

## **Sommario**

***L**a tesi, articolata in sei capitoli, si propone di presentare il problema del layout fieristico in un'ottica alquanto innovativa. L'attenta e prolungata ricerca effettuata a inizio tesi e riportata nei primi capitoli ha infatti portato alla luce numerose pubblicazioni vicine al problema, ma nessuna che fosse direttamente focalizzata su di esso. Per pubblicazioni vicine al problema si intendono quegli scritti incentrati sui problemi di Knapsack, di Cutting, di Packing e sul Facility Layout Problem.*

*La seconda parte della tesi è invece volta a delineare una panoramica esauriente dei modelli matematici studiati per l'ottimizzazione del layout fieristico, nonché a presentare i risultati della loro implementazione attraverso il software MP-Xpress.*

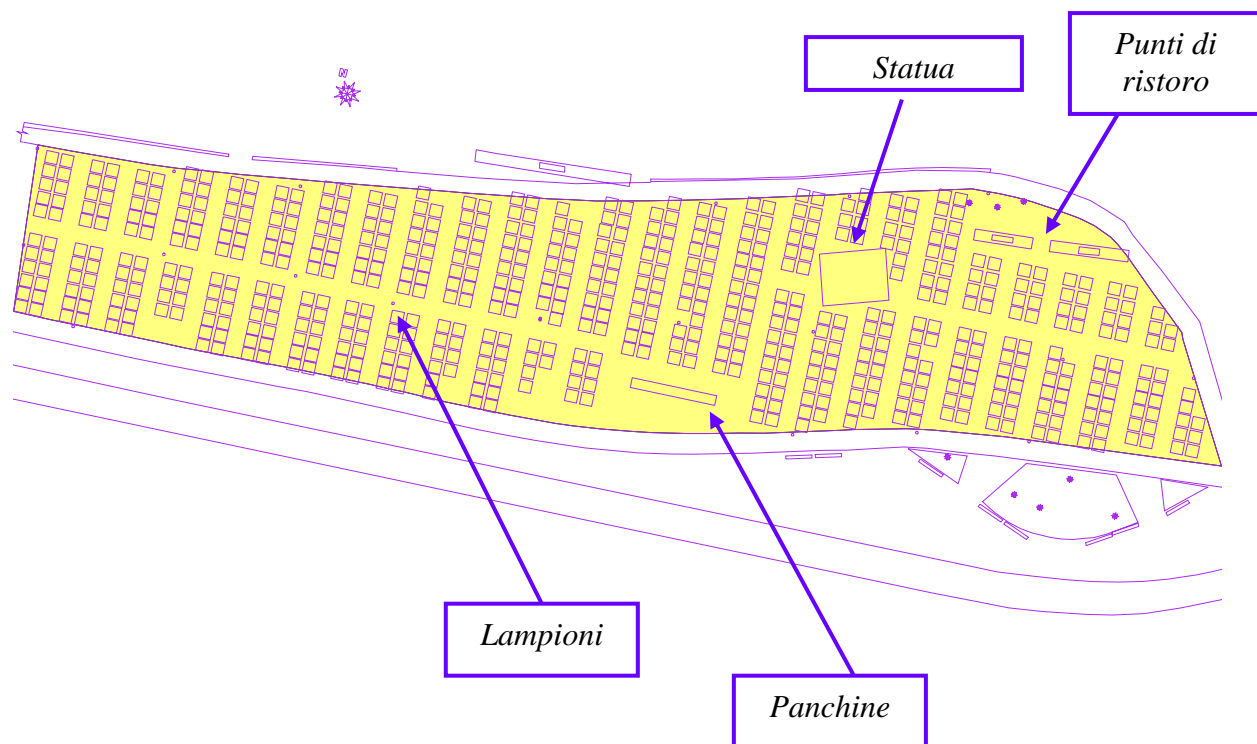
*Proprio l'implementazione ha permesso di provare il corretto funzionamento dei modelli che, nati per risolvere il problema in una famosa fiera dell'artigianato a Fortaleza (Brasile), sono stati qui implementati per le aree espositive delle Fiere di Reggio Emilia. L'implementazione dei modelli matematici su due contesti così diversi ha permesso di provare la loro generalizzabilità; essi possono essere applicati in qualsiasi contesto fieristico e in aree espositive di qualsiasi forma e grandezza.*

## Introduzione

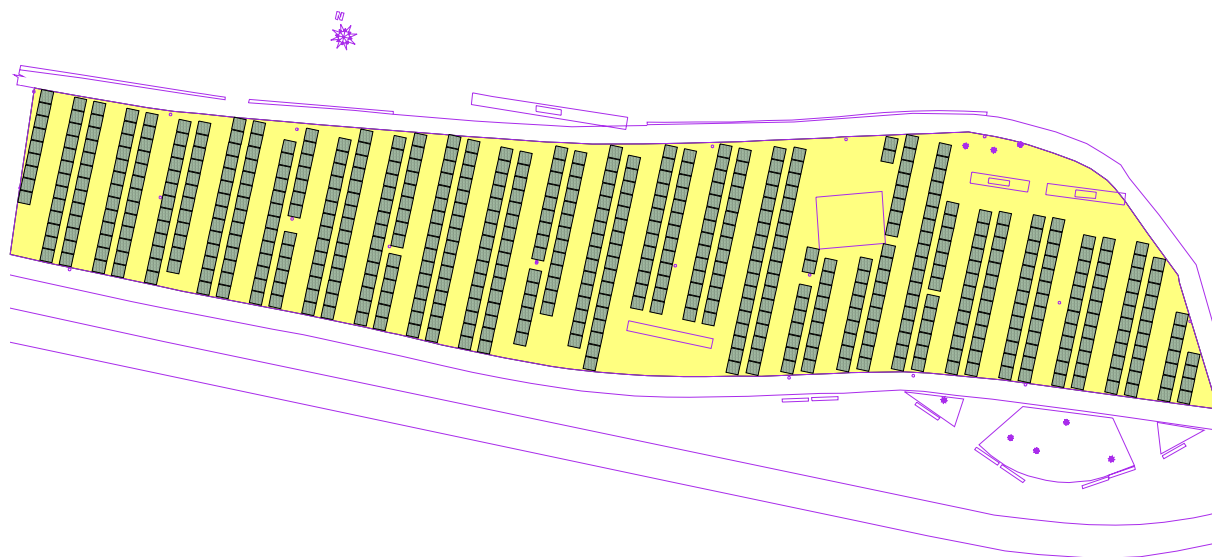
**I**l flusso portante della tesi è il layout fieristico; in particolare esso è nato da un problema di ottimizzazione presente in un contesto reale, ovvero una famosa fiera di artigianato che da anni è presente sul lungomare di Fortaleza (Brasile). La fiera accoglie ogni giorno moltissimi artisti che producono oggetti d'artigianato di qualsiasi natura e di alta qualità ed è posizionata ai piedi della lunghissima spiaggia della città. Ogni giorno circa seicento artigiani si recano all'altezza della spiaggia di Meireles e assemblano e smontano i loro stand. L'area che circonda la fiera è una delle zone più attive della città ed è piena di noti hotel e ristoranti. E' stata l'importanza di questa fiera, data dal grande numero di persone che vi lavorano e dall'enorme affluenza di visitatori, e la complessità dell'ambiente entro cui essa è collocata, che hanno spinto l'Università di Ceará a trasformare la disposizione degli stand dei cosiddetti Juarez in un vero e proprio problema di ottimizzazione. Questo grande ammontare di stand ha infatti bisogno di una disposizione meno caotica e più razionale che renda possibile l'inserimento di più artigiani possibile.

Lo scopo della ricerca è quindi quello di massimizzare il numero di stand che possono essere contenuti nella superficie adibita alla fiera, cercando di ottenere un layout accettabile, ovvero un layout che consideri anche tutti gli spazi necessari per il passaggio delle persone e per il montaggio/smontaggio degli stand. Tutto ciò tenendo ben presente che la superficie disponibile per l'esposizione è di forma irregolare e non convessa ( all'interno e attorno alla superficie sono infatti presenti numerosi ostacoli: palme, sculture, panchine..) e che gli stand sono tutti uguali e rettangolari. Proprio queste ultime considerazioni, rendono il problema generalizzabile ad ogni altro contesto fieristico e quindi particolarmente interessante. Tutto il percorso di ricerca si è infatti concluso con l'implementazione di modelli matematici sulle fiere di Reggio Emilia, esperimento che sottolinea come la tesi possa essere utilizzata anche in uno scenario molto vicino all'ambiente in cui è stata sviluppata.

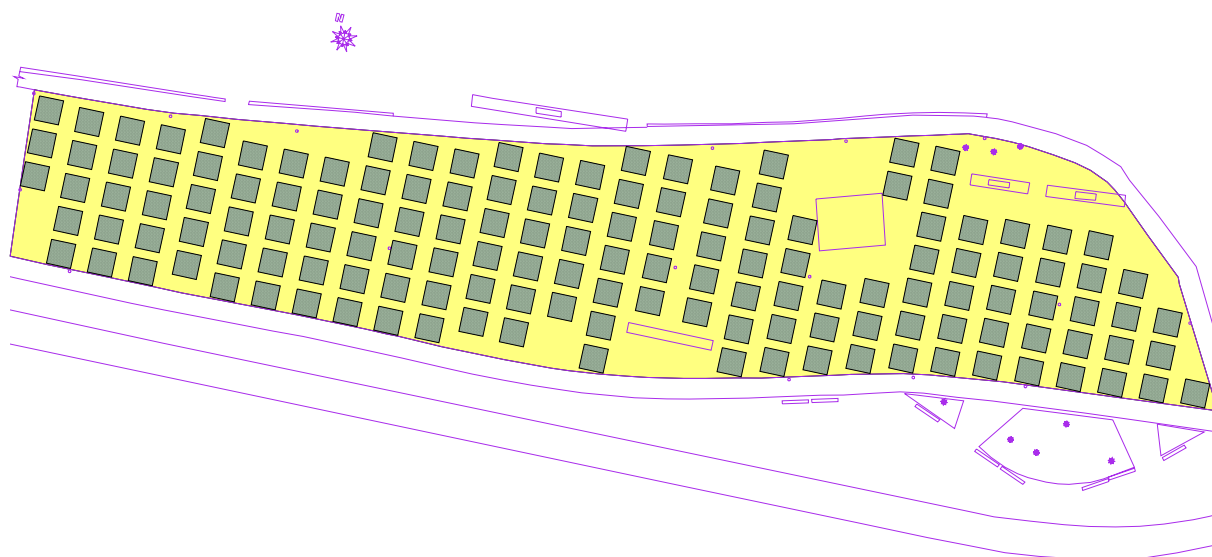
Nelle immagini seguenti è riportata la disposizione degli stand presente alla fiera di Fortaleza inizialmente, nonché due dei layout creati grazie all'implementazione dei modelli che verranno presentati nella tesi.



*Figura 1: Veduta aerea della fiera di Fortaleza. La disposizione degli stand è quella attuale. Alcuni stand fuoriescono dalla superficie d'esposizione (colorata in giallo) e vi sono ampi spazi inutilizzati per la presenza di lampioni, panchine, sculture e punti di ristoro.*



*Figura 2: Una delle soluzioni trovate dopo l'implementazione dei modelli. Gli stand sono tutti completamente contenuti nell'area espositiva, si è riempita anche l'area intorno agli ostacoli ed è stato preso in considerazione il fatto che nessun artigiano volesse stare dietro alla statua.*



*Figura 3: Nuovo layout fieristico calcolato supponendo di poter piazzare gli stand in gruppi di quattro.*

La prima cosa che è stata fatta per avvicinarsi al problema è stata quella di analizzare il concetto di layout dal punto di vista logistico, andando in particolare a riferirsi al layout per eccellenza, ovvero a quello industriale. Tutto questo è riportato nel capitolo 1, il quale si conclude sottolineando il divario tra layout industriale e layout fieristico: quest'ultimo non può essere paragonato al primo in quanto non esiste flusso di materiale tra uno stand e un altro, cosa invece di primaria importanza tra un macchinario ed un altro in uno stabilimento. Si noti inoltre che per il layout industriale sono stati trovati molti riferimenti bibliografici, mentre per quanto riguarda il layout fieristico non è stata trovata alcuna pubblicazione già esistente incentrata su tale argomento.

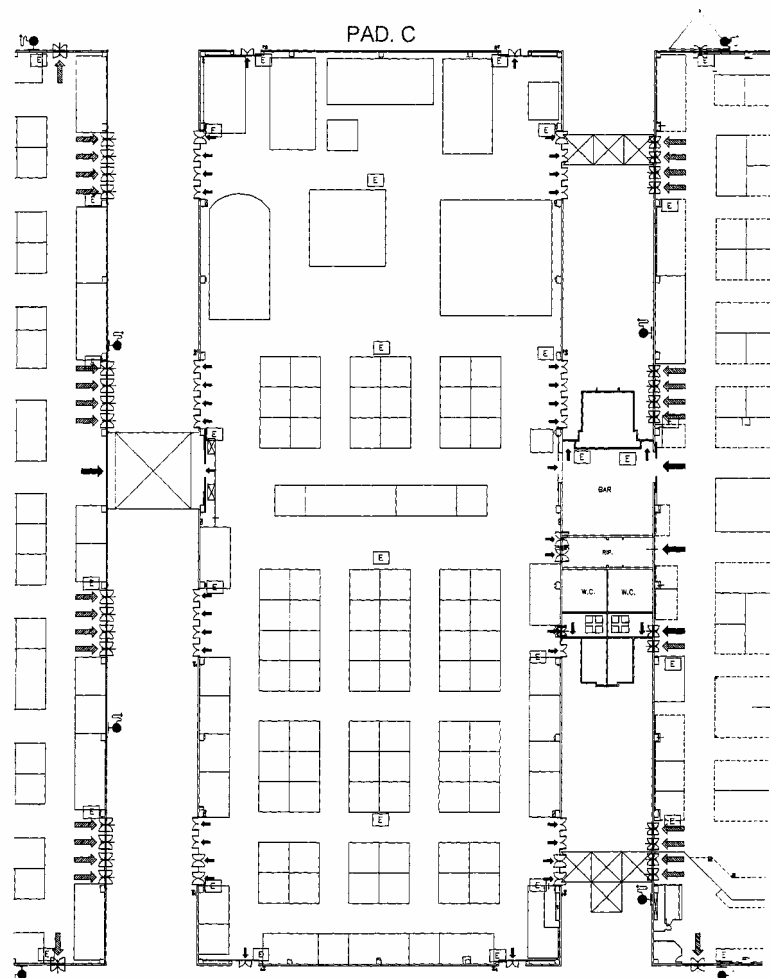
Si è poi pensato di continuare il lavoro, andando a cercare nel vasto campo della ricerca operativa: nel capitolo 2 sono stati presi in considerazione decine di articoli inerenti a problemi sull'ottimizzazione combinatoria che si pensava potessero in qualche modo essere vicini al problema del layout fieristico, ovvero i problemi di *Cutting* e di *Packing*. Essi sono stati presi in considerazione a partire dalle prime formulazioni matematiche fino agli ultimi approcci pubblicati.

L'area della ricerca operativa è stata ulteriormente analizzata andando a indagare sul Facility Layout Problem, ovvero sull'allocazione di un certo spazio disponibile a una varietà di attività che hanno diverse relazioni tra di loro. Riguardo quest'ultimo, nel capitolo 3, sono stati riportati i vari modelli matematici che lo risolvono, tra i quali spicca la formulazione del Quadratic Assignment Problem.

Con il capitolo 3 si conclude la parte di ricerca bibliografica della tesi: essa è stata molto interessante perché ha fatto emergere molti approcci, modelli e formulazioni utili per avvicinarsi al problema del layout fieristico, ma allo stesso tempo non ha portato alla luce alcuna recensione già effettuata sull'argomento. Proprio quest'ultima cosa ha reso molto stimolante la tesi aumentando la convinzione di essere in procinto di scrivere qualcosa di nuovo, o comunque di aver guardato un problema in un'ottica diversa dal solito.

Finita la ricerca bibliografica, nel capitolo 4 è stato spiegato in modo dettagliato come è stato affrontato il problema del layout fieristico ( *FLOP - Fair Layout Problem* ) e sono stati illustrati e spiegati uno a uno i modelli ideati per la risoluzione. In particolare i modelli sono stati inseriti dal più semplice al più difficile, dal più scarno al più generale. Nel capitolo 5 sono riportati tutti i modelli descritti nel capitolo 4, riscritti in linguaggio Mosel, linguaggio necessario per l'esecuzione degli stessi sul software Xpress-Mp.

Nel capitolo 6 sono presentati tutti i risultati ottenuti dall'implementazione, considerando che le matrici date in input ai programmi sotto forma di file dati (.dat), sono matrici ottenute dalle piantine delle Fiere di Reggio Emilia. Queste ultime sono state prese come esempio in modo da sottolineare come il problema del FLOP, anche se nato in un contesto estero, possa essere utilizzato in qualsiasi contesto fieristico e quindi anche in una realtà molto vicina all'ambiente di sviluppo della tesi.



*Figura 4: Piantina di un padiglione delle Fiere di Reggio Emilia*

Importante precisare che i file dati in input sono solo a titolo di esempio; i modelli sono infatti generalizzabili a qualsiasi tipo di superficie e a qualsiasi tipo di situazione.

La tesi termina con le conclusioni che accolgono al loro interno un riassunto degli obiettivi raggiunti e una riflessione sull'applicabilità e l'utilità dei modelli.

## Capitolo 1

## DAL LAYOUT INDUSTRIALE AL LAYOUT FIERISTICO

### 1.1 IL LAYOUT NELLE IMPRESE

**L**a letteratura è ricca di tematiche riguardanti lo studio e la progettazione del layout nell'ambito delle industrie; è proprio da questa raccolta di informazioni ormai già ampiamente discusse e approfondite che partirà questo capitolo, per poi arrivare all'estensione di tale problema all'ambito fieristico.

L'American Institute of Industrial Engineers ha definito il layout come

“*un' organizzazione della produzione che ha per oggetto la progettazione, la messa in opera, la manutenzione e il miglioramento di sistemi integrati di uomini, macchine e materiali; facendo uso di metodi e tecniche tratti dalle scienze matematiche, fisiche e sociali, oltre che dai criteri dell'analisi economica, essa deve definire gli obiettivi di tali sistemi integrati, valutare preventivamente e controllare i risultati ottenuti*”

Esso quindi non contempla la progettazione e l'attuazione della disposizione ottimale delle sole attrezzature industriali, ma anche della mano d'opera, delle scorte, dei trasporti interni.

E' interessante ricordare anche un'altra definizione diffusa negli Stati Uniti che sottolinea le strettissime relazioni che intercorrono tra layout e materials handling:














“*Pianificare e integrare i percorsi delle parti componenti il prodotto, in modo da ottenere la più efficace ed economica interdipendenza tra l'uomo, le attrezzature e la movimentazione dei materiali, a partire dalla ricezione, attraverso la produzione, fino ad arrivare alla spedizione dei prodotti finiti.*”

Riguardo alla relazione tra i due fattori appena citati è stato stimato che il 30-75 % dei costi di un'impresa manifatturiera sono attribuibili alla movimentazione di materiale (*handling*) e al layout (Chiang and Kouvelis , 1996). Un buon layout aiuterebbe quindi a diminuire i costi, ma questo perché facilita l'introduzione di tecniche di riduzione dei

tempi di set-up e di team di lavoro, aiuta a ridurre il più possibile i tempi di trasporto materiale, la quantità di work in progress e i lead times. Detto ciò è facile pensare: basta cercare il layout migliore per risolvere molti problemi presenti in un'azienda. Ma il problema è proprio qui: qual è il layout migliore?

In termini molto generali si può dire che il layout ottimale è quello che consente di soddisfare nel modo migliore le esigenze di tutti gli interessati, cioè la direzione dell'azienda, i suoi dipendenti e i suoi azionisti. E' vero che ciascuno di questi gruppi ha le sue ragioni per desiderare un layout ottimale, ma è altrettanto vero che accontentare tutti non è facile e porta a dover considerare un numero pressoché infinito di fattori interdipendenti influenti sulla sistemazione planimetrica degli impianti che rende ancora oggi difficile riuscire ad avere una soluzione su base scientifica per un problema di ottimizzazione del lay-out.

Ecco un elenco dei principali fattori di cui si deve tener conto nella progettazione del lay-out industriale:

-  Complessità e tipologia del processo produttivo
-  Volumi di produzione
-  Domanda del mercato
-  Trasporti interni di materiali e persone
-  Flusso dei materiali
-  Efficace utilizzo dello spazio disponibile
-  Massima utilizzazione degli impianti
-  Efficace utilizzo della manodopera
-  Condizioni ambientali soddisfacenti e di massima sicurezza
-  Capitali disponibili per gli investimenti
-  Flessibilità e capacità di adeguamento a variazioni di produzione
-  Previsione di ampliamenti futuri
-  .....

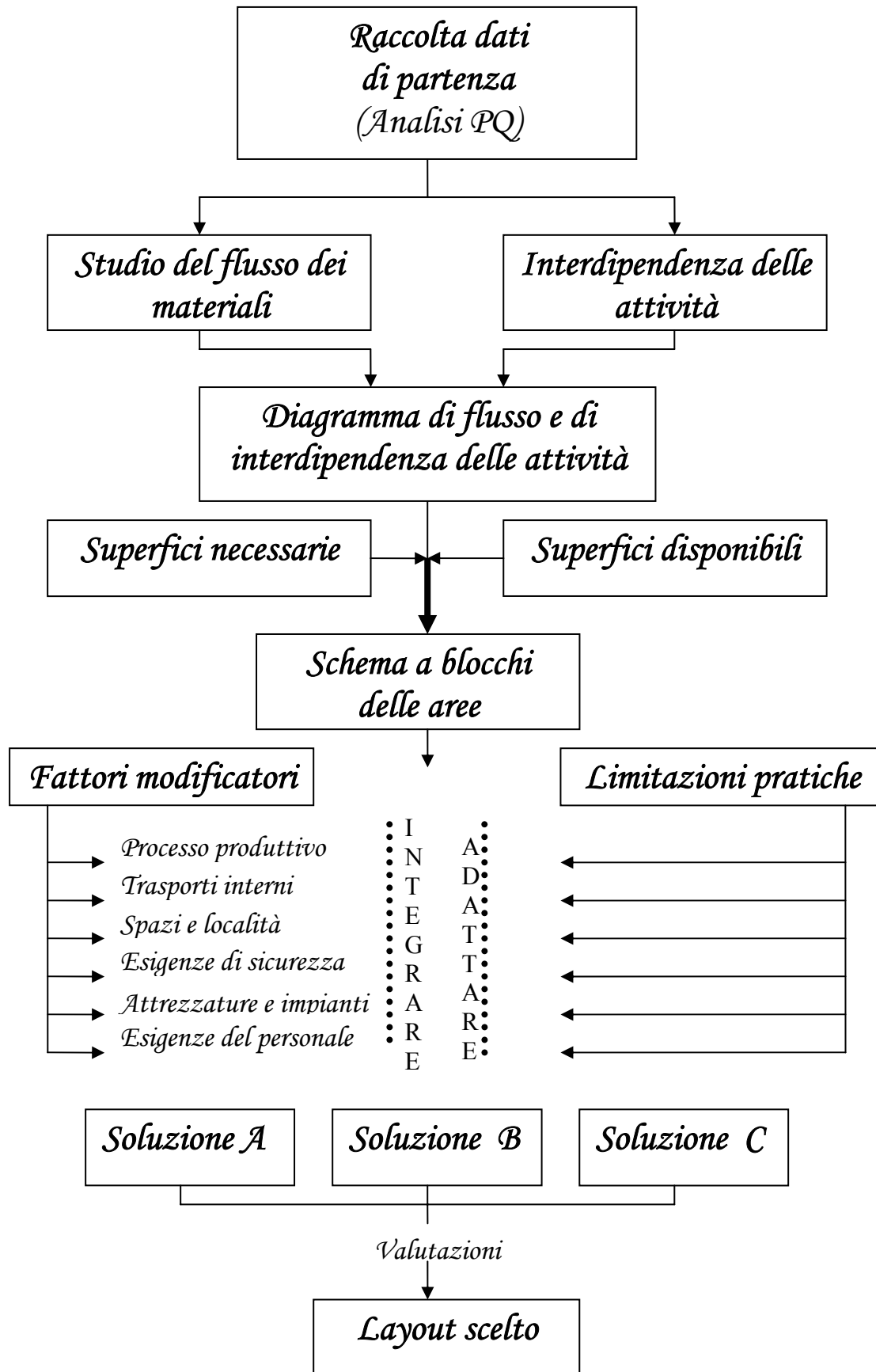
E' quindi importante sottolineare l'importanza delle implicazioni logistiche quali l'ottimizzazione dei flussi, le distanze da percorrere, la composizione e i volumi di scorte, i sistemi di trasporto interno, essenziali per le decisioni nella disposizione reciproca tra le diverse macchine. Soffermendosi in particolare sull'implicazione dei "flussi", si ricordi che non ci si deve fermare a considerare il mero flusso "fisico",



ma nel calcolare quanto un reparto o un macchinario debba distanziarsi da un altro impianto, si deve tener conto anche dell'enorme quantità di informazioni ovvero pratiche d'ufficio che vengono scambiate o comunque delle attività tra persone che devono essere fortemente integrate tra loro.

L'insieme dei dati da raccogliere ed elaborare sono spesso di varia natura e in quantità notevoli: dall'elenco di voci (o articoli) da produrre e da immagazzinare e relative quantità, alla successione delle operazioni richieste da ogni prodotto (ciclo di lavorazione); dai volumi, pesi e caratteristiche dei materiali da trasportare lungo il processo di lavorazione, al numero, al tipo e alle caratteristiche delle macchine e degli impianti occorrenti; dalla manodopera e personale necessario alle esigenze di servizi generali e dei magazzini. Dopo la raccolta dei dati di partenza, lo studio di un layout ricerca le soluzioni possibili (queste dipendono spesso in modo rilevante dai diversi tentativi di ottimizzazione dei flussi logistici e dai diversi sistemi di trasporto interni adottabili) e sceglie la soluzione migliore. Infatti al di là delle complessità e difficoltà, spesso notevoli, nella raccolta ed elaborazione dei dati, lo studio sistematico del layout dovrebbe poter prevedere almeno a livello generale alcune possibili soluzioni alternative.





Nel grafico seguente si riporta un possibile approccio al progetto di un layout industriale:



Schema 1.1 – Approccio al progetto di un layout industriale

Nonostante tutti questi vincoli, si è cercato in questi anni di rispondere all' esigenza di integrare processo, lavoro umano e flusso di attività di ogni azienda trovando alcune tipologie di lay-out diversificate tra loro a seconda del tipo di organizzazione del lavoro adottata e delle caratteristiche del processo e del flusso dei materiali.

I classici tipi di layout studiati e diffusi per far fronte alle problematiche esposte sono i seguenti:

-  Layout a punto fisso
-  Layout per processo
-  Layout per linea di prodotto
-  Layout a celle

E' interessante notare che non sempre si sceglie di applicare un solo layout tra queste tipologie. I layout ibridi, cioè due o più tipi di layout in una sola struttura, sono molto comuni, se non addirittura la norma. Una struttura con layout ibridi comporta, rispetto ad una struttura con un unico layout, maggiori difficoltà di progettazione, costi di allestimento più elevati e maggiori problemi di manutenzione. In compenso è quasi sempre impegnata grazie ad una maggior varietà di prodotti o servizi che si adattano ad una classe più estesa di potenziali clienti.

Di seguito sono riportate le caratteristiche e illustrati i grafici dei layout sopra citati.

	<i>Logica</i>	<i>Tipo di produzione</i>
<i>Layout a punto fisso</i>	I materiali, o il componente principale del prodotto, rimangono in una posizione prefissata nello stabilimento e gli attrezzi, il macchinario e il personale, come pure gli altri elementi facenti parte del prodotto, confluiscono verso tale posizione.	<ul style="list-style-type: none"> <li>❖ Produzione di un articolo di grandi dimensioni ( per es. grandi turbine, eliche, motori marini)</li> <li>❖ Produzione su Commessa</li> <li>❖ Bassi volumi</li> <li>❖ Domanda variabile</li> <li>❖ Gamma prodotti variabile</li> </ul>

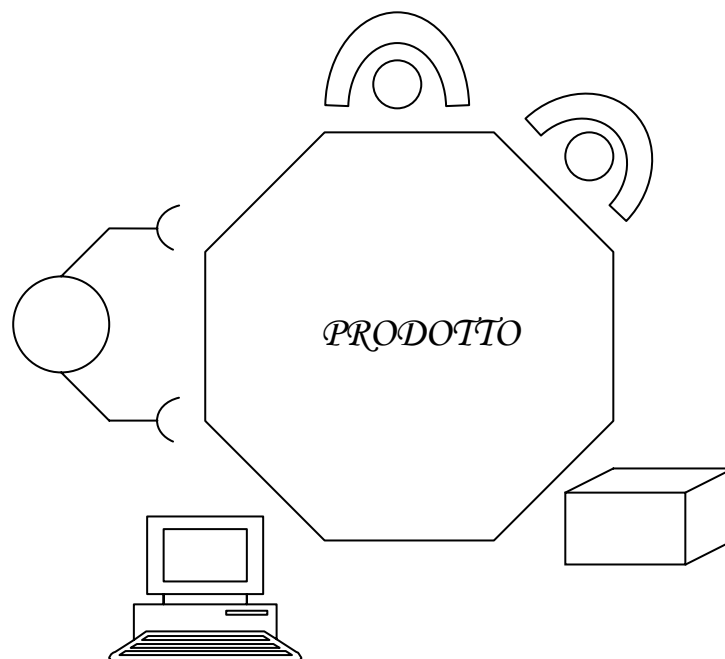


Figura 1.1 – Layout a punto fisso



	<i>Logica</i>	<i>Tipo di produzione</i>
<i>Layout per prodotto</i>	Ogni area dello stabilimento è destinata alla produzione di un solo prodotto o di prodotti simili tra loro.	<ul style="list-style-type: none"> <li>❖ Caratteristico delle grandi industrie automobilistiche</li> <li>❖ Prodotto standardizzato costruito in grandi quantità</li> <li>❖ Produzione continua</li> <li>❖ Volumi elevati</li> <li>❖ Domanda stabile</li> <li>❖ Gamma di prodotti limitata</li> </ul>

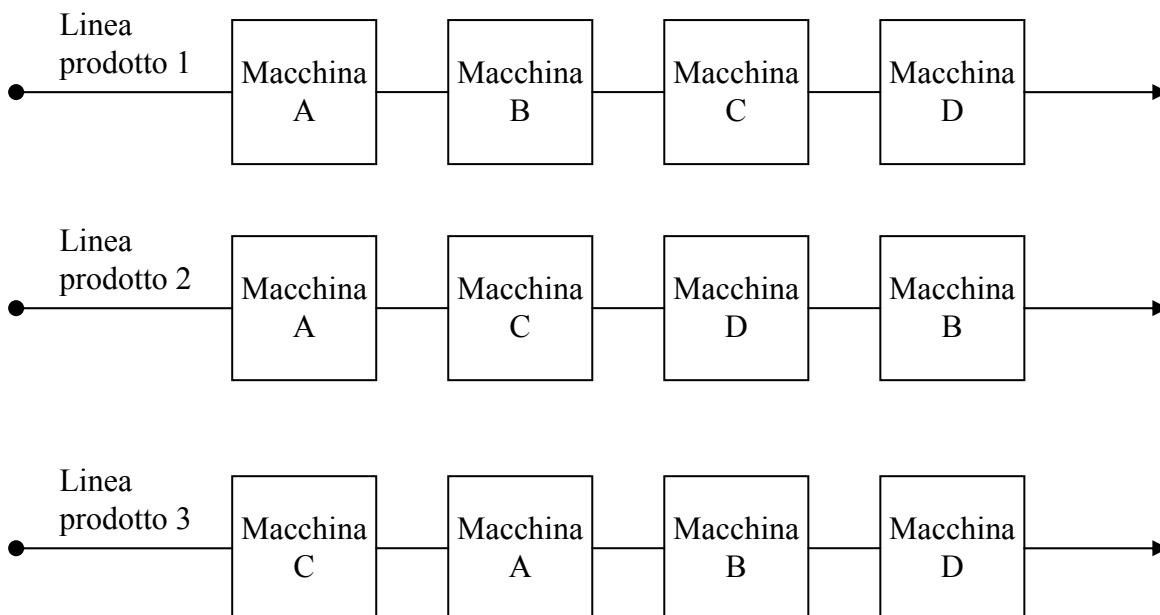


Figura 1.3 – Esempio di layout per prodotto

	<i>Logica</i>	<i>Tipo di produzione</i>
<i>Layout per gruppi (Group Technology/ Layout misto)</i>	Divisione dell'officina in sezioni o "gruppi" di macchine funzionalmente diverse tra loro in modo che ogni gruppo realizzi tutte le operazioni su una famiglia (insieme dei pezzi aventi determinate caratteristiche in comune), o serie di pezzi simili.	❖ Produzione di pezzi e articoli di media e piccola serie

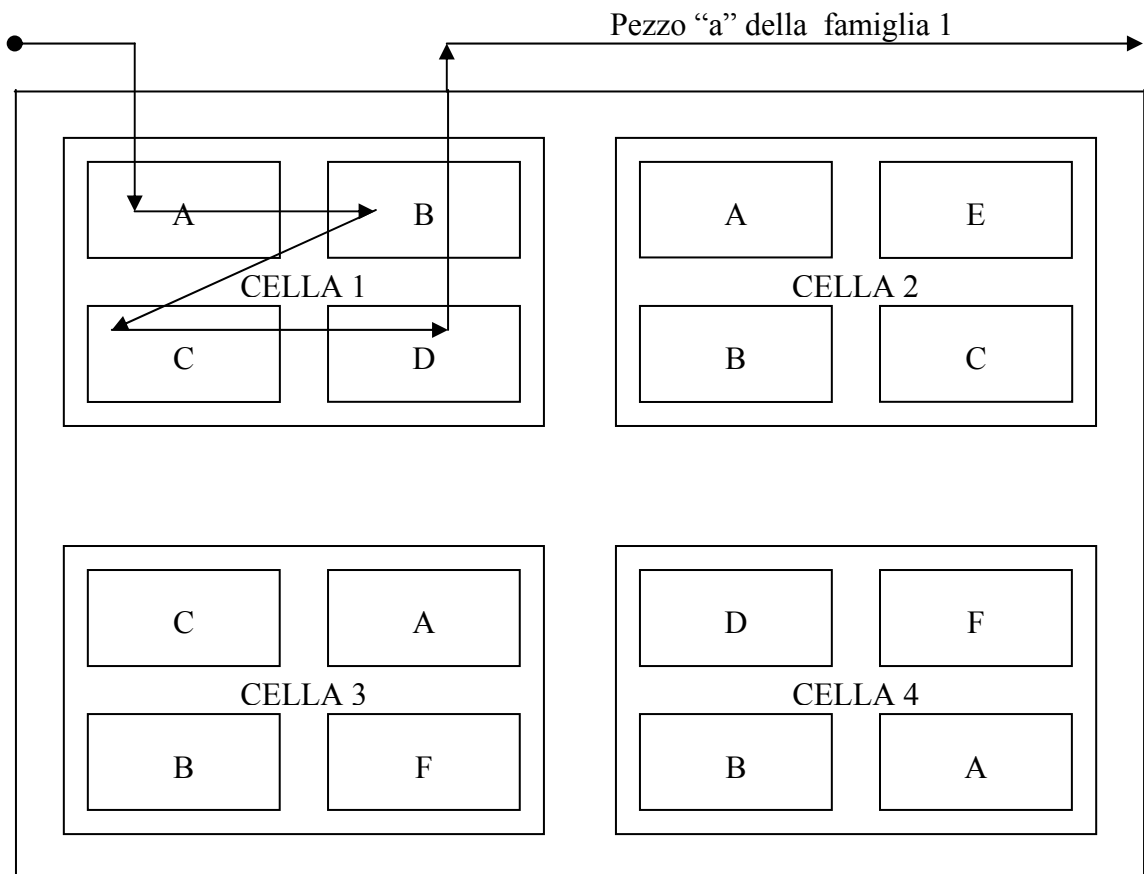










Figura 1.4 – Esempio di layout per gruppi

## 1.2 IL LAYOUT NELLE FIERE

Il layout non ha una funzione importantissima solo nell'ambito industriale, ma anche nel terziario. Questo perché l'obiettivo principale di un'attività è quello di migliorare tutti gli aspetti del servizio al cliente sia che si stia parlando di un'azienda di produzione sia che si parli di un centro commerciale o ancora di un ospedale; il layout è uno dei fattori più importanti per garantire la massima efficienza. Riferendosi in particolare alle attività fieristiche esso costituisce la spina ordinatrice, assumendo valenze sia funzionali che estetiche. Da un lato infatti assicura la soluzione dei problemi di flusso e la garanzia dei servizi indispensabili; dall'altro, è l'elemento in cui si realizza l'applicazione di regole prestabilite, quali altezze, larghezze ed aperture imponendo un principio di razionalità e di rispetto delle armonie compositive.

Elenchiamo anche in questo caso i principali fattori di cui si deve tener conto nella progettazione di un layout espositivo:

-  Efficace utilizzo dello spazio disponibile
-  Ottimizzazione del percorso previsto per il visitatore
-  Progettazione degli spazi in allineamento con il numero di visitatori previsti
-  Condizioni ambientali soddisfacenti e di massima sicurezza (allineamento con le normative di sicurezza nei luoghi pubblici)
-  Investimenti di capitali solo se necessari
-  Disposizione planimetrica in armonia con l'ambientazione richiesta dal prodotto esposto
-  Previsione di ampliamenti futuri
-  ...

E' facile quindi notare che un layout espositivo non è completamente riconducibile ad un layout industriale. Le due caratteristiche principali che rendono vera questo assunto sono le seguenti:

- i. Nel layout espositivo non si deve tener conto del flusso di materiale tra un punto dell'ambiente e l'altro, aspetto invece fondamentale nell'ambito di un' azienda manifatturiera



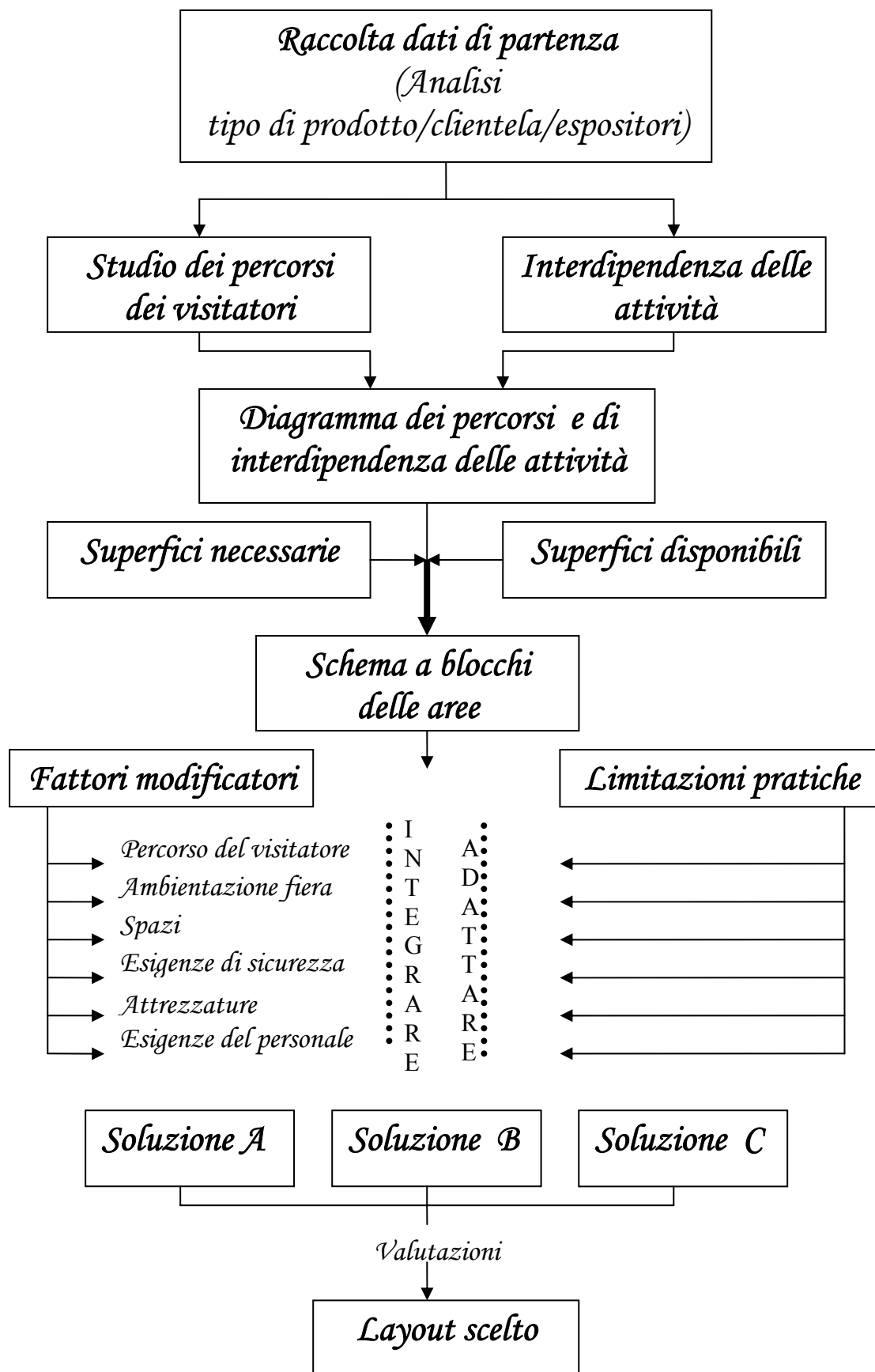
- ii. Il layout espositivo deve sempre fare i conti con un' esigenza di tipo "estetico" in quanto la planimetria deve rispettare la natura dell'oggetto esposto. Il layout industriale invece deve semplicemente essere funzionale.

Un po' brutalmente si potrebbe comunque ricondurre un layout comunemente riferito ad una linea produttiva ad una fiera: si potrebbe considerare un cliente che entra in una fiera come materia prima, mentre un cliente che esce come un prodotto finito. In questa visione la produzione consisterebbe nella trasformazione di un cliente a "mani vuote" in uno a "mani piene" e la fiera assisterebbe il cliente con un layout per linea di prodotti o un layout a celle: il visitatore camminando passa da un prodotto all'altro, comprando da uno stand ad un altro. Ogni cliente uscirebbe quindi come un prodotto unico, in quanto arricchito da un mix di oggetti scelti secondo il proprio gusto e le proprie necessità. Tutto ciò rimane comunque una visione platonica dettata unicamente dal voler ricondurre il layout fieristico ad un campo già ampiamente studiato come quello industriale; in letteratura infatti non sono presenti studi specifici sul layout fieristico. Eppure la progettazione di quest'ultimo non deve avvenire in modo aleatorio, ma al contrario in modo metodico e puntuale.

Un layout fieristico implica ad esempio uno studio dettagliato dei tragitti da far compiere al visitatore, come ad esempio avviene per le traiettorie dei carrelli automatici in un magazzino. La disposizione planimetrica degli stand può essere studiata in modo da ottenere un percorso obbligato o fortemente consigliato dove si cerca di far passare il visitatore davanti ad ogni stand (è questo il caso ad esempio di una fiera d'antiquariato, dove ogni espositore mostra prodotti di così alto valore che non si possono non vedere), oppure in modo da creare un percorso libero dove il cliente si diverte a girare anche dieci volte attorno allo stesso stand.

La progettazione di una fiera impone l'ideazione di un tracciato architettonico che faccia uso consapevole di quinte prospettiche, di una successione di chiusure e aperture spaziali calibrate e in grado di bilanciare la funzionalità del percorso espositivo. Nel progetto globale della manifestazione sono perciò da curarsi con la stessa attenzione il "pieno" degli stand e il "vuoto" dei percorsi.

Si può cercare di seguire un metodo nella progettazione di un layout fieristico, seguendo, come si è fatto per il layout industriale, l'approccio mostrato nello Schema 2. Di seguito, la figura 1.5 mostra la pianta di un layout fieristico.



Schema 1.2 – Approccio al progetto di un layout fieristico



Figura 1.5 – Pianta di un layout fieristico<sup>1</sup>

<sup>1</sup> Layout fieristico della fiera “Move, mostra del video entertainment” della società BolognaFiere . Preso dal sito [www.move.bolognafiere.it](http://www.move.bolognafiere.it)

Tutto ciò che si è detto finora sui layout fieristici rimane una grande considerazione effettuata senza alcun supporto bibliografico; dopo attente e prolungate ricerche infatti, non sono riuscita a trovare alcuna documentazione specifica focalizzata direttamente su questo argomento, nessuna pubblicazione che aiuti a risolvere i problemi logistici esistenti attorno a una fiera, nessun approccio “ingegneristico” che descriva in modo puntuale e dettagliato la progettazione del layout in tale ambito.

Nei prossimi capitoli si cercherà quindi di individuare alcune tipologie di layout fieristico, prendendo spunto da una parte della letteratura della ricerca operativa.

## **Bibliografia:**

- M. Boario, M. De Martini, E. Di Meo, G.M. Gros-Pietro  
**Manuale di logistica**, Volumi 1 e 2  
Utet
- W.C. Chiang, P.Kouvelis  
**Improved tabu search heuristic for solving facility layout design problems**  
International Journal of Production Research (1996) 34, 2565-2585
- C. Hicks  
**A Genetic Algorithm tool for optimising cellular or functional layouts in the capital goods industry**  
*International Journal of Production Economics, Volume 104, Issue 2, December 2006, Pages 598-614*
- E. Knod, R. Schonberger  
**Gestione della produzione**  
McGraw-Hill

## Capitolo 2

### CUTTING AND PACKING PROBLEMS (C&P)

**I** “cutting and packing problems” consistono nel mettere un dato insieme di oggetti (*items*) di piccole dimensioni in uno o più contenitori (naturalmente di dimensioni maggiori rispetto ai primi) senza che essi si sovrappongano tra di loro e in modo da minimizzare o massimizzare una data funzione obiettivo. Si tratta di un problema di ottimizzazione combinatoria con molte importanti applicazioni nelle industrie del legno, del vetro, dell'acciaio e della pelle; e ancora è utilizzato con buoni risultati nello studio dell'impaginazioni dei quotidiani e nel caricamento di autocarri e container. Per molti decenni inoltre i problemi di packing e cutting hanno interessato l'attenzione di molti ricercatori nelle aree della ricerca operativa, dell'industria manifatturiera e della computer science.

Questi problemi possono essere classificati basandosi su diversi criteri. La dimensione del problema ne è un esempio (oltre ad essere il criterio più usato): molti problemi sono definiti su una, due o tre dimensioni; in questo scritto ci si fermerà alle due dimensioni.

Cutting and Packing problems a una dimensione:

✚ Knapsack 01 (KP)

✚ Bin packing problem (BPP)

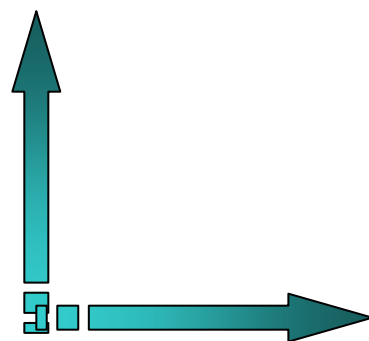


Cutting and Packing problem a due dimensioni:

✚ Two-dimensional bin packing problem (2BPP)

✚ Two-dimensional strip packing problem (2SPP)

✚ Two-dimensional Knapsack Problem (2KP)



## 2.1 KNAPSACK PROBLEM

Si supponga che un alpinista, prima di intraprendere una nuova escursione debba riempire il suo zaino scegliendo tra diversi oggetti quelli che potrebbero essergli più utili. Questo *knapsack problem* può essere formulato matematicamente numerando gli oggetti da 1 a  $n$  e creando un vettore di variabili binarie  $x_j$  ( $j = 1, \dots, n$ ) che hanno il seguente valore:

$$x_j = \begin{cases} 1 & \text{se l'oggetto } j \text{ è scelto} \\ 0 & \text{altrimenti} \end{cases}$$

Inoltre se  $p_j$  è la misura del profitto dato dall'oggetto  $j$ ,  $w_j$  la sua dimensione e  $c$  la dimensione dello zaino, il nostro problema sarà quello di selezionare tra tutti, i vettori binari  $x$  che soddisfano il seguente vincolo:

$$\sum_{j=1}^n w_j x_j \leq c$$

e che massimizzano la seguente funzione obiettivo:

$$\sum_{j=1}^n p_j x_j$$

Il problema può essere più stimolante se lo si pensa in questo modo: si vuole investire un capitale di  $c$  Euro e si considerano  $n$  possibili investimenti. Sia  $p_j$  il profitto che ci si aspetta di ottenere dall'investimento  $j$ , e  $w_j$  l'ammontare di Euro che esso richiede. E' evidente che la risoluzione di questo problema indicherà l'investimento migliore.

Per generalizzare il problema diciamo che di solito gli oggetti sono chiamati *items* e considerati di una quantità pari ad  $n$ . Il valore e la dimensione del  $j$ -esimo oggetto sono chiamati rispettivamente *profitto* e *peso* e indicati con  $p_j$  e  $w_j$  ( $j=1, \dots, n$ ).

Il problema finora descritto è rappresentativo di una varietà di problemi *knapsack* nei quali è dato un insieme di oggetti, ciascuno dei quali associati ad un profitto ed un peso, e nei quali è richiesto di trovare uno o più sottoinsiemi nei quali la somma dei pesi non superi ( o eguagli) un valore dato (*bound*) e la somma dei valori sia massimizzata.

I problemi *knapsack* sono stati studiati intensamente, soprattutto negli ultimi decenni. Esso infatti non è solo interessante dal punto di vista teorico, ma anche dal punto di

vista pratico. I problemi di knapsack sono infatti usati per problemi di budget, di carico dei veicoli, di taglio di materiali solo per menzionare le principali applicazioni.

### 2.1.1 KNAPSACK 01

Il *knapsack01* (o *Binario*) è: dato un insieme di  $n$  oggetti (items) e un contenitore (knapsack), con

$p_j$  = profitto dell'oggetto  $j$ ,

$w_j$  = peso dell'oggetto  $j$ ,

$c$  = capacità del contenitore

scegliere un sottoinsieme di oggetti tale da:

$$\text{maximize } z = \sum_{j=1}^n p_j x_j \quad (1)$$

$$\text{subject to } \sum_{j=1}^n w_j x_j \leq c \quad (2)$$

$$x_j \in \{0,1\}, \quad j \in N = \{1, \dots, n\} \quad (3)$$

Dove:

$$x_j = \begin{cases} 1 & \text{se l'oggetto } j \text{ è scelto} \\ 0 & \text{altrimenti} \end{cases}$$

Assumiamo ora, senza perdita di generalità, che

$$p_j, w_j \text{ e } c \text{ sono interi positivi,} \quad (4)$$

$$\sum_{j=1}^n w_j < c, \quad (5)$$

$$w_j \leq c \text{ con } j \in N \quad (6)$$

Se l'assunzione (4) è violata, le frazioni possono essere risolte moltiplicando per un opportuno fattore, tralasciando il caso di numeri irrazionali), mentre i valori negativi possono essere risolti come segue (Glover, 1965):

1. Per ogni  $j \in N^0 = \{j \in N: p_j \leq 0, w_j \geq 0\}$  do  $x_j := 0$

2. Per ogni  $j \in N^1 = \{j \in N: p_j \geq 0, w_j \leq 0\}$  do  $x_j := 1$



Sia inoltre  $N^- = \{j \in N: p_j < 0, w_j > 0\}$ ,  $N^+ = N \setminus (N^0 \cup N^1 \cup N^-)$ , e

$$\begin{cases} y_j = 1 - x_j, \bar{p}_j = -p_j, \bar{w}_j = -w_j & \text{per } j \in N^- \\ y_j = x_j, \bar{p}_j = p_j, \bar{w}_j = w_j & \text{per } j \in N^+ \end{cases}$$

Si risolva il rimanente problema:

$$\begin{aligned} \text{Maximize } z &= \sum_{j \in N^- \cup N^+} \bar{p}_j y_j + \sum_{j \in N^1 \cup N^-} p_j \\ \text{subject to } &\sum_{j \in N^- \cup N^+} \bar{w}_j y_j \leq c - \sum_{j \in N^1 \cup N^-} w_j, \\ &y_j \in \{0,1\}, \quad j \in N^- \cup N^+ \end{aligned}$$

Se i dati in input violano l'assunzione (5) allora  $x_j = 0$  per ogni  $j \in N$ ; se violano l'assunzione (6) allora  $x_j = 0$  per ogni  $j$  tale che  $w_j > c$ .

Se non in altro modo specificato noi supporremo sempre che gli *items* siano ordinati in ordine decrescente secondo il valore del profitto per unità di peso, cioè:

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \dots \geq \frac{p_n}{w_n}$$

Dato ad ogni problema un'istanza  $I$ , indichiamo il valore di ogni soluzione ottima con  $z(I)$  o semplicemente con  $z$ .

Si è sempre considerato il problema nella versione di massimizzazione; la forma di minimizzazione del problema appare in questo modo:

$$\begin{aligned} \text{Minimize } &\sum_{j=1}^n p_j y_j \\ \text{subject to } &\sum_{j=1}^n w_j y_j \geq q \\ &y_j \in \{0,1\}, \quad j \in N \end{aligned}$$

Quest'ultimo modello può facilmente essere trasformato nella forma di massimizzazione ponendo  $y_j = 1 - x_j$  e risolvendo la (1), la (2) e la (3) con  $c =$

$\sum_{j=1}^n w_j - q$ . Si ponga come  $z_{max}$  il valore della soluzione di ogni problema, il problema

di minimizzazione avrà come soluzione il valore  $z_{min} = \sum_{j=1}^n p_j - z_{max}$ . (Intuitivamente,

si massimizza il profitto degli oggetti non contenuti nel contenitore).

### Rilassamento lineare:

Il primo rilassamento di un problema Knapsack è il cosiddetto “*Linear programming relaxation*”, cioè il *continuous Knapsack problem*  $C(KP)$ , ottenuto da (1), (2), (3) rimuovendo i vincoli di integralità sulle variabili  $x_j$ :

$$\begin{aligned} \text{maximize } z &= \sum_{j=1}^n p_j x_j \\ \text{subject to } &\sum_{j=1}^n w_j x_j \leq c \\ &0 \leq x_j \leq 1 \quad j = 1, \dots, n \end{aligned}$$

Si supponga che gli oggetti, ordinati come detto sopra, siano consecutivamente inseriti nel contenitore fino a quando non si trova il primo oggetto  $s$  che non ci sta. Esso è chiamato *critical item*:

$$s = \left\{ j : \sum_{i=1}^j w_i > c \right\},$$

Si noti che, viste le assunzioni (4), (5), (6), si ha  $1 < s \leq n$ .

Il problema  $C(KP)$  può essere risolto grazie a un metodo scoperto da Dantzig (1957).

### Rilassamento Lagrangiano:

Un modo alternativo di rilassare il KP è attraverso l'approccio Lagrangiano. Dato un fattore non negativo  $\lambda$ , il *rilassamento lagrangiano* del KP ( $L(KP, \lambda)$ ) è:

$$\begin{aligned} & \text{maximize } \sum_{j=1}^n p_j x_j + \lambda \left( c - \sum_{j=1}^n w_j x_j \right) \\ & \text{subject to } x_j \in \{0,1\}, \quad j = 1, \dots, n \end{aligned}$$

La funzione obiettivo può essere riformulata in questo modo:

$$z(L(KP, \lambda)) = \sum_{j=1}^n \tilde{p}_j x_j + \lambda c \quad (7)$$

Dove  $\tilde{p}_j = p_j - \lambda w_j$  per  $j = 1, \dots, n$  e la soluzione ottima di  $L(KP, \lambda)$  è facilmente determinabile, in un tempo  $O(n)$ , come:

$$\tilde{x}_j = \begin{cases} 1 & \text{se } \tilde{p}_j > 0, \\ 0 & \text{se } \tilde{p}_j < 0. \end{cases} \quad (8)$$

(Quando  $\tilde{p}_j = 0$ , il valore di  $x_j$  è irrilevante).

Quindi, definendo  $J(\lambda) = \{ j: \tilde{p}_j / w_j > \lambda \}$ , il valore della soluzione di  $L(KP, \lambda)$  è:

$$z(L(KP, \lambda)) = \sum_{j \in J(\lambda)} \tilde{p}_j + \lambda c$$

Per ogni  $\lambda \geq 0$ , c'è un upper bound in  $z(KP)$  che, comunque, non potrà mai essere migliore del bound di Dantzig. Anzi la (8) ci dà la soluzione del rilassamento continuo di  $L(KP, \lambda)$ , così:

$$z(L(KP, \lambda)) = z(C(L(KP, \lambda))) \geq z(C(KP))$$

Il valore di  $\lambda$  che produce il valore minimo di  $z(L(KP, \lambda))$  è  $\lambda^* = p_s / w_s$ . Con questo valore, in realtà, abbiamo  $\tilde{p}_j \geq 0$  per  $j = 1, \dots, s-1$  e  $\tilde{p}_j \leq 0$  per  $j = s, \dots, n$ . Così  $J(\lambda^*) \subseteq \{1, \dots, s-1\}$ . Da qui  $\tilde{x}_j = \bar{x}_j$  per  $j \in N \setminus \{s\}$  (dove  $(\bar{x}_j)$  è definita dal teorema 2.1) e dalla (7) e dalla (8),  $z(L(KP, \lambda^*)) = \sum_{j=1}^{s-1} (p_j - \lambda^* w_j) + \lambda^* c = z(C(KP))$ .

Inoltre si noti che, per  $\lambda = \lambda^*$ ,  $\tilde{p}_j$  diventa:

$$p_j^* = p_j - w_j \frac{p_s}{w_s};$$

$|p_j^*|$  è il decremento che otteniamo in  $z(L(KP, \lambda^*))$  ponendo  $\tilde{x}_j = 1 - \tilde{x}_j$ , e da qui un *lower bound* sul corrispondente decremento della soluzione continua (da quando l'ottimo  $\lambda$  cambia imponendo le condizioni sopra descritte).

Altri rilassamenti lagrangiani del problema KP sono stati studiati da Maculan (1983) e da Fisher (1981).

### Teorema 2.1

La soluzione ottima  $\bar{x}$  di C(KP) è

$$\bar{x}_j = 1 \quad \text{per } j = 1, \dots, s-1$$

$$\bar{x}_j = 0 \quad \text{per } j = s+1, \dots, n,$$

$$\bar{x}_s = \frac{\bar{c}}{w_s},$$

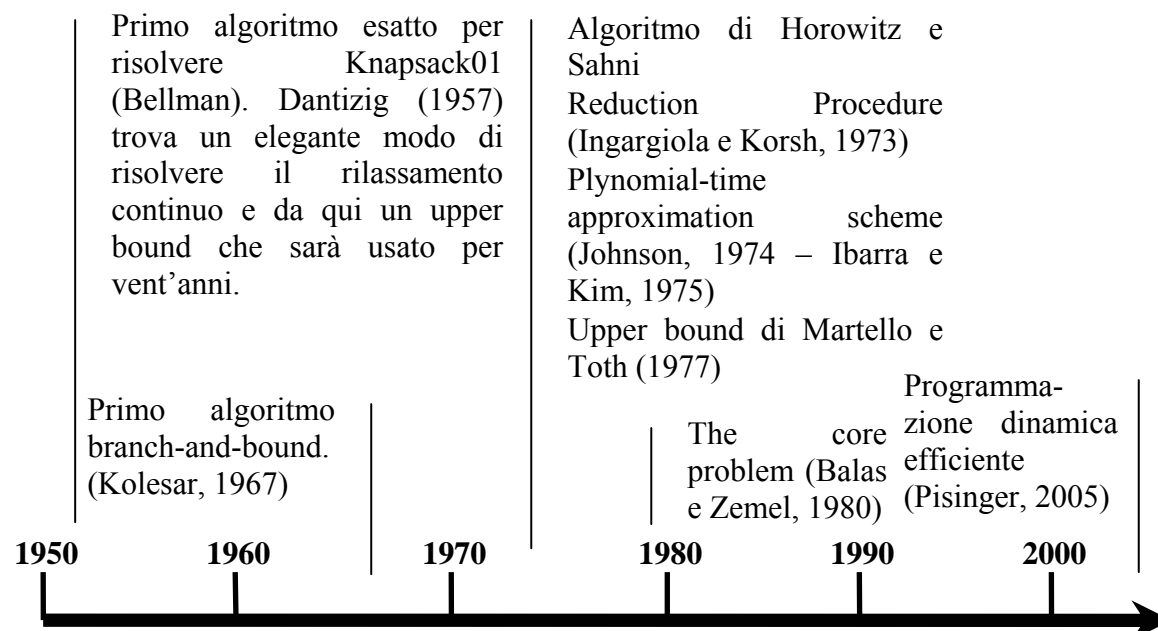
dove:

$$\bar{c} = c - \sum_{j=1}^{s-1} w_j$$

Il Knapsack Problem 01 è il più importante *Knapsack problem* e uno dei più studiati problemi di programmazione discreta. La ragione di questo interesse così grande deriva principalmente intera da tre fattori:

- Può essere visto come un semplice problema di Programmazione Intera Lineare
- Appare come un sottoproblema in molti problemi più complessi
- Può essere utile in molte situazioni pratiche

Ecco un piccolo riassunto delle migliori scoperte in ordine cronologico:



## 2.2 BIN-PACKING PROBLEM

I *bin-packing problem (BPP)* possono essere descritti, usando la terminologia dei problemi di knapsack. Dati  $n$  items e  $n$  *knapsacks* (*o bins*), con

$w_j$  = peso dell'item  $j$ ,  
 $c$  = capacità di ogni bin,

assegnare ogni item a un bin in modo che il peso totale degli oggetti in ogni contenitore non ecceda la capacità  $c$  e che il numero di bins usati sia minimo. Una possibile formulazione matematica del problema è:

$$\begin{aligned} \text{minimize } z &= \sum_{i=1}^n y_i \\ \text{subject to } &\sum_{j=1}^n w_j x_{ij} \leq c y_i, \quad i \in N = \{1, \dots, n\}, \\ &\sum_{i=1}^n x_{ij} = 1, \quad j \in N, \\ &y_i \in \{0, 1\}, \quad i \in N, \\ &x_{ij} \in \{0, 1\}, \quad i \in N, j \in N, \end{aligned}$$

Dove:

$$y_i = \begin{cases} 1 & \text{se il bin } i \text{ è usato;} \\ 0 & \text{altrimenti,} \end{cases} \quad (1)$$

$$x_{ij} = \begin{cases} 1 & \text{se l'item } j \text{ è assegnato al contenitore } i; \\ 0 & \text{altrimenti.} \end{cases} \quad (2)$$

$$c \text{ è un intero positivo,} \quad (3)$$

$$w_j \leq c \quad \text{per } j \in N. \quad (4)$$

Se l'assunzione (3) è violata e i  $w_j$  sono tutti interi,  $c$  può essere sostituito con  $\lfloor c \rfloor$ . Se un item viola l'assunzione (4), l'istanza è inaccettabile. Tuttavia, non c'è un modo semplice per trasformare un'istanza in modo da poter trattare pesi negativi.

Per semplicità assumiamo anche che, per ogni soluzione ammissibile, siano usati i bins di indice più basso (per es.  $y_i \leq y_{i+1}$  per  $i = 1, \dots, n-1$ ).

Quasi la totalità della letteratura sui BPP riguarda l'approssimazione degli algoritmi e la loro performance. Si segnala in particolare lo studio di Coffman, Garey e Johnson (1984).

### Algoritmi approssimati

Il più semplice approccio approssimativo al bin packing problem è l'algoritmo *Next-Fit* (NF). Il primo *item* è assegnato al bin 1. Gli items 2, ..., n sono poi considerati al crescere dell'indice: ogni item è assegnato al bin corrente se è adatto; altrimenti, è assegnato a un nuovo bin, che diventa quello corrente. La complessità dell'algoritmo è chiaramente  $O(n)$ . E' facile infatti provare che, per ogni istanza  $I$  del BPP, la soluzione  $NF(I)$  fornita dall'algoritmo soddisfa il bound:

$$NF(I) \leq 2 z(I)$$

Dove  $z(I)$  indica la soluzione ottima. Inoltre esistono istanze per le quali il rapporto  $NF(I)/z(I)$  è arbitrariamente fermato a 2, cioè la peggior performance del rapporto di NF è  $r(NF) = 2$ . Si noti che, per un problema di minimo, il peggior caso di performance del rapporto di un algoritmo A approssimato è definito come il più piccolo numero reale  $r(A)$  tale che:

$$\frac{A(I)}{z(I)} \leq r(A) \text{ per tutte le istanze } I,$$

dove  $A(I)$  indica la soluzione ottenuta da A.

Un algoritmo migliore, il *First-Fit* (FF), considera gli items all'incrementare degli indici e assegna ogni item al più basso indice di bin inizializzato nel quale entra; solo quando l'oggetto corrente non viene assegnato a qualche bin inizializzato, viene introdotto un nuovo contenitore.

Un altro algoritmo, *Best-Fit* (BF), è ottenuto dal FF assegnando il corrente item al bin accettabile (se c'è) che ha la più piccola capacità residua (rompendo il vincolo del bin con indice più basso).

Si assuma ora che gli oggetti siano sistemati in modo che:

$$w_1 \geq w_2 \geq \dots \geq w_n,$$

e che in seguito siano applicati gli algoritmi NF oppure FF, oppure BF. Gli algoritmi risultanti sono chiamati *Next-fit Decreasing (NFD)*, *First-Fit Decreasing* e *Best-Fit Decreasing (BFD)*, rispettivamente.

La complessità di questi algoritmi e i casi peggiori di performance che si possono raggiungere sono stati studiati da Johnson, Demers, Ullman, Garey e Graham (1974). Ecco la tabella riassuntiva ( $r^\infty$  è l' *asymptotic worst-case performance ratio*. Esso è molto usato e per un algoritmo approssimato A, è definito come il minor numero reale  $r^\infty(A)$  tale che, per un qualche intero positivo  $K$ :  $\frac{A(I)}{z(I)} \leq r^\infty(A)$  per ogni istanza  $I$  che soddisfi  $z(I) \geq K$ ):

Algoritmo	Complessità	$r^\infty$
<i>NF</i>	$O(n)$	2.000
<i>FF</i>	$O(n \log n)$	1.700
<i>BF</i>	$O(n \log n)$	1.700
<i>NFD</i>	$O(n \log n)$	1.691
<i>FFD</i>	$O(n \log n)$	1.222
<i>BFD</i>	$O(n \log n)$	1.222

### Rilassamento Lineare:

Per il nostro modello di BPP, il rilassamento continuo C(BPP) del problema è:

$$\begin{aligned}
 &\text{minimize } z = \sum_{i=1}^n y_i \\
 &\text{subject to } \sum_{j=1}^n w_j x_{ij} \leq c y_i, \quad i \in N = \{1, \dots, n\}, \\
 &\quad \sum_{i=1}^n x_{ij} = 1, \quad j \in N, \\
 &\quad 0 \leq y_i \leq 1, \quad i \in N, \\
 &\quad 0 \leq x_{ij} \leq 1, \quad i \in N, j \in N
 \end{aligned}$$

Il rilassamento può essere immediatamente risolto ponendo  $x_{ij} = 1$ ,  $x_{ij} = 0$  ( $j \neq i$ ) e  $y_i = w_i / c$  per  $i \in N$ . Da qui dunque:

$$z(C(BPP)) = \sum_{i=1}^n w_i / c$$

## 2.3 TWO-DIMENSIONAL BIN PACKING PROBLEM

Quasi tutti i problemi di impaccamento a due dimensioni fanno parte dei cosiddetti problemi NP-difficili, cioè quei problemi per i quali non è mai stato trovato un algoritmo polinomiale.

Inoltre, finora si sono classificati i problemi di *packing* a seconda del numero di dimensioni. I problemi a due dimensioni possono anche essere classificati in base alla forma degli oggetti da impaccare; nelle applicazioni industriali è particolarmente presente il *two-dimensional rectangle packing problem*. In esso è richiesto di collocare un insieme di oggetti rettangolari in un numero minimo di unità rettangolari più grandi, minimizzando quindi anche lo spazio inutilizzato. Inoltre nelle industrie del legno e del vetro, pezzi di forma rettangolare devono essere tagliati a partire da grandi lamine di materiale. Nel contesto dei magazzini, i prodotti devono essere messi sugli scaffali. Nell'impaginazione dei giornali, gli articoli devono essere ben posizionati nelle pagine. In tutte queste applicazioni industriali, il problema dell'impaccamento si presenta



spesso con vincoli lievemente differenti; molte varianti del problema sono state considerate in letteratura. In particolare importanti vincoli per il problema sono:

✚ L'orientamento

Si assume di solito che ogni rettangolo abbia una certa orientazione fissa, oppure che ogni rettangolo possa essere ruotato di  $90^\circ$ . La rotazione dei rettangoli non è ammessa per esempio nell'impaginazione dei giornali, quando gli item da tagliare sono decorati o non isomorfi (legno, lamiera).

✚ I tagli a ghigliottina

Effettuare un taglio a ghigliottina significa che gli items sono stati ottenuti attraverso una sequenza di tagli lato-a-lato paralleli ai lati del contenitore. Ciò è spesso imposto da limitazioni tecniche di macchine da taglio automatiche (ad esempio seghe verticali o circolari) o di alcuni materiali. Ecco due esempi di piazzamenti di items con o senza il vincolo dei tagli a ghigliottina:

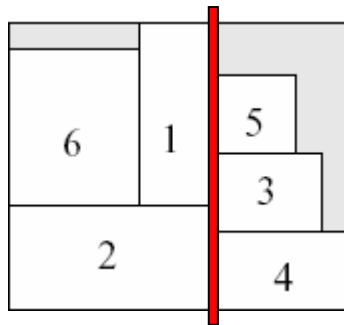


Figura 2.1a: taglio a ghigliottina

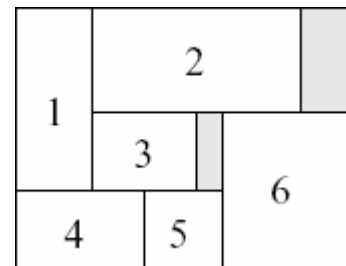


Figura 2.1b: taglio non a ghigliottina

✚ L'impaccamento a livelli

Il piazzamento degli items è ottenuto posizionandoli da sinistra a destra in righe formanti diversi livelli. Il primo livello è il fondo del contenitore, e ogni livello seguente è la linea orizzontale coincidente con il lato superiore dell'item più alto impaccato a livello inferiore.

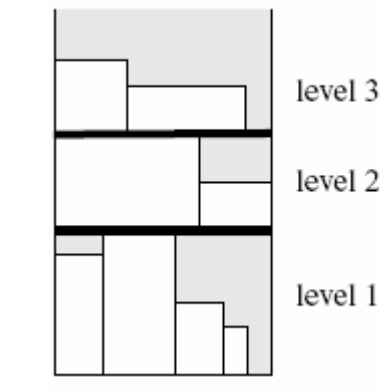


Figura 2.2: Esempio di level packing

#### ✚ Disegno a scacchi (Checkerboard pattern)

I *checkerboard patterns*, conosciuti anche come *1-group patterns* (Gilmore e Gomory, 1965) fanno parte della classe speciale dei tagli a ghigliottina a due fasi (*two-stage guillotine patterns*) che non necessitano di essere ritagliati. Essi possono infatti essere prodotti ruotando la sega di  $90^\circ$ , dopo i tagli fatti in una prima fase. Le strisce ottenute nella prima fase sono tutte tagliate nella seconda fase producendo gli items desiderati. Tagli di questo tipo richiedono meno tempo per le macchine, e sono particolarmente interessanti nei settori con alta domanda quando le macchine rappresentano il collo di bottiglia della produzione.

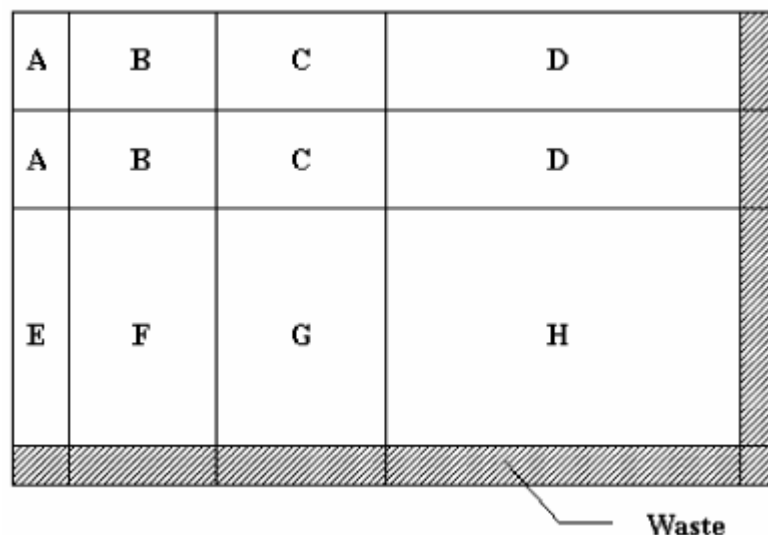


Figura 2.3: Checkerboard pattern

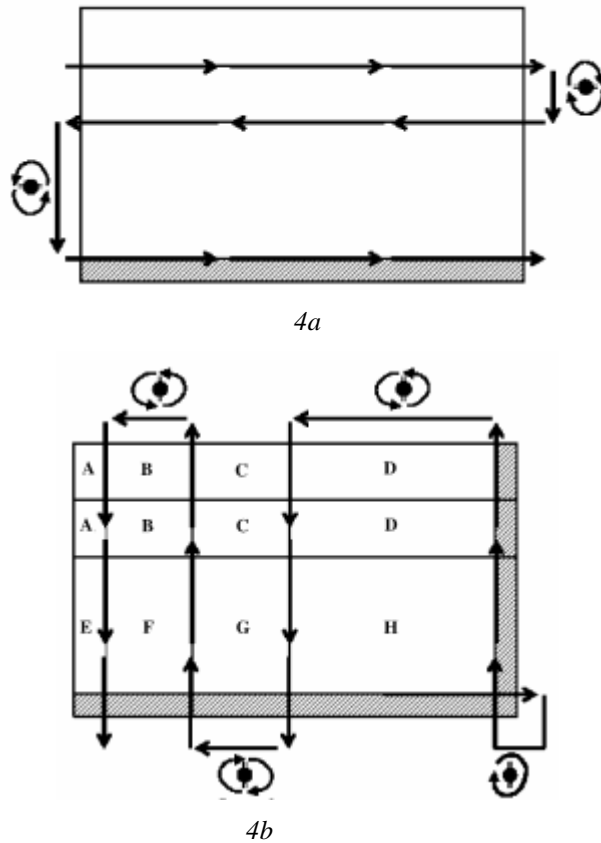


Figura 2.4: Esempio di tagli a scacchi: prima (4a) e seconda (4b) fase.

#### ✚ Packing Problem Unconstrained, Constrained o Doubly constrained

Indicato con  $x_i$  i pezzi da impaccare e con  $P_i$  e  $Q_i$  rispettivamente il numero minimo e massimo di pezzi del tipo  $i$  da poter impaccare, si ha che  $0 \leq P_i \leq x_i \leq Q_i$  e che in accordo con tali valori, si possono distinguere tre tipi di problemi:

1. *Unconstrained*:  $\forall i, P_i = 0, Q_i = \lfloor L * W / l_i * w_i \rfloor$
2. *Constrained*:  $\forall i, P_i = 0, Q_i < \lfloor L * W / l_i * w_i \rfloor$
3. *Doubly constrained*:  $\exists i, P_i > 0; \exists j, Q_j < \lfloor L * W / l_j * w_j \rfloor$

Dove  $L$  e  $W$  indicano rispettivamente la larghezza e l'altezza del contenitore, mentre  $w_i$  e  $l_i$  la larghezza e la lunghezza dell'  $i$ -esimo item.

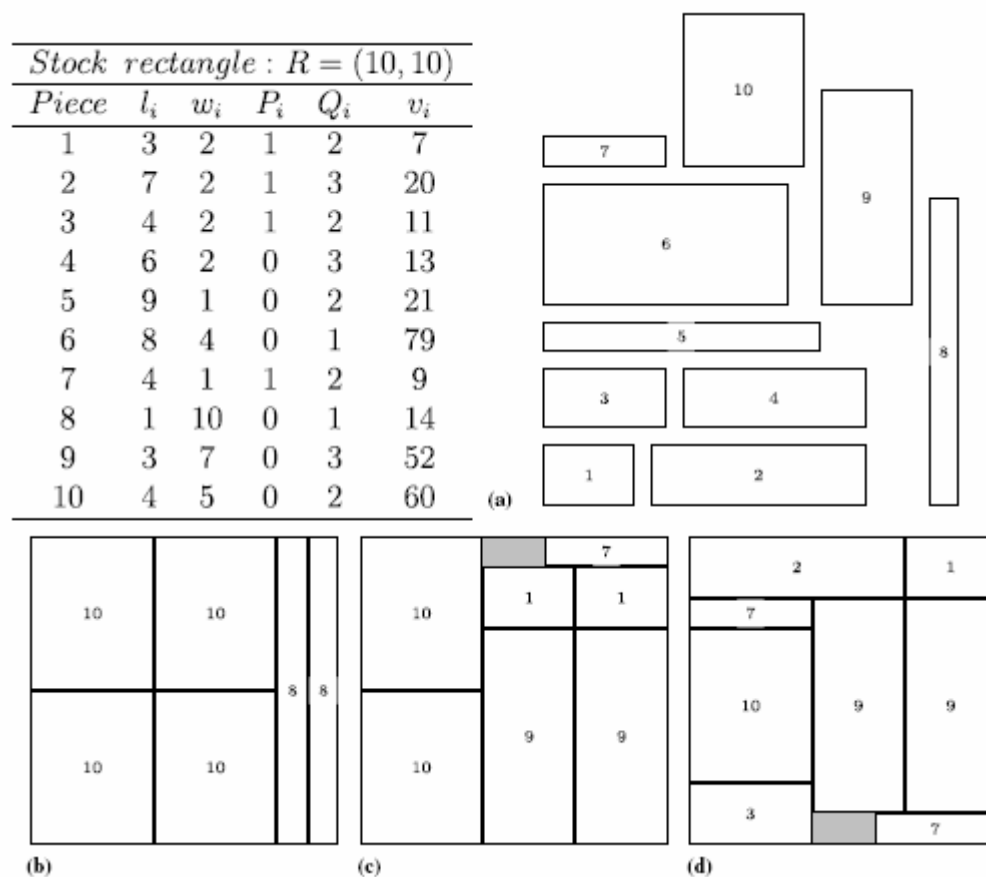


Figura 2.5: Esempio di un stock di rettangoli di misura  $10 \times 10$  in cui collocare i pezzi sopra numerati da 1 a 10. La figura mostra le soluzioni ottime per i problemi unconstrained (b), constrained (c) e doubly constrained (d).

Per semplicità, si definiranno i problemi successivi assumendo che ogni item abbia una orientazione fissa e che il vincolo del taglio a ghigliottina non sia imposto se non diversamente specificato.

Inoltre ci si concentrerà sugli algoritmi cosiddetti *off-line*, ovvero algoritmi nei quali si ha completa conoscenza di tutti i dati in input. Gli algoritmi *on-line*<sup>2</sup> sono invece algoritmi nei quali si “impaccano” gli items in ordine dell’ arrivo in input (senza conoscere l’item successivo).

La descrizione del problema in modo più formale è la seguente:

si ha un insieme di  $n$  oggetti rettangolari (*rectangular items*)  $j \in J = \{1, \dots, n\}$ , ognuno dei quali definito da una larghezza  $w_j$  e un’altezza  $h_j$ , e un numero illimitato di

<sup>2</sup> Sugli algoritmi on-line ci si riferisca a: J.Csirik, G.Woeginger, On-line packing and covering problems, in: Online algorithms, Springer Lecture Notes in Computer Science, vol. 1442, 1996, pp. 147-177.

contenitori rettangolari (*bins*) aventi tutti uguale larghezza  $W$  e altezza  $H$ . L'obiettivo è quello di impaccare tutti gli oggetti minimizzando il numero di contenitori usati. E' richiesto di sistemare gli oggetti ortogonalmente senza sovrapporli (il lato " $w$ " di ogni item deve essere parallelo al lato " $W$ " del contenitore). Si assumerà inoltre, senza perdita di generalità, che i dati in input siano interi positivi, e che  $w_j \leq W$  e  $h_j \leq H$  ( $j = 1, \dots, n$ ).

E' interessante notare che il problema è una generalizzazione del (one-dimensional) *Bin Packing Problem (BPP)*. Quest'ultimo è infatti una caso speciale di quello a due dimensioni, che compare quando  $h_j = H$  per ogni  $j \in N$ .

Il primo tentativo di costruire un modello del *packing problem* a due dimensioni è stato fatto da Gilmore e Gomery (1965), attraverso un' estensione del loro approccio al *packing* a una dimesione. Essi proposero un approccio a generazione di colonne basato sull'enumerazione di tutti i sottoinsiemi di *items (patterns)* che possono essere impaccati all'interno di un singolo contenitore. Sia  $A_j$  una colonna di vettori binari di  $n$  elementi  $a_{ij}$  ( $i = 1, \dots, n$ ) che assumono valore 1 se l' item  $i$  appartiene al  $j$ -esimo *pattern*, e valore 0 altrimenti. L'insieme di tutti i *pattern* accettabili viene rappresentato attraverso la matrice  $A$ , composta da tutte le possibili colonne  $A_j$  ( $j = 1, \dots, M$ ) e il corrispondente modello matematico, di tipo set partitioning, è:

$$\begin{aligned} \min \quad & \sum_{j=1}^M x_j \\ \text{subject to} \quad & \sum_{j=1}^M a_{ij} x_j = 1 \quad (i = 1, \dots, n) \\ & x_j \in \{0, 1\} \quad (j = 1, \dots, M) \end{aligned}$$

Dove  $x_j$  assume valore 1 se il pattern  $j$  appartiene alla soluzione, 0 altrimenti. Si osservi che il modello mostrato, è un valido modello anche per il *packing* a una dimensione, con l'unica differenza che le colonne  $A_j$  soddisfano tutte il vincolo

$$\sum_{i=1}^n a_{ij} h_i \leq H .$$

A causa dell'immenso numero di colonne che possono apparire in  $A$ , l'unico modo di affrontare il modello è quello di generare dinamicamente colonne quando ce n'è bisogno. Ma mentre per il 1BP Gilmore e Gomory sono riusciti ad avere un approccio di programmazione dinamica alla generazione di colonne, grazie all'associazione del

problema a un K01, per il 2BPP essi hanno avuto difficoltà nell'associarlo al corrispondente problema a due dimensioni. Così essi hanno trattato il problema nel caso in cui gli oggetti devono essere impaccati in righe che formano livelli.

### 2.3.1 LEVEL PACKING

Come si è appena detto, molti degli algoritmi approssimati per il 2BPP suppongono di impaccare gli oggetti “a livelli”. Il primo livello è il fondo del contenitore, e gli oggetti sono impaccati appoggiando la loro base su di esso. Il livello successivo è determinato dalla linea orizzontale “disegnata” sull'estremità superiore dell' oggetto più alto impaccato a livello inferiore.

Tre classiche strategie per l'impaccamento a livelli sono state rilevate da alcuni famosi algoritmi. In ogni caso, gli items sono inizialmente numerati in ordine decrescente di altezza e impaccati nella corrispondente sequenza. Se si indica con  $j$  l'item corrente, e con  $s$  l'ultimo livello creato, le tre strategie richiamate sono le seguenti:

- *Next-Fit Decreasing Height (NFDH) strategy*: l'item  $j$  è impaccato giustificato a sinistra al livello  $s$ , se ci sta. Altrimenti è creato un nuovo livello ( $s:=s+1$ ), e  $j$  è impaccato giustificato a sinistra in esso;
- *First-Fit Decreasing Height (FFDH) strategy*: l'item  $j$  è impaccato giustificato a sinistra al primo livello in cui riesce a stare, se c'è. Se nessun livello può contenere  $j$ , è creato un nuovo livello inizializzato come in *NFDH*;
- *Best-Fit Decreasing Height (BFDH) strategy*: l'item  $j$  è impaccato giustificato a sinistra in quel livello, tra quelli dove riesce a stare, per il quale lo spazio in orizzontale inutilizzato è minimo. Se nessun livello può contenere  $j$ , è creato un nuovo livello inizializzato come in *NFDH*.

Le strategie appena specificate sono illustrate nella seguente immagine:

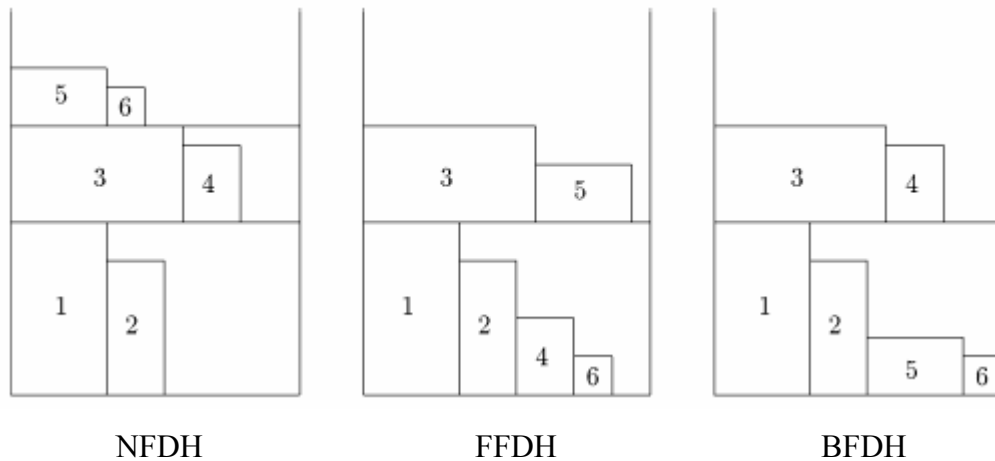


Figura 2.6: Rappresentazione delle strategie NFDH, FFDH e BFDH

### Two-dimensional level bin packing problem (2LBP)

Si chiami con 2LBP (*two-dimensional level bin packing problem*) il problema a due dimensioni ristretto al tipo di impaccamento a livelli:

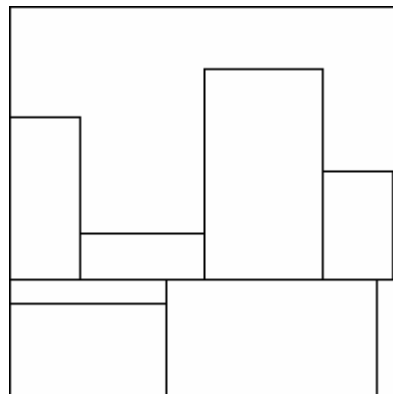


Figura 2.7: Level packing

Si supponga inoltre, senza perdita di generalità, che:

- i. In ogni livello, l'oggetto più a sinistra sia il più alto;
- ii. In ogni contenitore, il livello più in basso sia il più alto;
- iii. Gli oggetti siano posti e rinumerati per valori decrescenti di  $h_j$

Si dirà che l'oggetto più a sinistra di un livello e il livello più in basso di un contenitore inizializzano il livello o il contenitore.

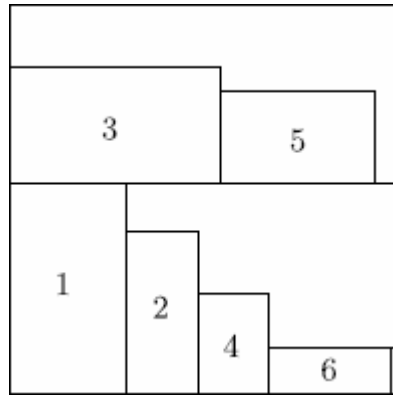


Figura 2.8: Level packing normalizzato

Si può efficientemente costruire il modello del problema 2LBP assumendo che ci siano  $n$  potenziali livelli (l'  $i$ -esimo livello è associato all' item  $i$  che lo inizializza), e  $n$  potenziali contenitori (il  $k$ -esimo contenitore è associato al potenziale  $k$ -esimo livello che lo inizializza). Sia quindi  $y_i, i \in J$  una variabile binaria che assume valore 1 se l'item  $i$  inizializza il livello  $i$ , e valore 0 viceversa. Allo stesso modo sia  $q_k, k \in J$  una variabile binaria che assume valore 1 se il livello  $k$  inizializza il contenitore  $k$ , e valore 0 viceversa. Il modello del problema può essere il seguente:

$$\min \sum_{k=1}^n q_k \quad (1)$$

$$\text{subject to } \sum_{i=1}^{j-1} x_{ij} + y_j = 1 \quad (j=1, \dots, n), \quad (2)$$

$$\sum_{j=i+1}^n w_j x_{ij} \leq (W - w_i) y_i \quad (i=1, \dots, n-1), \quad (3)$$

$$\sum_{k=1}^{i-1} z_{ki} + q_i = y_i \quad (i=1, \dots, n), \quad (4)$$

$$\sum_{i=k+1}^n h_i z_{ki} \leq (H - h_k) q_k \quad (k=1, \dots, n-1), \quad (5)$$

$$y_i, x_{ij}, q_k, z_{ki} \in \{0, 1\} \quad \forall i, j, k \quad (6)$$

Dove  $x_{ij}, i \in J \setminus \{n\}$  e  $j > i$  (rispettivamente  $z_{ki}, k \in J \setminus \{n\}$  e  $i > k$ ) assume valore 1 se l'item  $j$  è impaccato a livello  $i$  (rispettivamente il livello  $i$  è allocato al contenitore  $k$ ), e valore 0 altrimenti. Inoltre:



- Le restrizioni  $j > i$  e  $i > k$  seguono dalle assunzioni (i)-(iii).
- Le equazioni (2) e (4) impongono, rispettivamente, che ogni *item* sia impaccato esattamente una volta, e che ogni livello usato sia assegnato a un unico contenitore.
- Le equazioni (3) e (5) impongono, rispettivamente il vincolo di larghezza di ogni livello usato e il vincolo di altezza di ogni contenitore usato.

Esperimenti computazionali hanno mostrato che il modello è abbastanza utile nella pratica. Il loro uso diretto su un risolutore commerciale di ILP produce risultati molto buoni (e, in molti casi, la soluzione ottima) per istanze realistiche con tempi di CPU brevi. Inoltre alcune varianti del problema 2LBP possono facilmente essere effettuate modificando alcuni vincoli, o aggiungendo altri vincoli lineari al modello. Il modello matematico può anche essere usato per produrre lower bounds, rilassando i requisiti di integralità delle variabili.

### 2.3.2 NON-LEVEL ALGORITHMS

Oltre all'impaccamento a livelli, ci sono stati altri approcci al problema di packing a due dimensioni nella letteratura. La principale strategia “non a livelli” è conosciuta come *Bottom-left (BF)*, e consiste nell'impaccare l'item corrente nella posizione più bassa possibile, giustificata a sinistra. Essa è stata analizzata da Baker (1980) et al., Jakobs (1996) e Liu e Teng (1999). Di questi ultimi l'algoritmo che ha conseguito risultati migliori è stato quello di Baker.

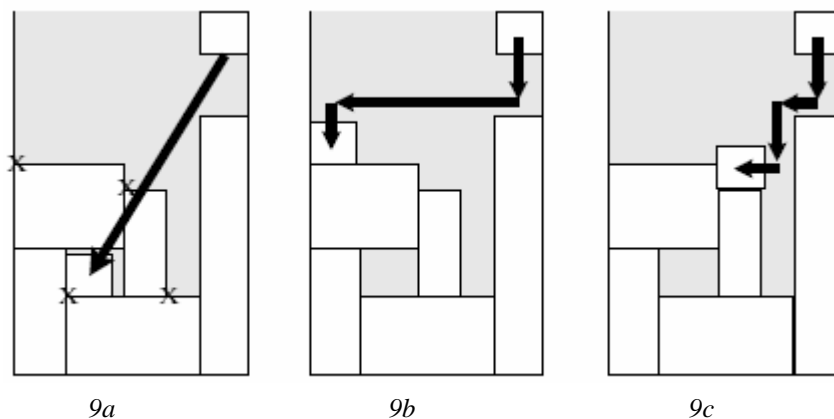


Figura 2.9: Strategie Bottom-Left: Baker et al.(9a), Jakobs (9b), Liu&Teng(9c)

Anche Barkey e Wang (1987) hanno proposto il loro approccio BL per il caso in cui c'è un finito numero di contenitori. Il loro algoritmo *Finite Bottom-Left (FBL)* sistema gli items in ordine decrescente di larghezza. L'item corrente è poi impaccato nella posizione più bassa di ogni contenitore inizializzato, giustificato a sinistra; se nessun contenitore può collocare l'item, ne è inizializzato uno nuovo.

Lodi et al. (1999) hanno proposto l' *alternate directions (AD)*. L'algoritmo inizializza  $L$  contenitori ( $L$  è il lower bound) impaccando sul loro fondo un sottoinsieme di items, seguendo una politica decrementale best-fit. Gli items rimanenti sono impaccati, in un contenitore alla volta, a *strisce*, alternativamente da sinistra a destra e da destra a sinistra. Non appena nessun item può essere impaccato in nessuna delle due direzioni, un nuovo bin inizializzato o un nuovo bin già parzialmente riempito diventa quello corrente. L'algoritmo ha complessità  $O(n^3)$ . Il metodo è illustrato nella seguente figura:

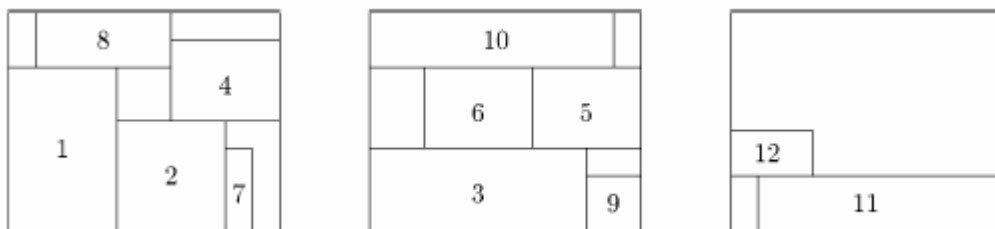


Figura 2.10: Algoritmo AD

Beasley (1985) ha considerato il *two dimensional cutting problem* nel quale a ogni oggetto è associato un profitto, e l'obiettivo è quello di impaccare il sottoinsieme di oggetti avente profitto massimo in un unico contenitore (*cutting stock problem*). Esso ha una formulazione ILP basata sulla rappresentazione discreta dello spazio geometrico e l'uso delle coordinate nelle quali gli oggetti possono essere collocati, vale a dire:

$$x_{ipq} = \begin{cases} 1 & \text{se l'item } i \text{ è posizionato con il suo angolo} \\ & \text{sinistro di base in } (p, q), \\ 0 & \text{altrimenti} \end{cases}$$

Per  $i = 1, \dots, n$ ,  $p = 0, \dots, W - w_i$  e  $q = 0, \dots, H - h_i$ .

Un modello simile, nel quale le coordinate  $p$  e  $q$  sono affrontate attraverso variabili di decisione distinte, è stato introdotto da Hadjiconstantinou e Christofides (1995). Entrambi i modelli sono usati per fornire upper bounds attraverso rilassamenti Lagrangiani e l'ottimizzazione di subgradienti.

Un approccio completamente diverso è stato recentemente proposto da Fekete e Schepers (1997), attraverso una caratterizzazione grafo-teoretica dell'impaccamento di un insieme di oggetti in un unico contenitore.

Sia  $G_w = (V, E_w)$  ( e rispettivamente  $G_h = (V, E_h)$ ) un grafo intervallato avente un vertice  $v_i$  associato a ogni item  $i$  nel packing e un lato tra due vertici  $(v_i, v_j)$  se e solo se le proiezioni degli items  $i$  e  $j$  sull'asse orizzontale (rispettivamente sul verticale) si sovrappongono. E' stato dimostrato appunto da Fekete e Schepers che se il packing è accettabile allora:

- i. Per ogni insieme  $S$  di solidi di  $G_w$  ( rispettivamente  $G_h$ ),  $E_{v_i \in S} w_i \leq W$  ( rispettivamente  $E_{v_i \in S} h_i \leq H$ );
- ii.  $E_w \cap E_h = \{\}$ .

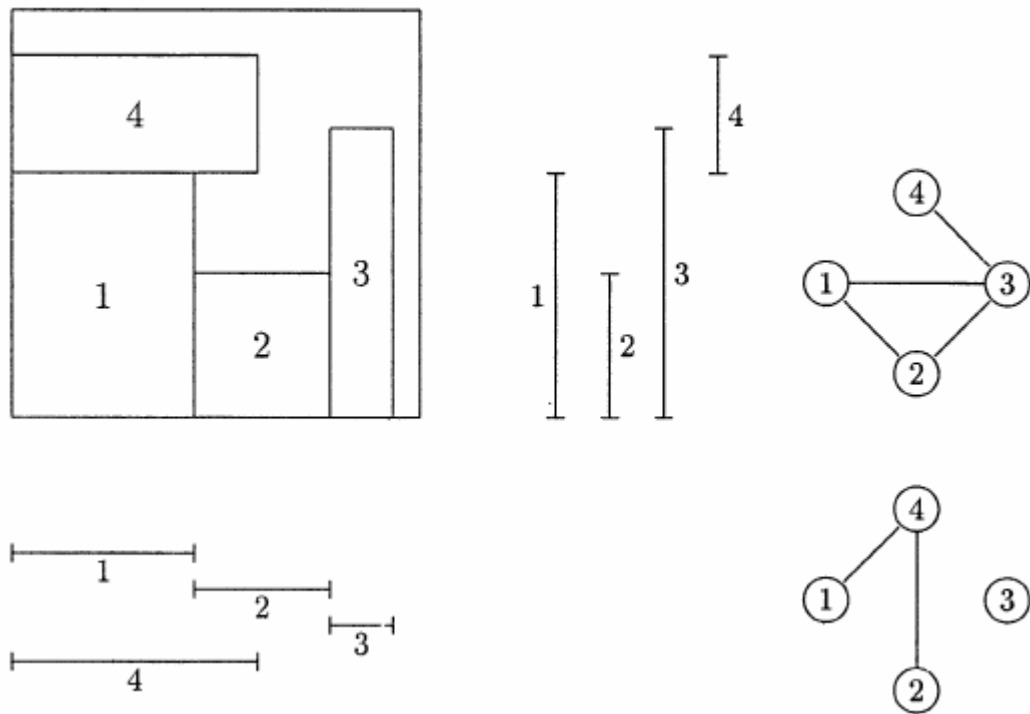


Figura 2.11: L'approccio di Fekete e Schepers

### 2.3.3 ALGORITMI METAURISTICI

Le tecniche metauristiche sono oggi un frequente strumento usato per trovare la soluzione approssimata dei problemi di ottimizzazione combinatoria. Ci si può riferire a Aarts e Lenstra (1997) e a Glover e Laguna (1997) per una introduzione generale all'argomento.

Dowsland (1993) presentò uno dei primi approcci metauristici al 2BPP. Il suo algoritmo *simulated annealing* esplora sia le soluzioni accettabili che le soluzioni nelle quali gli items si sovrappongono. Durante la ricerca, la funzione obiettivo è perciò l'eliminazione totale delle aree sovrapposte, e il vicinato contiene tutte le soluzioni corrispondenti allo spostamento verticale o orizzontale degli items. Appena una nuova soluzione migliore di quella corrente è trovata, l'upper bound è fissato alla sua altezza.

Un'estensione dell'approccio di Dowsland al 2BP è stato proposto da Færø et al. (1999). Essi usano un simile vicinato e una simile strategia di ricerca all'interno di un approccio *guided local search*. Dato un lower bound e un upper bound sul valore della soluzione ottima, se questi non coincidono, l'algoritmo assegna casualmente gli items impaccati nel contenitore di indice più alto ad altri contenitori. La nuova soluzione è di solito non ammissibile, così la nuova funzione obiettivo è la totale eliminazione delle aree sovrapposte, più un termine che penalizza, durante la ricerca, i modelli inammissibili. Il vicinato è esplorato attraverso gli spostamenti degli items. Minimizzare la nuova funzione obiettivo corrisponde a trovare una soluzione ammissibile che coinvolge un contenitore in meno. Il processo è iterato fino a quando l'upper bound non eguaglia il lower bound, oppure fino a quando non si raggiunge un limite di tempo prefissato. Speciali tecniche per ridurre la complessità computazionale per l'esplorazione di questo grande vicinato sono state implementate.

Lodi et al. (1999) hanno sviluppato un algoritmo tabu search per il 2BPP. La caratteristica principale di quest'ultimo è l'adozione di uno schema di ricerca e un vicinato che sono indipendenti dallo specifico packing problem da risolvere. La struttura può anche essere usata per ogni variante del 2BPP ed è stata facilmente estesa anche al problema a tre dimensioni. Partendo da una soluzione accettabile, le mosse la modificano, attraverso un euristico costruttivo, cambiando l'impaccamento di un sottoinsieme  $S$  di items al momento impaccati in  $k$  diversi contenitori, e cercando di

riempire uno specificato *target bin* (un contenitore che può essere facilmente riempito). Sia  $S_i$  l'insieme degli items momentaneamente impaccati nel contenitore  $i$ : il target bin  $t$  è quello che minimizza, tra tutti i bins  $i$ , la funzione:

$$\varphi(S_i) = \alpha \frac{\sum_{j \in S_i} w_j h_j}{WH} - \frac{|S_i|}{n}$$

( $\alpha$  è un peso positivo pre-stabilito). Tale formula dà appunto la misura della facilità con cui un contenitore può essere riempito.

Una volta selezionato il *target bin*, il sottoinsieme  $S$  è definito in modo da includere un item  $j$ , nel target bin e gli oggetti contenuti attualmente dagli altri  $k$  bins. Il nuovo packing per  $S$  è ottenuto eseguendo un appropriato algoritmo euristico  $A$  su  $S$ . Il valore di  $k$ , che definisce la dimensione e la struttura del vicinato corrente, è automaticamente aggiornato durante la ricerca in modo da evadere dall'ottimo locale. Se la mossa impacca gli items  $S$  in  $k$  (o meno) contenitori, cioè l'item  $j$  è stato rimosso dal target bin, un nuovo item è stato selezionato, un nuovo insieme  $S$  è definito, e una nuova mossa è stata eseguita. Altrimenti  $S$  è cambiato selezionando un diverso insieme di  $k$  bins, o un diverso item  $j$  dal target bin ( se tutte le possibili configurazioni di  $k$  bins sono state provate per il corrente item  $j$  ). Se l'algoritmo si blocca, cioè il target bin non è riempito, il vicinato si allarga aumentando il valore di  $k$  fino a un limite superiore prefissato. C'è una *tabu list* e un *tabu tenure* per ogni valore di  $k$ . Un alto valore di  $k$  implica una potente ricombinazione, ma anche l'evidente inconveniente di un elevato tempo computazionale per l'esplorazione del vicinato. L'unica parte che dipende dallo specifico problema è l'euristico costruttivo, usato per ottenere la prima soluzione e per ricombinare gli items ad ogni mossa.

#### 2.3.4 ALGORITMI ESATTI

Martello e Vigo (1998) hanno proposto un algoritmo esatto per il 2BPP. Gli items sono inizialmente sistemati in ordine decrescente rispetto alla loro area, e una procedura di riduzione prova a determinare il packing ottimale di alcuni contenitori, riducendo quindi la dimensione dell'istanza. La prima soluzione in carica, di valore  $z^*$ , è poi ottenuta euristicamente. L'algoritmo si basa su uno schema *two-level branching*:

- I. *outer branch-decision tree*: a ogni nodo decisionale, un item è assegnato a un contenitore senza specificare la sua posizione attuale;
- II. *inner brach-decision tree*: è determinato un packing accettabile (se ce n'è) per gli items assegnati momentaneamente a un contenitore euristicamente o attraverso un algoritmo che enumera tutti i possibili modelli.

L' *outer branch-decision tree* è ricercato in un modo detto *depth-first*, facendo uso di lower bound. Quando è possibile stabilire che ulteriori items non ancora assegnati possono essere messi in un dato contenitore inizializzato, questo bin è chiuso: un bin inizializzato e non chiuso è chiamato *active*. Al livello  $k$  ( $k = 1, \dots, n$ ), l'item  $k$  è assegnato, a turno, a tutti i contenitori inizializzati e, possibilmente, a un nuovo bin ( se il numero di bins richiesto dalla soluzione parziale corrente è minore di  $z^* - 1$  ). L'accettabilità dell'assegnamento di un item  $k$  a un contenitore già contenente un insieme  $I$  di items è controllata euristicamente. Un lower bound  $L(I)$  è calcolato per l'istanza  $I$  definita dagli items correntemente assegnati al contenitore: se  $L(I) > 1$  segue un *backtracking*. Altrimenti, sono applicati a  $I$  algoritmi euristici: se un single-bin packing accettabile è trovato, si riprende l'enumerazione. Se non lo si trova, lo schema inner branching elenca tutti i possibili modi di impaccare gli items  $I$  in un unico bin attraverso il principio *left-most downward*: ad ogni livello, l'item successivo è posizionato, a turno, in tutte le posizioni dove esso ha il lato sinistro adiacente al lato destro di un altro item o al lato sinistro del contenitore, e il lato inferiore adiacente al lato superiore di un altro item o al lato inferiore del contenitore. Non appena si trova un packing accettabile per tutti gli items  $I$ , si riprende l'enumerazione. Se un tale packing non esiste, si esegue un *outer backtracking*.

### 2.3.5 ULTIME RECENSIONI

I packing problems a due dimensioni continuano a essere studiati. E' ancora in stampa l'articolo che parla di due nuovi metodi esatti per risolvere il *two-dimensional orthogonal packing problem* redatto da Clautiaux, Carlier e Moukrim (2006). Essi propongono procedure di riduzione e un nuovo metodo esatto soprannominato *TSBP* (*two-step bin packing*). Il metodo riduce sostanzialmente il numero di nodi visitati del metodo di Martello e Vigo (1998). Questo può essere spiegato per il fatto che per molte istanze non ammissibili, il problema rilassato non ha soluzioni. Inoltre essi propongono metodi per risolvere parte delle ridondanze che ricorrono nel metodo branch&bound e nuovi lower bounds che possono essere usati con risultati positivi.

Sempre Clautiaux, Carlier e Moukrim, riferendosi al *bin packing problem* con “aggiunta di varianti” hanno presentato un nuovo articolo che comparirà presto su Operations Research Letters. In esso propongono un nuovo metodo esatto per il problema. Esso si basa su una decomposizione iterativa dell’insieme di items all’interno di due disgiunti sottoinsiemi. L’algoritmo, confrontato con alcuni *benchmarks* della letteratura conferma l’efficienza del metodo.

Interessante anche l’articolo di Cui e Zhang (2006), anche questo ancora in stampa, sull’*unconstrained two-dimensional cutting problem of rectangular pieces*. Esso descrive un nuovo algoritmo two-stage, ovvero come tagliare generici blocchi in due fasi, con altri due o più fasi richiesti per tagliare i blocchi in pezzi: in un primo momento tagli verticali dividono il foglio di materiale in segmenti, poi tagli orizzontali dividono i segmenti in blocchi. Un generico blocco contiene generiche strisce, e ogni taglio su un blocco produce una di queste strisce. L’algoritmo presentato ricorre a una programmazione dinamica per determinare il layout delle strisce per ogni blocco, risolve un problema di knapsack per ottenere il layout dei blocchi su ogni segmento e il layout del segmento su ogni foglio di materiale. I risultati computazionali indicano che l’algoritmo è molto efficiente e ha tempi di esecuzione ragionabili.

Solo un anno fa inoltre alcuni ricercatori dell’università della Danimarca (Pisinger, Sigurd, 2005) hanno pubblicato un articolo sul packing a due dimensioni considerando due nuove variabili: il cosiddetto *two-dimensional variable sized bin packing problem (2DVSBP)* è il problema di impaccare un insieme di items in un insieme di bins rettangolari aventi *diverse* misure e *diversi costi*, e l’obiettivo è quello di minimizzare il costo dei contenitori usati per impaccare gli oggetti. Nello scritto uscito lo scorso anno su Discrete Optimization gli autori presentano una formulazione intera lineare del problema e introducono diversi lower bound. In esso è inoltre sviluppato un algoritmo esatto basato sul *branch-and-price*.

## 2.4 TWO DIMENSIONAL STRIP PACKING PROBLEM

Dato un insieme di  $J = \{1, \dots, n\}$  di  $n$  oggetti rettangolari (items), ognuno dei quali avente larghezza  $w_j$  e altezza  $h_j$  ( $j \in J$ ), e una striscia di larghezza  $W$  e di altezza infinita, il *two-dimensional strip packing problem (2SPP)* consiste nell’allocazione

ortogonale di tutti gli items, senza sovrapposizione, nella striscia minimizzando l'altezza totale occupata dall'impaccamento. Tale problema compare in diversi contesti reali, come nel taglio di rulli di carta o di tessuto, nel taglio di legno, metallo o vetro da pezzi si stocks standardizzati, nell'allocazione di memoria nei computer, per elencare solo alcuni degli usi più comuni.

Anche questo problema, come il *two-dimensional bin packing problem* è NP-difficile, e può facilmente essere visto come una trasformazione del problema (one-dimensional) *Bin Packing Problem*, che al contrario è un problema “facile”. Data infatti qualsiasi istanza del 1BPP, si ottenga una istanza del 2SPP definendo  $h_j = 1$  per  $j \in J$ : il valore della soluzione ottima di tale istanza è ottima anche per l'istanza del 1BPP.

#### 2.4.1 ALGORITMI ESATTI

Il miglior algoritmo esatto del problema è stato presentato da Martello, Monaci e Vigo (2003). In esso si assume che gli items abbiano un'orientazione fissa, ovvero che non possano essere ruotati. Si suppone inoltre che tutti i dati in input siano positivi interi, e che  $w_j \leq W$  ( $j = 1, \dots, n$ ). Essi, per determinare la soluzione esatta del 2SPP, hanno inserito particolari lower bounds (ottenuti alcuni da semplici considerazioni geometriche, altri da un nuovo rilassamento del problema) in un “adattamento” dell'algoritmo branch-and-bound proposto da Scheithauer (1997) e Martello et al. (2003) per riempire un unico contenitore bidimensionale. L'albero decisionale funziona in questo modo: ad ogni nodo decisionale, la potenziale soluzione corrente, che impacca gli items di un sottoinsieme  $I \subset J$ , è incrementato selezionando a turno ogni item  $j \in J \setminus I$ , e generando nodi discendenti allocando  $j$  in tutte le sue posizioni ammissibili. E' stato provato in Martello et al. (2000) che gli items di  $I$  definiscono un “involucro” che separa le due regioni dove gli items  $J \setminus I$  possono o no essere posizionati, e ciò è sufficiente per generare nodi corrispondenti al piazzamento dell'angolo inferiore sinistro di  $j$  nei punti dove la pendenza dell'involucro cambia da verticale a orizzontale. Questi sono chiamati *corner points* e possono essere determinati in un tempo  $O(|I| \log |I|)$ :



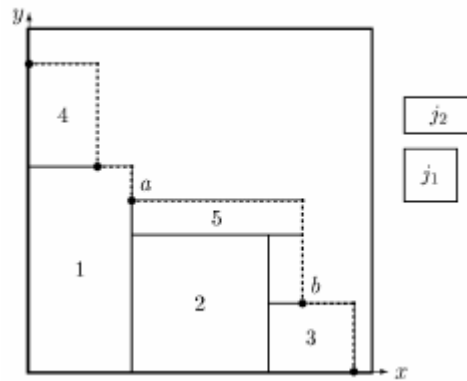


Figura 2.12:  $L$  "involucro" associato agli items di  $I=\{1,2,3,4,5\}$ .

I corner points sono indicati con un puntino nero.

Lo schema branch-and-bound è stato poi migliorato in due modi: attraverso una tecnica per evitare la generazione multipla di nodi decisionali che producono lo stesso modello, e attraverso l'aggiunta di efficaci algoritmi di approssimazione. Si considerino, ad un dato livello  $k$  dell'albero decisionale, due punti d'angolo  $a, b$  e due items  $j_1, j_2 \in J \setminus I$ : la regola di base potrebbe generare quattro nodi, corrispondenti a quattro coppie (corner point, item):  $(a, j_1), (a, j_2), (b, j_1), (b, j_2)$ . La coppia indichi il nodo. Tra i figli di  $(a, j_1)$  si può poi generare un nodo corrispondente alla coppia  $(b, j_2)$ , e, allo stesso livello  $k+1$ , si può generare, tra i figli di  $(b, j_2)$ , un identico nodo corrispondente a  $(a, j_1)$ . Per evitare situazioni di questo genere, che non si presentano solo tra i figli, ma anche, più generalmente, tra discendenti di nodi fratelli, Martello et al. Hanno adottato la seguente tecnica. Per ogni potenziale nodo, corrispondente al piazzamento di un item  $j$  in un angolo  $a$ , essi determinano gli involucri che si dovrebbero generare dall'insieme di item  $\{j\} \cup (I \setminus \{i\})$ , per ogni  $i \in I$ . Se esiste un item  $i \in I$ , correntemente piazzato, in un corner point  $b$ , per il quale l'involucro corrispondente a  $\{j\} \cup (I \setminus \{i\})$  include  $b$  tra i suoi punti d'angolo, sappiamo che il corrente nodo potenziale potrebbe produrre un modello identico a quello prodotto cambiando l'ordine nel quale  $i$  e  $j$  sono considerati nel processo di *branching*. Ecco perché il nodo è generato solo se  $j > i$ .

Per trovare una buona soluzione iniziale ammissibile per il nodo padre, o per migliorare la soluzione corrente per i nodi discendenti, sono implementati alcuni classici algoritmi euristici della letteratura (Coffman et al. (1980); Baker et al. (1980)...).

### 2.4.2 ALGORITMI METAURISTICI

Un approccio metauristico interessante è quello presentato da Iori, Martello e Monaci (2003). Il 2SPP è risolto attraverso una combinazione tra un algoritmo tabu search e un algoritmo genetico.

L'algoritmo di tipo tabu search di riferimento è l'algoritmo 2BP\_TS proposto da Lodi, Martello e Vigo (1999) eseguito indicando con  $z^*$  il valore della soluzione corrente del 2SPP, inizialmente calcolata attraverso algoritmi BUILD<sup>3</sup> e  $TP_{2SP}$ <sup>4</sup>, e applicando particolari procedure di post-ottimizzazione descritte dagli autori. L'algoritmo genetico è invece un algoritmo sviluppato dagli stessi e indicato con 2SP\_GA. Esso si basa sulla permutazione della struttura dei dati che rappresenta l'ordine nel quale gli items sono impaccati nella striscia. La dimensione della popolazione è costante ma vengono prodotti nuovi individui attraverso criteri di riproduzione. E' inoltre ottenuta una diversificazione attraverso l'immigrazione e la mutazione. La ricerca è inoltre intensificata attraverso ricerca locale. Infine nell'algoritmo, viene considerata la storia dell'evoluzione, in modo da intensificare la ricerca in aree promettenti, e da evitarla nei minimi locali.

Sui due algoritmi metauristici sono stati effettuati vari esperimenti computazionali sia su istanze prese dalla letteratura sia su istanze create casualmente: sia il 2SP\_TS che il 2SP\_GA hanno mostrato un buon comportamento empirico. In particolare si è potuto notare come l'approccio genetico abbia migliori prestazioni nel caso di istanze di piccole dimensioni e con piccoli valori della misura W della striscia da riempire, mentre l'opposto accade per l'altro algoritmo. Tuttavia le performance migliori sono state ottenute sperimentando l' *hybrid algorithm*, ovvero la combinazione dei due algoritmi sopra citati.

Oltre al metodo sopra menzionato, nella tabella che segue è riportato un elenco di alcuni di altri recenti sviluppi metauristici per il 2SPP. In particolare in essa sono stati distinti:

---

<sup>3</sup> Algoritmo BUILD: algoritmo che parte dalla soluzione del rilassamento del 1CBP e costruisce una soluzione ammissibile per il 2SPP congiungendo i tagli per ricostruire l'item di origine. Si veda a proposito Martello, Monaci e Vigo: An exact approach to the Strip-Packing problem

<sup>4</sup> Algoritmo  $TP_{2SP}$ : inizializza la striscia ad una lunghezza L, e considera gli items a seconda dell'ordine in cui vengono dati. Per poi impaccarli secondo un preciso metodo. A riguardo si legga Iori, Martello, Monaci: Metaheuristic Algorithms for the Strip Packing Problem

- quattro sottotipi di *Strip Packing Problem*:
  - RF: I pezzi possono essere ruotati di  $90^\circ$  ma non sono richiesti tagli a ghigliottina
  - RG: I pezzi possono essere ruotati di  $90^\circ$  e sono richiesti tagli a ghigliottina
  - OF: L'orientamento dei pezzi è fisso ma non sono richiesti tagli a ghigliottina
  - OG: L'orientamento dei pezzi è fisso e sono richiesti tagli a ghigliottina
  
- tre gruppi di approccio alla soluzione dei metauristici (Hopper e Turton, 2000/2001):
  - 1) Il metodo del primo gruppo usa una codifica delle soluzioni. Di solito, una soluzione genera una sequenza di piazzamenti per i pezzi. La ricerca è svolta dal relativo euristico nello spazio delle soluzioni e tipicamente usa operatori indipendenti dal problema.
  - 2) Gli approcci per le soluzioni del secondo gruppo sono in una posizione intermedia. Mentre, da un lato, soluzioni codificate, contengono già informazioni geometriche o di layout, è richiesto un procedimento addizionale di piazzamento per il posizionamento finale. Tipici di questo gruppo è una codifica specifica al problema di riferimento, spesso basata su grafi.
  - 3) L'approccio del gruppo 3 non usa la codifica. La ricerca avviene direttamente nello spazio dei layout completamente definiti, che sono poi manipolati come detto da specifici operatori.
  
- tre tipi di metauristici:
  - GA: genetic algorithms
  - SA: simulated annealing
  - TS: tabu search

No.	Authors, source	SPP subtype	Group of approaches	Type of metaheuristic
1	Jakobs (1996)	RF	1	GA
2	Dagli and Poshyanonda (1997) and Poshyanonda and Dagli (2004)	RF	1	GA
3	Liu and Teng (1999)	RF	1	GA
4	Hopper and Turton (2000)	RF	1	GA
5	Hopper and Turton (2000)	RF	1	SA
6	Hopper and Turton (2000)	RF	1	NE *
7	Mumford-Valenzuela et al. (2003)	RG	1	GA
8	Kröger (1993, 1995)	RG	2	Parallel GA
9	Schnecke (1996)	RG	2	Parallel GA
10	Ratanapan and Dagli (1997, 1998)	RF	3	GA
11	Iori et al. (2002) and Monaci (2001)	OF	Hybrid	Hybrid, combines GA and TS

Figura 2.13: Metauristici per il 2SPP con pezzi rettangolari.

\* : NE sta per naive evolution, cioè un GA con mutazione ma senza crossover.

### 2.4.3 LEVEL PACKING

Così come si è visto per il *bin packing problem*, Lodi, Martello e Monaci (2002) hanno proposto l'impaccamento a livelli anche per il 2SPP. In particolare, modificando la funzione obiettivo e eliminando tutti i vincoli e le variabili relativi al *bin packing problem* visto nella sezione 2.3.1, il 2LSP (*two-level strip packing problem*) può essere descritto nel modo seguente:

$$\begin{aligned}
 & \min \sum_{i=1}^n h_i y_i \\
 & \text{subject to } \sum_{i=1}^{j-1} x_{ij} + y_j = 1 \quad (j = 1, \dots, n), \\
 & \sum_{j=i+1}^n w_j x_{ij} \leq (W - w_j) y_i \quad (i = 1, \dots, n-1), \\
 & y_i, x_{ij} \in \{0, 1\} \quad \forall i, j.
 \end{aligned}$$

### 2.5 TWO-DIMENSIONAL KNAPSACK PROBLEM

Per descrivere il problema lo si guardi dapprima come un *rectangle packing problem*, ovvero un problema di knapsack a due dimensioni aventi gli oggetti e i contenitori di forma rettangolare.

E' dato un insieme  $N = \{1, \dots, n\}$  di "piccoli" rettangoli (*items*), dove ogni item  $j \in N$  ha una larghezza  $w_j$ , un' altezza  $h_j$  e un profitto  $p_j$ . E' inoltre dato un contenitore rettangolare (*knapsack*) di larghezza  $W$  e altezza  $H$ . L'obiettivo è quello di mettere nel contenitore un sottoinsieme di items che massimizzi il profitto, col vincolo che gli oggetti non si sovrappongano e che ognuno di essi abbia il lato  $h_j$  parallelo al lato  $H$  del contenitore ( con tale requisito il problema viene definito *orthogonal packing problem without rotation* ). Si noti che 2KP è una generalizzazione del già citato *1-dimensional Knapsack problem* , visto come caso particolare avente  $h_j = H$  per ogni  $j \in N$ .

Anche il *two-dimensional Knapsack Problem* è un problema NP-difficile e in letteratura sono stati studiati molti metodi per risolverlo, a partire dal fondamentale studio di Gilmore e Gomory (1965). Tuttavia, sembra non essere conosciuto nessun algoritmo approssimato che garantisca una *worst-case performance*; ciò è sorprendente dato che alcuni risultati di questo genere sono invece noti per il connesso problema di impaccamento a due dimensioni.

Un naturale rilassamento del 2KP è il seguente KP:

$$\begin{aligned} \max \quad & \sum_{j \in N} p_j x_j, \\ & \sum_{j \in N} (w_j h_j) x_j \leq WH, \\ & x_j \in \{0,1\} \quad (j \in N) \end{aligned}$$

nel quale ogni item  $j$  ( $j \in N$ ) ha un profitto  $p_j$  e peso  $w_j h_j$ , uguale all'area del rettangolo  $j$  nel 2KP, e capacità del contenitore  $WH$ , uguale all'area del contenitore nel 2KP. Per una istanza  $I$  del 2KP, sia  $OPT(I)$  la soluzione ottima e  $U_{KP}(I)$  l'upper bound su  $OPT(I)$  corrispondente alla soluzione ottima della formulazione sopra descritta. Il rapporto di performance peggiore è definito come:

$$r(U_{KP}) := \inf_I \left\{ \frac{OPT(I)}{U_{KP}(I)} \right\}$$

In particolare il *worst-case ratio* trovato finora è  $r_{UK} = 1/3$ .

Tale rilassamento, ripreso da Caprara e Monaci (2003) costituisce per questi ultimi la base di partenza di quattro nuovi algoritmi studiati per la risoluzione del 2KP: essi rappresentano l'approccio esatto al problema più recente.

Oltre a Caprara e Monaci, altri studiosi si sono occupati del problema:

### Beasley

Basley (2003) ha presentato un algoritmo euristico di tipo genetico per la risoluzione del problema; esso è basato su una nuova e non lineare formulazione del problema ed è rappresentato da una popolazione euristica (*Population heuristics, PH*) dove la popolazione delle soluzioni al problema si evolve progressivamente. La nuova *PH* proposta si discosta dalle “semplici” popolazioni euristiche precedenti e può essere sintetizzata nel seguente modo:

- a) Genera una popolazione iniziale.
- b) Valuta il fitness e l'unfitness degli individui

REPEAT

- c) Seleziona due individui genitori dalla popolazione (a random)
- d) Ricombina i genitori producendo un solo figlio, attraverso il crossover.

- e) Muta il figlio; uno ogni dieci figli viene mutato.
- f) Valuta il fitness e l'unfitness del figlio e miglioralo usando uno schema euristico di miglioramento
- g) Sostituisci un membro della popolazione con il figlio usando uno schema di rimpiazzamento, a meno che il figlio non sia un duplicato di un membro della popolazione, in tale caso scartalo.

UNTIL

- h) Decidi di fermarti dopo aver riportato la miglior soluzione incontrata.

#### Hadjiconstantinou and Christofides

Hadjiconstantinou and Christofides (1995) hanno presentato una procedura esatta *tree-search* per risolvere il knapsack a due dimensioni. Essi hanno considerato il caso in cui c'è un numero massimo di volte in cui un pezzo può essere usato. L'algoritmo limita la dimensione della *tree-search* usando un bound derivante da un rilassamento Lagrangiano della formulazione binaria del problema. Un'ottimizzazione del subgradiente è poi usata per ottimizzare il bound. Le performance computazionali dell'algoritmo indicano che la precedente è una procedura veramente capace di risolvere ottimamente il problema nel caso di medie dimensioni.

#### R.Alvarez-Valdes, F.Parreño, J.M.Tamarit

R.Alvarez-Valdes, F.Parreño, J.M.Tamarit (2006) presentano un algoritmo tabu search completo di mosse basate sulla riduzione e l'inserzione di pezzi, di procedure di intensificazione e diversificazione basate su una memoria a lungo termine. In particolare l'efficienza delle mosse è basata su una strategia *merge and fill*. I risultati computazionali mostrano che tutte queste idee funzionano bene in particolare per i problemi *constrained e doubly constrained*.

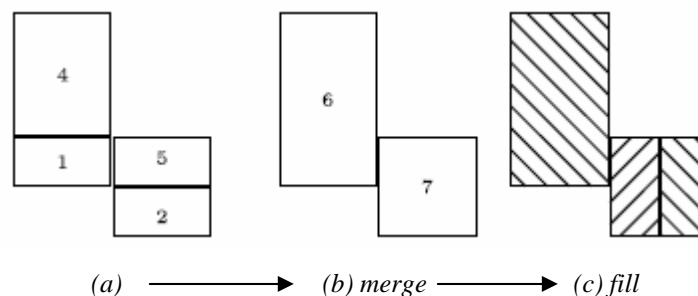


Figura 2.14: Strategia merge and fill

E. Hadjiconstantinou, M. Iori

E. Hadjiconstantinou, M. Iori (2006) presentano un nuovo approccio euristico per risolvere il problema, semplice ma molto efficiente: nella fase preliminare, vengono calcolati semplici upper bounds e soluzioni iniziali sono ottenute attraverso *greedy algorithms*. In seguito è eseguita una ricerca genetica, che usa la selezione dei genitori, la teoria etilista, l'immigrazione, e diversi operatori crossover. L'approccio genetico diviene ibrido dopo aver usato un euristico on-line. Risultati computazionali mostrano che l'algoritmo trova la miglior soluzione in un tempo computazionale molto veloce e che in molti casi migliora gli altri metaeuristici presenti in letteratura.



## Bibliografia:

### *Knapsack Problems*

- A. Caprara, M. Monaci  
**On the 2-Dimensional Knapsack Problem**  
*Operations Research Letters* (2003) 32, 5-14
- G.B.Dantzig  
**Discrete variable extremum problems**  
*Operations Research* (1957) 5, 266-277
- M.L.Fisher  
**The Lagrangian relaxation method for solving integer programming problems**  
*Management Science* (1981) 27, 1-18
- E.Hadjiconstantinou, N.Christofides  
**An exact algorithm for general, orthogonal, two-dimensional knapsack problems**  
*European Journal of Operational Research* (1995) 83, 39-56
- P.J. Kolesar  
**A Branch and Bound Algorithm for the Knapsack Problem**  
*Management Science* (1967) 13, 723-735
- N.Maculan  
**Relaxation Lagrangienne: le problème du knapsack 0-1.**  
*Canadian Journal of Operational Research and Information Processing* (1983) 21, 315-327
- S. Martello, P. Toth.  
**An upper bound for the zero-one knapsack problem and a branch and bound algorithm.**  
*European Journal of Operational Research* (1977) 1, 169–175.
- S. Martello, P. Toth Eds.  
**Knapsack Problems-Algorithms and Computer Implementations**  
John Wiley & Sons (1990)
- D.Pisinger  
**Where are the hard Knapsack Problems?**  
*Computers and Operational Research* (2005) 32, 2271-2282

### *Bin Packing Problems*

- E.Aarts, J.K.Lenstra (Eds.)  
**Local Search in Combinatorial Optimization**  
Wiley, Chichester, 1997.
- R.Alvarez-Valdes, F.Parreño, J.M.Tamarit  
**A tabu search algorithm for a two-dimensional non-guillotine cutting problem**  
*European Journal of Operational Research*, (2006) *In press*
- J.E.Beasley  
**An exact two-dimensional non-guillotine cutting tree search procedure**  
*Operational Research* (1985) 33, 49-64

- J.E.Beasley  
**A population heuristic for constrained two-dimensional non-guillotine cutting**  
*European Journal of Operational Research*, (2003)
- B.S. Baker, E.G.Coffman, R.L. Rivest  
**Orthogonal packing in two dimensions**  
*SIAM Journal on Computing* (1980) 9, 846-855
- O.Barkey, P.Y. Wang  
**Two dimensional finite bin packing algorithms**  
*J.Oper.Res.Soc.* (1987) 38, 423-429
- F.Clautiaux, J.Carlier, A.Moukrim  
**A new exact method for the two-dimensional orthogonal packing problem**  
*European Journal of Operational Research* (2006) *In press*
- F. Clautiaux, J.Carlier, A.Moukrim  
**A new exact method for the two-dimensional bin-packing problem with fixed orientation**  
*Operations Research Letters* (2006) *In press*
- Jr. Coffman, E.G., M.R. Garey, D.S. Johnson  
**Performance bounds for level-oriented two-dimensional packing algorithms**  
*SIAM Journal on Computing* (1980) 9, 801-826
- E.G.Coffman Jr, M.R.Garey, D.S.Johnson  
**Approximation algorithms for bin-packing-an updated survey**  
Algorithm Design for Computer System Design. G.Ausiello, M.Lucertini, P.Serafini (eds), Springer, Vienna (1984), 49-106.
- Y. Cui, X. Zhang  
**Two-stage general block patterns for the two-dimensional cutting problem**  
*Computers and Operations Research* (2006) *In press*
- K.Dowsland  
**Some experiments with simulated annealing techniques for packing problems**  
*European Journal of Operational Research* (1993) 68, 389-399.
- O.Færø, D.Pisinger, M.Zachariasen  
**Guided local search for three-dimensional bin packing problem**  
Technical paper, DIKU, University of Copenhagen, 1999.
- S.P.Fekete, J.Schepers  
**On more-dimensional packing I: Modeling**  
Technical paper ZPR97-288, Mathematisches Institut, Universität zu Köln, 1997.
- P.C. Gilmore, R.E. Gomory  
**Multistage cutting problems of two and more dimensions**  
*Operations Research* (1965) 13, 94-119
- F.Glover, M.Laguna  
**Tabu Search**  
Kluwer Academic Publishers, Boston, 1997.
- E.Hadjiconstantinou, M.Iori  
**An hybrid genetic algorithm for the two-dimensional single large object placement problem**  
*European Journal of Operational Research*, (2006) *In press*

- S. Imahori, M. Yagiura and H. Nagamochi  
**Practical Algorithms for Two-dimensional Packing**  
*Mathematical Engineering Technical Reports*
- S.Jakobs  
**On genetic algorithms for the packing of polygons**  
*European Journal of Operational Research* (1996) 88, 165
- D.s.Johnson, A.Demers, J.D.Ullman, M.R.Garey, R.L. Graham  
**Worst-case performance bounds for simple one-dimensional packing algorithms.**  
*SIAM Journal on Computing* (1974) 3, 299-325
- D.Liu, and Teng, H.  
**An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles**  
*European Journal of Operational Research* (1999) 112, 413
- A. Lodi, S. Martello, M. Monaci  
**Two-dimensional packing problems: A survey**  
*European Journal of Operational Research* 141 (2002) 241-252
- A.Lodi, S.Martello, D.Vigo  
**Approximation algorithms for the oriented two-dimensional bin packing problem**  
*European Journal of Operational Research* (1999) 112, 158-166
- A.Lodi, S.Martello, D.Vigo  
**Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems**  
*INFORMS Journal on Computing* (1999) 11, 345-357.
- A. Lodi, S. Martello, D. Vigo  
**Recent advances on two-dimensional bin packing problems**  
*Discrete Applied Mathematics* (2002) 123, 379-396
- S.Martello, D.Pisinger, D.Vigo  
**The three dimensional bin packing problem**  
*Operations Research*, (2000) 48, 256-267
- S.Martello, D.Vigo  
**Exact solution of the two-dimensional finite bin packing problem**  
*Management Science* (1998) 44, 388-399.
- D. Pisinger, M. Sigurd  
**The two-dimensional bin packing problem with variable bin sizes and costs**  
*Discrete Optimization* (2005) 2, 154-167
- H.H.Yanasse, D.M.Katsurayama  
**Checkerboard pattern: proposals for its generation**  
*International Transactions in Operational Research* (2005) 12, 21-45

#### *Strip Packing Problem*

- A. Bortfeldt  
**A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces**  
*European Journal of Operational Research* (2006) 172, 814-837

- M.Iori, S.Martello, M.Monaci  
**Metaheuristic Algorithms for Strip Packing Problem**  
Optimization and Industry: New Frontiers, Kluwer Academic Publisher  
(Pardalos P. e Korotkich V eds.) (2003)
- S. Martello, M. Monaci, D.Vigo  
**An exact Approach to the Strip-Packing Problem**  
*INFORMS Journal on Computing* (2003) 15, 310-319
- G.Scheithauer  
**Equivalence and dominance for problems of optimal packing of rectangles**  
*Ricerca Operativa* (1997) 83, 3-34

## Capitolo 3

### FACILITY LAYOUT PROBLEM

**I**l *facility layout problem (FLP)* si riferisce all'allocazione di un certo spazio disponibile a una varietà di attività che hanno diverse relazioni tra di loro. Esso è un problema di ottimizzazione combinatoria che si presenta in una varietà di problemi come ad esempio nella progettazione dei layout di ospedali, scuole, e aeroporti; nei problemi di progettazione di impianti elettrici; nei magazzini; nella progettazione di turbine elettriche; etc. Tuttavia la maggior parte delle volte il *facility layout problem (FLP)* è definito come la determinazione dell'organizzazione fisica di un sistema di produzione. Dove posizionare gli impianti e le attrezzature e una loro efficiente progettazione sono problemi strategici e fondamentali che ogni industria di produzione si trova a dover affrontare. La ragione principale, come già detto precedentemente, è che i costi di material handling coprono una buona percentuale dei costi totali di un'azienda. Tompkins e White (1984) hanno stimato che l'8% del prodotto lordo nazionale degli Stati Uniti è stato speso e continua ad esser speso annualmente per nuovi impianti e attrezzature, ciò include anche la modifica di quelli già esistenti. Francis e White (1974) hanno dichiarato che i costi attribuiti alle operazioni di spostamento dei materiali coprono dal 20 al 50 % dei costi operativi totali. Un layout dei macchinari efficiente può ridurre questi costi dal 10 al 30 % all'anno. Un layout inefficiente può infatti portare a un accumulo di giacenze di work-in-progress, un sovraccarico dei sistemi di movimentazione dei materiali, tempi di setup inefficienti. Inoltre, il *facility layout problem* rappresenta un investimento costoso e a lungo termine. Anche le modifiche richiedono grandi spese perché non possono essere effettuate facilmente: il re-layout delle strutture non è solo un consumo di tempo, ma interrompe le attività dei lavoratori e il flusso dei materiali. E' per questo che l'impatto strategico della progettazione del layout non dovrebbe essere ignorato.

Negli ultimi anni, molti ricercatori hanno proposto una varietà di procedure per risolvere il problema: solo negli ultimi venti anni sono stati pubblicati circa 140 articoli inerenti a quest'area della ricerca. In particolare sviluppare una soluzione ottima per l'allocazione di attrezzature dalle forme rettangolari, usando l'adiacenza come criterio di interesse nel selezionare le attrezzature che devono condividere i confini. Molte delle procedure recenti inoltre richiedono interazioni con il progettista degli impianti e necessitano di un buon progetto come scheletro iniziale.

Data una matrice di flusso tra diversi impianti e l'area allocata a ciascuno di essi, la ricerca del layout ottimo è stata messa a fuoco sviluppando diverse tecniche. La maggior parte delle procedure sono state classificate come “*single-row*” quando gli impianti sono sistemati linearmente su di una fila, oppure *multi-row facilities* quando gli impianti sono sistemati linearmente in due o più file. Due classi di modelli e algoritmi sono inoltre stati usati per risolvere sia i problemi di layout *single-row* che quelli di tipo *multi-row*: procedure ottimali e euristici. Di seguito saranno elencate le principali.

### 3.1 MODELLI PER IL FACILITY LAYOUT PROBLEM

#### 3.1.1 QAP MODEL

Sebbene molti ricercatori abbiano usato altre formulazioni per risolvere il facility layout problem, il *quadratic assignment problem (QAP)* è ancora la formulazione più utilizzata: essa usa il baricentro della locazione come punto di riferimento per valutare la distanza tra due attrezzature. Inoltre molto spesso, le formulazioni diverse dal QAP sono basate su varianti del QAP o usano il QAP come struttura di riferimento.

Koopmans e Beckman (1957) sono stati i primi a formulare il facility layout problem come QAP. Il nome “quadratic assignment problem” è stato scelto perché la funzione obiettivo è una funzione polinomiale di secondo grado rispetto alle variabili, e i vincoli sono uguali ai vincoli del problema dell'assegnamento. L'obiettivo del QAP è quello di trovare l'assegnamento ottimo di  $n$  facilities (impianti, dipartimenti o stazioni macchina) in  $n$  siti in modo da minimizzare il costo della movimentazione dei materiali espresso come il prodotto del flusso di lavoro per la distanza. Il QAP può essere formulato come segue:

$$\min \sum_{i=1}^n \sum_{j=1}^n \sum_{s=1}^n \sum_{t=1}^n A_{ijst} x_{ij} x_{st} \quad (1)$$

$$s.t. \sum_{i=1}^n x_{ij} = 1 \quad \forall j, \quad (2)$$

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i, \quad (3)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j, \quad (4)$$

Dove

$$A_{ijst} = \begin{cases} w_{is} d_{jt} & \text{se } i \neq s \text{ oppure } j \neq t \\ F_{ij} & \text{se } i = s \text{ oppure } j = t \\ \infty & \text{se } i = s \text{ e } j \neq t \end{cases} \quad (5)$$

$w_{is}$  è il flusso tra gli impianti  $i$  e  $s$ ,  $w_{ii} = 0$ ,  $d_{jt}$  la distanza tra le posizioni  $j$  e  $t$ ,  $d_{jj} = 0$ ,  $F_{ij}$  il costo fisso per locare l'impianto  $i$  nella posizione  $j$ , e

$$x_{ij} = \begin{cases} 1 & \text{se l'impianto } i \text{ è allocato alla posizione } j, \\ 0 & \text{altrimenti} \end{cases} \quad (6)$$

Il vincolo (2) è la restrizione che solo un impianto può essere piazzato in una determinata posizione, mentre il vincolo (3) assicura che ogni posizione può essere assegnata a un solo impianto. L'obiettivo è quello di minimizzare il flusso totale tra gli impianti  $i$  ( $i=1, \dots, n$ ) e  $s$  ( $s=1, \dots, n$ ).

La funzione obiettivo può anche essere completata con un fattore che considera il costo sostenuto se l'impianto  $i$  è piazzato nella posizione  $s$ ; la funzione diviene pertanto:

$$\min \sum_{i=1}^n \sum_{j=1}^n \sum_{s=1}^n \sum_{t=1}^n A_{ijst} x_{ij} x_{st} + \sum_{i=1}^n \sum_{s=1}^n c_{is} x_{is} \quad (7)$$

Il QAP è un problema NP-difficile (Sahni e Gonzalez, 1976): finora l'istanza più grande per la quale è stata trovata una soluzione ottima ha una dimensione di  $n \approx 30$ . Per questa ragione sono stati molti gli euristici proposti per risolvere il problema. In particolare Lawler (1963) ha presentato una formulazione generale del QAP, del quale sostiene che la formulazione di Koopmans-Beckmann sia un caso particolare, e proposto un metodo per calcolare la soluzione ottima seguendo una logica di partizione. Egli ha inoltre dimostrato l'equivalenza del problema QAP con un problema di

assegnamento lineare avente certi vincoli addizionali: un QAP con  $n^2$  variabili  $x_{ij}$  può essere reso lineare definendo  $n^4$  variabili  $y_{ijpq}$ , dove

$$y_{ijpq} = x_{ij}x_{pq} . \quad (8)$$

Christofides et al.(1980) hanno proposto una tecnica di bounding basata sull'estrazione dalla formulazione del problema QAP, di un ampio problema di assegnamento lineare (che può essere risolto all'ottimo), lasciando il rimanente problema QAP di dimensioni più ridotte possibili. La soluzione del residuo QAP può poi essere risolto con una procedura separata. Questo metodo *2-step* produce bounds migliori rispetto a quelli prodotti all'applicazione diretta di algoritmi di bounding al QAP originale.

### 3.1.2 GRAPH THEORY MODEL

Nell'approccio dei grafi ogni dipartimento o macchinario (ignorando l'area e la forma del dipartimento all'inizio) è definito come un nodo all'interno di una rete di un grafo. Questi dipendono dall'adiacenza predefinita e opportuna di ogni coppia di macchinari. In altre parole, si può dire che nell'approccio dei grafi, si assume che si conosca l'opportuna vicinanza della locazione di ogni coppia di attrezzature. Come accade per il QAP, anche in questo caso problemi di aree diverse, perfino di piccole dimensioni, non possono essere risolti all'ottimo. Numerose pubblicazioni su questo argomento sono state pubblicate; in esse sono stati analizzati diversi modelli e algoritmi. In particolare una rassegna degli approcci dei grafi più emergenti si può trovare in Foulds (1991) e Hassan e Hogg (1987).

### 3.1.3 MIP MODEL

Anche metodi basati sul *Mixed Integer Programming* sono stati presi in considerazione per risolvere il FLP. Il primo a formulare questo metodo è stato Montreuil (1990) : un obiettivo basato sulla distanza è usato nella rappresentazione di un layout che è una estensione del discreto QAP. Heragu e Kusiak (1991) hanno sviluppato un caso speciale di questo MIP. Lacksonen (1994) hanno proposto un algoritmo *two-step* per risolvere il FLP assumendo variabile l'area che può risolvere il dynamic facility layout e rettangolare l'area di tutti i dipartimenti. Lacksonen (1997) ha poi esteso il modello proposto per trattare aree disuguali e costi di ridisposizione. Tuttavia, il modello può essere risolto all'ottimo solo nel caso di problemi di piccole dimensioni. Kim e Kim



(1999) hanno considerato il problema di allocare i punti di input e output (I/O) di ogni dipartimento per un dato blocco di layout che l'obiettivo di minimizzare la distanza totale di trasporto. E' stato proposto un nuovo algoritmo branch-and-bound che sembra migliorare l'efficienza anche per problemi di grandi dimensioni. Anche se, la soluzione simultanea del problema del layout e dei punti di I/O non è ancora stato risolto. Barbosa-Povoa et al. (2001) hanno proposto un approccio di programmazione matematica per il problema del layout generalizzato.

Un MIP dettagliato è stato presentato anche da Montreuil(1990). Sebbene il suo approccio riserbi molte promesse, al momento sono risolvibili solo FLP di dimensione sei o minori. L'obiettivo è basato sulla distanza rettilinea di flusso e di tempo tra il baricentro e i dipartimenti.

### **3.2 METODOLOGIE DI RISOLUZIONE**

Varie metodologie di risoluzione sono disponibili per risolvere il facility layout problem.

#### **3.2.1 METODI ESATTI**

Sono stati usati metodi branch-and-bound per trovare una soluzione ottima del FLP formulato come assegnamento quadratico poichè il QAP coinvolge solo variabili binarie. In letteratura sono però riportate soluzioni ottime per problemi di dimensione fino a 16. Per  $n > 16$  diventa impossibile per il computer risolvere il problema e, di conseguenza, anche un computer molto potente non può affrontare un'istanza di grandi dimensioni.

#### **3.2.2 EURISTICI**

Gli algoritmi euristici possono essere classificati come algoritmi di tipo *costruttivo* poichè una soluzione viene costruita da uno schizzo, ma anche di tipo *migliorativo* poichè la soluzione iniziale viene migliorata. I metodi *costruttivi* sono considerati gli approcci più semplici e tradizionali per risolvere il QAP da un punto di vista concettuale e di implementazione, ma la qualità delle soluzioni prodotte non è di solito soddisfacente. I metodi basati sul *miglioramento* partono da una soluzione ammissibile e provano a migliorarla attraverso interscambi tra singoli assegnamenti. Questi due tipi di approcci (costruttivi e migliorativi) possono facilmente essere combinati tra loro. Ad

esempio CRAFT è un algoritmo migliorativo molto usato che usa doppi interscambi. Una lista degli euristici più conosciuti usati per risolvere il FLP è quella riportata in tabella 3.1. Questi euristici sono classificati come algoritmi basati sull' *adiacenza* o sulla *distanza*. La differenza tra questi due tipi di algoritmi si trova nella funzione obiettivo. La funzione obiettivo per gli algoritmi di adiacenza è data come:

$$\max \sum_i \sum_j (r_{ij}) x_{ij} \quad (9)$$

dove  $x_{ij}$  vale 1 se il dipartimento  $i$  è adiacente al dipartimento  $j$  e 0 altrimenti. Il principio di base che sta dietro a questa funzione obiettivo è che i costi di movimentazione dei materiali è ridotto significativamente se i due dipartimenti hanno i confini adiacenti.

La funzione obiettivo degli algoritmi basati sulla distanza è invece data come:

$$\min(TC) = \frac{1}{2} * \sum_{\substack{i=1 \\ i \neq k}}^n \sum_{\substack{j=1 \\ j \neq l}}^n \sum_{k=1}^n \sum_{l=1}^n C_{ik} * D_{jl} * X_{ij} * X_{kl} \quad (10)$$

$X_{ij} = 1$  se l'impianto  $i$  è assegnato alla posizione  $j$

$X_{ij} = 0$  se l'impianto  $i$  non è assegnato alla posizione  $j$

$C_{ik}$  = costo per posizionare l'impianto  $i$  nella posizione  $k$

$D_{jl}$  = distanza tra la posizione  $j$  e  $l$

La filosofia da sottolineare dietro a questa funzione obiettivo è che la distanza aumenta il costo totale dei viaggi. Visto che tutti gli indici sono sommati da 1 a  $n$ , ogni assegnamento è contato due volte; da qui la necessità di moltiplicare per  $\frac{1}{2}$ .

$C_{ik}$  può essere sostituito con  $F_{ik}$  ( = flusso tra gli impianti  $i$  e  $k$  ) a seconda dell'obiettivo.

S.No	References	Name of package
1	Dr. Gordan Armour	CRAFT
2	Seehof and Evans	ALDEP
3	Dr. Moore James	CORELAP
4	Michael P. Deisenroth	PLANET
5	Teichholz Eric	COMP2
6	Kaiman Lee	COMPROPLAN COMSBUL
7	Robert C. Lee	CORELAP8
8	Robert Dhillon	DOMINO
9	Teichholz Eric	GRASP
10	Dr. Johnson T.E.	IMAGE
11	Dr. Warnecke	KONUVER
12	Dr. Warnecke	LAYADAPT
13	Raimo Matto	LAYOPT
14	John S. Gero	LAYOUT
15	Dr. Love R.F.	LOVE*
16	Dr. Warnecke	MUSTLAP2
17	Dr. Vollman Thomas	OFFICE
18	McRoberts K.	PLAN
19	Anderson David	PREP
20	Moucka Jan	RG and RR
21	Dr. Ritzman L.P.	RITZMAN*
22	Dr. Warnecke	SISTLAPM
23	Prof. Spillers	SUMI
24	Hitchings G.	Terminal Sampling Procedure
25	Johnson	SPACECRAFT
26	Tompkins and Reed	COFAD
27	Hassan, Hogg and Smith	SHAPE
28	Banerjee et al.	QLAARP
29	Tam	LOGIC
30	Bozer, Meller, and Erlebacher	MULTIPLE
31	Tate and Smith	FLEX-BAY
32	Foulds and Robinson	DA (Adjacency Based)
33	Montreuil, Ratliff and Goetschalckx	MATCH (Adjacency Based)
34	Goetschalckx	SPIRAL (Adjacency Based)
35	Balkrishnan et al.	FACOPT

Tabella 3.1 – Raccolta di articoli sugli algoritmi euristici usati per la risoluzione del FLP

### 3.2.3 METAURISTICI

Vari metaeuristici come ad esempio SA e GA sono attualmente usati per approssimare la soluzione di FLP di grandi dimensioni. La tecnica SA deriva dalla teoria della meccanica statistica ed è basata sull'analogia tra la fusione dei solidi e la risoluzione di problemi di ottimizzazione. Burkard e Rendl (1984) hanno studiato un algoritmo SA per il QAP. Una raccolta molto recente degli articoli basati sugli algoritmi SA è riportata nella tabella 3.2.

S. No.	Reference	Year	QAP	MIP	Heuristic
1	Kirkpatrick et al.	1983	√		Simulated annealing
2	Burkard and Rendl	1984	√		Simulated annealing
3	Wilhelm and Ward	1987	√		Simulated annealing
4	Kaku and Thomson	1986	√		Simulated annealing
5	Connolly	1990	√		Simulated annealing
6	Laursen	1993	√		Simulated annealing
7	Tam	1992		√	Simulated annealing
8	Heragu and Alfa	1992			√ Simulated annealing
9	Kouvelis et al.	1992	√		Simulated annealing
10	Jajodia et al.	1992			√ Simulated annealing
11	Shang	1993	√		SA and AHP
12	Souilah	1995		√	Simulated annealing
13	Peng et al.	1996	√		Simulated annealing
14	Meller and Bozer	1996			√ Simulated annealing
15	Azadivar and Wang	2000	√		Simulated annealing
16	Baykasoglu and Gindy	2001	√		Simulated annealing
17	Misevicius	2003	√		Simulated annealing
18	Balakrishnan et al.	2003	√		√ SA and GA

Tabella 3.2 – Raccolta di articoli sugli algoritmi SA usati per la risoluzione del FLP

Durante l'ultimo decennio anche l'algoritmo genetico ha guadagnato molta attenzione; esso utilizza una codifica binaria di individui come stringhe di lunghezza fissa. GA cerca iterativamente l'ottimo globale, senza esaurire lo spazio delle soluzioni, in un processo parallelo che inizia da un piccolo insieme di soluzioni ammissibili (popolazione) e che genera nuove soluzioni in un qualche modo aleatorio. La performance del GA dipende dal problema perché la sistemazione dei parametri e lo schema di rappresentazione dipendono dalla natura del problema. Tavakkoli-Moghaddam e Shayan (1998) hanno analizzato l'adequatezza degli algoritmi genetici per risolvere il FLP. La tabella 3.3 fornisce recenti articoli sui FLP basati sui GA.

Gli algoritmi Tabu-Search (TS) sono altre procedure iterative progettate per risolvere problemi di ottimizzazione. Helm e Hadley (2000) hanno applicato algoritmi TS per la

risoluzione del FLP. Il metodo è ancora molto studiato, ed è in continua evoluzione e miglioramento. Recentemente, alcuni articoli sono apparsi dove un algoritmo ant colony è stato usato per risolvere FLP di grandi dimensioni. Talbi et al. (2001) hanno usato un algoritmo ant colony per risolvere il QAP.

S. No.	Reference	Year	QAP	MIP	Heuristic	
1	Tam	1992		√	Genetic algorithm	
2	Banerjee and Zhou	1995		√	Genetic search	
3	Tate and Smith	1995	√		GA	
4	Kochhar and Heragu	1998		√	√	Extension of GA
5	Islier	1998				GA
6	Rajshekaran et al.	1998		√	√	GA
7	Mak et al.	1998		√		GA
8	Mckendall et al.	1999		√	√	GA nested approach
9	Kochhar and Heragu	1999			√	GA
10	Gau and Meller	1999		√	√	GA
11	Azadivar and Wang	2000	√			GA and simulation algorithm
12	Al-Hakim	2000				GA
13	Ahuja	2000	√			Genetic algorithm
14	Wu and Appleton	2002		√		GA
15	Lee, Han and Roh	2003		√		GA, Dijkstra algorithm
16	Balakrishnan et al.	2003	√		√	GA and SA

*Tabella 3.3 – Raccolta di articoli sugli algoritmi GA usati per la risoluzione del FLP*

### 3.2.4 ALTRI APPROCCI

Altri approcci che sono altrettanto usati per il FLP sono la rete neurale e la fuzzy logic. Una raccolta degli articoli su queste metodologie applicate nella risoluzione del FLP sono elencate in tabella 3.4.

S. No.	Reference	Year	QAP	MIP	Heuristic	Techniques
1	Dutta and Sahu	1982	√		√	
2	Murtagh et al.	1982	√		√	
3	Foulds	1983	√			Graph theory
4	Herroelen and Vangils	1985				Flow dominance theory
5	Fortenberry and Fox	1985		√		Pair-wise exchange
6	Hammouch and Webster	1985				Graph theory (theoretical approach)
7	Foulds and Giffin	1985			√	Graph theory
8	Green and Al_Hakim	1985		√	√	
9	Rosenblatt	1986	√			Dynamic programming
10	Kaku and Thomson	1986	√			Simulated annealing
11	Hassan et al.	1986		√	√	Construction
12	Foulds et al.	1986	√			Graph theory
13	Grobelyny	1987			√	Fuzzy approach
14	Evans et al.	1987	√			Fuzzy set theory
15	Urban	1987	√		√	
16	Rosenblatt and Lee	1987	√		√	
17	Jacobs	1987	√			Graph theory
18	Montreuil et al.	1987				Graph theory
19	Hassan and Hogg	1987				Graph theory
20	Grobelyny	1988			√	Fuzzy approach
21	Kaku et al.	1988	√		√	
22	Kumar et al.	1988				Expert system, pattern recognition
23	Smith and Macleod	1988	√			L. R. and B and B
24	Malakooti and Tsurushima	1989				Expert system, rule based
25	Malakooti	1989	√		√	
26	Heragu and Kusiak	1988		√	√	
27	Heragu and Kusiak	1990		√		Knowledge approach
28	Abdou and Dutta	1990				Expert system
29	Houshyar and McGinis	1990	√		√	Cut approach
30	Al-Hakim	1991				Graph theory
31	Heragu and Kusiak	1991		√	√	Unconstrained opt.
32	Kaku et al.	1991		√	√	
33	Hassan and Hogg	1991		√		Graph theory
34	Logendran	1991		√	√	
35	Burkard et al.	1991	√			QAP_LIB
36	Camp et al.	1992		√	√	Penalty function
37	Leung	1992			√	Graph theory
38	Kaku and Rachamadya	1992	√		√	
39	Rosenblatt and Golany	1992	√		√	
40	Goetschalckx	1992	√		√	Graph theory
41	Harmonosky and Tothoro	1992	√		√	Pairwise, construction
42	Askin and Mitwasi	1992		√	√	
43	Balakrishnan et al.	1992	√		√	
44	Al-Hakim	1992				Graph theory
45	Lacksonan and Ensore	1993	√			B and B, cutting plane, D.P.

S. No.	Reference	Year	QAP	MIP	Heuristic	Techniques
50	Rao and Rakshit	1994		√		Fuzzy based
51	Urban	1993	√	√		
52	Montreuil et al.	1993	√			Graph theory, LP
53	Bozer et al.	1994		√		
54	Boswell	1994		√		Graph theory based
55	Sirinaovakul	1994		√		Knowledge based expert
56	Langevin et al.	1994	√	√		
57	Trethway and Footle	1994		√		
58	White	1996	√			Lagrangian relaxation
59	Badiru and Arif	1996				Fuzzy theory
60	Chiang and Kouvelis	1996		√		Tabu Search
61	Watson and Giffin	1997		√		Vertex splitting algo.
62	Meller	1997	√	√		
63	Lacksonan	1997	√	√		Branch and bound
64	Bozer and Meller	1997		√		
65	Sarker et al.	1998	√	√		
	Zetu et al.	1998				Virtual reality(Theoretical approach)
66	Urban	1998	√			Dynammic programming
67	Chan and Sha	1999	√	√		
68	Smith and Helm	1999				Virtual reality (Theoretical approach)
69	Dweiri	1999				Fuzzy based
70	Helm and Hadley	2000	√	√		Tabu-search based
71	Knowles and Corne	2002	√			Multi-obj. approach
72	Kim and Kim	2000	√	√		
73	Barbosa-Povoa et al.	2001	√	√		
74	Al-Hakim	2001				Maximally planer graph
75	Wang and Sarker	2002	√	√		
76	Chan, Chan and Ip	2002	√	√		
77	Diponegoro and Sarker	2003	√	√		
78	Castillo and Peters	2003	√	√		Extended distance based

Tabella 4 – Raccolta di articoli su altri approcci usati per la risoluzione del FLP

### 3.3 ORIENTAMENTI CORRENTI E FUTURI CAMPI D'AZIONE

Osservando tutte le tabelle sopra riportate, si può concludere che la ricerca sul FLP non è convergente, ma in qualche modo divergente. Al momento, AI può essere usata per sviluppare euristici che risolvano FLP di grandi dimensioni; inoltre è necessaria più ricerca nelle funzioni multi-obiettivo per raggiungere criteri di layout più rilevanti.

Ogni due anni il *Material Handling Institute of America*, insieme ad altre industrie sponsorizzatrici e alle agenzie di governo, organizza un consorzio sulla ricerca riguardo le novità sui sistemi di trasporto materiali dove i ricercatori possono presentare le proprie ricerche. È stato scoperto che c'è una mancanza di applicazione della

*concurrent engineering* nel FLP in accordo con la scelta del sistema di movimentazione materiali che mostra come la progettazione del layout dei macchinari sia indipendente dalla scelta del sistema di movimentazione. E' stato calcolato che lo stesso layout non è appropriato per tutti i periodi, poiché la domanda non rimane la stessa. Da qui, la ricerca dovrebbe indirizzarsi verso un layout delle attrezzature dinamico e non statico.

Per la risoluzione del FLP sta emergendo soprattutto la ricerca nell'applicazione di meta-euristici come gli algoritmi SA, GA e TS. Tuttavia, il risultato finale dipende anche dalla soluzione iniziale presa (popolazione). Quindi, la ricerca deve anche sviluppare validi euristici che generino buone soluzioni iniziali.



## Bibliografia:

- AP. Borbosa-Povoa, R. Mateus, AQ. Novais  
**Optimal two dimensional layout of industrial facilities**  
*International Journal of Production Research* (2001) 39, 2567-2593
- RE. Burkard, F. Rend  
**A thermodynamically motivated simulation procedure for combinatorial optimization problems**  
*European Journal of Operational Research* (1984) 17, 169-174
- W. Chiang, C. Chiang  
**Intelligent local search strategies for solving facility layout problems with the quadratic assignment problem formulation**  
*European Journal of Operational Research* 106 (1998) 457-488
- N. Christofides, A. Mingozzi, P. Toth  
**Contributions to the quadratic assignment problem**  
*European Journal of Operational Research* (1980) 18, 243-247
- LR Foulds  
**Graph theory and applications**  
 Springer, Berlin Heidelberg New York (1991)
- RL Francis, JA White  
**Facility layout and location: an analytical approach**  
 Prentice Hall, Englewood Cliffs, NJ (1974)
- MMD. Hassan, GL Hogg  
**A review of graph theory applications to the facilities layout problem**  
*Omega* (1987) 15, 291-300
- SA Helm, SW Hadley  
**Tabu search based heuristics for multi floor facility layout**  
*Int.J. Prod Res* (2000) 38; 365-383
- S. Heragu, A. Kusiak  
**Efficient models for the facility layout problems**  
*European Journal of Operational Research* (1991) 53, 1-13
- A. Houshyar, B. White  
**Comparison of solution procedures to the facility location problem**  
*Computers industrial Engng.* Vol.32 (1997) pp. 77-87
- JY Kim, YD Kim  
**A branch-and-bound algorithm for locating input and output points of departments on the block layout**  
*Journal of Operational Research Soc.* (1999) 50, 517-525
- TC Koopmans, M. Beckmann  
**Assignment problems and the location of economic activities**  
*Econometrica* (1957) 25, 53-76
- TA. Lacksonen  
**Static and dynamic facility layout problems with varying areas**  
*Journal of Operational Research Soc.* (1994) 45, 59-69
- TA. Lacksonen  
**Pre-processing for static and dynamic facility layout problems**  
*Int J Prod Res* (1997) 35, 1095-1106
- E.L. Lawler  
**The Quadratic Assignment Problem**  
*Management Science* (1963) 9, 586-599

- B.Montreuil  
**A modelling framework for integrating layout design and flow network design**  
*Proceedings of the material handling research colloquium*, Hebron, KY, (1990)  
pp 43-58
- S.P.Singh, R.R.K.Sharma  
**A review of different approaches to the facility layout problems**  
*Int.J. Manuf Technol* (2006) 30; 425-433
- EG Talbi, O.Roux, C.Fonlupt, D.Robillard  
**Parallel ant colonies for quadratic assignment problem**  
*Future Generation Computer System* (2001) 17, 441-449
- R.Tavakkoli-Moghaddain, E. Shanyan  
**Facilities layout design by genetic algorithms**  
*Computer Industrial Engineering* (1998) 35, 527-530
- JA.Tompkins, JA White  
**Facilities planning**  
Wiley, New York (1984)

## Capitolo 4

### MODELLI E ALGORITMI PER L'OTTIMIZZAZIONE DI LAYOUT FIERISTICI

**L**o scopo del capitolo è quello di riuscire a trovare un modello che riesca ad ottimizzare i layout fieristici; data una superficie  $S$  per l'esposizione di forma irregolare e non convessa, e un numero  $n$  di stand uguali e rettangolari, si vuole massimizzare il numero di stand contenuti in  $S$  cercando allo stesso tempo di ottenere un layout che possa rispondere alle esigenze di un contesto fieristico. Perché il layout sia realmente applicabile alla superficie è infatti necessario che si verifichino le seguenti condizioni:

- Gli stand non devono sovrapporsi
- Gli stand devono essere completamente contenuti in  $S$ , ovvero i confini degli stessi non possono oltrepassare quelli dell'area disponibile per l'esposizione
- Gli stand devono essere disposti in modo che i clienti possano facilmente accedervi e visitarli

Oltre a questi, si possono considerare altri vincoli addizionali a seconda delle necessità del richiedente; ad esempio si può considerare il caso in cui ogni stand abbia un profitto e lo scopo sia quello di massimizzare il profitto totale.

Il problema è di grande interesse nel campo dell'ottimizzazione combinatoria, soprattutto perché mancano riferimenti bibliografici o ricerche già effettuate su di esso. Un problema simile già ampiamente studiato è quello concernente l'ottimizzazione dei layout industriali; si usa il termine “simile” e non il termine “uguale” perché nell'ambito delle fiere non bisogna tenere conto del flusso: tra uno stand e uno altro non vi è nessun flusso di materiale come invece accade tra un macchinario ed un altro in un'impresa manifatturiera.

Il problema del layout fieristico sembra molto vicino anche ai problemi di impaccamento (*packing*); in particolare esso può essere visto come un problema di impaccamento a due dimensioni (*two-dimensional packing problem*, Vedi cap.2) e risulta essere molto utile nella pratica. Appunto perché il problema generalizza problemi di *packing* già conosciuti e studiati, viene automatico dire che si tratta di un problema NP-difficile (considerando il caso più generale in cui ogni stand può essere ruotato in ogni direzione possibile). La cosa che complica ulteriormente il problema in questione rispetto ai problemi di *packing* trattati nei capitoli precedenti è il fatto che si ha a che

fare con una superficie espositiva irregolare. L'area in figura 4.1 è quella che sarà presa come campione in tutti gli esempi del capitolo:

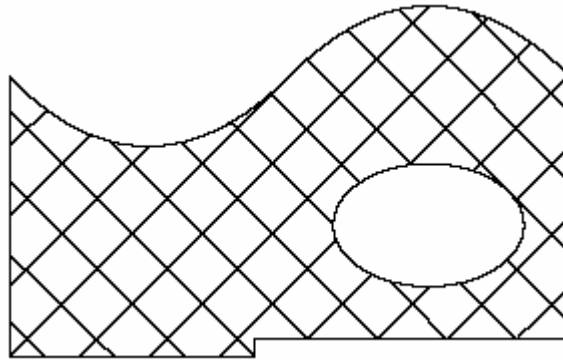


Figura 4.1: esempio di un'area di esposizione non convessa e irregolare

La ricerca sarà circoscritta all'impaccamento a due dimensioni; inoltre si assumerà che gli stand abbiano tutti lo stesso orientamento e che siano tutti allineati lungo una certa direzione che rimarrà fissa, in modo da lasciare abbastanza spazio tra una serie di stand allineati ( che d'ora in poi sarà chiamata *strip* ) e un'altra, per facilitare il passaggio dei clienti. Queste restrizioni ci permettono di concentrarci su un problema più semplice, che rimane comunque in un contesto reale; infatti nelle fiere è comune usare una disposizione degli stand di questo tipo.

In figura 4.2 è rappresentato un possibile impaccamento a due dimensioni relativo alla superficie presentata precedentemente. Un impaccamento di questo tipo è chiamato in letteratura *two-dimensional two-staged packing*; si supponga che si voglia tagliare l'area per l'esposizione per produrre una serie di aree più piccole rappresentanti gli stand, ciò può essere fatto grazie a una prima fase di tagli a ghigliottina paralleli al lato rappresentante l'altezza e a una seconda fase in cui si eseguono tagli, sempre a ghigliottina, paralleli al lato rappresentante la larghezza. Nella seconda fase è necessario effettuare un taglio per ogni striscia ottenuta nella prima fase.

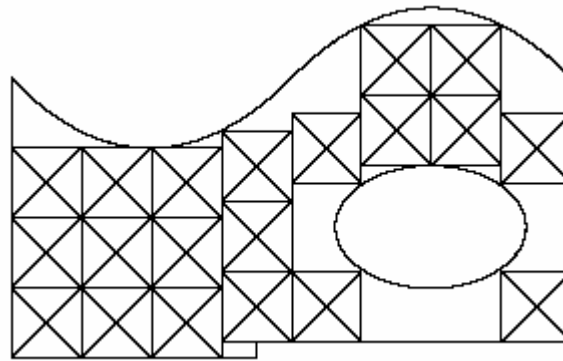


Figura 4.2: esempio di two-staged feasible packing

Una richiesta aggiuntiva potrebbe essere di piazzare gli stand allineati anche ad una seconda direzione, perpendicolare alla prima, così da formare una seconda serie di vicoli per i clienti. In figura 4.3 è raffigurato un esempio di questo tipo di impaccamento; esso è conosciuto in letteratura col nome di *checkerboard packing* (vedi Cap.2) ; in questo caso le aree rappresentanti gli stand possono essere tagliati dall'area d'esposizione applicando due soli tagli a ghigliottina, uno parallelo all'altezza e uno alla larghezza ed entrambi sulla stessa superficie di partenza.

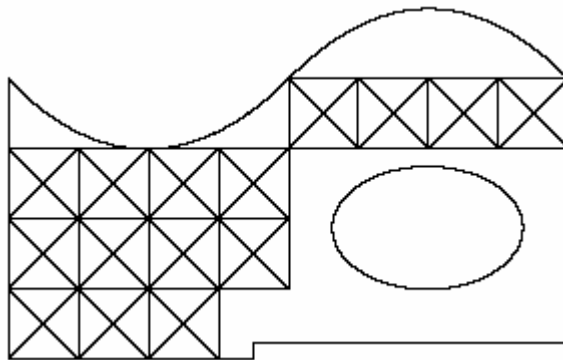


Figura 4.3: esempio di checkerboard feasible packing

Il capitolo si concentrerà nella risoluzione di questi due particolari problemi di impaccamento.

#### 4.1 DESCRIZIONE DEL PROBLEMA

E' data una superficie non convessa, chiamata superficie *principale* o di *esposizione*, e un insieme di stand rettangolari identici. Si considerano le seguenti assunzioni:

- *Assunzione 1:* L'area principale è limitata e in particolare è contenuta in un rettangolo di larghezza  $W$  e altezza  $H$
- *Assunzione 2:* Gli stand sono identici e hanno una forma rettangolare di larghezza  $w$  e altezza  $h$
- *Assunzione 3:* Non è ammessa alcuna rotazione degli stand
- *Assunzione 4:* Gli stand sono piazzati su *strip* che hanno lo stesso orientamento, ovvero, gli stand potrebbero immaginariamente essere ottenuti tagliando la superficie con tagli a *ghigliottina* paralleli al lato di altezza  $H$  del rettangolo che contiene la superficie principale.

Si chiamerà quindi *Fair Lay-Out Optimization Problem (FLOP)* il problema di impaccare il massimo numero possibile di stand nella superficie principale, in modo che gli stand siano completamente contenuti in essa, che non si sovrappongano e che non violino le assunzioni 1-4.

Le assunzioni 1, 2 e 3 sono strettamente dipendenti dall'orientamento nel quale si considera la superficie principale; quest'ultimo può essere imposto dalla particolare configurazione della superficie oppure essere scelto in modo arbitrario.

Secondo l'assunzione 2 si noti che si prenderà in esame il problema nel quale tutti gli stand sono uguali e nel quale si vuole massimizzarne il numero. Un problema connesso potrebbe essere quello di avere stand di diverse dimensioni e profitti, volendo massimizzare il profitto totale. Questo problema non sarà considerato nel capitolo in quanto diverso dalla situazione reale che si sta cercando di risolvere.

L'assunzione 4 ci dice che gli stand devono essere tutti allineati. Si ricordi che un taglio a ghigliottina è un taglio che va da un lato a quello opposto di una superficie. In questo caso i tagli andranno effettuati in modo perpendicolare al lato di altezza  $H$  della superficie d'esposizione.

Altre possibili richieste che ci si potrebbe trovare a dover soddisfare sono le seguenti:

- *Assunzione 5:* Ogni stand ha un retro e un fronte. Il fronte avere almeno una distanza principale  $a$  dagli altri stand piazzati davanti: ciò è utile per permettere il passaggio dei visitatori ma anche per lasciare agli artigiani lo spazio necessario per caricare e scaricare la loro merce all'inizio e alla fine della giornata. Il dietro necessita almeno di una distanza di servizio  $b$  dagli altri stand, in modo che gli artigiani abbiano uno spazio adatto in cui stare durante le ore di lavoro. La distanza di servizio deve essere minore o uguale a quella principale ( $b \leq a$ ). In alcuni contesti in cui non è richiesto alcun spazio di servizio, si può supporre  $b=0$ .
- *Assunzione 6:* Il retro e il fronte di ogni stand deve essere accessibile anche dall'esterno della superficie principale (cioè, sia clienti che artigiani devono poter camminare anche all'esterno della superficie d'esposizione)

L'assunzione 5 obbliga a considerare distanze ben definite tra gli stand (in questo caso tra una *strip* e un'altra) in modo da permettere il passaggio di clienti e artigiani: da qui deriva la creazione di vicoli (*alley*) tra ogni coppia di colonne. Ogni colonna si trova ad avere un vicolo *principale* di fronte e un vicolo *di servizio* dietro di sé.

Nell'assunzione 6, si considera di avere aree di accesso per i clienti e per gli artigiani lungo tutto il bordo della superficie. Gli stand che sono piazzati ai bordi della superficie principale non necessitano di avere tutte entrambe le distanze  $a$  e  $b$  poiché l'accesso o dei clienti o degli artigiani può essere effettuato dall'esterno. Quest'ultima considerazione è data ad esempio dal caso della fiera di Fortaleza dove la superficie di esposizione confina con una spiaggia e con larghe pavimentazioni di dimensioni maggiori di  $a$ . Se si volesse riferire il problema ad una fiera situata al chiuso, o comunque circondata da barriere architettoniche (es. muri) sarebbe sufficiente modificare il vincolo appena esposto.

Un vincolo ulteriore può essere quello di effettuare un impaccamento *checkerboard*:

- *Assunzione 7:* Gli stand non devono solamente essere piazzati a strisce (*strip*), ma devono anche essere piazzati alla stessa altezza.

Nelle prossime sezioni del capitolo saranno considerate queste tre diverse versioni del problema:

*FLOP1*: lo scopo è impaccare il numero massimo di stand sulla superficie di esposizione, in modo che gli stand siano completamente contenuti in essa, che non si sovrappongano e che non violino le assunzioni 1-4. Questa situazione è la più semplice e la più vicina ai problemi di packing della letteratura.

*FLOP2*: il problema è come il *FLOP1*, ma deve soddisfare anche le assunzioni 5 e 6. Esso si avvicina già a un caso reale nel quale, come nel *FLOP1*, gli stand sono piazzati in modo *two-dimensional two-staged*.

*FLOP3*: il problema è come il *FLOP2*, ma l'impaccamento deve soddisfare anche l'assunzione 7. Anche questo può rappresentare un caso reale, nel quale gli stand sono piazzati in modo *checkerboard*.

#### 4.2 UNO SCHEMA APPROSSIMATIVO PER IL FLOP

Si crei una matrice  $\phi$ , composta da quadrati di dimensione  $[\delta \times \delta]$  che copra totalmente il rettangolo di dimensioni  $[W \times H]$  contenente la superficie di esposizione. D'ora in poi, usando un approccio granulare, si esprimeranno tutte le dimensioni della superficie principale e di quella degli stand come multipli di questi (piccoli) valori  $\delta$ . In particolare è definita la seguente dimensione:

$$X_\delta = \left\lfloor \frac{X}{\delta} \right\rfloor \quad (1)$$

dove  $X = \{W, H, w, h\}$ . La matrice  $\phi$  ha perciò dimensioni  $[W_\delta \times H_\delta]$ . Un elemento della matrice assume valore 1 se il corrispondente quadratino  $[\delta \times \delta]$  può essere usato completamente per l'impaccamento degli stand:

$$\phi_{ij} = \begin{cases} 1 & \text{se il quadrato } (i, j) \text{ può essere completamente usato per l'impaccamento} \\ 0 & \text{altrimenti} \end{cases} \quad (2)$$

per ogni  $j = 1, \dots, W_\delta, i = 1, \dots, H_\delta$ .

Si noti che per  $\delta \ll W$ , o ancora meglio  $\delta \ll w$ , l'approssimazione è molto bassa. In figura 4.4 è rappresentata una possibile matrice  $\phi$  associata all'area rappresentata in figura 4.1. In particolare nella figura si ha  $W_\delta = 16$ ,  $H_\delta = 10$  e  $w_\delta = h_\delta = 2$  (giusto per



curiosità, nel caso reale che si vuole risolvere  $w_\delta = h_\delta = 20$ , quindi l'approssimazione è 100 volte migliore rispetto a quella in figura).

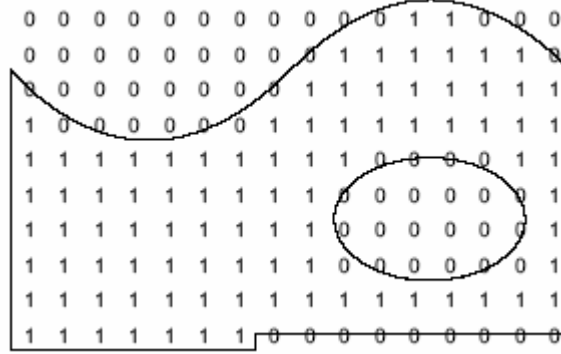


Figura 4.4: matrice associata all'area di esposizione di figura 4.1

Ora si numerino le colonne da 1 a  $W_\delta$ . Si dirà che una colonna è scelta per l'impaccamento, se una striscia di stand è piazzata con l'angolo sinistro dello stand all'inizio della colonna. Per esempio, l'impaccamento di figura 4.2 associato alla matrice  $\phi$  della figura 4.4 sceglierebbe le colonne 1, 3, 5, 7, 9, 11, 13, 15, 17. I problemi di ottimizzazione combinatoria che si analizzeranno determinano le colonne da scegliere per impaccare le strisce di stand.

La matrice sopra definita sarà usata nel capitolo per determinare i modelli e gli algoritmi per il problema di ottimizzazione che si sta discutendo.

### 4.3 MODELLI MATEMATICI E ALGORITMI PER IL FLOP1

Data una matrice  $\phi$  di dimensioni  $[W_\delta \times H_\delta]$ , si costruisca una seconda matrice  $\theta$  di dimensioni  $(W_\delta - w_\delta + 1) \times H_\delta$  imponendo

$$\theta_{ij} = \sum_{h=j}^{j+w_\delta-1} \phi_{ih} \quad (3)$$

per  $j = 1, \dots, W_\delta - w_\delta + 1$ ,  $i = 1, \dots, H_\delta$ . Si noti che se lungo la colonna  $j$  ci sono  $h_\delta$  posizioni  $\theta_{ij}$  consecutive di valore  $w_\delta$ , allora si può impaccare uno stand in quella colonna. Dopodichè si conti per ogni colonna il numero di stand impaccati definendo il profitto come:

$p_j = n^\circ$  di stand che possono essere impaccati con l'angolo sinistro nella colonna  $j$

per  $j = 1, \dots, W_\delta - w_\delta + 1$ . Si noti che le ultime colonne  $w_\delta$  non sono definite in  $\theta$ , poiché nessuno stand può essere impaccato in quelle posizioni senza eccedere  $W_\delta$ . In figura 4.5 è rappresentata la matrice  $\theta$  corrispondente alla matrice  $\phi$  definita in figura 4.4. Al di sotto della matrice sono riportati i profitti  $p_j$  associati alle colonne; per esempio se si sceglie la colonna 1 si possono impaccare 3 stand, mentre se si sceglie la colonna 15 se ne può impaccare solo uno.

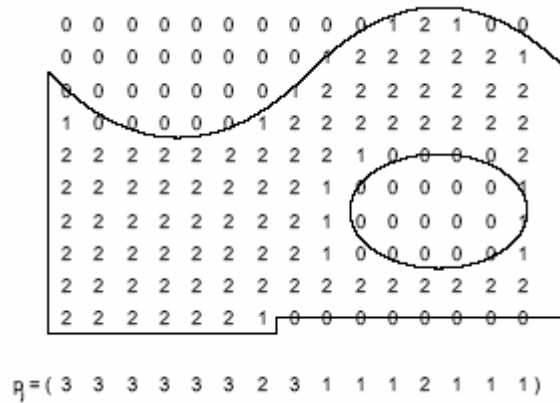


Figura 4.5: Matrice  $\theta$  per il calcolo dei profitti  $p_j$  associati a ogni a colonna  $j=1, \dots, W_\delta - w_\delta + 1$ .

Si noti che la costruzione dei profitti  $p_j$  impiega tempo  $O(W_\delta H_\delta)$ , cioè un tempo strettamente proporzionale al tempo richiesto per leggere la matrice.

I profitti possono essere usati per sviluppare algoritmi e per trovare soluzioni approssimate per il FLOP. E' importante notare che questi algoritmi sono efficienti poiché non hanno bisogno di tempi esponenziali, in quanto la matrice è totalmente unimodulare (Vedi paragrafo 4.3.2).

#### 4.3.1 MODELLO MATEMATICO PER IL FLOP1

Si ricordi che nel *FLOP1* si considerano solo le assunzioni 1-4 descritte nel paragrafo 4.1, e perciò non c'è nessuna richiesta che riguardi la distanza tra due *strips* diverse di stand. Per creare un modello per il *FLOP1* è definita la variabile:

$$x_j = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (4)$$

per  $j = 1, \dots, W_\delta - w_\delta + 1$

Il modello risulta dunque essere il seguente:

$$(MM1) \quad \text{Max } z = \sum_{j=1}^{W_\delta - w_\delta + 1} p_j x_j \quad (5)$$

$$\sum_{j=k}^{k - w_\delta + 1} x_j \leq 1 \quad \forall k = 1, \dots, W_\delta - w_\delta + 1 \quad (6)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, W_\delta \quad (7)$$

Il modello (5)-(7) è un classico modello di impaccamento e risolve correttamente il problema. In particolare il vincolo (6) impone che gli stand non si sovrappongano. Si ha bisogno di  $W_\delta - w_\delta + 1$  variabili binarie e di  $W_\delta - w_\delta + 1$  disuguaglianze. Si noti che il modello MM1 non ci dice esplicitamente a quale altezza può essere posizionato ogni stand; tuttavia quest'informazione può essere facilmente ottenuta una volta saputa quale colonna può essere presa. Il modello è costruito in modo che per ogni colonna scelta, si parte a impaccare gli stand dal fondo verso l'alto, fino a quando non possono più essere posizionati nuovi stand. Il modello MM1 applicato alla superficie di esposizione rappresentata in Figura 4.1 e alla matrice definita in Figura 4.4 produce l'impaccamento raffigurato in Figura 4.2 con  $z = 20$ .

#### 4.3.2 CASI IN CUI LA MATRICE E' TOTALMENTE UNIMODULARE

Si ricordi che, dato un modello ILP, la soluzione del corrispondente rilassamento lineare è sempre intera se tutta la parte destra del modello è intera e se la matrice dei vincoli è totalmente unimodulare (TUM). Si ricordi anche che le condizioni sufficienti affinché una matrice  $A$  sia totalmente unimodulare (Papadimitriou e Steiglitz, 1988) sono le seguenti:

- i.  $a_{ij} \in \{-1, 0, 1\}$
- ii. Ogni colonna non ha più di due elementi non nulli
- iii. Esiste una partizione  $(A_1, A_2)$  delle righe tale che per ogni colonna con due elementi non nulli, questi appartengono uno a  $A_1$ , e l'altro a  $A_2$  se e solo se sono concordi in segno

Si chiami  $A$  la matrice associata ai vincoli (6):  $A$  è totalmente unimodulare. La matrice  $A$  può infatti essere disegnata come in Figura 4.6, dove si può notare che per ogni riga c'è un gruppo di  $w_\delta$  elementi di valore 1, mentre tutti gli altri elementi hanno valore 0.

1	1	...	1	1	0	0	...	0	0		
0	1	1	...	1	1	0	0	...	0	0	
0	0	1	1	...	1	1	0	0	...	0	0
...											
0	0	...	0	0	1	1	...	1	1	0	0
0	0	...	0	0	1	1	...	1	1	0	0
0	0	...	0	0	1	1	...	1	1	1	1

Figura 4.6: Matrice  $A$  associata ai vincoli (6)

Ora sia  $a'_i$  il vettore associato alla riga  $i$  di  $A$ , per  $i = 1, \dots, W_\delta - w_\delta + 1$ , e si modifichi  $A$  nel seguente modo lineare: per ogni riga  $a'_i$  si ponga  $a'_i = a'_i - a'_{i+1}$ , per  $i=1, \dots, W_\delta - w_\delta + 1$ . In questo modo, partendo da  $A$  si ottiene la matrice  $\tilde{A}$  di Figura 4.7. E' facile notare che la matrice  $\tilde{A}$  soddisfa le condizioni sufficienti per essere una matrice TUM. In particolare la colonna avente più di due elementi non nulli può essere divisa in  $A_1 = A$  e  $A_2 = \emptyset$ .

1	0	0	...	0	0	-1	0	0	...	0	0		
0	1	0	0	...	0	0	-1	0	0	...	0	0	
0	0	1	0	0	...	0	0	-1	0	0	...	0	0
...													
0	0	...	0	0	1	0	0	...	0	0	-1	0	0
0	0	...	0	0	1	0	0	...	0	0	-1	0	0
0	0	...	0	0	1	1	...	1	1	1	1	1	1

Figura 4.7: Matrice  $\tilde{A}$  ottenuta da  $A$  dopo alcune operazioni lineari sulle righe.

Si può quindi dire che  $A$  è totalmente unimodulare e di conseguenza il problema  $FLOPI$  può essere risolto in modo polinomiale, sostituendo il vincolo (7) con il rispettivo rilassamento lineare:

$$0 \leq x_j \leq 1 \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (8)$$

e risolvendo il relativo modello lineare (5), (6) e (8).

#### 4.4 MODELLI MATEMATICI PER IL FLOP2

Si supponga ora che ci debba essere una distanza minima  $a_\delta$  tra ogni coppia consecutiva di *strip* di stand, ma che non ci sia bisogno della distanza di sicurezza. Ciò significa, come scritto nell'assunzione 4 del paragrafo 4.1, che  $b_\delta = a_\delta$ . Questo nuovo caso, in cui si tiene conto di una distanza tra due colonne di stand, può essere descritta dal seguente modello:

$$(MM2) \quad \text{Max} \quad \sum_{j=1}^{W_\delta - w_\delta + 1} p_j x_j \quad (9)$$

$$s.t. \quad \sum_{j=k}^{k+w_\delta+a_\delta-1} x_j \leq 1 \quad \forall k = 1, \dots, W_\delta - 2w_\delta - a_\delta + 2 \quad (10)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (11)$$

Il modello (9)-(11) ha quindi bisogno di  $W_\delta - w_\delta + 1$  variabili binarie  $W_\delta - 2w_\delta - a_\delta + 2$  disuguaglianze. La distanza  $a_\delta$  viene considerata nel vincolo (10); essa incrementa il numero delle variabili (cioè del numero di variabili che copre una certa colonna). La dimostrazione riguardo alla totale unimodularità vista nel paragrafo 4.3.2 può essere applicato direttamente alla matrice MM2; in questo modo il problema può essere risolto in modo polinomiale sostituendo la (11) con la seguente:

$$0 \leq x_j \leq 1 \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (12)$$

e risolvendo il relativo modello lineare (9), (10) e (12).

Si noti che il numero di variabili del modello MM2 è uguale a quello di MM1. Questo perché è ammissibile che l'ultima *strip* di stand confini col bordo dell'area di esposizione; l'accesso a questa colonna di stand è infatti possibile anche senza lasciare la distanza  $a_\delta$  poiché la superficie confina, in questo caso, con una spiaggia o con pavimentazioni e non con altre barriere architettoniche. L'ultima *strip* può quindi essere piazzata nella colonna  $W_\delta - w_\delta + 1$ .

La logica del modello non perderebbe comunque significato se applicato ad una fiera in cui l'area espositiva è delimitata, ad esempio, da muri. In questo caso infatti il modello sarebbe ancora valido: bisognerebbe solo limitare le variabili al seguente intervallo

$$x_j \in \{0,1\} \quad \forall j = 1, \dots, W_\delta - w_\delta - a_\delta + 1 \quad (13)$$

e cambiare di conseguenza le equazioni (9) e (10).

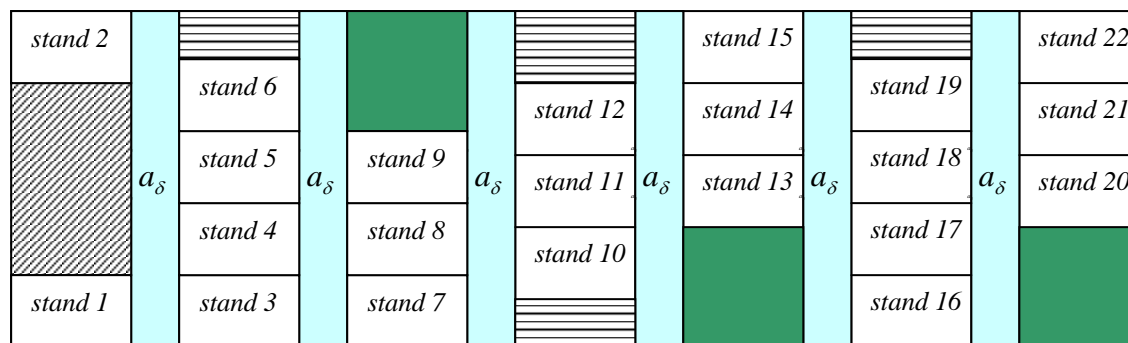


Figura 4.8: Esempio di un layout che tiene conto di una distanza  $a_\delta$  tra una strip di stand e un'altra. Gli spazi che non possono essere occupati da stand sono stati riempiti di verde o tratteggiati in nero senza alcun significato in particolare.

#### 4.4.1 UN MODELLO MATEMATICO PER IL FLOP2 CON UN ACCESSO

Il modello MM2 risolve i problemi in cui ogni colonna di stand ha una distanza minima di valore  $a_\delta$ , sia sulla sinistra che sulla destra, dalle colonne precedenti e successive di stand. Si prende ora in considerazione la possibilità che due *strip* possano essere adiacenti, ma che comunque mantengano almeno un accesso. Così, in questa nuova ottica, ogni colonna di stand deve avere una distanza di valore  $a_\delta$  o dalla colonna precedente o dalla colonna successiva. In questo caso, come specificato nell'assunzione 4 del paragrafo 4.1, si pone  $b_\delta = 0$ .

Il problema risultante può essere trasformato in un modello usando un insieme di variabili associate a una *single strip* (cioè una strip composta da un'unica colonna di stand) e un insieme di variabili associate a una *double strip* (cioè una strip composta da due colonne di stand piazzate una strettamente vicina all'altra).

In figura 4.9 sono rappresentate graficamente le differenze tra i problemi MM1 e MM2.

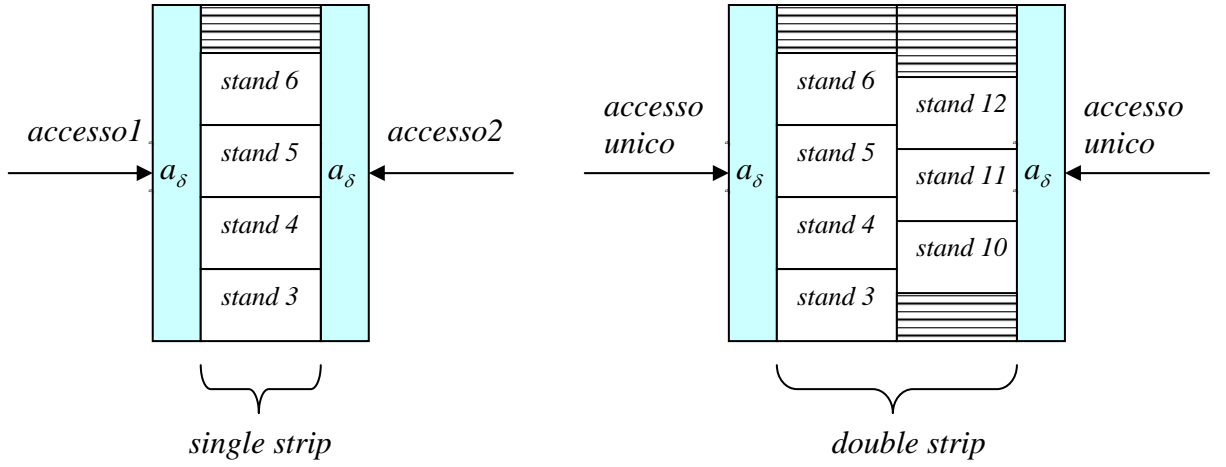


Figura 4.9: differenza tra MM1, in cui ogni stand ha due accessi e vi sono single strip, e MM2, in cui ogni stand ha un solo accesso e possono anche essere presenti double strip.

Si possono quindi definire le seguenti variabili:

$$x_j^1 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + a_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (14)$$

$$\text{per } j = 1, \dots, W_\delta - w_\delta + 1$$

$$x_j^2 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + w_\delta + a_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (15)$$

$$\text{per } j = 1, \dots, W_\delta - 2w_\delta + 1$$

Il modello del problema è il seguente:

$$(MM3) \quad \text{Max} \quad \sum_{j=1}^{W_\delta - w_\delta + 1} p_j^1 x_j^1 + \sum_{j=1}^{W_\delta - 2w_\delta + 1} p_j^2 x_j^2 \quad (16)$$

$$\text{s.t.} \quad \sum_{j=k-w_\delta-a_\delta}^k x_j^1 + \sum_{j=k-2w_\delta-a_\delta}^k x_j^2 \leq 1 \quad \forall k = w_\delta + a_\delta, \dots, W_\delta - w_\delta + 1 \quad (17)$$

$$x_j^1 \in \{0,1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (18)$$

$$x_j^2 \in \{0,1\} \quad \forall j = 1, \dots, W_\delta - 2w_\delta + 1 \quad (19)$$

dove  $p_j^1 = p_j$  per  $j=1, \dots, W_\delta - w_\delta + 1$  e  $p_j^2 = p_j + p_{j+w_\delta}$  per  $j=1, \dots, W_\delta - 2w_\delta + 1$  sono i nuovi profitti dei due tipi di *strips*. Naturalmente, nella disequazione (17) gli indici negativi sono semplicemente trascurati nella formula.

Anche in questo caso si è in presenza di una matrice TUM e il problema è risolto all'ottimo considerando il rilassamento lineare di MM3.

Come nel caso precedente, anche questo modello, dopo opportune modifiche, può essere usato nel caso in cui l'area di esposizione sia circondata da muri.

#### 4.4.2 MODELLO MATEMATICO PER IL FLOP2 CON DISTANZA DI SERVIZIO FISSA

Si supponga che, per ogni *strip*, ci debba essere una distanza  $a_\delta$  dalla *strip* precedente, o dalla *strip* successiva. Inoltre, tra ogni coppia di *strip* consecutive è necessario ci sia esattamente una distanza  $b_\delta$  per l'accesso di servizio. Si sta parlando cioè del caso in cui ogni stand ha un accesso principale di larghezza minima  $a_\delta$  e un accesso di servizio di larghezza uguale a  $b_\delta$ ; esso è interessante, ad esempio, quando gli stand possono essere costruiti usando solo due tipologie di struttura, ovvero quella singola o quella doppia. Tutto ciò implica una piccola modifica delle assunzioni 1-4 del paragrafo 4.1: bisogna considerare la distanza di servizio non *almeno* uguale a  $b_\delta$ , ma *esattamente* uguale a  $b_\delta$ .

Analogamente a quanto fatto nel caso precedente, si può creare un modello per questo problema definendo le variabili:

$$x_j^1 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + a_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (20)$$

$$\text{per } j = 1, \dots, W_\delta - w_\delta + 1$$

$$x_j^2 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + b_\delta + w_\delta + a_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (21)$$

$$\text{per } j = 1, \dots, W_\delta - 2w_\delta - b_\delta + 1$$



Il modello del problema è il seguente:

$$(MM4) \quad \text{Max} \quad \sum_{j=1}^{W_\delta - w_\delta + 1} p_j^1 x_j^1 + \sum_{j=1}^{W_\delta - 2w_\delta - b_\delta + 1} p_j^2 x_j^2 \quad (21)$$

$$s.t. \quad \sum_{j=k-w_\delta-a_\delta}^k x_j^1 + \sum_{j=k-2w_\delta-a_\delta-b_\delta}^k x_j^2 \leq 1 \quad \forall k = w_\delta + a_\delta, \dots, W_\delta - w_\delta + 1 \quad (22)$$

$$x_j^1 \in \{0, 1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (23)$$

$$x_j^2 \in \{0, 1\} \quad \forall j = 1, \dots, W_\delta - 2w_\delta - b_\delta + 1 \quad (24)$$

dove  $p_j^1 = p_j$  per  $j = 1, \dots, W_\delta - w_\delta + 1$  e  $p_j^2 = p_j + p_{j+w_\delta+b_\delta}$  per  $j = 1, \dots, W_\delta - 2w_\delta - b_\delta + 1$  sono i nuovi profitti dei due tipi di *strip*. Ancora una volta si è in presenza di una matrice totalmente unimodulare. Inoltre, come nel caso precedente, gli indici negativi presenti nel vincolo (22) vengono semplicemente trascurati.

Infine si noti che il modello (21)-(24) uguaglia il modello (16)-(19) per  $b_\delta = 0$ .

#### 4.4.3 UN MODELLO MATEMATICO GENERALE PER IL FLOP2

Il modello descritto in questa sezione è quello che si riferisce al caso più generale del *FLOP2*. Si supponga che, per ogni *strip*, ci debba essere una distanza minima di  $a_\delta$  o rispetto alla *strip* precedente o rispetto a quella successiva. Inoltre, tra ogni coppia di *strip* consecutive ci deve essere una distanza minima di servizio pari a  $b_\delta$ .

Sono introdotte le seguenti variabili:

$$x_j^1 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + b_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (25)$$

$$x_j^2 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + a_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (26)$$

per  $j = 1, \dots, W_\delta - w_\delta + 1$

Il modello del problema è il seguente:

$$(MM5) \quad \text{Max} \quad \sum_{j=1}^{W_\delta - w_\delta + 1} p_j x_j^1 + \sum_{j=1}^{W_\delta - w_\delta + 1} p_j x_j^2 \quad (27)$$

$$s.t. \quad \sum_{j=k-w_\delta-b_\delta}^k x_j^1 + \sum_{j=k-w_\delta-a_\delta}^k x_j^2 \leq 1 \quad \forall k = w_\delta + b_\delta, \dots, W_\delta - w_\delta + 1 \quad (28)$$

$$x_j^1 \in \{0,1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (29)$$

$$x_j^2 \in \{0,1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1 \quad (30)$$

Si noti che il modello (27)-(30) è equivalente, per quanto riguarda il valore della soluzione trovata, al modello MM3 quando  $b_\delta = 0$ . Anche nel modello MM5, come in tutti i modelli precedenti, la matrice è TUM.

#### 4.5 MODELLI MATEMATICI PER IL FLOP3

Si consideri ora il problema *FLOP3*: esso richiede il posizionamento degli stand *a scacchi* (*checkerboard packing*), ovvero gli stand devono essere allineati sia verticalmente che orizzontalmente.

Si prenda nuovamente in considerazione la matrice  $\phi$  definita nel paragrafo 4.3 e si crei una nuova matrice  $\rho$  contenente i profitti che verranno poi utilizzati per il modello del problema; lo scopo di quest'ultimo sarà quello di sapere se uno stand può essere piazzato con il suo angolo inferiore sinistro in una particolare posizione  $(i, j)$  per  $i=1, \dots, W_\delta - w_\delta + 1$  e  $j=1, \dots, H_\delta - h_\delta + 1$ .

Sono definiti i profitti  $r_{ij}$  come:

$$r_{ij} = \left\lfloor \frac{\sum_{h=i}^{i+w_\delta-1} \sum_{k=j}^{j+h_\delta-1} \phi_{kh}}{w_\delta h_\delta} \right\rfloor \quad (31)$$

Ogni profitto  $r_{ij}$  assume valore 1 se lo stand può essere piazzato nella posizione  $(i, j)$  e 0 altrimenti. Dopodichè il profitto può essere calcolato in un tempo pari a  $O(W_\delta H_\delta)$ . In Figura 4.10 è rappresentata la matrice  $\rho$  ottenuta dall'esempio di Figura 4.4.

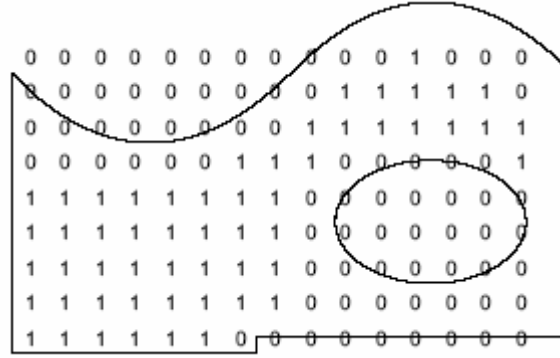


Figura 4.10: Matrice  $\rho$  per il calcolo dei profitti  $r_{ij}$  per  $i=1, \dots, W_\delta - w_\delta + 1$  e  $j=1, \dots, H_\delta - h_\delta + 1$ .

Si cercherà ora di adattare il modello MM1 del paragrafo 4.3.1 ad una situazione in cui viene richiesto un impaccamento di tipo *checkerboard*. Si supponga dunque (come in MM1) che non vi sia alcuna distanza tra due *strip* consecutive di stand.

Sono definite le seguenti variabili:

$$x_i = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta \text{ è piazzata in } i \\ 0 & \text{altrimenti} \end{cases} \quad (32)$$

per  $i=1, \dots, W_\delta - w_\delta + 1$

$$y_j = \begin{cases} 1 & \text{se una strip di altezza } h_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (33)$$

per  $j=1, \dots, H_\delta - h_\delta + 1$ .

Il modello del problema risulterebbe quindi in questo modo:

$$(MMC1) \quad \text{Max} \quad \sum_{i=1}^{W_\delta - w_\delta + 1} \sum_{j=1}^{H_\delta - h_\delta + 1} r_{ij} x_i y_j \quad (34)$$

$$\text{s.t.} \quad \sum_{i=k}^{k+w_\delta-1} x_i \leq 1 \quad \forall k = 1, \dots, W_\delta - 2w_\delta + 2 \quad (35)$$

$$\sum_{j=k}^{k+h_\delta-1} y_j \leq 1 \quad \forall k = 1, \dots, H_\delta - 2h_\delta + 2 \quad (36)$$

$$x_i \in \{0, 1\} \quad \forall i = 1, \dots, W_\delta - w_\delta + 1 \quad (37)$$

$$y_j \in \{0, 1\} \quad \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (38)$$

Il modello appena descritto richiede  $2(W_\delta - w_\delta + 1)$  variabili binarie e  $2(W_\delta - 2w_\delta + 2)$  disuguaglianze e risolve correttamente il problema. Tuttavia il problema ha una funzione obiettivo bilineare; essa può diventare lineare definendo la seguente variabile:

$$z_{ij} = \begin{cases} 1 & \text{se uno stand è piazzato in } (i, j) \\ 0 & \text{altrimenti} \end{cases} \quad (39)$$

per  $i=1, \dots, W_\delta - w_\delta + 1$  e  $j=1, \dots, H_\delta - h_\delta + 1$ .

La variabile  $z_{ij}$  assume valore 1 se  $x_i = y_j = 1$ . Il modello risultante è il seguente:

$$(MMC1') \quad \text{Max} \quad \sum_{i=1}^{W_\delta - w_\delta + 1} \sum_{j=1}^{H_\delta - h_\delta + 1} r_{ij} z_{ij} \quad (40)$$

$$\text{s.t.} \quad \sum_{j=1}^{H_\delta - h_\delta + 1} z_{ij} \leq \left\lfloor \frac{H_\delta}{h_\delta} \right\rfloor x_i \quad \forall i = 1, \dots, W_\delta - w_\delta + 1 \quad (41)$$

$$\sum_{i=1}^{W_\delta - w_\delta + 1} z_{ij} \leq \left\lfloor \frac{W_\delta}{w_\delta} \right\rfloor y_j \quad \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (42)$$

$$\sum_{i=k}^{k+w_\delta-1} x_i \leq 1 \quad \forall k = 1, \dots, W_\delta - 2w_\delta + 2 \quad (43)$$

$$\sum_{j=k}^{k+h_\delta-1} y_j \leq 1 \quad \forall k = 1, \dots, H_\delta - 2h_\delta + 2 \quad (44)$$

$$z_{ij} \in \{0, 1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1, \forall i = 1, \dots, H_\delta - h_\delta + 1 \quad (45)$$

$$x_i \in \{0, 1\} \quad \forall i = 1, \dots, W_\delta - w_\delta + 1 \quad (46)$$

$$y_j \in \{0, 1\} \quad \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (47)$$

Il modello (40)-(47) risolve correttamente il problema, anche se richiede un numero di variabili binarie superiore al modello precedente e pari a  $(H_\delta - h_\delta)(W_\delta - w_\delta)$ .

Il rilassamento lineare del modello MMC1' può essere ottenuto sostituendo i vincoli (41)-(42) con seguenti:

$$z_{ij} \leq x_i \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (48)$$

$$z_{ij} \leq y_j \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (49)$$

La differenza principale del modello *MMC1'* rispetto a tutti i modelli precedenti è che la sua matrice non è TUM.

#### 4.5.1 UN MODELLO GENERALE PER IL FLOP3

Come fatto per il problema FLOP2, anche in questo caso si vuole creare un modello che tenga in considerazione alcune particolari situazioni descritte nei paragrafi precedenti; tuttavia non ci si comporterà come precedentemente, ovvero non si adatterà il modello *MMC1'* ad ogni singola situazione, ma ci si riferirà ad un unico caso che sia generale ma anche interessante per applicazioni reali. Si supponga dunque che ogni stand debba essere adiacente a due accessi principali, uno lungo il lato parallelo alle  $x$  e uno lungo il lato parallelo alle  $y$ . Il modello del nuovo problema può essere creato combinando insieme i modelli MM3 e *MMC1'* precedentemente visti.

Si definiscono le variabili:

$$x_i^1 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + a_\delta \text{ è piazzata in } i \\ 0 & \text{altrimenti} \end{cases} \quad (50)$$

$$\text{per } i=1, \dots, W_\delta - w_\delta + 1$$

$$x_i^2 = \begin{cases} 1 & \text{se una strip di larghezza } w_\delta + w_\delta + a_\delta \text{ è piazzata in } i \\ 0 & \text{altrimenti} \end{cases} \quad (51)$$

$$\text{per } i=1, \dots, W_\delta - 2w_\delta + 2$$

$$y_j^1 = \begin{cases} 1 & \text{se una strip di altezza } h_\delta + c_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (52)$$

$$\text{per } j=1, \dots, H_\delta - h_\delta + 1$$

$$y_j^2 = \begin{cases} 1 & \text{se una strip di altezza } h_\delta + h_\delta + c_\delta \text{ è piazzata in } j \\ 0 & \text{altrimenti} \end{cases} \quad (53)$$

$$\text{per } j=1, \dots, H_\delta - 2h_\delta + 2$$

Il modello ( $MMC2'$ ) risulta essere il seguente:

$$\text{Max} \sum_{i=1}^{W_\delta - w_\delta + 1} \sum_{j=1}^{H_\delta - h_\delta + 1} r_{ij} z_{ij} \quad (54)$$

$$\text{s.t.} \sum_{i=k-w_\delta-a_\delta+1}^k x_i^1 + \sum_{i=k-2w_\delta-a_\delta+1}^k x_i^2 \leq 1 \quad \forall k = w_\delta + a_\delta, \dots, W_\delta - w_\delta + 1 \quad (55)$$

$$\sum_{j=k-h_\delta-c_\delta+1}^k y_j^1 + \sum_{j=k-2h_\delta-c_\delta+1}^k y_j^2 \leq 1 \quad \forall k = h_\delta + c_\delta, \dots, H_\delta - h_\delta + 1 \quad (56)$$

$$z_{ij} - x_i^1 - x_i^2 - x_{i-w_\delta}^2 \leq 0 \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (57)$$

$$z_{ij} - y_j^1 - y_j^2 - y_{j-h_\delta}^2 \leq 0 \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (58)$$

$$z_{ij} \in \{0,1\} \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (59)$$

$$x_i \in \{0,1\} \quad \forall i = 1, \dots, W_\delta - w_\delta + 1 \quad (60)$$

$$y_j \in \{0,1\} \quad \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (61)$$

In Figura 4.11 è rappresentato un esempio di packing che può essere prodotto dal modello  $MMC2'$  appena descritto.

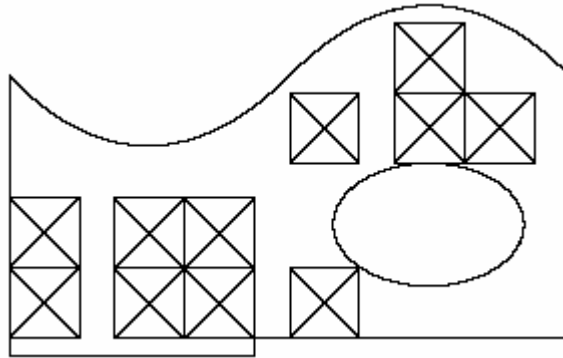


Figura 4.11: Un packing di tipo checkerboard ottenuto dal modello  $MMC2'$ .

Si noti che, come nel caso precedente, anche questo modello non ha una matrice totalmente unimodulare.

**Bibliografia:**

- R. Baldacci, M. Dell'Amico  
**Fondamenti di ricerca operativa**  
Pitagora Editrice Bologna (2002)
- C.H. Papadimitriou e K. Steiglitz  
**Combinatorial Optimization: Algorithms and Complexity**  
Prentice Hall, Englewood Cliffs, N.J., 1988
- A.E. Fernandez Muritiba, M. Iori e M. Negreiros  
**Models and Algorithms for Optimizing Fair Lay – Outs**  
*Rapporto Tecnico* OR/06/8, DEIS, Università di Bologna (2006).

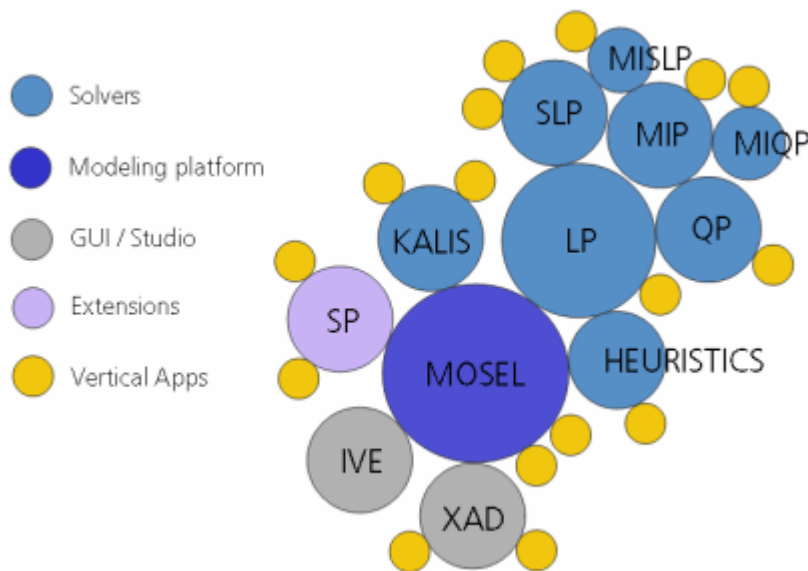
## Capitolo 5

### IMPLEMENTAZIONE DEI MODELLI

**P**er completare il lavoro svolto finora, si è deciso di implementare alcuni modelli descritti nel paragrafo precedente; a questo proposito, è stato usato il software di ottimizzazione Xpress-MP della Dash.

#### 5.1 IL SOFTWARE XPRESS-MP

Il software Xpress-MP è costituito da una serie di strumenti per la modellizzazione matematica e l'ottimizzazione, usati per risolvere problemi di programmazione lineari, interi, quadratici, non lineari, stocastici. Esso è disponibile su tutte le comuni piattaforme dei computer e in diverse *capacità* per permettere la risoluzione di problemi di varie dimensioni, supporta numerose interfacce incluse diverse librerie APIs accessibili in C, C++, Java, VB, .NET e da riga di comando. Nella figura seguente sono rappresentati i diversi prodotti Xpress-Mp, utili in diversi campi applicativi:



*Figura 5.1: Prodotti di Xpress-MP*

Per mancanza di inerenza con l'argomento sviluppato nella tesi, non si spiegherà il campo di utilizzo di ogni prodotto rappresentato, ma ci si soffermerà solamente a quello utilizzato per l'implementazione, ovvero Xpress-Mosel.



## 5.2 XPRESS-MOSEL

Xpress-Mosel è uno strumento che aiuta a formulare il problema, risolverlo attraverso un motore di risoluzione e analizzare la soluzione trovata: tutto ciò facendo uso di un linguaggio di programmazione specifico e multi funzionale progettato proprio a questo scopo. I programmi Mosel vengono compilati; ciò li rende più veloci e nasconde le proprietà intellettuali presenti al loro interno agli utilizzatori finali. Essi inoltre possono essere eseguiti interattivamente o eseguiti all'interno di una applicazione. Mosel è completamente aperto a tutte le estensioni da parte degli utenti; tale architettura aperta e modulare permettere di non restringere l'ambiente di sviluppo solamente a particolari tipi di problemi. La distribuzione Mosel include inoltre l'estensione di librerie (chiamate *moduli*), ognuna delle quali provvede al diretto controllo di Xpress-Optimizer.

### 5.2.1 IL LINGUAGGIO MOSEL

Il linguaggio Mosel è facile da imparare e da utilizzare e riduce i costi che sorgono durante lo sviluppo di modelli di ottimizzazione nati da bozze e durante la comprensione di modelli creati da altre persone. Esso permette inoltre di modificare i modelli e di mantenerli nel tempo. L'interfaccia grafica Xpress-IVE rende il processo ancora più facile.

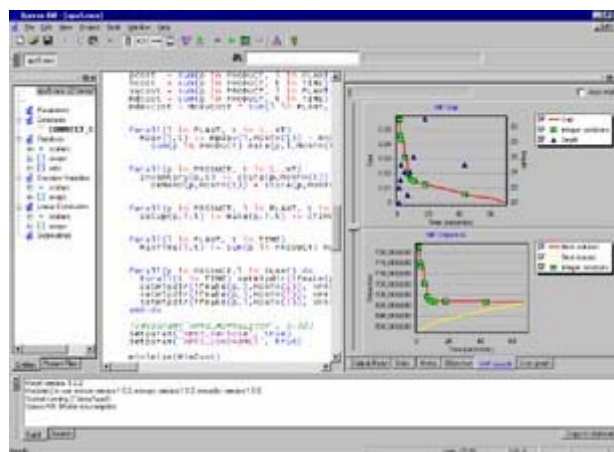


Figura 5.2: L'interfaccia Xpress-IVE: essa comprende un editor Mosel, un compilatore e un esecutore.

Un supporto completo per ranges arbitrari, insiemi di indici e oggetti irregolari consente di esprimere in modo chiaro e conciso problemi molto ampi e sofisticati. Lo sviluppo dei modelli è inoltre aiutato da un Mosel *debugger* che supporta la tipica funzione di *debug* usata per analizzare l'esecuzione del modello.

### 5.3 I MODELLI IN LINGUAGGIO MOSEL

Come detto in precedenza, si è deciso di non implementare tutti i modelli descritti nel capitolo precedente, ma solo i più significativi e quelli per i quali l'implementazione può essere utile per una situazione reale. Di seguito sono riportati alcuni modelli matematici riscritti in linguaggio Mosel, completi delle relative note (si riconoscono perché precedute da un punto esclamativo) utili per capire la funzione di ogni parte del programma.

#### 5.3.1 MODELLO 1

Il modello 1 è l'implementazione del modello matematico per la risoluzione del FLOP1, indicato con la sigla MM1 e spiegato nel paragrafo 4.3.1.

```

model modello1
uses "mmxprs";

declarations
W = 16 ! larghezza della superficie di esposizione
w = 2 ! larghezza degli stand
H = 10 ! altezza della superficie di esposizione
h = 3 ! altezza degli stand
WIDTH = 1..W
HEIGHT = 1..H
widthr = 1..(W-w+1)
heightr = 1..(H-h+1)
A: array ( WIDTH, HEIGHT ) of integer
A0: array ( HEIGHT, WIDTH) of integer
B: array ( widthr, HEIGHT ) of integer
P: array ( widthr ) of integer
VAR: array (widthr) of mpvar
end-declarations

initializations from 'tabella.dat'
A0
end-initializations

writeln("Begin running model")

! lettura della matrice data in input e trasformazione
! della stessa in una matrice A su cui poter lavorare
forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  
```

```

end-do
end-do

! creazione della matrice B
forall ( x in widthr ) do
  forall ( y in HEIGHT) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( i in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(i,j)
        end-do
      end-do
      if (COUNT=w*h) then B(x,y):=1
        else B(x,y):=0
      end-if
    else B(x,y):=0
    end-if
  end-do
end-do

! creazione degli array dei profitti
forall (x in widthr) do
  P(x):=0
  forall (y in heightr) do
    if ( B(x,y)=1 ) then P(x):=P(x)+1
    forall ( j in (y+1)..(y+h-1) )do
      if (j<H) then B(x,j):=0
    end-if
  end-do
end-if
end-do
end-do

! vincolo di linearità della variabile: vincolo possibile
! grazie alla presenza di una matrice TUM
forall (x in widthr) VAR(x)<= 1

! funzione obiettivo
z:= sum(x in widthr) (P(x)* VAR(x))

! primo e unico vincolo del modello MMI, assicura
! la non sovrapposizione degli stand
forall (k in 1..(W-w+1-w+1)) do
  v1(k):= sum(x in k..(k+w-1))(VAR(x))<= 1
end-do

maximize(z)

```

```

! scrittura a video del valore della funzione obiettivo
writeln("z=", getobjval)

! scrittura a video dei valori assunti dalla variabile
forall (i in widthr) do
  writeln("x(", i, ")=", getsol(VAR(i)))
end-do

! scrittura a video della posizione in cui piazzare tutti
! gli stand possibili
forall (x in widthr) do
  if (getsol(VAR(x))=1) then
    forall (y in heightr) do
      if (B(x,y)= 1) then
        writeln ("C'è uno stand in (", x, ", ", y, ")")
      end-if
    end-do
  end-if
end-do

writeln("End running model")

end-model

```

Il Modello 1 è il modello più semplice tra tutti quelli implementati, e permette di definire alcuni elementi che saranno presenti anche nei modelli successivi:

- La matrice A0 è presa da input (in particolare deve essere presa da un file .dat) ed è la matrice che è stata creata dall'utente: essa presenta degli uni nelle posizioni in cui è possibile posizionare uno stand, e degli zeri nelle posizioni rimanenti. Essa è l'analogo della matrice  $\phi$  spiegata nel paragrafo 4.2.
- La matrice A è la matrice su cui il programma Mosel andrà a lavorare ed è ottenuta trasformando la matrice A0; in particolare scambiando le righe con le colonne ovvero, detto in altre parole, effettuando su A0 una rotazione di 90° in senso orario. Ciò è necessario perché il programma legge la matrice riga per riga: in questo modo la matrice data in input sarebbe memorizzata seguendo un verso di lettura errato.
- La matrice B è sempre una matrice di uni e zeri: essa viene creata mettendo un uno nella posizione in cui è possibile mettere con l'angolo inferiore sinistro uno stand.
- L'array P è creato per ogni colonna e dice il profitto di ciascuna di esse.

Spiegato ciò è possibile soffermarsi sui passaggi caratterizzanti il Modello1:

- La funzione obiettivo del modello sceglie le colonne che permettono di ottenere il maggior profitto.
- Il vincolo  $v1$  è la trascrizione del vincolo

$$\sum_{j=k}^{k-w_\delta+1} x_j \leq 1 \quad \forall k = 1, \dots, W_\delta - w_\delta + 1$$

e garantisce la non sovrapposizione degli stand: per ogni colonna  $x_j$  controlla che una *strip* di stand non sia posizionata su di un'altra.

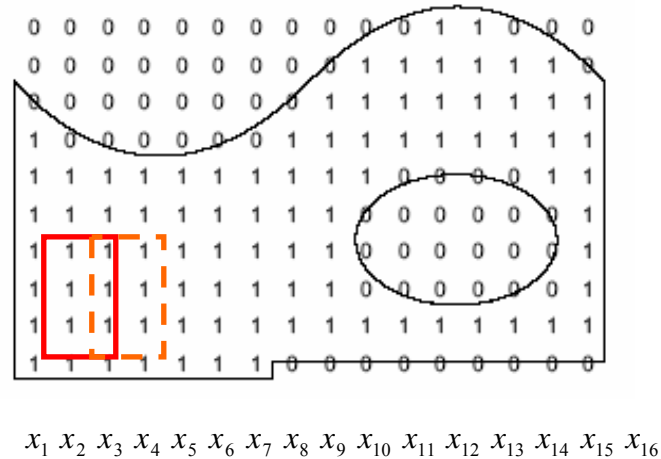


Figura 5.3: Esempio grafico del vincolo  $v1$ . Se lo stand è  $[2 \times 3]$  e viene scelta la strip  $x_2$ , non può essere scelta la strip  $x_3$  perché gli stand si sovrapporrebbero (in figura lo stand disegnato in rosso si sovrapporrebbe allo stand tratteggiato). Da qui il vincolo  $x_2 + x_3 \leq 1$ .

- La variabile non è più binaria come nel modello MM1, ma è lineare:

$$\text{forall } (x \text{ in widthr}) \text{ VAR}(x) \leq 1.$$

Ciò è stato possibile poiché si è in presenza di una matrice totalmente unimodulare (vedi paragrafo 4.3.2)

### 5.3.2 MODELLO 2

Il modello 2 è l'implementazione di un modello matematico per la risoluzione del FLOP2, in particolare del modello indicato con la sigla MM2 e spiegato nel paragrafo 4.4. Esso tiene conto di una distanza minima  $a$  necessaria tra due *strip* di stand.

```

model modello2
uses "mmxprs";

declarations
  W = 16
  w = 2
  H = 10
  h = 3
  a = 1 ! distanza necessaria tra due stand
  WIDTH = 1..W
  HEIGHT = 1..H
  widthr = 1..(W-w+1)
  heightr = 1..(H-h+1)
  A: array ( WIDTH, HEIGHT ) of integer
  A0: array ( HEIGHT, WIDTH) of integer
  B: array ( widthr, HEIGHT ) of integer
  P: array ( widthr ) of integer
  VAR: array (widthr) of mpvar
end-declarations

initializations from 'tabella.dat'
  A0
end-initializations

writeln("Begin running model")

forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  end-do
end-do

forall ( x in widthr ) do
  forall ( y in HEIGHT ) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( i in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(i,j)
        end-do
      end-do
    end-do
  end-do
end-do

```

```

        end-do
        if (COUNT=w*h) then B(x,y):=1
            else B(x,y):=0
        end-if
        else B(x,y):=0
    end-if
end-do
end-do

forall (x in widthr) do
    P(x):=0
    forall (y in heightr) do
        if ( B(x,y)=1 ) then P(x):=P(x)+1
            forall ( j in (y+1)..(y+h-1) )do
                if (j<H) then B(x,j):=0
            end-if
        end-do
    end-if
end-do
end-do

forall (x in widthr) VAR(x)<= 1

z:= sum(x in widthr) (P(x)* VAR(x))

! primo e unico vincolo del modello MM2, assicura
! la non sovrapposizione degli stand e il rispetto
! della distanza a
forall (k in 1..(W-w+1-w+1-a)) do
    v1(k):= sum(x in k..(k+w+a-1))(VAR(x))<= 1
end-do

maximize(z)

writeln("z=", getobjval)

forall (i in widthr) do
    writeln("x(",i,")=", getsol(VAR(i)))
end-do

forall (x in widthr) do
    if (getsol(VAR(x))=1) then
        forall (y in heightr) do
            if (B(x,y)= 1) then
                writeln ("C'è uno stand in (", x,",",y,")")
            end-if
        end-do
    end-if
end-do
end-do

```

```
writeln("End running model")

end-model
```

Per quanto riguarda il modello 2, vale tutto ciò che si è detto sul modello 1, tranne per un vincolo:

- Il vincolo  $vI$  del modello 1 segue la stessa logica del rispettivo vincolo del modello 2, ma tiene anche in considerazione una distanza  $a$  che deve essere lasciata tra il posizionamento di uno stand e il successivo:

$$\sum_{j=k}^{k+w_\delta+a_\delta-1} x_j \leq 1 \quad \forall k = 1, \dots, W_\delta - 2w_\delta - a_\delta + 2$$

Posizionato uno stand, non bisogna lasciare solamente una distanza pari alla sua larghezza prima di posizionarne un altro, ma una distanza pari a  $w_\delta + a_\delta - 1$ .

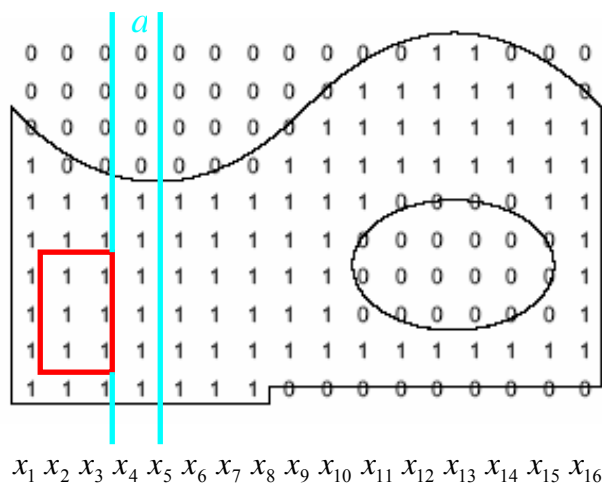


Figura 5.4: Esempio grafico del vincolo  $vI$ . Se lo stand è  $[2 \times 3]$ , la distanza  $a$  da lasciare tra una strip e l'altra è 1 e viene scelta la strip  $x_2$ , non può essere scelta né la strip  $x_3$  né la strip  $x_4$  (in figura lo stand è disegnato in rosso mentre la distanza  $a$  in azzurro).

$$Da \text{ qui il vincolo } x_2 + x_3 + x_4 \leq 1.$$



### 5.3.3 MODELLO 3

Il modello 3 è l'implementazione di un altro modello matematico per la risoluzione del FLOP2, in particolare del modello indicato con la sigla MM3 e spiegato nel paragrafo 4.4.1. Esso permette la presenza non solo di *single strip*, ma anche di *double strip*. E' inoltre sempre presa in considerazione la distanza  $a$ .

```

model modello3
uses "mmxprs";

declarations
  W = 16
  w = 2
  H = 10
  h = 3
  a = 1
  WIDTH = 1..W
  HEIGHT = 1..H
  widthr = 1..(W-w+1)
  widthr2 = 1..(W-2*w+1)
  heightr = 1..(H-h+1)
  A: array ( WIDTH, HEIGHT ) of integer
  A0: array (HEIGHT, WIDTH) of integer
  B: array ( widthr, HEIGHT ) of integer
  P1: array ( widthr ) of integer
  P2: array (widthr) of integer
  VAR1: array (widthr) of mpvar
  VAR2: array (widthr2) of mpvar
  start1:integer
  start2:integer
  end1:integer
  end2:integer
end-declarations

initializations from 'tabella.dat'
  A0
end-initializations

writeln("Begin running model")

forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  end-do
end-do

```

```

forall ( x in widthr ) do
  forall ( y in HEIGHT) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( I in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(I,j)
        end-do
      end-do
      if (COUNT=w*h) then B(x,y):=1
        else B(x,y):=0
      end-if
    else B(x,y):=0
    end-if
  end-do
end-do

forall (x in widthr) do
  P1(x):=0
  forall (y in heightr) do
    if ( B(x,y)=1 ) then P1(x):=P1(x)+1
      forall ( j in (y+1)..(y+h-1) )do
        if (j<H) then B(x,j):=0
        end-if
      end-do
    end-if
  end-do
end-do

! creazione dell'array che indica i profitti delle double strip
forall (x in 1..W-2*w+1) do
  P2(x):=P1(x)+P1(x+w)
end-do

forall (x in widthr) VAR1(x)<= 1
forall (x in widthr2) VAR2(x)<= 1

z:= (sum(x in widthr) (P1(x)* VAR1(x)))+
    (sum(x in 1..(W-2*w+1)) (P2(x)* VAR2(x)))

! primo e unico vincolo del modello MM3, assicura
! la non sovrapposizione delle strip single e double
! e il rispetto della distanza a tra due strip
forall (k in (w+a)..(W-w+1)) do

  if (k-w-a+1>1) then start1:=k-w-a+1
    else start1:=1
  end-if

```

```

if (k-2*w-a+1>1) then start2:=k-2*w-a+1
    else start2:=1
end-if
if (W-w+1<k) then end1:=W-w+1
    else end1:=k
end-if
if (W-2*w+1<k) then end2:=W-2*w+1
    else end2:=k
end-if

v1(k):= (sum(x in start1..end1)(VAR1(x)))+
        (sum(x in start2..end2)(VAR2(x))) <= 1
end-do

maximize(z)

writeln("z=", getobjval)

forall (I in widthr) do
    writeln("x1(",I,")=", getsol(VAR1(i)))
end-do

forall (I in widthr2) do
    writeln("x2(",I,")=", getsol(VAR2(i)))
end-do

forall (x in widthr) do
    if ((getsol(VAR1(x))=1)) then
        forall (y in heightr) do
            if (B(x,y)= 1) then
                writeln ("Lo stand appartenente alla single strip è in (" , x," ",y,")")
            end-if
        end-do
    end-if
end-do

forall (x in widthr2) do
    if ((getsol(VAR2(x))=1)) then
        forall (y in heightr) do
            if (B(x,y)= 1) then
                writeln ("Lo stand appartenente alla double strip è in (" , x," ",y,")")
            end-if
        end-do
    end-if
end-do

writeln("End running model")

end-model

```

La prima cosa che si nota nel modello 3 è che esso presenta due variabili e due diversi array di profitti:

- VAR1 assume valore 1 se è scelta una *single strip* di stand
- VAR2 assume valore 1 se è scelta una *double strip* di stand
- L'array P1 conta il profitto delle *single strip*
- L'array P2 conta il profitto delle *double strip*

La funzione obiettivo quindi non può più essere come nei modelli precedenti, ma presenta due sommatorie:

$$\sum_{j=1}^{W_\delta - w_\delta + 1} p_j^1 x_j^1 + \sum_{j=1}^{W_\delta - 2w_\delta + 1} p_j^2 x_j^2$$

La logica di funzionamento del vincolo  $v1$  è in linea con i vincoli precedenti; l'unica cosa che anche in questo caso si nota è che sono presenti due sommatorie anziché una. Nel modello MM3 il vincolo compariva infatti in questo modo:

$$\sum_{j=k-w_\delta-a_\delta}^k x_j^1 + \sum_{j=k-2w_\delta-a_\delta}^k x_j^2 \leq 1 \quad \forall k = w_\delta + a_\delta, \dots, W_\delta - w_\delta + 1$$

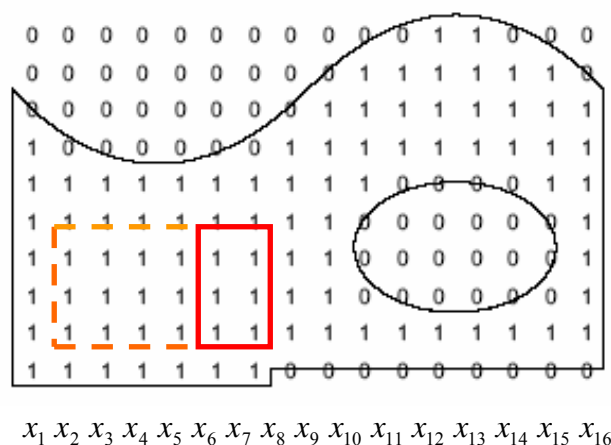


Figura 5.5: Esempio grafico del vincolo  $v1$ . Se lo stand è  $[2 \times 3]$ , la distanza  $a$  da lasciare tra una strip e l'altra è 1 e, per esempio, venisse scelta la strip  $x_6$ , non può essere scelta nessuna single strip in  $x_4$  e  $x_5$  e nessuna double strip in  $x_2, x_3, x_4, x_5$  e  $x_6$  (in figura lo stand è disegnato in rosso mentre la linea tratteggiata rappresenta il confine entro il quale non si può piazzare una double strip di stand).

Da qui il vincolo  $x_4^1 + x_5^1 + x_6^1 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \leq 1$ .

### 5.3.4 MODELLO 4

Il modello 4 è l'implementazione del modello matematico per la risoluzione del FLOP3, in particolare del modello indicato con la sigla MMC1' e spiegato nel paragrafo 4.5. Il modello 4 è stato implementato in diverse "versioni" per confrontare i risultati ottenuti. E' quindi utile riportare la formulazione matematica :

$$(MMC1') \quad \text{Max} \quad \sum_{i=1}^{W_\delta - w_\delta + 1} \sum_{j=1}^{H_\delta - h_\delta + 1} r_{ij} z_{ij} \quad (40)$$

$$s.t. \quad \sum_{j=1}^{H_\delta - h_\delta + 1} z_{ij} \leq \left\lfloor \frac{H_\delta}{h_\delta} \right\rfloor x_i \quad \forall i = 1, \dots, W_\delta - w_\delta + 1 \quad (41)$$

$$\sum_{i=1}^{W_\delta - w_\delta + 1} z_{ij} \leq \left\lfloor \frac{W_\delta}{w_\delta} \right\rfloor y_j \quad \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (42)$$

$$\sum_{i=k}^{k+w_\delta-1} x_i \leq 1 \quad \forall k = 1, \dots, W_\delta - 2w_\delta + 2 \quad (43)$$

$$\sum_{j=k}^{k+h_\delta-1} y_j \leq 1 \quad \forall k = 1, \dots, H_\delta - 2h_\delta + 2 \quad (44)$$

$$z_{ij} \in \{0, 1\} \quad \forall j = 1, \dots, W_\delta - w_\delta + 1, \forall i = 1, \dots, H_\delta - h_\delta + 1 \quad (45)$$

$$x_i \in \{0, 1\} \quad \forall i = 1, \dots, W_\delta - w_\delta + 1 \quad (46)$$

$$y_j \in \{0, 1\} \quad \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (47)$$

Dove :

$$z_{ij} = \begin{cases} 1 & \text{se uno stand è piazzato in } (i, j) \\ 0 & \text{altrimenti} \end{cases}$$

per  $i=1, \dots, W_\delta - w_\delta + 1$  e  $j=1, \dots, H_\delta - h_\delta + 1$ .

Si era inoltre detto che il modello (40)-(47) risolve correttamente il problema, anche se richiede un numero di variabili binarie superiore al modello precedente e pari a  $(H_\delta - h_\delta)(W_\delta - w_\delta)$ , e che il rilassamento lineare del modello MMC1' può essere ottenuto sostituendo i vincoli (41)-(42) con seguenti:

$$z_{ij} \leq x_i \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (48)$$

$$z_{ij} \leq y_j \quad \forall i = 1, \dots, W_\delta - w_\delta + 1, \forall j = 1, \dots, H_\delta - h_\delta + 1 \quad (49)$$

Rivisto il modello, è possibile riportare le diverse versioni di implementazione:

I. Modello 4\_1

Implementazione del modello (40)-(47)

```

model modello4_1
uses "mmxprs";

declarations
  W = 16
  w = 2
  H = 10
  h = 3
  a = 1
  WIDTH = 1..W
  HEIGHT = 1..H
  widthr = 1..(W-w+1)
  heightr = 1..(H-h+1)
  A: array (WIDTH, HEIGHT) of integer
  A0: array (HEIGHT, WIDTH) of integer
  R: array (widthr, heightr) of integer
  VARX1: array (widthr) of mpvar
  VARY1: array (heightr) of mpvar
  VARZ: array (widthr, heightr) of mpvar
  DIVH:integer
  DIVW:integer
end-declarations

initializations from 'tabella.dat'
  A0
end-initializations

writeln("Begin running model")

forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  end-do
end-do

forall ( x in widthr ) do
  forall ( y in heightr ) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( i in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(i,j)
        end-do
      end-do
    end-do
  end-do
end-do

```

```

        if (COUNT=w*h) then R(x,y):=1
            else R(x,y):=0
        end-if
    else R(x,y):=0
    end-if
end-do
end-do

! il modello è stato implementato sia in versione PLI, ovvero con
! tutte le variabili binarie, sia in versione PL
! versione PL:
! forall (x in widthr) VARX1(x) <=1
! forall (x in heightr) VARY1(x) <=1
! forall (x in heightr, y in widthr) VARZ(x,y) <=1
! versione PLI:
forall (x in widthr) VARX1(x) is_binary
forall (y in heightr) VARY1(y) is_binary
forall (x in widthr, y in heightr) VARZ(x,y) is_binary

z:= (sum(x in widthr, y in heightr) (R(x,y)* VARZ(x,y)))

! la funzione floor restituisce l'estremo inferiore
DIVH := floor(H/h)
DIVW := floor(W/w)

! I vincoli v1 e v2 assicurano la non sovrapposizione degli stand
forall (k in 1..(W-2*w+2)) do
    v1(k):= (sum(x in k..(k+w-1))(VARX1(x))) <= 1
end-do

forall (k in 1..(H-2*h+2)) do
    v2(k):= (sum(y in k..(k+h-1))(VARY1(y))) <= 1
end-do

! I vincoli v3 e v4 impongono che non si possono posizionare più stand
! di quanti ve ne possono stare nella matrice
forall (x in widthr) do
    v3(x):= (sum(y in 1..(H-h+1))(VARZ(x,y))) <= DIVH*VARX1(x)
end-do

forall (y in heightr) do
    v4(x):= (sum(x in 1..(W-w+1))(VARZ(x,y))) <= DIVW*VARY1(y)
end-do

maximize(z)

writeln("z=", getobjval)

forall (i in widthr) do
    writeln("x1(",i,")=", getsol(VARX1(i)))

```

```

end-do

forall (j in heightr) do
  writeln("y1(",j,")=", getsol(VARY1(j)))
end-do

forall (i in widthr) do
  forall (j in heightr) do
    writeln("z(",i,j,")=", getsol(VARZ(i,j)))
  end-do
end-do

forall (x in widthr r) do
  forall (y in heightr) do
    if ((getsol(VARZ(x,y))=1))and ((R(x,y)= 1)) then
      writeln ("C'è uno stand in (", x," ",y,"")
    end-if
  end-do
end-do

writeln("End running model")

end-model

```

Come si può notare, nel modello 4 sono presenti tre variabili (VARX, VARY, VARZ); questo perché d'ora in poi non si ragionerà più per *strip*, ma per *posizioni*:

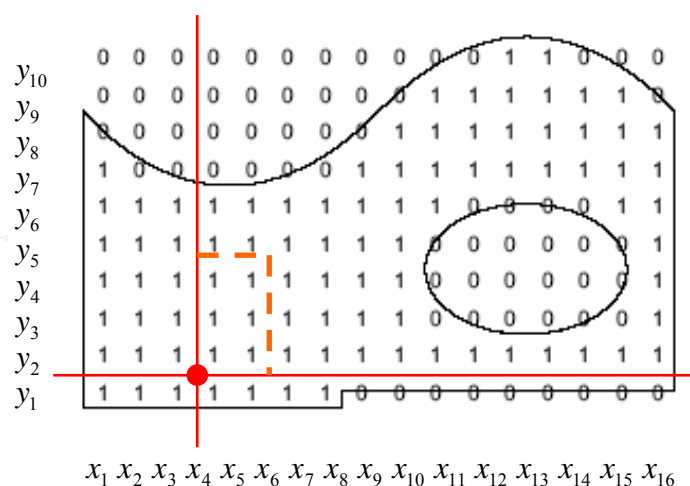


Figura 5.6: Schema rappresentate la nuova logica di posizionamento; gli stand non sono più distribuiti a *strip*, ma per *posizioni*. Non vi è più un'unica variabile  $x$ , ma anche una variabile  $y$  e una variabile  $z$ .

Il puntino rosso indica la variabile  $z$ .



- I vincoli  $v1$  e  $v2$  sono i vincoli che assicurano la non sovrapposizione degli stand. Da notare il fatto che la non sovrapposizione debba essere espressa tramite due equazione e non tramite una sola come nei modelli precedenti; un vincolo per la variabile  $x$  e uno per la variabile  $y$ .
- I vincoli  $v3$  e  $v4$  controllano che non si posizionino più stand di quanti ne possano stare; tale numero è dato sulla larghezza della superficie di esposizione dal rapporto  $\left\lfloor \frac{W_\delta}{w_\delta} \right\rfloor$ , mentre sull'altezza dal rapporto  $\left\lfloor \frac{H_\delta}{h_\delta} \right\rfloor$ .

## II. Modello 4\_2

Implementazione del modello (40), (43)-(47), (48)-(49)

```

model modello4_2
uses "mmxprs";

declarations
W = 16
w = 2
H = 10
h = 3
a = 1
WIDTH = 1..W
HEIGHT = 1..H
widthr = 1..(W-w+1)
heightr = 1..(H-h+1)
A: array (WIDTH, HEIGHT) of integer
A0: array (HEIGHT, WIDTH) of integer
R: array (widthr, heightr) of integer
VARX1: array (widthr) of mpvar
VARY1: array (heightr) of mpvar
VARZ: array (widthr, heightr) of mpvar

end-declarations

initializations from 'tabella.dat'
A0
end-initializations

writeln("Begin running model")

```

```
forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  end-do
end-do
```

```
forall ( x in widthr ) do
  forall ( y in heightr) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( i in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(i,j)
        end-do
      end-do
      if (COUNT=w*h) then R(x,y):=1
        else R(x,y):=0
      end-if
    else R(x,y):=0
    end-if
  end-do
end-do
```

*! Anche in questo modello si è provata sia la versione seguente  
! con tutte le variabili binarie, sia la versione lineare*

```
forall (x in widthr) VARX1(x) is_binary
forall (y in heightr) VARY1(y) is_binary
forall (x in widthr, y in heightr) VARZ(x,y) is_binary
```

```
z:= (sum(x in widthr,y in heightr) (R(x,y)* VARZ(x,y)))
```

*! I vincoli v1 e v2 assicurano la non sovrapposizione degli stand*

```
forall (k in 1..(W-2*w+2)) do
  v1(k):= (sum(x in k..(k+w-1))(VARX1(x))) <= 1
end-do
```

```
forall (k in 1..(H-2*h+2)) do
  v2(k):= (sum(y in k..(k+h-1))(VARY1(y))) <= 1
end-do
```

*! Il vincolo v5 impone che la variabile VARZ non può assumere  
! valore 1 se la variabile VARY1 è 0*

```
forall (x in widthr) do
  forall (y in heightr) do
    v5(x,y):= VARZ(x,y) <= VARY1(y)
  end-do
end-do
```

```

! Il vincolo v5 impone che la variabile VARZ non può assumere
! valore 1 se la variabile VARX1 è 0
forall (x in widthr) do
  forall (y in heightr) do
    v6(x,y):= VARZ(x,y) <= VARX1(x)
  end-do
end-do

maximize(z)

writeln("z=", getobjval)

forall (i in widthr) do
  writeln("x1(",i,"")=", getsol(VARX1(i)))
end-do

forall (j in heightr) do
  writeln("y1(",j,"")=", getsol(VARY1(j)))
end-do

forall (i in widthr) do
  forall (j in heightr) do
    writeln("z(",i,j,"")=", getsol(VARZ(i,j)))
  end-do
end-do

forall (x in widthr) do
  forall (y in heightr) do
    if ((getsol(VARZ(x,y))=1))and (R(x,y)= 1) then
      writeln ("C'è uno stand in (", x,"","y,"")")
    end-if
  end-do
end-do

writeln("End running model")

end-model

```

Il modello 4\_2 presenta due nuovi vincoli: il vincolo v5 e il vincolo v6. Questi due vincoli servono a imporre le seguenti relazioni tra variabili:

$x$	$y$		$z$
0	0	$\rightarrow$	0
0	1	$\rightarrow$	0
1	0	$\rightarrow$	0
1	1	$\rightarrow$	1

### III. Modello 4\_3

Implementazione del modello (40)-(44), (46)-(48), con aggiunta del vincolo di linearità della variabile  $z$ , ovvero:

$$0 \leq z_{ij} \leq 1 \quad \forall j = 1, \dots, W_{\delta} - w_{\delta} + 1, \forall i = 1, \dots, H_{\delta} - h_{\delta} + 1 \quad (45.2)$$

```

model modello4_3
uses "mmxprs";

declarations
  W = 16
  w = 2
  H = 10
  h = 3
  a = 1
  WIDTH = 1..W
  HEIGHT = 1..H
  widthr = 1..(W-w+1)
  heightr = 1..(H-h+1)
  A: array (WIDTH, HEIGHT) of integer
  A0: array (HEIGHT, WIDTH) of integer
  R: array (widthr, heightr) of integer
  VARX1: array (widthr) of mpvar
  VARY1: array (heightr) of mpvar
  VARZ: array (widthr, heightr) of mpvar
  DIVH: integer
  DIVW: integer

end-declarations

initializations from 'tabella.dat'
  A0
end-initializations

writeln("Begin running model")

forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1) := A0 (x, y)
  end-do
end-do

forall ( x in widthr ) do
  forall ( y in heightr ) do
    if ((y <= H-h+1) and (x <= W-w+1)) then

```

```

COUNT:=0
forall ( i in x..(x+w-1) ) do
  forall ( j in y..(y+h-1) ) do
    COUNT:=COUNT + A(i,j)
  end-do
end-do
if (COUNT=w*h) then R(x,y):=1
  else R(x,y):=0
end-if
else R(x,y):=0
end-if
end-do
end-do

forall (x in widthr) VARX1(x) is_binary
forall (y in heightr) VARY1(y) is_binary
forall (x in widthr, y in heightr) VARZ(x,y)<=1

z:= (sum(x in widthr,y in heightr) (R(x,y)* VARZ(x,y)))

DIVH := floor(H/h)
DIVW := floor(W/w)

forall (k in 1..(W-2*w+2)) do
  v1(k):= (sum(x in k..(k+w-1))(VARX1(x))) <= 1
end-do

forall (k in 1..(H-2*h+2)) do
  v2(k):= (sum(y in k..(k+h-1))(VARY1(y))) <= 1
end-do

forall (x in widthr) do
  v3(x):= (sum(y in 1..(H-h+1))(VARZ(x,y))) <= DIVH*VARX1(x)
end-do

forall (y in heightr) do
  v4(x):= (sum(x in 1..(W-w+1))(VARZ(x,y))) <= DIVW*VARY1(y)
end-do

maximize(z)

writeln("z=", getobjval)

forall (i in widthr) do
  writeln("x1(",i,"")=", getsol(VARX1(i)))
end-do

forall (j in heightr) do
  writeln("y1(",j,"")=", getsol(VARY1(j)))
end-do

```

```

forall (i in widthr) do
  forall (j in heightr) do
    writeln("z(",i,j,")=", getsol(VARZ(i,j)))
  end-do
end-do

forall (x in widthr) do
  forall (y in heightr) do
    if ((getsol(VARZ(x,y))=1))and (R(x,y)= 1) then
      writeln ("C'è uno stand in (", x, ",", y, ")")
    end-if
  end-do
end-do

writeln("End running model")

end-model

```

#### IV. Modello 4\_4

Implementazione del modello (40), (43)-(44), (46)-(47), (48)-(49) e (45.2).

```

model modello4_4
uses "mmxprs";

declarations
  W = 16
  w = 2
  H = 10
  h = 3
  a = 1
  WIDTH = 1..W
  HEIGHT = 1..H
  widthr = 1..(W-w+1)
  heightr = 1..(H-h+1)
  A: array (WIDTH, HEIGHT) of integer
  A0: array (HEIGHT, WIDTH) of integer
  R: array (widthr, heightr) of integer
  VARX1: array (widthr) of mpvar
  VARY1: array (heightr) of mpvar
  VARZ: array (widthr, heightr) of mpvar
end-declarations

```

```

initializations from 'tabella.dat'
A0
end-initializations

writeln("Begin running model")

forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  end-do
end-do

forall ( x in widthr ) do
  forall ( y in heightr) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( i in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(i,j)
        end-do
      end-do
      if (COUNT=w*h) then R(x,y):=1
        else R(x,y):=0
      end-if
    else R(x,y):=0
    end-if
  end-do
end-do

forall (x in widthr) VARX1(x) is_binary
forall (y in heightr) VARY1(y) is_binary
forall (x in widthr, y in heightr) VARZ(x,y)<= 1

z:=(sum(x in widthr, y in heightr) (R(x,y)* VARZ(x,y)))

forall (k in 1..(W-2*w+2)) do
  v1(k):=(sum(x in k..(k+w-1))(VARX1(x))) <= 1
end-do

forall (k in 1..(H-2*h+2)) do
  v2(k):=(sum(y in k..(k+h-1))(VARY1(y))) <= 1
end-do

forall (x in widthr) do
  forall (y in heightr) do
    v5(x,y):= VARZ(x,y) <= VARY1(y)
  end-do
end-do

```

```

forall (x in widthr) do
  forall (y in heightr) do
    v6(x,y):= VARZ(x,y) <= VARX1(x)
  end-do
end-do

maximize(z)

writeln("z=", getobjval)

forall (i in widthr) do
  writeln("x1(",i,")=" , getsol(VARX1(i)))
end-do

forall (j in heightr) do
  writeln("y1(",j,")=" , getsol(VARY1(j)))
end-do

forall (i in widthr) do
  forall (j in heightr) do
    writeln("z(",i,j,")=" , getsol(VARZ(i,j)))
  end-do
end-do

forall (x in widthr) do
  forall (y in heightr) do
    if ((getsol(VARZ(x,y))=1))and (R(x,y)= 1) then
      writeln ("C'è uno stand in (", x,"",y,"")
    end-if
  end-do
end-do

writeln("End running model")

end-model

```

### 5.3.5 MODELLO 5

Il modello 5 permette l'implementazione del modello matematico *MMC2'* descritto nel paragrafo 4.5.1. Esso è una formulazione che risolve il problema FLOP3 nel modo più generale possibile: sono ammesse sia *single strip* che *double strip* e si richiede di rispettare sia una distanza *a* sull'asse delle x che una distanza *c* sull'asse delle y tra uno stand e il successivo.



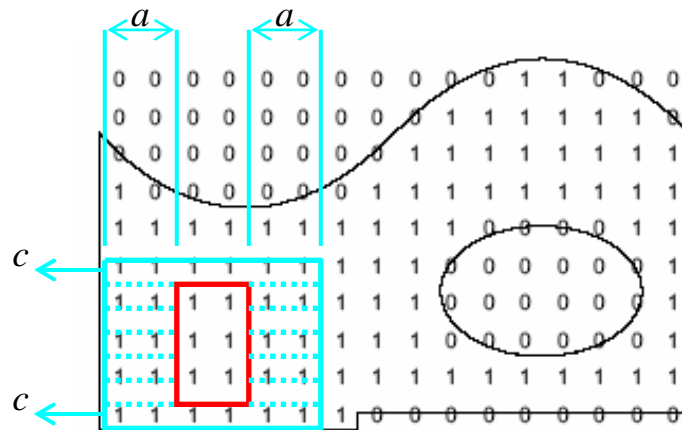


Figura 5.6: Esempio grafico delle distanze  $a$  e  $c$ . Se si sceglie lo stand  $[2 \times 3]$  rappresentato in rosso in figura, la parte tratteggiata in turchese rappresenta l'area entro cui non è possibile posizionare nessun altro stand.

```

model modello5
uses "mmxprs";

declarations
    W = 16
    w = 2
    H = 10
    h = 3
    a = 1
    c = 1
    WIDTH = 1..W
    HEIGHT = 1..H
    widthr = 1..(W-w+1)
    widthr2 = 1..(W-2*w+1)
    heightr = 1..(H-h+1)
    heightr2 = 1..(H-2*h+1)
    A: array (WIDTH, HEIGHT) of integer
    A0: array (HEIGHT, WIDTH) of integer
    R: array (widthr, HEIGHT) of integer
    VARX1: array (widthr) of mpvar
    VARX2: array (widthr2) of mpvar
    VARY1: array (heightr) of mpvar
    VARY2: array (heightr2) of mpvar
    VARZ: array (widthr, heightr) of mpvar
end-declarations
    
```

```

initializations from 'tabella.dat'
A0
end-initializations

writeln("Begin running model")

forall (x in HEIGHT) do
  forall (y in WIDTH) do
    A(y, H-x+1):=A0 (x, y)
  end-do
end-do

forall ( x in widthr ) do
  forall ( y in heightr ) do
    if ((y <= H-h+1) and (x<=W-w+1)) then
      COUNT:=0
      forall ( i in x..(x+w-1) ) do
        forall ( j in y..(y+h-1) ) do
          COUNT:=COUNT + A(i,j)
        end-do
      end-do
      if (COUNT=w*h) then R(x,y):=1
        else R(x,y):=0
      end-if
    else R(x,y):=0
    end-if
  end-do
end-do

forall (x in widthr) VARX1(x) is_binary
forall (y in heightr) VARY1(y) is_binary
forall (x in widthr2) VARX2(x) is_binary
forall (y in heightr2) VARY2(y) is_binary

forall (x in widthr, y in heightr) VARZ(x,y) is_binary

z:= (sum(x in widthr,y in heightr) (R(x,y)* VARZ(x,y)))

! Il vincolo v1 assicura la non sovrapposizione degli stand
! sull'asse delle x
forall (k in (w+a)..(W-w+1)) do

  if (k-w-a+1>1) then start1:=k-w-a+1
    else start1:=1
  end-if
  if (k-2*w-a+1>1) then start2:=k-2*w-a+1
    else start2:=1
  end-if

```

```

if (W-w+1<k) then end1:=W-w+1
    else end1:=k
end-if

if (W-2*w+1<k) then end2:=W-2*w+1
    else end2:=k
end-if

v1(k):= (sum(x in start1..end1)(VARX1(x)))+
    (sum(x in start2..end2)(VARX2(x))) <= 1
end-do

! Il vincolo v2 assicura la non sovrapposizione degli stand
! sull'asse delle y
forall (k in (h+c)..(H-h+1)) do

    if (k-h-c+1>1) then start3:=k-h-c+1
        else start3:=1
    end-if

    if (k-2*h-c+1>1) then start4:=k-2*h-c+1
        else start4:=1
    end-if

    if (H-h+1<k) then end3:=H-h+1
        else end3:=k
    end-if

    if (H-2*h+1<k) then end4:=H-2*h+1
        else end4:=k
    end-if

    v2(k):= (sum(y in start3..end3)(VARY1(y)))+
        (sum(y in start4..end4)(VARY2(y))) <= 1

end-do

! Il vincolo v3 assicura la giusta relazione tra le variabili
! VARZ, VARX1 e VARX2
forall (x in widthr, y in heightr) do
    if x<=w then
        v3(x,y):= VARZ(x,y)-VARX1(x)-VARX2(x)<=0
    else
        if x>W-2*w+1 then
            v3(x,y):= VARZ(x,y)-VARX1(x)-VARX2(x-w)<=0
        else
            v3(x,y):= VARZ(x,y)-VARX1(x)-VARX2(x-w)-VARX2(x)<=0
        end-if
    end-if
end-do

```

```

! Il vincolo v4 assicura la giusta relazione tra le variabili
! VARZ, VARY1 e VARY2
forall (x in widthr, y in heightr) do
  if y<=h then
    v4(x,y):= VARZ(x,y)-VARY1(y)-VARY2(y)<=0
  else
    if y>H-2*h+1 then
      v4(x,y):= VARZ(x,y)-VARY1(y)-VARY2(y-h)<=0
    else
      v4(x,y):= VARZ(x,y)-VARY1(y)-VARY2(y)-VARY2(y-h)<=0
    end-if
  end-if
end-do

maximize(z)

writeln("z=", getobjval)

forall (i in widthr) do
  writeln("x1(",i,")", getsol(VARX1(i)))
end-do

forall (j in heightr) do
  writeln("y1(",j,")", getsol(VARY1(j)))
end-do

forall (i in widthr2) do
  writeln("x2(",i,")", getsol(VARX2(i)))
end-do

forall (j in heightr2) do
  writeln("y2(",j,")", getsol(VARY2(j)))
end-do

forall (i in widthr) do
  forall (j in heightr) do
    writeln("z(",i,j,")", getsol(VARZ(i,j)))
  end-do
end-do

forall (x in widthr) do
  forall (y in heightr) do
    if ((getsol(VARZ(x,y))=1))and ((R(x,y)= 1)) then
      writeln ("C'è uno stand in (" , x, ", ", y, ")")
    end-if
  end-do
end-do

```

```
writeln("End running model")

end-model
```

Il modello 5 presenta cinque variabili:

- VARX1 assume valore 1 se è scelta una *strip* di larghezza  $w+a$
- VARX2 assume valore 1 se è scelta una *strip* di larghezza  $w+w+a$
- VARY1 assume valore 1 se è scelta una *strip* di altezza  $h+c$
- VARY2 assume valore 1 se è scelta una *strip* di altezza  $h+h+c$
- VARZ assume valore 1 se è scelta una certa posizione  $(i, j)$

Inoltre, come già detto nel capitolo 4, il modello 5 può essere visto come la combinazione dei modelli 3 e 4. In particolare:

- I vincoli  $v1$  e  $v2$  assicurano la non sovrapposizione degli stand e funzionano come il vincolo  $v1$  del modello 3; l'unica differenza è che in questo caso è necessario sviluppare il vincolo sia lungo l'asse orizzontale che lungo l'asse verticale.
- I vincoli  $v3$  e  $v4$  funzionano come i vincoli  $v5$  e  $v6$  del modello 4\_2, con l'unica differenza che vi sono due variabili in più.

Il modello 5 termina la successione di algoritmi per i layout fieristici considerati in questo scritto.

## Capitolo 6 RISULTATI

I modelli presentati nel capitolo precedente sono tutti stati implementati sulle fiere di Reggio Emilia; è stata cioè data in input al programma una matrice di uni e di zeri rappresentante la superficie espositiva delle Fiere di Reggio Emilia. Si sottolinea che quest'ultima è stata presa solo a titolo d'esempio, in quanto i modelli sono generalizzabili a molti altri contesti e tipi di superfici.

### 6.1 DESCRIZIONE DELLA SUPERFICIE ESPOSITIVA PRESA COME ESEMPIO: PADIGLIONE C DELLE FIERE DI REGGIO EMILIA

La zona fiere di Reggio Emilia, rappresentata in figura 6.1, è composta da quattro padiglioni (di cui tre di uguali dimensioni, ovvero i padiglioni B, C e D) e da un'area esterna.



Figura 6.1: Piantina delle fiere di Reggio Emilia allestite in occasione di un particolare evento.

In particolare i modelli sono stati implementati sul padiglione C:

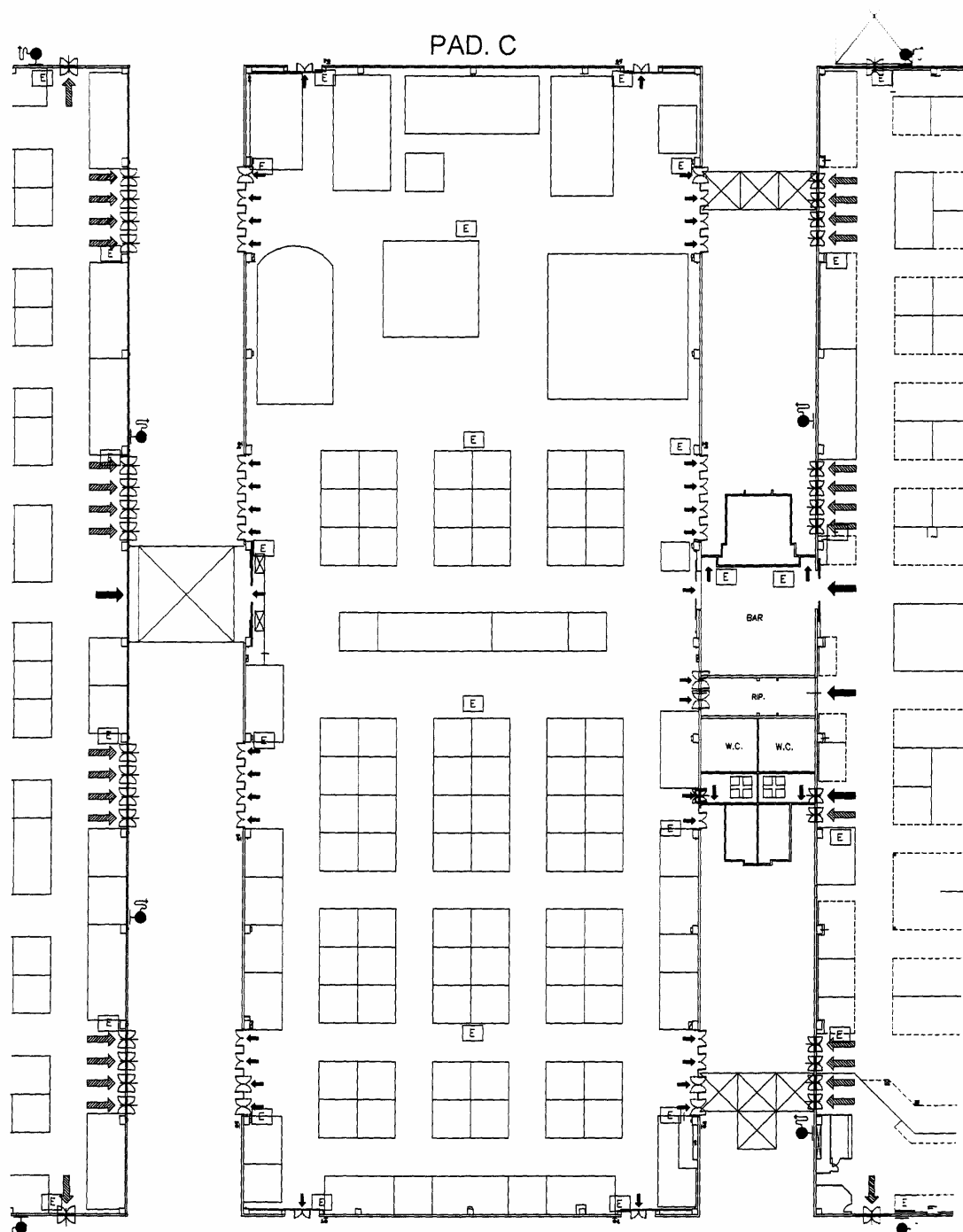


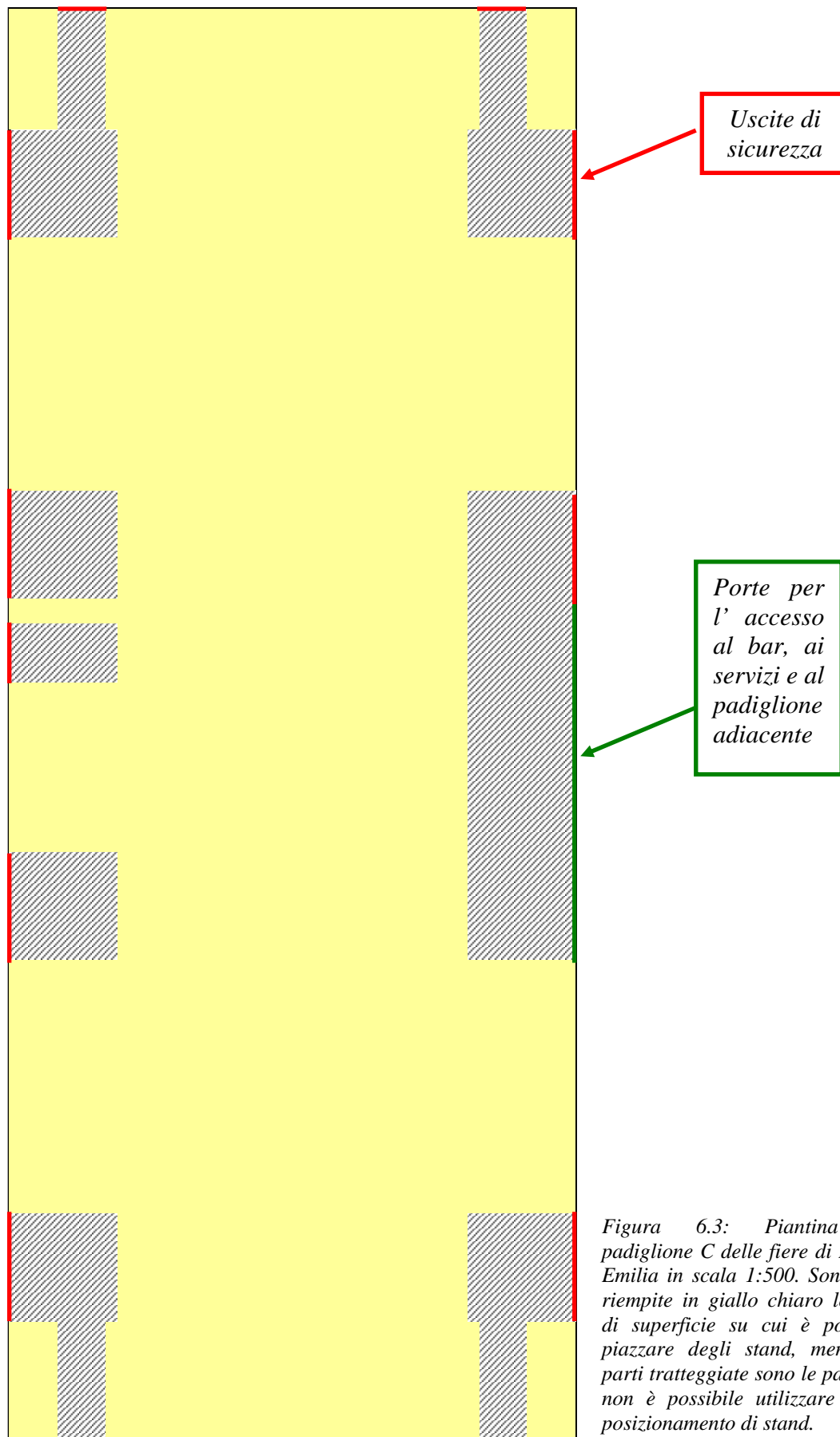
Figura 6.2: Piantina del Padiglione C delle Fiere di Reggio Emilia, allestito con un particolare layout.

I dati principali riguardanti le Fiere di Reggio Emilia che sono stati presi in considerazione sono i seguenti:

- *Dimensioni del padiglione C: 119,80 x 47,60 metri*
- *Dimensioni dei moduli costituenti gli stand espositivi: 4 x 4 metri*
- *Dimensione minima dei corridoi tra gli stand: 3 metri*
- *Modulo di sicurezza, ovvero larghezza minima che deve essere adibita alle porte di sicurezza: 0,60 metri*
- *Dimensione del corridoio che adduce alla porta di sicurezza: almeno uguale alla somma dei moduli di sicurezza ( D.M. 19 agosto 1996)*

Dati alla mano, è stata costruita la matrice di uni e zeri relativa alla superficie espositiva considerata; di seguito è riportata la piantina del padiglione, dove le parti in giallo chiaro rappresentano le parti in cui la matrice presenta gli uni, mentre le parti tratteggiate sono quelle che rappresentano gli elementi della matrice uguali a zero.





*Figura 6.3: Piantina del padiglione C delle fiere di Reggio Emilia in scala 1:500. Sono state riempite in giallo chiaro le parti di superficie su cui è possibile piazzare degli stand, mentre le parti tratteggiate sono le parti che non è possibile utilizzare per il posizionamento di stand.*

## **6.2 RISULTATI DELL'IMPLEMENTAZIONE DEI MODELLI**

I primi tre modelli sono stati implementati con una approssimazione di un metro; più precisamente ciò significa che la matrice  $\emptyset$  di uni e di zeri è stata creata considerando ogni elemento come corrispettivo di un “quadrato” di superficie espositiva di grandezza  $1m \times 1m$ . La matrice risultante è pertanto una matrice di  $119 \times 47$  elementi (le misure del capannone sono state approssimate per difetto).

Gli ultimi due modelli sono stati invece implementati con una approssimazione di quattro metri; la matrice risultante conta 29 elementi in altezza e 11 in larghezza.

La cosa importante da sottolineare è che tali approssimazioni sono state prese solo per rendere rappresentabili risultati graficamente e per alcuni limiti sul numero di variabili imposti dalla licenza studenti del software Xpress-MP. I modelli sono infatti utilizzabili in qualsiasi tipo di superficie e con qualsiasi tipo di approssimazione, anche molto più piccola di quella presa in esame al momento.

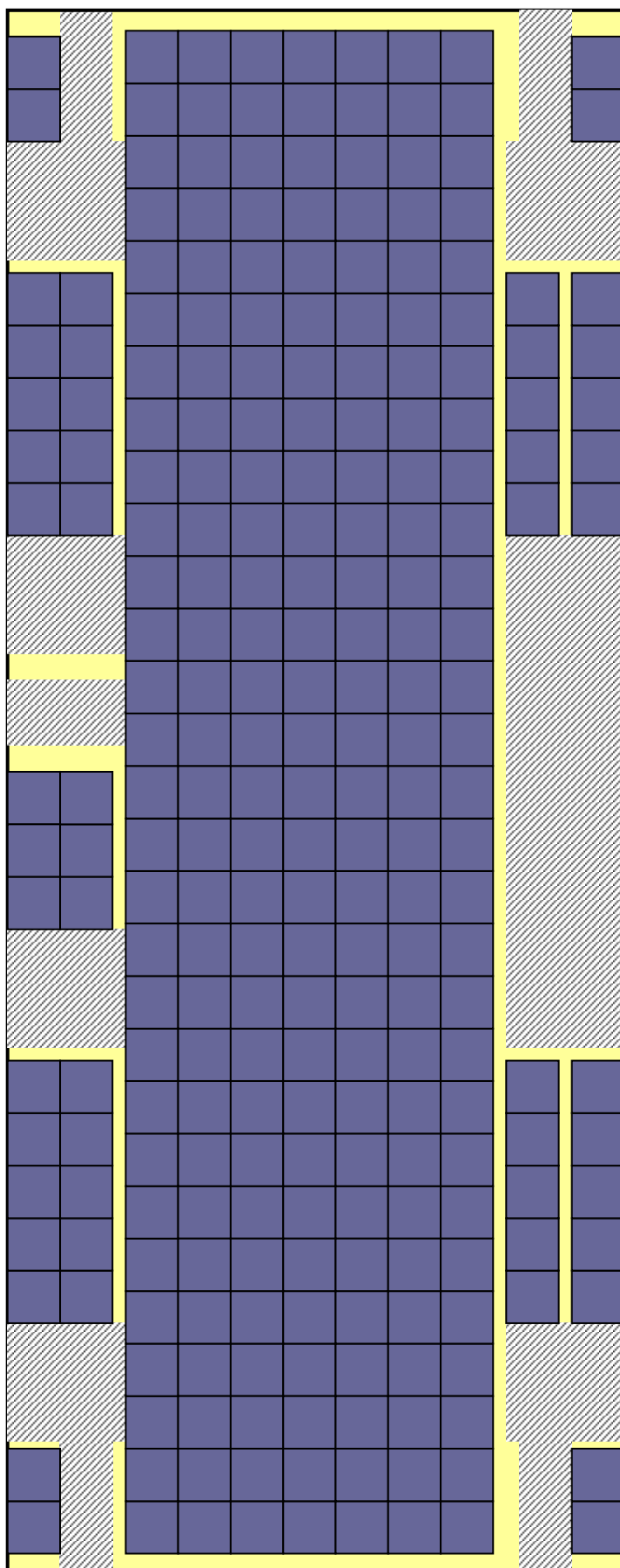
### **6.2.1 RISULTATI DEI PRIMI TRE MODELLI**

Se si implementa il modello 1, il layout risultante è quello mostrato in figura 6.4. Non ci si spaventi vedendo il gran numero di stand impaccati nella superficie senza alcuno spazio tra di essi: il modello 1 è infatti solo il primo passo del percorso fatto nella tesi e presenta solo il vincolo di non sovrapposizione degli stand.

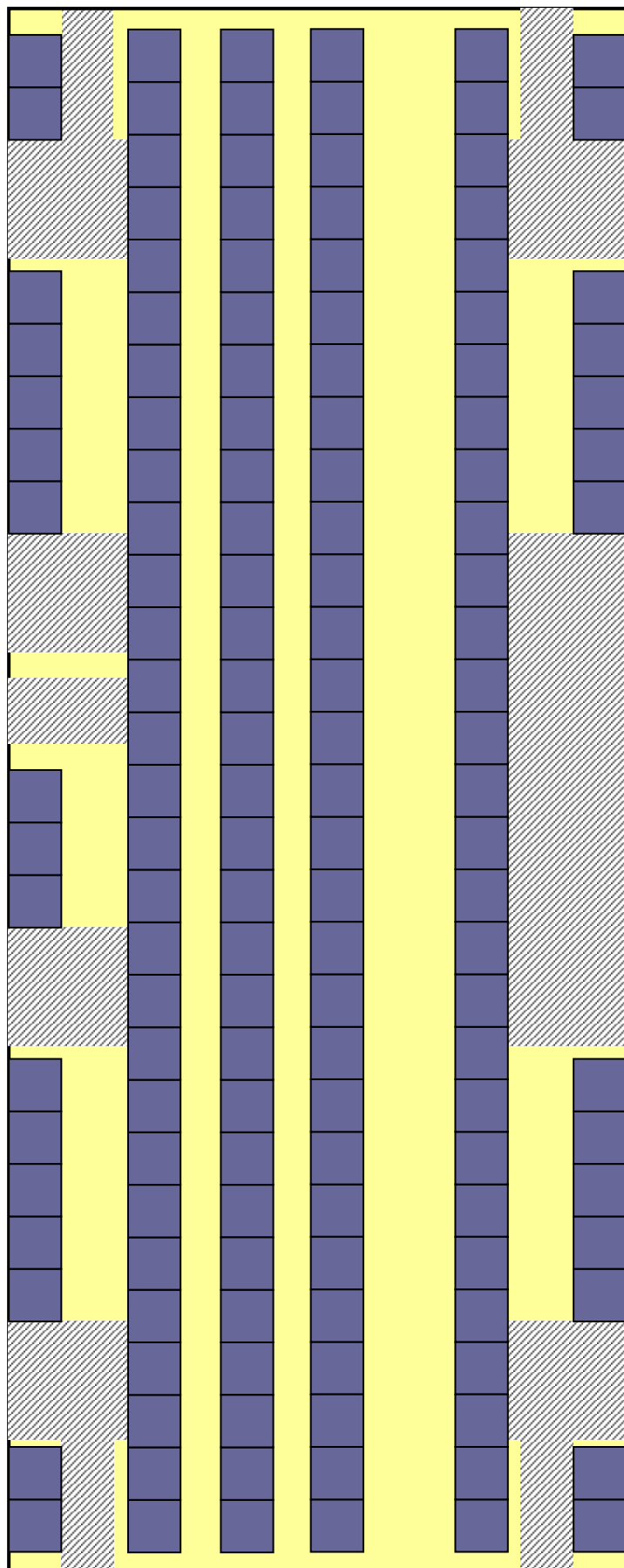
Guardando la figura 6.5 si può invece osservare un layout risultante che inizia a tener conto dei corridoi: esso presenta infatti una distanza minima tra una strip di stand e un'altra di 3 metri.

La figura 6.6 mostra il layout risultante dal modello 3: in esso è inserita la possibilità di posizionare anche double strip, ovvero due colonne attaccate di stand.

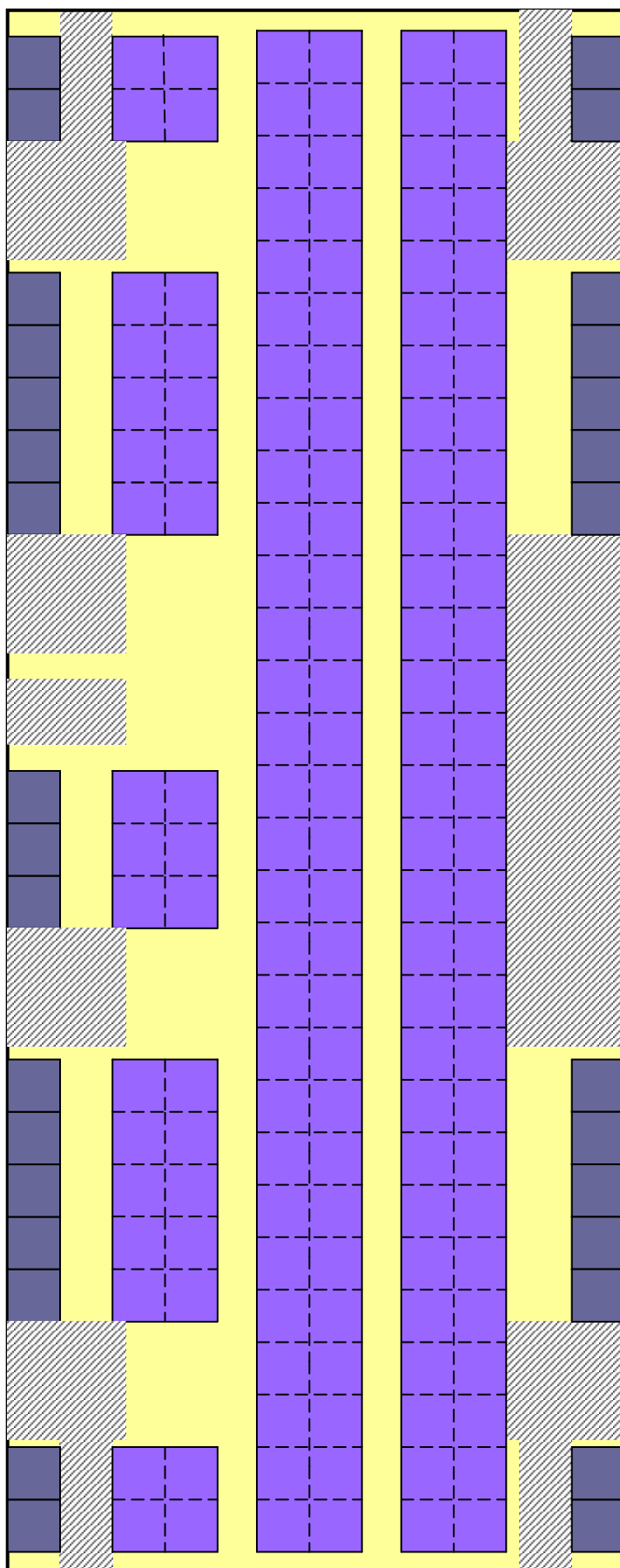
Si noti che per facilitare il passaggio dei visitatori si è deciso, in tutti i modelli, di non consentire il posizionamento degli stand direttamente accanto alle pareti più piccole del capannone, ma di lasciare da esse una distanza minima.



*Figura 6.4: Piantina del padiglione C delle fiere di Reggio Emilia in scala 1:500. Layout risultante dall'implementazione del modello 1. Il numero totale degli stand è 257.*



*Figura 6.5: Piantina del padiglione C delle fiere di Reggio Emilia in scala 1:500. Layout risultante dall'implementazione del modello 2. Il numero totale degli stand è 147.*



*Figura 6.6: Piantina del padiglione C delle fiere di Reggio Emilia in scala 1:500. Layout risultante dall'implementazione del modello 3. Le single strip sono rappresentate il viola scuro, mentre le double strip in viola chiaro. Il numero totale degli stand è 193.*

## 6.2.2 RISULTATI DEGLI ULTIMI DUE MODELLI

Come già detto precedentemente, i modelli 4 e 5 sono stati implementati su una matrice di  $29 \times 11$  elementi. Si ricordi che la differenza principale tra questi ultimi due modelli e i primi tre è che essi non posizionano più gli stand scegliendo le *strip*, ma posizioni ben precise della matrice. La differenza può essere ben visibile leggendo gli output dei programmi: essi sono riportati di seguito in forma abbreviata per problemi di spazio.

```

Begin running model
z=193
x1(1)=1 → è stata scelta la strip  $x_1^1$ 
x1(2)=0
x1(3)=0
x1(4)=0
x1(5)=0
...
x2(1)=0
x2(2)=0
x2(3)=0
x2(4)=0
x2(5)=0
x2(6)=0
x2(7)=0
x2(8)=0
x2(9)=1 → è stata scelta la strip  $x_9^2$ 
...
Lo stand appartenente alla single
strip è in (1,1)
Lo stand appartenente alla single
strip è in (1,5)
Lo stand appartenente alla single
strip è in (1,20)
...
Lo stand appartenente alla double
strip è in (9,1)
Lo stand appartenente alla double
strip è in (9,5)
Lo stand appartenente alla double
strip è in (9,20)
...
End running model

```

Figura 6.7a

```

Begin running model
z=124
z(1,1)=1 → è stata scelta la
variabile  $z_{1,1}$ 

z(1,2)=1
z(1,3)=0
z(1,4)=0
z(1,5)=1
z(1,6)=0
z(1,7)=1
z(1,8)=1
z(1,9)=0
...
z(11,28)=1 → è stata scelta la
variabile  $z_{11,28}$ 

z(11,29)=1

C'è uno stand in (1,1)
C'è uno stand in (1,2)
C'è uno stand in (1,5)
C'è uno stand in (1,7)
C'è uno stand in (1,8)
C'è uno stand in (1,13)
C'è uno stand in (1,14)
C'è uno stand in (1,20)
C'è uno stand in (1,22)
C'è uno stand in (1,23)
C'è uno stand in (1,25)
...
C'è uno stand in (11,28)
C'è uno stand in (11,29)
End running model

```

Figura 6.7b

Figura 6.7: Differenza tra gli output dei primi tre modelli (6.7a) e gli output degli ultimi due (6.7b)

Il layout risultante dai modelli 4 e 5 è rappresentato graficamente nelle figure 6.8 e 6.9. Il modello 4 è l'analogo del modello 1, ovvero assicura solo la non sovrapposizione di stand, mentre il modello 5 è il più generale e tiene conto di una certa distanza obbligatoria tra uno stand e il successivo (sia in verticale che in orizzontale). Si ricordi anche che l'impaccamento studiato è di tipo *checkerboard*, quindi tutti gli stand devono essere allineati. Si noti che nel modello 5 non è stata lasciata alcuna distanza tra gli stand e la parete più piccola, poiché il passaggio è garantito ugualmente.

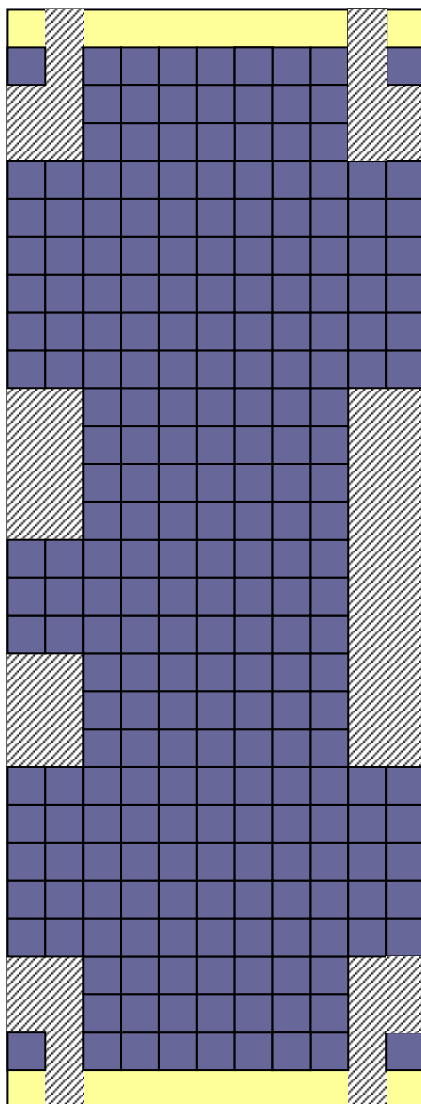


Figura 6.8: Piantina del padiglione delle Fiere di Reggio Emilia in scala 1:800. Layout risultante dall'implementazione del modello 4. Il numero totale degli stand è 243.

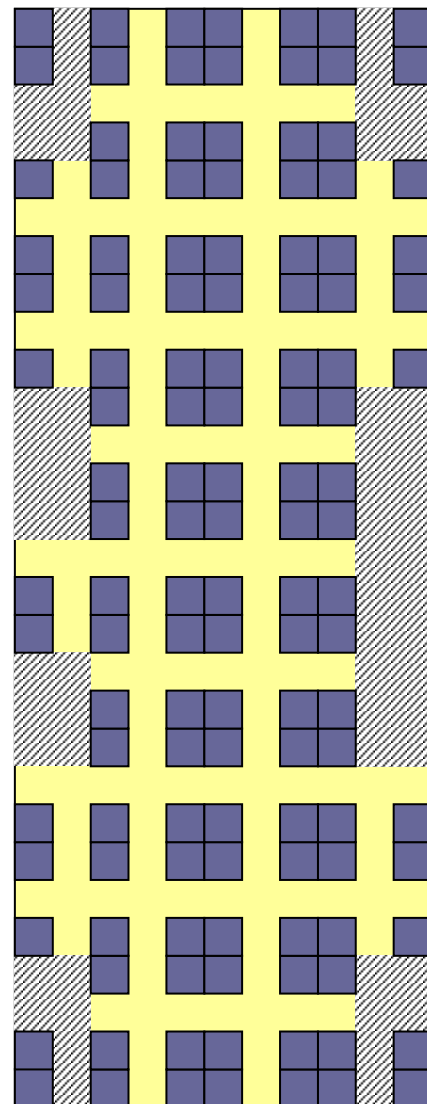
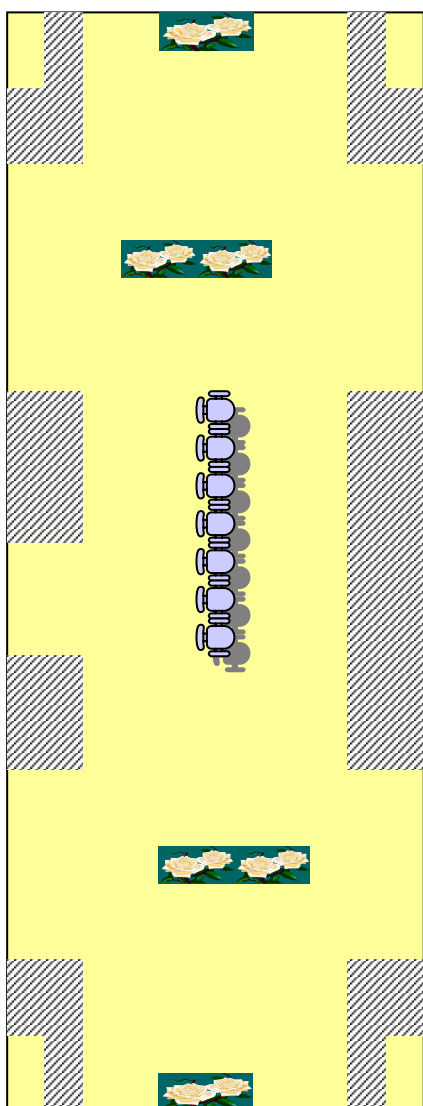
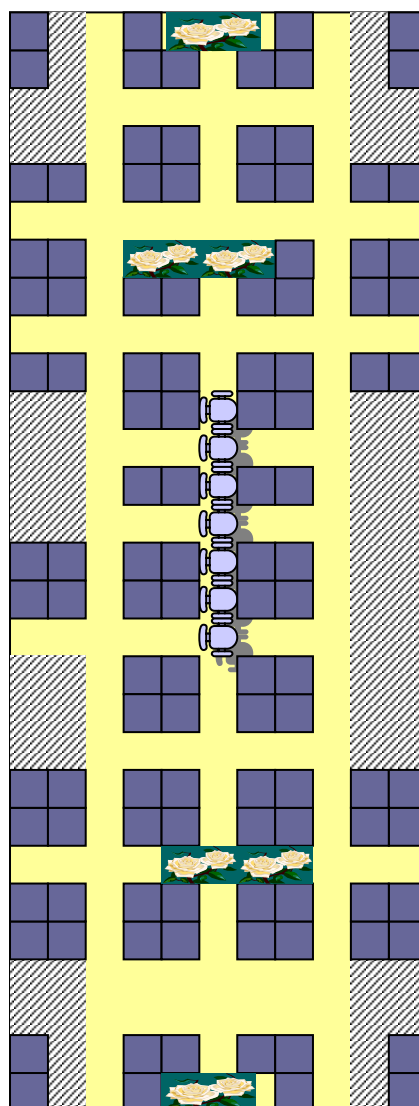


Figura 6.9: Piantina del padiglione delle Fiere di Reggio Emilia in scala 1:800. Layout risultante dall'implementazione del modello 5. Il numero totale degli stand è 114.

Finora è stata presa in esame una superficie espositiva nella quale gli unici spazi non utilizzabili per il posizionamento di stand sono quelli che bisogna lasciare nelle vicinanze delle porte di sicurezza o delle porte che permettono l'accesso ad altri spazi. Si supponga ora di immettere all'interno della superficie espositiva un maggior numero di *ostacoli*: i modelli, in particolare il più generale, ovvero il modello 5, sono funzionali ugualmente.



*Figura 6.10: Piantina del padiglione delle Fiere di Reggio Emilia in scala 1:800. Inserimento di alcuni elementi aggiuntivi all'interno dell'area espositiva (fioriere e sedie).*



*Figura 6.11: Piantina del padiglione delle Fiere di Reggio Emilia in scala 1:800. Layout risultante dall'implementazione del modello 5 sulla piantina di figura 6.10. Il numero totale di stand è 113.*



## Conclusioni

**P**rima di parlare di obiettivi raggiunti, ripercorriamo velocemente tutta la strada fatta. Dopo aver individuato nel *FLOP (Fair Layout Problem)* il cuore della tesi, si è intrapreso un percorso di ricerca nella letteratura, dapprima in ambito logistico e poi nell'area della ricerca operativa, allo scopo di trovare qualche supporto bibliografico significativo. In ambito logistico sono stati trovati solo testi inerenti al layout industriale, ma nessun testo focalizzato sul layout fieristico. Anche per quanto riguarda l'ambito della ricerca operativa, non sono state trovate pubblicazioni indirizzate al *FLOP*, ma sono comunque stati trovati problemi simili. Il problema del posizionamento di stand può essere infatti visto come caso particolare dei problemi di impaccamento; un ampio capitolo della tesi è incentrato sui problemi di *Knapsack*, di *Cutting* e di *Packing*.

La ricerca bibliografica ha portato alla scoperta di un altro problema già ampiamente discusso in letteratura e che si avvicina a quello della tesi; si tratta del *Facility Layout Problem (FLP)*. In particolare una tipologia particolare di quest'ultimo, ovvero il *Quadratic Assignment Problem (QAP)*, presenta caratteri molto simili agli ultimi modelli presentati nel capitolo 4 (modelli *MMC1*, *MMC1'* e *MMC2'*). Se da un lato gli algoritmi studiati per il *FLP* possono essere prese come riferimento e come spunto per alcuni sviluppi futuri del *FLOP*, non potranno mai essere presi *in toto*, poiché nei primi vi è il flusso tra una posizione e l'altra come *driver* principale, mentre nel secondo non vi è alcun problema di movimentazione di materiali.

Una volta finita la parte di ricerca bibliografica sono stati affrontati i modelli per la risoluzione del *FLOP*; questi ultimi sono stati inseriti nel capitolo 4 per gradi crescenti di difficoltà. Il primo modello è il più semplice ma il meno reale, l'ultimo il più complesso ma anche il più generale.

La parte più interessante è però stata la trascrizione di tali modelli in linguaggio Mosel e la loro successiva implementazione con il software MP-Xpress. Proprio l'implementazione ha dimostrato la validità di tutti i modelli presentati nel capitolo 4; sono stati tutti correttamente compilati ed eseguiti dal software di ottimizzazione. In particolare si è dimostrato che le matrici associate alle variabili dei primi quattro modelli sono totalmente unimodulari, e proprio per questo i vincoli di interezza delle variabili possono essere sostituiti dai vincoli di minore o uguale; che tutti i modelli producono l'output istantaneamente o in breve tempo (al massimo è richiesto qualche

minuto per il modello 5); che tutti i modelli funzionano correttamente e restituiscono un output corretto e in linea con le richieste. L'unico limite che è stato posto all'implementazione è quello riguardante l'approssimazione effettuata sulla matrice di uni e di zeri: per i primi tre modelli è stata creata una matrice di  $119 \times 47$  elementi, dove cioè ogni suo elemento era il corrispettivo di un quadrato di superficie di  $1m \times 1m$ , mentre per gli ultimi due modelli è stata creata una matrice di  $29 \times 11$  elementi. Tuttavia questa limitazione è stata effettuata solo ed unicamente per consentire una rappresentazione grafica dei risultati nella tesi e per alcuni limiti sul numero di variabili imposti dalla licenza studenti di MP-Xpress. Tutto ciò significa che non sussiste alcun problema nell'implementare i modelli con matrici di dimensioni ben più elevate, e quindi con approssimazioni più piccole e numero di variabili più grande.

A parte tutte queste considerazioni, ciò che fa di questa tesi uno scritto interessante è la generalizzabilità del lavoro svolto. I modelli esposti possono infatti essere utili in qualsiasi contesto fieristico, in aree espositive di qualsiasi forma (sia regolare che irregolare): gli algoritmi matematici descritti sono infatti estremamente versatili e bastano piccoli cambiamenti per poterli adattare a qualsiasi esigenza. La conferma di tutto ciò è data dal fatto che i modelli siano stati correttamente usati in una fiera come quella di Fortaleza, localizzata in uno spazio aperto e irregolare, ma anche nel contesto fieristico di Reggio Emilia caratterizzato da spazi chiusi e regolari.

In sintesi, i punti di forza della tesi sono l'innovazione del punto di vista col quale è affrontato l'argomento dei layout fieristici e la generalizzabilità dei modelli matematici presentati.

## **Ringraziamenti**

Nel licenziare questo lavoro, desidero esprimere la mia gratitudine al Professor Manuel Iori, dimostratosi sempre molto disponibile e puntuale nelle risposte, e al Presidente Valter Franceschini e all'Architetto Manuela Maccagnani delle Fiere di Reggio Emilia, per la loro cordialità.

Un ringraziamento particolare a tutta la mia famiglia, in particolare ai miei genitori, che mi sono stati da sempre molto vicini e mi hanno aiutato in ogni situazione.

Un grazie di cuore anche a Gio per avermi sostenuto ed essermi stato molto accanto in questi ultimi due anni.

Un pensiero speciale va inoltre alla mia nipotina Alessandra.

### Bibliografia generale:

- E.Aarts, J.K.Lenstra (Eds.)  
**Local Search in Combinatorial Optimization**  
Wiley, Chichester, 1997.
- R.Alvarez-Valdes, F.Parreño, J.M.Tamarit  
**A tabu search algorithm for a two-dimensional non-guillotine cutting problem**  
*European Journal of Operational Research*, (2006) *In press*
- R.Baldacci, M.Dell'Amico  
**Fondamenti di ricerca operativa**  
Pitagora Editrice Bologna (2002)
- J.E.Beasley  
**An exact two-dimensional non-guillotine cutting tree search procedure**  
*Operational Research* (1985) 33, 49-64
- J.E.Beasley  
**A population heuristic for constrained two-dimensional non-guillotine cutting**  
*European Journal of Operational Research*, (2003)
- B.S. Baker, E.G.Coffman, R.L. Rivest  
**Orthogonal packing in two dimensions**  
*SIAM Journal on Computing* (1980) 9, 846-855
- O.Barkey, P.Y. Wang  
**Two dimensional finite bin packing algorithms**  
*J.Oper.Res.Soc.* (1987) 38, 423-429
- M. Boario, M. De Martini, E. Di Meo, G.M. Gros-Pietro  
**Manuale di logistica**, Volumi 1 e 2  
Utet
- AP. Borbosa-Povoa, R. Mateus, AQ. Novais  
**Optimal two dimensional layout of industrial facilities**  
*International Journal of Production Research* (2001) 39, 2567-2593
- A. Bortfeldt  
**A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces**  
*European Journal of Operational Research* (2006) 172, 814-837
- RE. Burkard, F.Rend  
**A thermodynamically motivated simulation procedure for combinatorial optimization problems**  
*European Journal of Operational Research* (1984) 17, 169-174
- A. Caprara, M. Monaci  
**On the 2-Dimensional Knapsack Problem**  
*Operations Research Letters* (2003) 32, 5-14
- W. Chiang, C. Chiang  
**Intelligent local search strategies for solvine facility layout problems with the quadratic assignment problem formulation**  
*European Journal of Operational Research* 106 (1998) 457-488
- W.C. Chiang, P.Kouveli  
**Improved tabu search heuristic for solvine facility layout design problems**  
*International Journal of Production Research* (1996) 34, 2565-2585

- N.Christofides, A.Mingozzi, P.Toth  
**Contributions to the quadratic assignment problem**  
*European Journal of Operational Research* (1980) 18, 243-247
- F.Clautiaux, J.Carlier, A.Moukrim  
**A new exact method for the two-dimensional orthogonal packing problem**  
*European Journal of Operational Research* (2006) *In press*
- F. Clautiaux, J.Carlier, A.Moukrim  
**A new exact method for the two-dimensional bin-packing problem with fixed orientation**  
*Operations Research Letters* (2006) *In press*
- Jr. Coffman, E.G., M.R. Garey, D.S. Johnson  
**Performance bounds for level-oriented two-dimensional packing algorithms**  
*SIAM Journal on Computing* (1980) 9, 801-826
- E.G.Coffman Jr, M.R.Garey, D.S.Johnson  
**Approximation algorithms for bin-packing-an updated survey**  
Algorithm Design for Computer System Design. G.Ausiello, M.Lucertini, P.Serafini (eds), Springer, Vienna (1984), 49-106.
- Y. Cui, X. Zhang  
**Two-stage general block patterns for the two-dimensional cutting problem**  
*Computers and Operations Research* (2006) *In press*
- G.B.Dantzig  
**Discrete variable extremum problems**  
*Operations Research* (1957) 5, 266-277
- K.Dowsland  
**Some experiments with simulated annealing techniques for packing problems**  
*European Journal of Operational Research* (1993) 68, 389-399.
- O.Færø, D.Pisinger, M.Zachariasen  
**Guided local search for three-dimensional bin packing problem**  
Technical paper, DIKU, University of Copenhagen, 1999.
- S.P.Fekete, J.Schepers  
**On more-dimensional packing I: Modeling**  
Technical paper ZPR97-288, Mathematisches Institut, Universität zu Köln, 1997.
- AE. Fernandez Muritiba, M. Iori e M. Negreiros  
**Models and Algorithms for Optimizing Fair Lay – Outs**  
*Rapporto Tecnico OR/06/8, DEIS, Università di Bologna* (2006).
- M.L.Fisher  
**The Lagrangian relaxation method for solving integer programming problems**  
*Management Science* (1981) 27, 1-18
- LR Foulds  
**Graph theory and applications**  
Springer, Berlin Heidelberg New York (1991)
- RL Francis, JA White  
**Facility layout and location: an analytical approach**  
Prentice Hall, Englewood Cliffs, NJ (1974)

- P.C. Gilmore, R.E. Gomory  
**Multistage cutting problems of two and more dimensions**  
*Operations Research* (1965) 13, 94-119
- F.Glover, M.Laguna  
**Tabu Search**  
Kluwer Academic Publishers, Boston, 1997.
- E.Hadjiconstantinou, N.Christofides  
**An exact algorithm for general, orthogonal, two-dimensional knapsack problems**  
*European Journal of Operational Research* (1995) 83, 39-56
- E.Hadjiconstantinou, M.Iori  
**An hybrid genetic algorithm for the two-dimensional single large object placement problem**  
*European Journal of Operational Research*, (2006) *In press*
- MMD. Hassan, GL Hogg  
**A review of graph theory applications to the facilities layout problem**  
*Omega* (1987) 15, 291-300
- SA Helm, SW Hadley  
**Tabu search based heuristics for multi floor facility layout**  
*Int.J. Prod Res* (2000) 38; 365-383
- S.Heragu, A.Kusiak  
**Efficient models for the facility layout problems**  
*European Journal of Operational Research* (1991) 53, 1-13
- C. Hicks  
**A Genetic Algorithm tool for optimising cellular or functional layouts in the capital goods industry**  
*International Journal of Production Economics*, Volume 104, Issue 2, December 2006, Pages 598-614
- E. Knod, R. Schonberger  
**Gestione della produzione**  
McGraw-Hill
- A.Houshyar, B.White  
**Comparison of solution procedures to the facility location problem**  
*Computers industrial Engng.* Vol.32 (1997) pp. 77-87
- JY Kim, YD Kim  
**A branch-and-bound algorithm for locating input and output points of departments on the block layout**  
*Journal of Operational Research Soc.* (1999) 50, 517-525
- P.J. Kolesar  
**A Branch and Bound Algorithm for the Knapsack Problem**  
*Management Science* (1967) 13, 723-735
- TC Koopmans, M.Beckam  
**Assignment problems and the location of economic activities**  
*Econometrica* (1957) 25, 53-76
- S. Imahori, M. Yagiura and H. Nagamochi  
**Practical Algorithms for Two-dimensional Packing**  
*Mathematical Engineering Technical Reports*

- M.Iori, S.Martello, M.Monaci  
**Metaheuristic Algorithms for Strip Packing Problem**  
 Optimization and Industry: New Frontiers, Kluwer Academic Publisher  
 (Pardalos P. e Korotkich V eds.) (2003)
- S.Jakobs  
**On genetic algorithms for the packing of polygons**  
*European Journal of Operational Research* (1996) 88, 165
- D.s.Johnson, A.Demers, J.D.Ullman, M.R.Garey, R.L. Graham  
**Worst-case performance bounds for simple one-dimensional packing algorithms.**  
*SIAM Journal on Computing* (1974) 3, 299-325
- TA. Lacksonen  
**Static and dynamic facility layout problems with varying areas**  
*Journal of Operational Research Soc.* (1994) 45, 59-69
- TA. Lacksonen  
**Pre-processing for static and dynamic facility layout problems**  
*Int J Prod Res* (1997) 35, 1095-1106
- E.L.Lawler  
**The Quadratic Assignment Problem**  
*Management Science* (1963) 9, 586-599
- D.Liu, and Teng, H.  
**An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles**  
*European Journal of Operational Research* (1999) 112, 413
- A. Lodi, S. Martello, M. Monaci  
**Two-dimensional packing problems: A survey**  
*European Journal of Operational Research* 141 (2002) 241-252
- A.Lodi, S.Martello, D.Vigo  
**Approximation algorithms for the oriented two-dimensional bin packing problem**  
*European Journal of Operational Research* (1999) 112, 158-166
- A.Lodi, S.Martello, D.Vigo  
**Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems**  
*INFORMS Journal on Computing* (1999) 11, 345-357.
- A. Lodi, S. Martello, D. Vigo  
**Recent advances on two-dimensional bin packing problems**  
*Discrete Applied Mathematics* (2002) 123, 379-396
- N.Maculan  
**Relaxation Lagrangienne: le problème du knapsack 0-1.**  
*Canadian Journal of Operational Research and Information Processing* (1983) 21, 315-327
- S. Martello, M. Monaci, D.Vigo  
**An exact Approach to the Strip-Packing Problem**  
*INFORMS Journal on Computing* (2003) 15, 310-319
- S.Martello, D.Pisinger, D.Vigo  
**The three dimensional bin packing problem**  
*Operations Research*, (2000) 48, 256-267

- S. Martello, P. Toth.  
**An upper bound for the zero-one knapsack problem and a branch and bound algorithm.**  
*European Journal of Operational Research* (1977) 1, 169–175.
- S. Martello, P. Toth Eds.  
**Knapsack Problems-Algorithms and Computer Implementations**  
John Wiley & Sons (1990)
- S.Martello, D.Vigo  
**Exact solution of the two-dimensional finite bin packing problem**  
*Management Science* (1998) 44, 388-399.
- B.Montreuil  
**A modelling framework for integrating layout design and flow network design**  
*Proceedings of the material handling research colloquium*, Hebron, KY, (1990) pp 43-58
- C.H. Papadimitriou e K. Steiglitz  
**Combinatorial Optimization: Algorithms and Complexity**  
Prentice Hall, Englewood Cliffs, N.J., 1988
- D.Pisinger  
**Where are the hard Knapsack Problems?**  
*Computers and Operational Research* (2005) 32, 2271-2282
- D. Pisinger, M. Sigurd  
**The two-dimensional bin packing problem with variable bin sizes and costs**  
*Discrete Optimization* (2005) 2, 154-167
- G.Scheithauer  
**Equivalence and dominance for problems of optimal packing of rectangles**  
*Ricerca Operativa* (1997) 83, 3-34
- S.P.Singh, R.R.K.Sharma  
**A review of different approaches to the facility layout problems**  
*Int.J. Manuf Technol* (2006) 30; 425-433
- EG Talbi, O.Roux, C.Fonlupt, D.Robillard  
**Parallel ant colonies for quadratic assignment problem**  
*Future Generation Computer System* (2001) 17, 441-449
- R.Tavakkoli-Moghaddain, E. Shanyan  
**Facilities layout design by genetic algorithms**  
*Computer Industrial Engineering* (1998) 35, 527-530
- JA.Tompkins, JA White  
**Facilities planning**  
Wiley, New York (1984)
- H.H.Yanasse, D.M.Katsurayama  
**Checkerboard pattern: proposals for its generation**  
*International Transactions in Operational Research* (2005) 12, 21-45



**Sitografia:**

- [www.dashoptimization.com](http://www.dashoptimization.com)
- [www.fierereggioemilia.it](http://www.fierereggioemilia.it)
- [www.move.bolognafiere.it](http://www.move.bolognafiere.it)
- [www.sciencedirect.com](http://www.sciencedirect.com)
- [www.springer.com](http://www.springer.com)