| 6 | L08 | A | MPS.1761 | 64140LL01 | Mtocicdo 100-4t | 47 | 20/04/02 | | | 1600 |
| 7 | L07 | A | MPS.1762 | 64021LL02 | Mtocicdo 50 i.e. | 125 | 25/04/02 | | | 1600 |
| 8 | L08 | A | MPS.1763 | 64140USOB | Mtocicdo 100-4t | 36 | 15/05/02 | | | 1600 |

*Tabella 6 - Esempio tabella ordini MPS da schedulare*

Altre informazioni importanti sono:

- il tempo di processo nominale (o tempo ciclo) dei modelli che sarà poi pesato con l'efficienza assegnata alla squadra nella linea al momento della produzione seguendo la tabella di efficienza (tab. 5).
- Per schedulare in modo efficace è indispensabile conoscere anche i calendari di lavoro ed il numero degli operatori che compongono un team.

I principali vincoli per la definizione dei piani di assemblaggio sono:

- MPS date: data concordata di consegna del lotto (*due date*).
- Sequenza: codice numerico di prima (macro) aggregazione dei lotti da schedulare derivante dalla pianificazione di medio termine; viene assegnato in fase di definizione del *Master Production Schedule* dall'Azienda e definisce gli ordini che devono essere prodotti su base settimanale.
- La data di rilascio (*release date*), corrispondente alla data di disponibilità dei componenti o almeno per i codici critici.

Le principali leve su cui può operare il pianificatore sono i pesi della funzione obiettivo somma di tre termini che prendono in considerazione il tempo di *chessboard* e di *setup* ed il rispetto della data di consegna (vedi formula 1). Si stabilisce in tal modo l'intenzione dell'Azienda a mantenere il rispetto dei tre vincoli. Il pianificatore prende inoltre decisioni riguardo:

- la composizione delle squadre;
- l'efficienza della squadra (assegnazione dell'opportuna tabella di efficienza);
- l'assegnazione dei modelli alle linee dedicate;

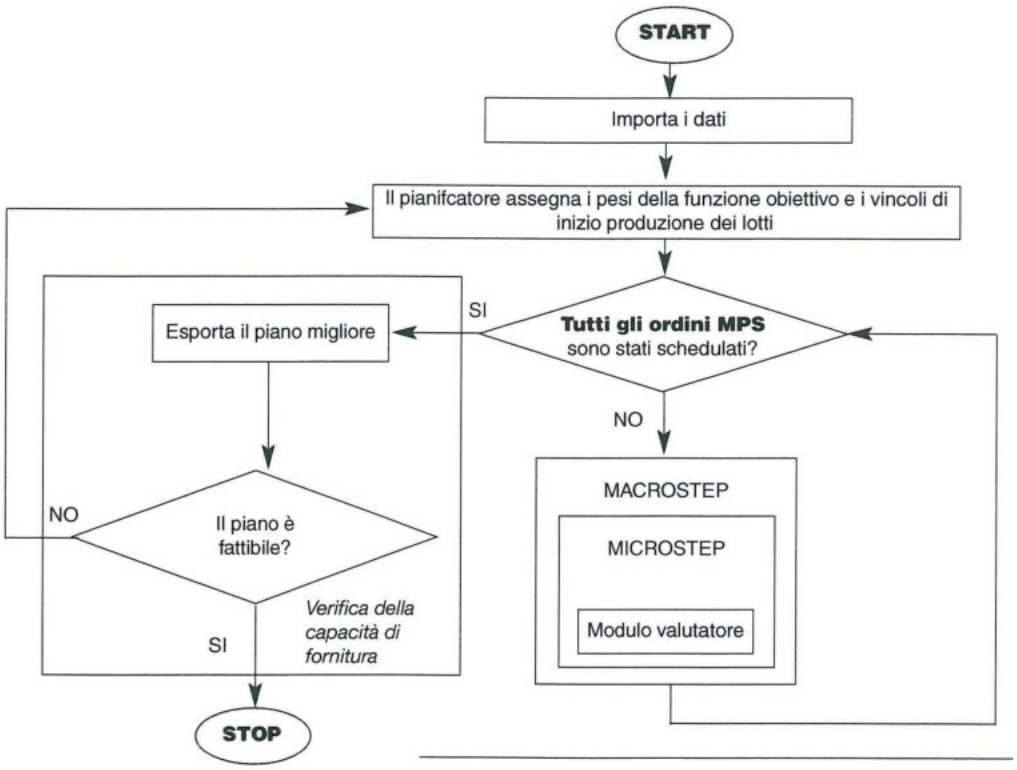



*Figura 15 - Approccio proposto: Macrostep, microstep, modulo valutatore e verifica di capacità*

- l'assegnazione forzata di vincoli sull'inizio di produzione di alcuni ordini (ad esempio ordini preferenziali di partenza).

### *6.2 Macrostep e microstep*

L' approccio risolutivo proposto, illustrato nel diagramma di flusso in figura 15, si basa su due differenti livelli, *un macrostep* e un *microstep*. Le due operazioni avvengono, sugli ordini MPS da schedulare per primi, secondo un numero di sequenza che è strettamente minore delle sequenze corrispondenti ai successivi ordini.

Il *macrostep* si occupa di:

1. Definire il subset di ordini MPS che hanno il più basso numero di sequenza non ancora processato e che devono essere realizzati dal team considerato.
2. Costruire con il *microstep* una nuova famiglia di soluzioni parziali:
3. Scegliere le soluzioni parziali che devono essere conservate per il prossimo *macrostep* con i corrispondenti valori della funzione obiettivo e lo stato finale delle linee.

Allo schedulatore giungono gli ordini MPS, disposti secondo una sequenza numerica, più tutti gli altri dati necessari alla schedulazione (figura 17), e procede poi in tal modo:

- si inizia operando dunque una selezione di questi ordini contenenti uno stesso numero di sequenza, riferendosi naturalmente al team che opera su tali lotti; nell' esempio di tabella 3 verrebbero considerati i primi 8 ordini aventi tutti sequenza 1600 (figura 16).

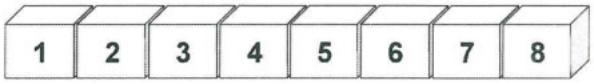



*Figura 16 - Selezione degli ordini*

- Viene operata all'interno di questi ordini una selezione su quelli che sono stati definiti dall'operatore come preferenziali o aventi comunque data di inizio preferenziale; questi ordini vengono stimati mediante il modulo valutatore ed ordinati secondo l'ordine "forzato" immesso manualmente dall'operatore.
- Sugli N ordini rimanenti aventi stessa sequenza, a questo punto il *macrostep* procede con la funzione *microstep* che costruisce delle soluzioni parziali

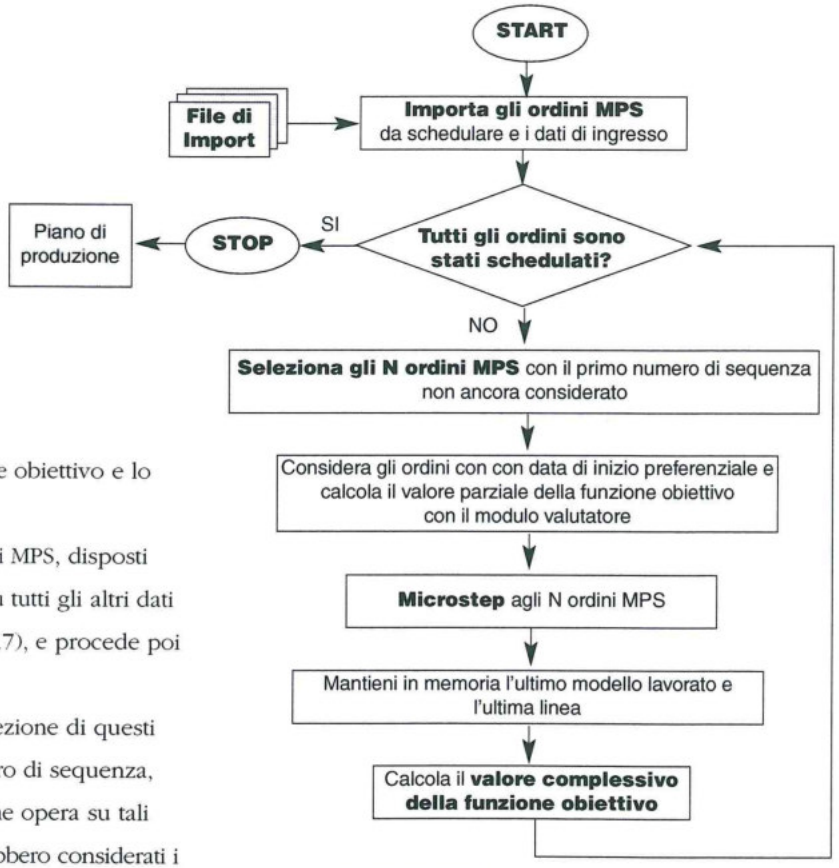



*Figura 17 - Diagramma di flusso del* macrostep

stimandole con il modulo valutatore ed assegnando gli stati finali delle linee.

- Ora il *macrostep* mantiene in memoria lo stato finale delle linee e l'ultimo modello lavorato, utili al *microstep* successivo.
- Viene dunque calcolato il valore complessivo della funzione obiettivo.
- Queste operazioni vengono effettuate ad ogni iterazione dell'algoritmo fino a che tutti gli ordini non vengono schedulati; quando tutti gli ordini in input per un dato team sono stati schedulati, si passa al team successivo fino a completamento.
- Una volta schedulati tutti gli ordini si giungerà dunque al piano di produzione migliore secondo i vincoli predeterminati.

Il *microstep* (figura 20) si occupa invece di:

1. Costruire una nuova famiglia di soluzioni parziali (permutazioni) aggiungendo gli ordini non preferenziali (vedi voce leve paragrafo 6.1) selezionati dal *macrostep*. Per generare la famiglia di soluzioni parziali si è scelto di utilizzare due metodi differenti a seconda che il numero di ordini MPS da sequenziare sia grande o piccolo (N < 10):

- per N <= 10 si è pensato di generare tutte le N! possibili permutazioni e valutarle tramite il modulo valutatore (paragrafo 6.3), scegliendo la permutazione migliore, cioè quella con il valore minimo della funzione obiettivo.
- Per N > 10 invece, si è scelto di applicare gli algoritmi genetici (paragrafo 4) secondo lo schema descritto in figura 10 e mediante l'utilizzo dell'operatore di *crossover* KX e la mutazione di 2 elementi come illustrato in figura 18 e figura 19.

La scelta di operare in modo differente secondo la quantità di ordini è obbligata perché se il numero degli ordini MPS all'interno di una data sequenza è grande, una generazione completa delle N! possibili sequenze (tabella 4) richiederebbe un tempo di computazione troppo elevato visto l'elevatissimo numero di iterazioni necessarie.

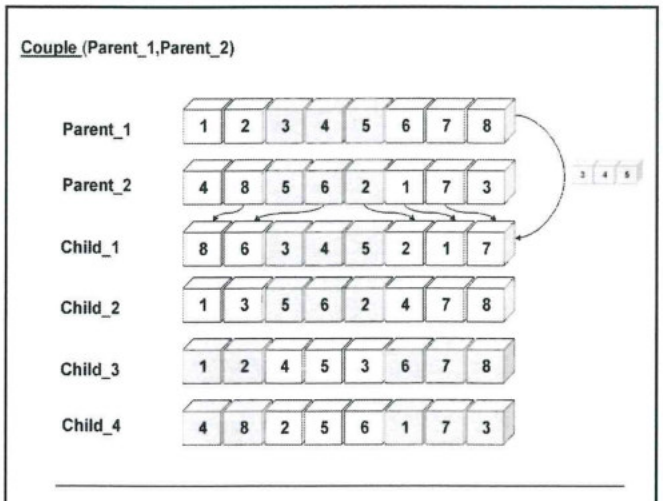



*Figura 18 - Crossover*

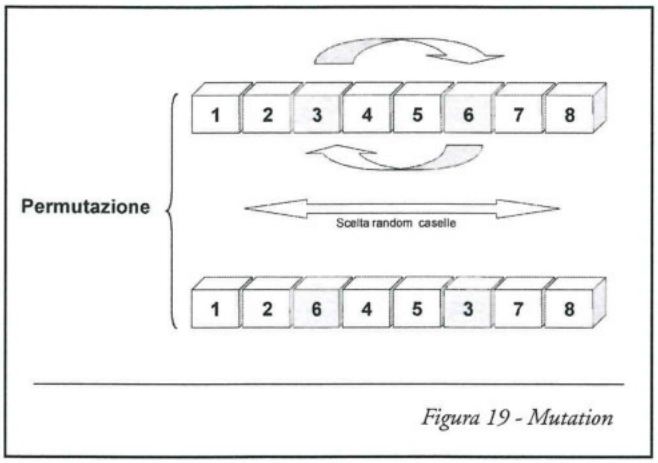



*Figura 19 - Mutation*

| 2! | 3! | 4! | 5! | 6! | 7! | 8! | 9! | 10! | 11! | 12! |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 24 | 120 | 720 | 5040 | 40320 | 362880 | 3628800 | 39916800 | 479001600 |

*Tabella 7 - N!*

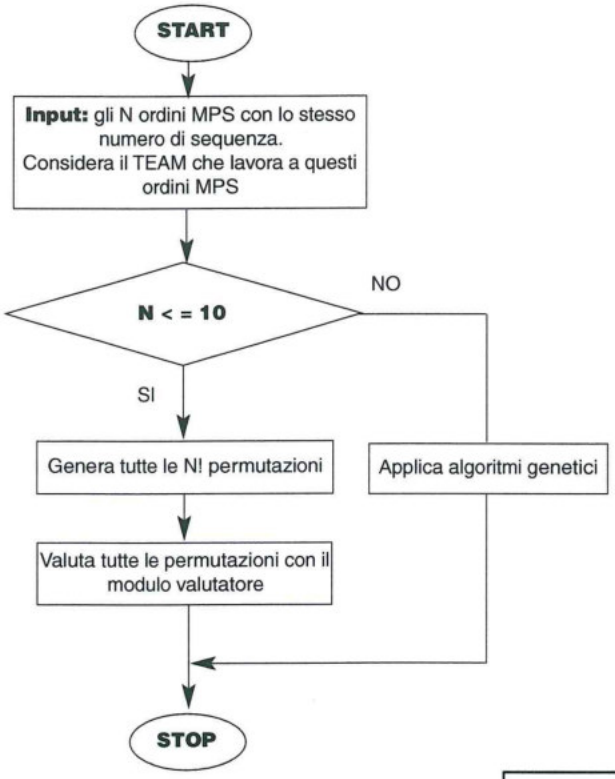



*Figura 20 - Diagramma di flusso del microstep*

2. Calcolare i valori assunti dalla funzioni obiettivo delle soluzioni tramite determinati valori (ritardi, costi, ecc.) collegati al criterio di ottimizzazione adottato;
3. Calcolare lo stato finale delle linee lasciate alla fine di ciascuna soluzione parziale.

Alla fine della schedulazione è presente il controllo della capacità di fornitura come ultima verifica della fattibilità del piano. Lo schedulatore propone dunque un piano schedulato e l'operatore, in conformità ad alcuni requisiti di ottimo, potrà forzare alcune variabili (pesi della funzione obiettivo, posizione e la data d'inizio dei singoli lotti) garantendo una buona flessibilità d'uso dello schedulatore e verificandone poi l'impatto fino al raggiungimento del piano ritenuto ottimo.

In figura 21 è illustrato il procedimento di costruzione delle soluzioni. Ogni sezione costituisce una soluzione parziale ottenuta con ordini MPS con uguale numero di sequenza. Ad ogni iterazione vengono conservate solo le soluzioni migliori (in base alla funzione obiettivo) per proseguire nel processo di schedulazione.

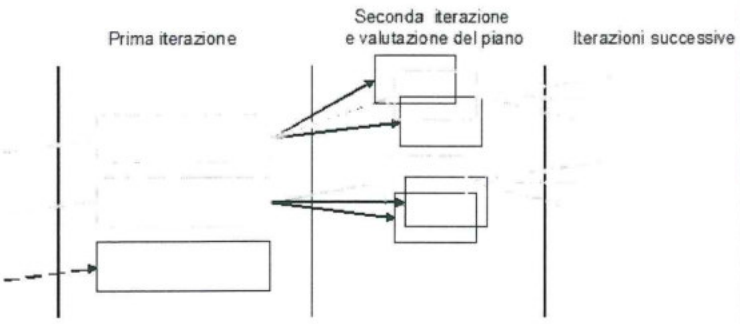



*Figura 21 - Processo di costruzione del piano schedulato*

### *6.3 Modulo valutatore (valutazione del piano tramite funzione obiettivo)*

La qualità di un piano schedulato si misura sulla base di indicatori quantitativi e dunque, a completamento della nuova architettura software, si è integrato un modulo valutatore capace di esprimere un "voto" al piano schedulato sulla base di una serie di indicatori di costo:

- Costi di *setup* delle linee.
- Costi di *chessboard* fra le linee.
- Costi di ritardo rispetto alla data concordata per la consegna.
- Costi d'anticipo rispetto alla data concordata per la consegna.

Il modulo valutatore si occupa della valutazione delle permutazioni generate nel *microstep,* quindi del calcolo degli stati finali delle linee interessate e del valore della funzione obiettivo, utilizzando i pesi della stessa funzione inseriti manualmente dal pianificatore. Assieme agli AG rappresenta il cuore del software di schedulazione implementato in quanto definisce quello che è il *fitness* del cromosoma, secondo quelli che sono i principi portanti degli algoritmi genetici.

A partire da una data permutazione vengono dunque considerati i singoli ordini in successione in modo da poter valutare che peso potrebbe avere uno spostamento di linea, un cambio di modello rimanendo su una stessa linea o semplicemente un cambio di nazionalità o di colore. Tutto ciò al fine di calcolare il valore della funzione obiettivo per ogni permutazione e scegliere la migliore all'interno del *microstep.*

La funzione obiettivo da minimizzare è la seguente:

**Funzione obiettivo =**

$$\alpha_i \bullet \sum SU_{i-1;i} + ß \cdot \sum CB_{i-1;i} \bullet \sum y_i \cdot T_i$$

*Formula 1 - Funzione Obiettivo*

- $\alpha_i$ = peso costo/tempo del tempo perso nel *set-up* della linea i-esima;
- ß = peso costo/tempo del tempo perso nel *chessboard* fra le linee;
- $Y_{i1}$ = peso costo/giorni d'anticipo rispetto alla *due date*;
- $Y_{i2}$ = peso costo/giorni di ritardo rispetto alla *due date;*
- $CB_{i-1;i}$ tempo perso nel chessboard per iniziare l'assemblaggio dell'ordine i-esimo;
- $SU_{i-1;i}$ tempo perso nell'attrezzaggio della linea per assemblare l'ordine i-esimo;
- $T_i$ = giorni di ritardo/anticipo di completamento del lotto rispetto alla due date;

con i indice di ordine e linea, in quanto il modello è associato alla linea e viceversa.

I dati in input del modulo valutatore (figura 22) sono:

- Lotti (ordinati secondo il risultato di una data permutazione): A1,...Ai..., An.
- Stato del team: ultima linea in cui ha lavorato il team.
- Stato della linea: ultimo modello lavorato.
- *Release date:* data più vicina nel tempo in cui può iniziare la produzione del lotto considerato Ai.
- *Due date:* data di consegna del lotto considerato Ai.
- Tempi ciclo di assemblaggio dei modelli.
- Valore della funzione obiettivo assegnato è 0.

Il modulo valutatore procede attraverso i seguenti passaggi:

- La prima operazione che viene compiuta è verificare se tutti gli elementi della permutazione siano stati considerati: in caso affermativo l'algoritmo si ferma perché è stato calcolato il valore della funzione obiettivo, in caso contrario viene preso il primo elemento della permutazione (lotto) non ancora considerato. Nel caso l'elemento considerato sia

effettivamente il primo, la funzione considera lo stato delle linee e del team memorizzati alla fine del *macrostep* precedente.

- Viene verificato se sarà prodotto un lotto di motocicli uguale al precedente. Questa operazione viene effettuata comparando il codice prodotto del lotto considerato con quello precedente.
- Se il codice prodotto è il medesimo allora procede trovando la linea su cui viene lavorato il lotto A(i).
- Se il codice non è lo stesso, si prosegue trovando la linea sia del lotto precedente A(j-1) che di quello considerato A(i).
- E' eseguita una verifica sull'uguaglianza delle linee:

- Se la linea è la medesima, calcola il tempo perso per l'attrezzaggio della linea basandosi sui dati d'input di data in cui ci si trova (al fine di conoscere il personale presente) e di tempo ciclo di un particolare modello; possono esserci due casi entrambi presi in considerazione:

1. Il modello è lo stesso: il tempo perso nel set-up della linea è nullo.
2. Il modello è differente: il tempo di set-up della linea è pari ad una moto d'esempio più due morse vuote.

- Se la linea è differente viene calcolato il tempo di chessboard richiamando i dati riguardanti il tempo ciclo di assemblaggio dei modelli del lotto A(i) e A(i-1) ed il personale presente la data in cui ci si trova.

- Si calcola il momento effettivo di inizio produzione del lotto A(i) sommando il tempo perso in *setup* e *chessboard* alla data ricevuta in input.
- Il modulo valutatore calcola il momento in cui il primo operatore della linea d'assemblaggio può iniziare ad assemblare il lotto successivo A(i+1).
- Ora è calcolato il tempo di completamento e quindi data e minuto di fine produzione del lotto A(i).

Il calcolo del tempo di completamento è eseguito invece mediante l'espressione:

$$c_i = \max(t_k, r_i) + SU_{i-1;i} + CB_{i-1;i} + P_i$$

*Formula 2 - Tempo di completamento*

- $t_k$ Tempo di linea libera;

$$ri = \max[(t_{now} + to_i); t_{pref}]$$

*Formula 3 - Release date del lotto i;*

- $t_{now}$ Tempo di lancio del processo di schedulazione;
- $to_i$ Peggior lead time per l'ordine dei componenti del lotto i;
- $t_{pref}$ Tempo preferito per l'inizio del lotto;
- $P_i$ Processing time del lotto i (dipende dal modello, dal numero di pezzi del lotto e dall'efficienza considerata).
- Verifica se la data di fine produzione è uguale alla *due date* del lotto; in caso contrario calcola i giorni di ritardo o anticipo rispetto alla stessa.
- L'ultima operazione che esegue il modulo valutatore è il calcolo del valore della funzione obiettivo da sommare al valore parziale applicando un peso α al tempo perso nell'eventuale set-up della line, un peso ß al tempo perso nell'eventuale *chessboard* fra due linee e i pesi y1 e y2 al tempo di ritardo, o d'anticipo, rispetto alla due date definiti dal pianificatore.

### *6.4 Dati in uscita*

Oltre a riottenere i dati di input utili alla visualizzazione grafica (fig. 23) si ottengono anche i seguenti dati:

- *Start date:* data e minuto di partenza della produzione del lotto considerato;
- *End date:* data e minuto di fine produzione del lotto considerato.

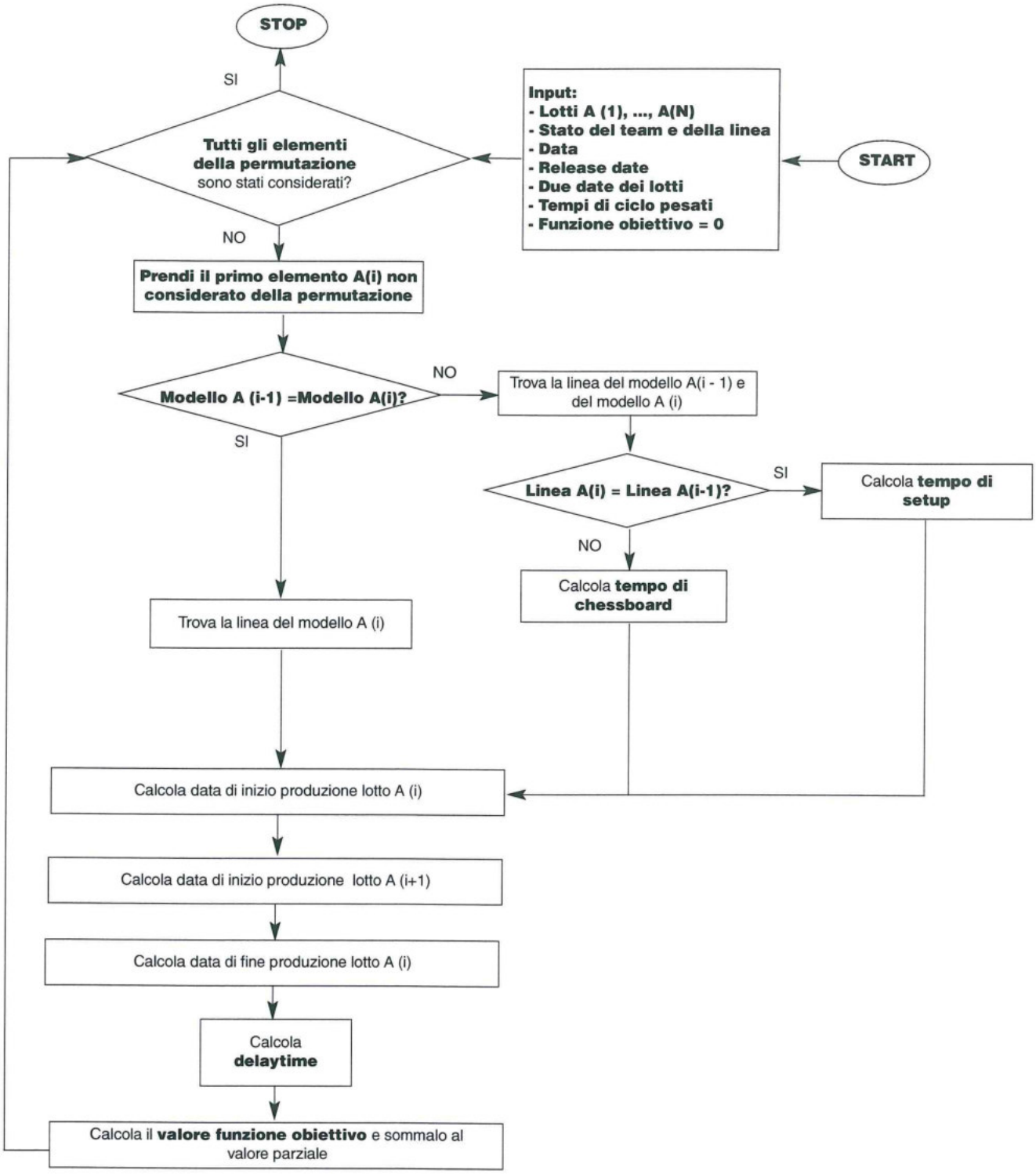
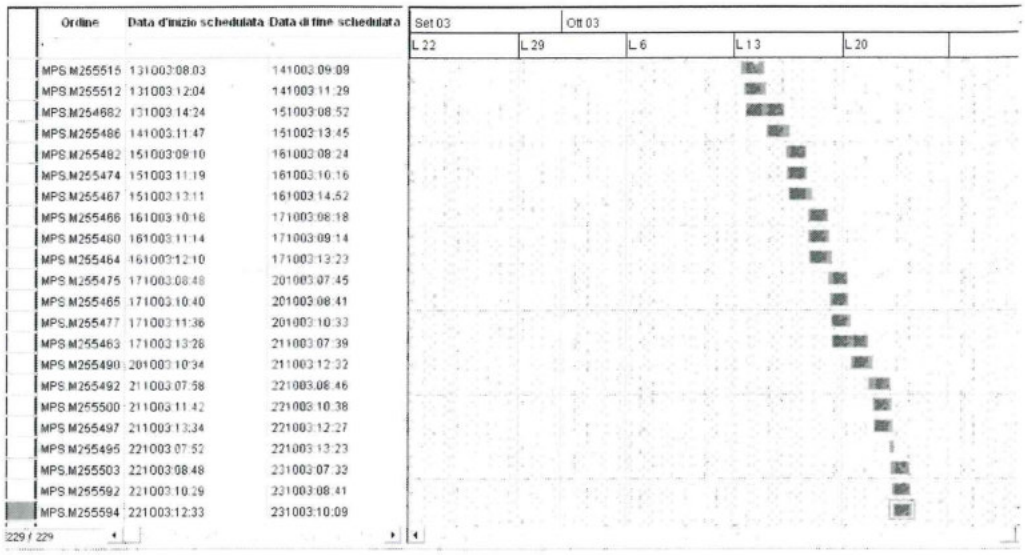



*Figura 22 - Diagramma di flusso del modulo valutatore*



| Ordine | Data d'inizio schedulata | Data di fine schedulata |
|---|---|---|
| MPS M255515 | 131003:08.03 | 141003:09:09 |
| MPS M255512 | 131003:12:04 | 141003:11:29 |
| MPS M254682 | 131003:14:24 | 151003:08:52 |
| MPS M255486 | 141003:11:47 | 151003:13:45 |
| MPS M255482 | 151003:09:10 | 161003:08:24 |
| MPS M255474 | 151003:11:19 | 161003:10:16 |
| MPS M255467 | 151003:13:11 | 161003:14:52 |
| MPS M255466 | 161003:10:16 | 171003:08:18 |
| MPS M255460 | 161003:11:14 | 171003:09:14 |
| MPS M255464 | 161003:12:10 | 171003:13:23 |
| MPS M255475 | 171003:08:48 | 201003:07:45 |
| MPS M255465 | 171003:10:40 | 201003:08:41 |
| MPS M255477 | 171003:11:36 | 201003:10:33 |
| MPS M255463 | 171003:13:28 | 211003:07:39 |
| MPS M255490 | 201003:10:34 | 211003:12:32 |
| MPS M255492 | 211003:07:58 | 221003:08:46 |
| MPS M255500 | 211003:11:42 | 221003:10:38 |
| MPS M255497 | 211003:13:34 | 221003:12:27 |
| MPS M255495 | 221003:07:52 | 221003:13:23 |
| MPS M255503 | 221003:08:48 | 231003:07:33 |
| MPS M255592 | 221003:10:29 | 231003:08:41 |
| MPS M255594 | 221003:12:33 | 231003:10:09 |

229 / 229



*Figura 23 - Piano schedulato visualizzato tramite Graphic User Interface (GUI)*

Gli ordini MPS sono inoltre ordinati secondo l'ottimizzazione dello schedulatore con una nuova sequenza; se ad esempio prima si avevano 8 ordini con numero di sequenza 1600 ora gli verranno assegnati i valori 1610,1620,1630,...1680.

## 7. Conclusioni

Il prototipo software realizzato seguendo le indicazioni progettuali descritte nel paragrafo 6 è stato testato con un piano di produzione quadrimestrale relativo a due linee di assemblaggio su cui operava in bassa stagione un solo *workteam*. L'elaborazione della schedulazione della produzione viene effettuata dal prototipo software in un intervallo di tempo compreso fra i 2'30" e i 3 minuti, a seconda dell' hardware su cui viene installato. Il tempo di computazione appare sicuramente adatto alle esigenze Aziendali in cui la pianificazione viene effettuata giornalmente o settimanalmente.

Il principale obiettivo dello schedulatore, ovvero la realizzazione di piani fattibili che tenessero conto della reale situazione dell'azienda, è stato pienamente raggiunto dimostrando significativi risparmi in termini di tempo perso in *setup* e *chessboard* e garantendo una piena aderenza alle date di consegna imposte come dato di ingresso al motore di schedulazione.

I reali vantaggi, oltre che nel tempo di computazione (bassissimo rispetto alla complessità del problema), si hanno in termini di flessibilità e consistenza del piano:

- la possibilità di cambiare i pesi della funzione obiettivo genera più piani di assemblaggio ottimizzati tenendo conto di differenti politiche produttive.
- la schedulazione a capacità finita verifica le reali capacità sia interne (efficienza dei *workteam*) che esterne (capacità di fornitura).

La generalità degli assunti presi in fase di progettazione rendono il prodotto software *general purpose* e customizzabile a realtà differenti permettendo ulteriori sviluppi futuri ed una sua applicazione in altri contesti, dove la gestione degli operai fra le linee ha un grande impatto nelle performance produttive.

[1] *La Ricerca Scatter, proposta da Glover (1977), prende in considerazione una popolazione di punti di riferimento e genera dei successori attraverso delle combinazioni lineari pesate .*

[2] *L'ottimizzazione Ant Colony è basata su modello probabilistico parametrizzato (utilizzato nella rappresentazione chimica del feromone) che costruisce le soluzioni incrementalmente aggiungendo i componenti della soluzione definita alla soluzione parziale considerata. Per quanto riguarda gli algoritmi formica si rimanda anche all'articolo "Ottimizzare la distribuzione merci. Il modello delle formiche" di Luca Maria Gambardella, pubblicato sul numero 7/2004 di QdM*

**Bibliografia**


- Arabeyre, J.,Fearnley J., Steiger F., Teather W., The airline Crew scheduling problem: a survey, *Transportation Science 3*, pp 140-163, 1969
- Darwin, C., *Origin of species*, 1859
- Eiben A.E., Schippers C.A., On evolutionary exploration and exploitation, *Fundamentae Informaticae*, Vol. 35, pp. 35-50, 1998.
- Fogel L.J., Owens A.J., Walsh M.J., *Artificial Intelligence through Simulated Evolution*, New York, John Wiley, 1966;
- Glover F., Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences*, 1977;
- Glover F., Kochenberger G. eds., *Handbook of Metaheuristics*, Kluwer Academic Publishers, 2002.
- Goldberg D.E., *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison-Wesley Publishing Company, 1989;
- Hall R.W., *Queuing Methods*, Prentice Hall, p. 215, 1991
- Holland J. H., *Adaptation in Natural and Artificial Systems.*, Ann Arbor, University of Michigan Press, 1975;
- Koza J.R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992;
- Levine D., Genetic Algorithms: a Practitioner's View, *INFORMS Journal of Computing*, Vol 9 No 3, pp 256-259, 1997;
- Pesenti R., Teoria della complessità, concetti fondamentali, appunti di lezione, http://www.unipa.it/~pesenti/, 2002
- Pesenti R., Euristiche ed approssimazioni, appunti di lezione, http://www.unipa.it/~pesenti/, 2002
- Portmann M.C. "Scheduling Methodology: Optimisation and Compu-search Approaches I", the *Planning and Scheduling of Production System: Methodologies and Applications*, Capitolo 9, ed. Artiba A., Elmaghraby S. E. Chapman & Hall, pp 271-300, 1997
- Rechenberg I., *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution*, Stuttgart, Frommann-Holzboorg, 1973;
- Reeves C.R., A Genetic Algorithm for Flowshop Sequencing, *Computers operational Research*, Vol. 22, N 1, pp 5-13,1995;
- Reeves C.R., Genetic Algorithms: No Panacea, but a Valuable Tool for the Operations Researcher, *INFORMS Journal of Computing*, Vol 9 No 3, pp 263-265, 1997;
- Roli A., An introduction to metaheuristics, Seminario: "Metaeuristiche e problemi di ottimizzazione combinatoria", http://www-lia.deis.unibo.it/Courses/AI/200203/ModuloB/ Seminari.html, 2003
- Ross P., What are Genetic Algorithms Good at?, *INFORMS Journal of Computing*, Vol 9 No 3, pp 260-262, 1997;
- Schwefel H.P., *Numerical Optimisation for Computer Models*, Wiley, 1981;
- Zitzler E., *A Tutorial on Evolutionary Tutorial on Evolutionary Multiobjective Optimisation*, ETH Zurich, 2002