

Indice

Introduzione	i
1 Problemi di packing e cutting nei layout	1
1.1 Layout fieristici	2
1.2 Cutting and Packing problems	4
1.2.1 Knapsack problem	6
1.2.2 Bin Packing Problem	13
1.2.3 Two Dimensional Bin-packing Problem	18
1.2.4 Two Dimensional Strip-packing Problem	34
1.2.5 Two Dimensional Knapsack Problem	40
1.3 Facility layout problem	44
1.3.1 Modelli per il facility layout problem	45
1.3.2 Metodologie di risoluzione	48
2 Ottimizzazione dei layout fieristici	55
2.1 Specifiche e contesti per il progetto	56
2.1.1 Modelli matematici	61
2.2 Studio e analisi di casi reali	65
2.2.1 Caso 1: Fortaleza handcraft fair	66
2.2.2 Caso 2: Regional fair in Romont	70
2.2.3 Considerazioni finali	80
3 Casi di studio e specifiche di progetto	83
3.1 Presentazione dei casi di studio	84
3.1.1 Fiera del Levante	85
3.1.2 Fiera di Bolzano	88

3.1.3	Expo Ferroviaria 2008	91
3.2	Criterio per la definizione e la scrittura delle matrici	92
4	Implementazione degli algoritmi e risultati computazionali	97
4.1	Fiera del Levante: soluzione	98
4.2	Fiera di Bolzano: soluzione	102
4.3	Expo Ferroviaria 2008: soluzione	103
4.4	Considerazioni finali	104
5	Conclusioni	109
	Bibliografia	111

Ringraziamenti

Ringrazio sentitamente il mio relatore, Prof.Ing.Silvano Martello, e i miei correlatori, Prof.Ing.Daniele Vigo e Ing.Manuel Iori, per la disponibilità e l'aiuto nello sviluppo del progetto di tesi e nella conseguente stesura.

Ringrazio soprattutto la mia Famiglia per il prezioso sostegno, la fiducia, l'aiuto costante e i numerosi sacrifici; ringrazio inoltre tutti gli Amici e le persone che in questi anni mi sono stati vicini.

Grazie a tutti.

*La stupidità deriva dall'avere una risposta per ogni cosa
La saggezza deriva dall'avere una domanda per ogni cosa*

(Milan Kundera)

Introduzione

Questa tesi ha come obiettivo quello di determinare algoritmi e metodi per ottimizzare il layout di esposizioni fieristiche, con particolare riferimento e applicazione a tre contesti fieristici eterogenei, manifestazioni di grande importanza (sia su scala nazionale che internazionale) che hanno luogo in varie città italiane. In particolare si cercherà di implementare gli algoritmi studiati e illustrati nei primi due capitoli ai tre casi sopracitati, in modo da raggiungere l'obiettivo dell'applicabilità di tali metodologie ai casi più generali possibile.

Lo scopo del lavoro consiste quindi nel massimizzare il numero di stand espositivi che possono essere contenuti negli spazi adibiti alla fiera, focalizzando il concetto in base ai problemi di spazio, allocazione e adeguamento.

Nel corso della trattazione, dopo una esauriente presentazione dei modelli matematici, verranno studiati casi reali già discussi e ultimati, in maniera da ricondurre i problemi a realtà simili, che fungono da punti di partenza. In particolare verranno studiati contesti quali la fiera di Fortaleza (Brasile) e di Romont (Svizzera) poichè rappresentano due esempi esaustivi e completi per procedere nel progetto.

La struttura della tesi è organizzata in 5 capitoli così suddivisi:

- *Capitolo 1*: introduzione e analisi dei problemi di layout e packing, e dei relativi modelli matematici utili per l'ottimizzazione. Verranno illustrate le metodologie alle quali è possibile ricondurre un problema di organizzazione di spazi espositivi, in cui si presentano numerosi ostacoli quali l'area irregolare di esposizione, la dimensione degli stand, gli spazi necessari al passaggio delle persone e all'eventuale gestione da parte degli organizzatori, l'ambiente in cui ci si trova a lavorare. Il tut-

to improntato sull'obiettivo della programmazione matematica e degli algoritmi della Ricerca Operativa.

- *Capitolo 2*: presentazione di casi di studio già analizzati e conclusi. Le manifestazioni fieristiche di Fortaleza e di Romont sono esempi molto calzanti di applicazione delle tecniche dell'ottimizzazione fieristica a queste tipologie di problemi. Partendo dallo studio di entrambi i casi e adattandone le caratteristiche al contesto in oggetto, si sono potuti ricavare algoritmi e modelli utili, da applicare in maniera più specifica ai casi presi in considerazione nella tesi.
- *Capitolo 3*: casi di studio oggetto della tesi. Applicazione degli algoritmi di ottimizzazione di layout fieristici con particolare riferimento a tre contesti diversi: la Fiera del Levante di Bari, la Fiera di Bolzano - Messe Bozen e la Expo Ferroviaria 2008 presso il Lingotto di Torino. Questo capitolo consiste in una panoramica dei tre contesti analizzati, descrivendone le soprattutto le specifiche tecniche per la progettazione, pur senza tralasciare notizie storiche e caratteristiche di ciascuna manifestazione.
- *Capitolo 4*: soluzioni e possibili ulteriori obiettivi. I metodi della ricerca operativa e della ottimizzazione combinatoria vengono qui messi in pratica grazie all'implementazione software degli algoritmi visti e studiati nei precedenti capitoli. Con riferimento alle alternative studiate nel terzo capitolo, verranno qui implementate le metodologie per l'applicazione degli algoritmi a tre proposte piuttosto differenti tra loro, in maniera tale da affrontare il *fair layout problem* in maniera più generica possibile.
- *Capitolo 5*: conclusioni e obiettivi raggiunti. Breve sintesi della strada percorsa nel progetto di tesi, con uno sguardo ai modelli e agli algoritmi implementati e alla loro applicazione ai casi presi in esame.

Capitolo 1

Problemi di packing e cutting nei layout

Il punto di partenza per l'analisi delle problematiche legate ai layout fieristici è rappresentato dalla progettazione in ambito industriale. Prima di proseguire è necessario fornire alcune definizioni.

Si definisce **layout** come

una organizzazione della produzione che ha come 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.

Quindi, in maniera più semplice e sintetica, il layout non è altro che un modo di organizzare la gestione di beni materiali affidandosi a metodi razionali, matematici e ingegneristici. Si tratta di trovare il giusto compromesso tra uomo e produzione, in termini di gestione ottimale di quest'ultima.

Se si pensa che uno dei fattori cruciali che maggiormente influenzano i costi di un'azienda (circa dal 30% al 75%) è proprio legato alle politiche di layout e di movimentazione di materiale, è bene cercare delle soluzioni ottimizzate per ovviare al problema e ridurre di conseguenza i costi che ne derivano.

Il layout ottimale è quello che consente di soddisfare il maggior numero possibile di soggetti interessati ad un certo aspetto di produzione o di organizzazione (come ad esempio i vari settori coinvolti nell'allestimento di una fiera), cercando di venire in contro alle esigenze di tutti gli interessati. Questo non è affatto un compito semplice: cercare di soddisfare adeguatamente ogni gruppo di persone addette ai numerosi settori di organizzazione, significa creare un layout che soddisfi i requisiti di tutti i settori coinvolti. Ovviamente ogni gruppo coinvolto ha le proprie esigenze, in base al proprio lavoro e occupazione svolti. Per questo nel progettare un layout ottimo occorre tenere conto di diversi fattori:

- complessità e tipologia del processo di produzione
- volumi e quantità di produzione
- flussi e trasporti interni di materiale
- massima utilizzazione degli impianti e delle attrezzature
- utilizzo efficace di spazi e ambienti disponibili
- condizioni ambientali e locali favorevoli
- previsioni future (possibili estensioni, differenziazioni, ecc.)

Tutti questi fattori dipendono in maniera più o meno forte tra di loro, ed è quindi fondamentale occuparsene, oltre che singolarmente, anche nel loro complesso; i principali riscontri dovuti a un'adeguata organizzazione del lavoro si avranno sulla planimetria delle aree adibite all'esposizione. In particolare si otterranno vantaggi in ambito logistico, ad esempio nell'ottimizzazione dei flussi, nelle distanze da coprire, nelle informazioni e dei dati da scambiare, nelle attività da svolgere e portare a termine.

1.1 Layout fieristici

In un contesto fieristico il layout assume un ruolo fondamentale, oltre che per l'organizzazione interna, anche per il servizio offerto a clienti e visitatori;

ricopre inoltre una funzione speciale per determinare l'efficienza, l'estetica e le funzionalità della fiera. In aggiunta alle caratteristiche viste per un layout prettamente industriale, in ambito fieristico (derivante appunto da quello industriale) si hanno altre importanti peculiarità quali:

- efficace utilizzo degli spazi e dell'ambiente
- ottimizzazione dei percorsi per clienti e per organizzatori
- progettazione di spazi e luoghi coerentemente con una previsione adeguata del numero di visitatori
- corretto sfruttamento delle risorse ambientali disponibili

In pratica, ciò che differenzia principalmente i layout industriale e fieristico risiede nel fatto che mentre nel primo caso ha molta importanza la buona gestione del flusso di materiale, nel secondo prevalgono esigenze di tipo estetico e accattivante, in modo da attrarre l'attenzione e suscitare l'interesse nel maggior numero possibile di persone. Considerando un po' brutalmente l'ambito fieristico come un'industria, si può associare il prodotto industriale al cliente della fiera: il cliente che entra in una fiera è la materia prima da lavorare, il cliente che esce dalla fiera è il prodotto ultimato. L'obiettivo consiste quindi nel produrre il maggior numero di beni materiali ossia, nel caso fieristico, nel soddisfare il maggior numero possibile di clienti.

Sebbene in letteratura non esistano studi specifici in ambito fieristico, ma solo industriale, è bene adattare questi studi in maniera da estrapolare una metodologia di progetto valida anche nel caso di fiere ed esposizioni; la progettazione deve avvenire in modo preciso e metodico prendendo in considerazione tutti i possibili aspetti e gli elementi coinvolti.

Naturalmente l'approccio con cui progettare un layout dipenderà molto dalla tipologia in cui si inserisce il contesto fieristico: una fiera dedicata al mercato automobilistico sarà ben diversa di una dedicata alle specialità gastronomiche! Tra gli elementi fondamentali che costituiscono un layout, massima importanza ha la *planimetria*, tramite la quale è possibile studiare la disposizione di stand, passaggi, flussi e percorsi. La progettazione planimetrica dovrà anche tenere conto delle capacità, ovvero del "funzionamento" a pieno

e a vuoto¹. Analogamente ai metodi usati per progettare layout industriali, si procederà applicando questo approccio a quello fieristico. Nello schema di figura 1.1 è mostrato un possibile insieme di passi. Grazie all'aiuto della Ricerca Operativa² è possibile applicare delle metodologie di progetto adeguate al caso di layout fieristici. Nei prossimi paragrafi verranno introdotti alcuni importanti algoritmi per risolvere problemi legati al *cutting* e *packing* di oggetti e più in generale per determinare il layout ottimo.

1.2 Cutting and Packing problems

L'ottimizzazione combinatoria è una branca della Ricerca Operativa cui fanno riferimento i problemi di cutting e packing, i quali consistono nel trovare una soluzione massimizzata o minimizzata per mettere insieme un dato insieme di oggetti in uno o più contenitori. Questa metodologia è molto utilizzata dalle industrie manifatturiere (legno, carta, vetro, acciaio, pelle, ecc.) così come nelle tipografie per la stampa di giornali e quotidiani, ma anche in molti altri settori della produzione industriale. Per molti anni si sono cercate soluzioni che minimizzassero il *throughput*³ e i tempi di latenza⁴ dei prodotti da ultimare, ma che nel contempo massimizzassero le quantità e i volumi di produzione delle aziende. Settori ingegneristici della Ricerca Operativa, della *Computer Science* o dell'industria manifatturiera si sono a lungo occupati di **problemi di cutting and packing (C & P)**.

Basandosi su diversi criteri, uno su tutti legato alla **dimensione** del problema, si possono suddividere gli approcci progettuali in tre dimensioni. Trattandosi di layout fieristici, il nostro interesse si soffermerà solamente su due dimensioni, relativamente all'area espositiva della fiera.

¹si intende nei casi di massime e minime presenze di visitatori, ma anche di flussi di materiale, personale ed eventuali mezzi coinvolti nell'organizzazione.

²settore della matematica che si occupa prevalentemente di problemi di ottimizzazione di risorse, con largo uso di supporti matematici e informatici per la determinazione e l'applicazione di algoritmi di ottimizzazione

³numero di unità completate in un certo lasso di tempo.

⁴tempo impiegato per la produzione di un bene materiale o di un servizio offerto.

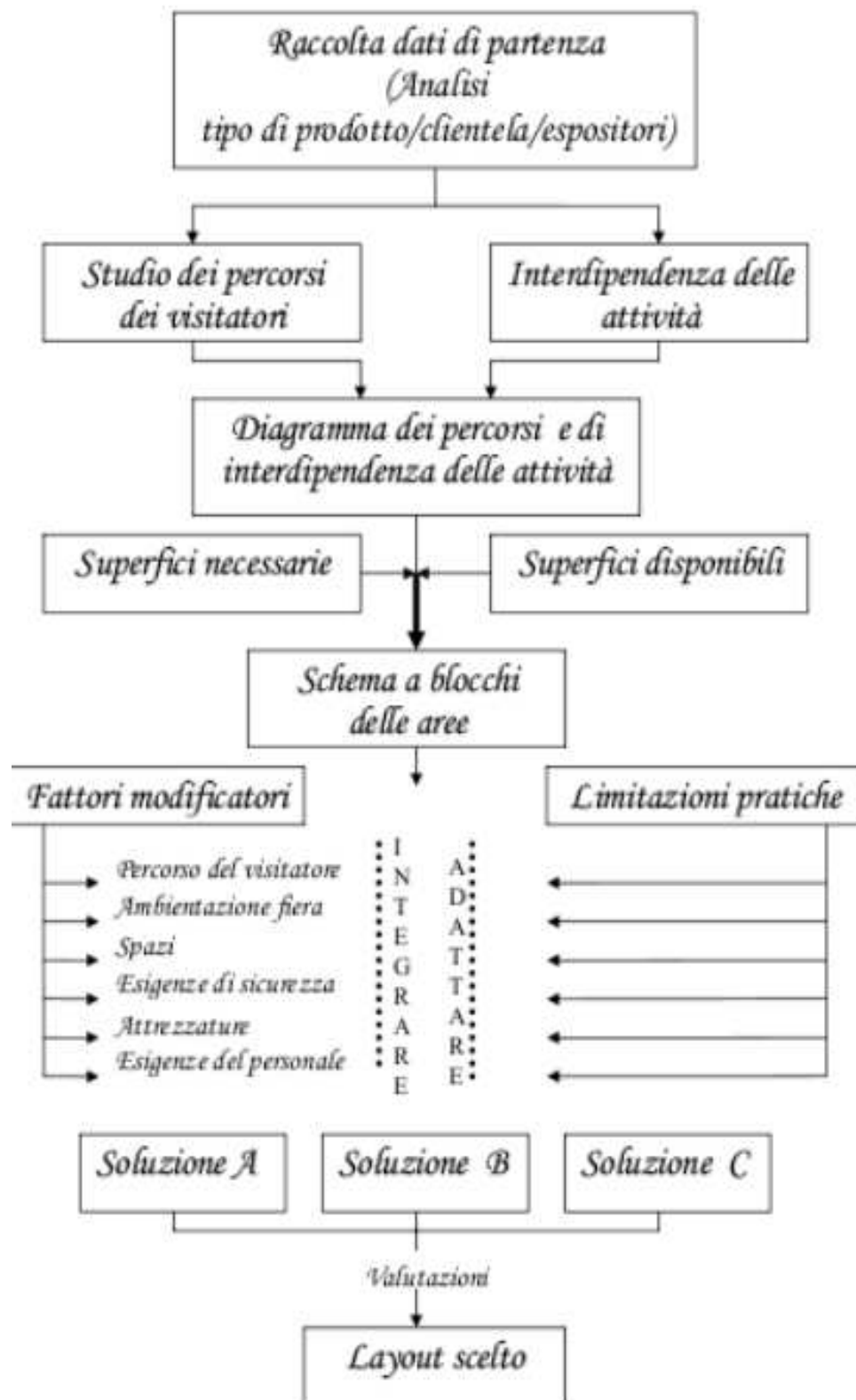


Figura 1.1: Semplificazione concettuale di un generico knapsack problem

• **C & P PROBLEMS AD UNA DIMENSIONE**

1. Knapsack algorithm 01 (KP)
2. Bin packing problem (BPP)

• **C & P PROBLEMS A DUE DIMENSIONI**

1. Two-dimensional Bin Packing Problem (2BPP)
2. Two-dimensional Strip Packing Problem (2SPP)
3. Two-dimensional Knapsack Problem (2KP)

E' necessario sottolineare che il *cutting and packing* si può considerare valido e attuabile nel caso in cui le dimensioni degli oggetti interessati (*items*) possano essere contenuti dagli oggetti contenitori (*containers*) e che entrambi items e containers non si sovrappongano tra loro.

1.2.1 Knapsack problem

Supponendo, tra varie alternative, il caso in cui si debbano scegliere degli oggetti compatibilmente con certi criteri prestabiliti, consideriamo nella fattispecie il caso in cui, per restare in tema, si debbano scegliere tra gli stands quelli adatti alle nostre specifiche di layout. Se numeriamo gli stand "idonei" da 1 a n e creiamo un vettore di variabili binarie $x_j (j = 1, \dots, n)$ in modo che valga

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

Se inoltre indichiamo con p_j il profitto dato dall'oggetto j (lo stand nel nostro caso), con w_j la sua dimensione e con c la dimensione del suo *container* (ossia il suo *costo*), allora l'obiettivo Knapsack sarà quello di selezionare, tra tutti, i vettori binari x che soddisfano il vincolo

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

e che massimizzano la funzione obiettivo

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

Per generalizzare il problema chiamiamo gli oggetti *items*, con una quantità pari ad n ; il valore e la dimensione dell'oggetto j –esimo sono chiamati rispettivamente *profitto* e *peso* e indicati con p_j e v_j ($j = (1, \dots, n)$). Naturalmente il discorso appena fatto riguarda un caso knapsack generico, il cui scopo consiste nel determinare uno o più sottoinsiemi per cui la somma dei pesi non superi (o eguagli) una certa quantità limite (detta *bound*) e la somma dei valori sia massimizzata. Questa tipologia di problema, riconducibile all'**algoritmo Knapsack**, ha riscontrato enorme interesse da entrambi i punti di vista teorico e pratico, adattandosi a varie forme di progettazione che richiedono l'ottimizzazione.

Di seguito si analizzeranno più in dettaglio le varie tipologie di *Cutting and Packing problems* in base alle definizioni precedentemente fornite per problemi a una e due dimensioni.

KNAPSACK 0-1 (BINARY)

Il *Knapsack problem 0-1* è il più importante dei problemi di ottimizzazione combinatoria. Esiste un esempio di problema associato allo knapsack che ne esplica in maniera piuttosto esauriente la funzione. Tale problema, detto “dello zaino”⁵ è posto nel modo seguente:

sia dato uno zaino che possa supportare un determinato peso e siano dati inoltre N oggetti, ognuno dei quali caratterizzato da un peso e un'utilità (ovvero un guadagno). Il problema si propone di scegliere quali di questi oggetti mettere nello zaino per ottenere la maggiore utilità senza eccedere nel peso sostenibile dallo zaino stesso,

il quale non è altro che un problema analogo al caso visto in precedenza. Detto ciò si può proseguire elencando tre motivi fondamentali per cui l'algoritmo ha assunto tale importanza:

- può essere trattato come un semplice problema di Programmazione Intera Lineare (ILP)

⁵in inglese *knapsack* indica proprio la parola *zaino*

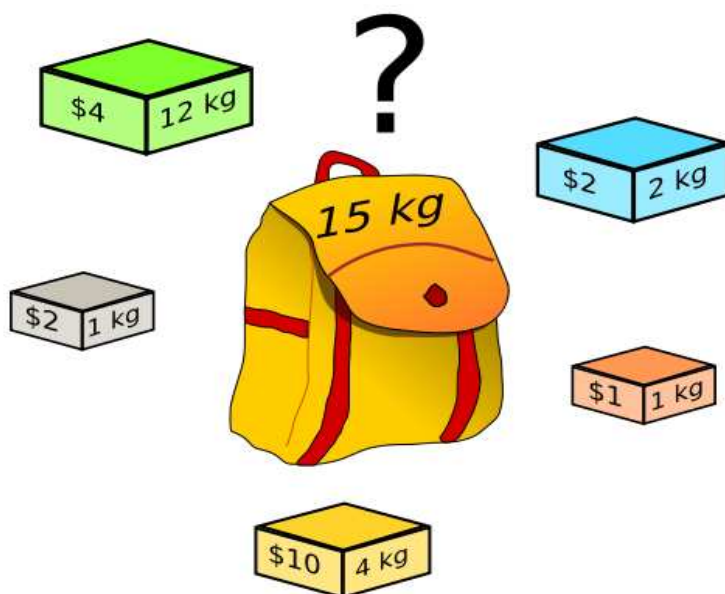


Figura 1.2: semplificazione concettuale di un generico knapsack problem

- appare come sottoproblema di problemi più complessi
- può essere utile in molte soluzioni pratiche

Le variabili d'interesse per lo sviluppo di questo algoritmo sono tutte binarie (da qui il nome “Knapsack 0-1”) e il succo del discorso può essere riassunto qui di seguito:

date le seguenti variabili

p_j = profitto dell'oggetto j

w_j = peso dell'oggetto j

c = capacità del contenitore

è opportuno scegliere un sottoinsieme di *items* tale da:

$$\text{massimizzare} \longrightarrow \max(z) = \sum_{j=1}^n p_j x_j \quad (1.1)$$

$$\text{soddisfacendo} \longrightarrow \sum_{j=1}^n w_j x_j \leq c \quad (1.2)$$

$$\text{dove } \longrightarrow x_j \in \{0, 1\}, j \in N = \{1, \dots, n\} \quad (1.3)$$

analogamente alle definizioni date in precedenza.

Se ora ipotizziamo che

$$p_j, w_j \text{ e } c \text{ sono interi positivi,} \quad (1.4)$$

$$\sum_{j=1}^n w_j \leq c, \quad (1.5)$$

$$w_j < c, \text{ con } j \in N, \quad (1.6)$$

e se consideriamo violata la prima ipotesi, allora si ottengono delle frazioni che possono essere moltiplicate per un opportuno fattore, mentre gli eventuali valori negativi verranno trattati come 0 o come 1 in base alle seguenti relazioni:

- $\forall j \in N^0 = \{j \in N : p_j \leq 0, w_j \geq 0\} \implies x_j = 0$
- $\forall j \in N^1 = \{j \in N : p_j \geq 0, w_j \leq 0\} \implies x_j = 1$

Definiamo inoltre N^- come $N^- = \{j \in N : p_j < 0, w_j > 0\}$, mentre per N^+ il valore sarà: $N^+ = N \setminus (N^0 \cup N^1 \cup N^-)$.

Infine definiamo:

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

Ora che tutte le variabili e i casi sono stati definiti, enunciamo il problema. Si vuole risolvere il problema che ha come obiettivo quello di massimizzare la funzione

$$z = \sum_{j \in N^- \cup N^+} \bar{p}_j y_j + \sum_{j \in N^1 \cup N^-} p_j$$

mentre come *constraint* del problema la funzione

$$\sum_{j \in N^- \cup N^+} \bar{w}_j y_j \leq c - \sum_{j \in N^1 \cup N^-} w_j.$$

Se i dati in input violano l'espressione (1.2) ipotizzata precedentemente allora $x_j = 0, \forall j \in N$, mentre se violano l'assunzione (1.3) allora si avrà $x_j =$

$0, \forall j \in N : w_j > c$.

Se non specificato in altro modo, si supporranno sempre gli *items* in ordine decrescente secondo l'ordine del profitto per unità di peso:

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

Data ad ogni problema una *istanza* I , si indicherà il valore di ogni soluzione ottima con $z(I)$ o più semplicemente con z . Considerando il problema nella versione minimizzata anzichè massimizzata, gli obiettivi si invertono secondo lo schema seguente:

$$\text{minimizzare} \longrightarrow \min(z) = \sum_{j=1}^n p_j y_j$$

$$\text{soddisfacendo} \longrightarrow \sum_{j=1}^n w_j y_j \leq q$$

$$\text{dove} \longrightarrow y_j \in \{0, 1\}, j \in N = \{1, \dots, n\}$$

dal quale si evince che il procedimento per tornare da una forma (minimizzata) all'altra (massimizzata) consiste nel porre $y_j = 1 - x_j$ e risolvendo la (1.1), la (1.2) e la (1.3) con $c = \sum_{j=1}^n w_j - q$.

Inoltre, ponendo come z_{max} il valore della soluzione massimizzata di ogni problema, la sua versione minimizzata sarà $z_{min} = \sum_{j=1}^n p_j - z_{max}$.

Rilassamento lineare

Togliendo il vincolo dell'integrità sulle variabili x_j , nelle proprietà (1.1), (1.2) e (1.3), si ottiene la prima variante del problema knapsack 0-1 ovvero il cosiddetto *rilassamento lineare* o *continuous Knapsack problem* $C(KP)$. In pratica un rilassamento è una qualunque tecnica che ha lo scopo di migliorare (ottimizzare) la funzione obiettivo in tutti i punti della regione ammissibile originale⁶:

$$\text{massimizzare} \longrightarrow \max(z) = \sum_{j=1}^n p_j x_j$$

⁶metodo improntato sull'ampliamento della regione ammissibile e/o sull'innalzamento della funzione obiettivo in corrispondenza della regione ammissibile originale

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

$$\text{dove} \longrightarrow 0 \leq x_j \leq 1, j = 1, \dots, n$$

Supponiamo inoltre che gli oggetti, ordinati come illustrato sopra, siano inseriti consecutivamente nel contenitore, finché non si trova un oggetto s che non può più essere inserito. Questo è un problema di *critical item*:

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

da cui si può notare, ricordando le (1.4), (1.5) e (1.6), che si ha $1 < s \leq n$. Tale metodo è stato risolto nel 1957 mediante un algoritmo scoperto da Dantzig.

Rilassamento lagrangiano

Un'alternativa al caso appena visto, di tipo lineare, è il rilassamento lagrangiano. In questo modo ci si pone lo scopo di eliminare un insieme di vincoli, tenendone però conto nella funzione obiettivo.

Dato un fattore non negativo λ , il *rilassamento lagrangiano* del KP, indicato con $(L(KP, \lambda))$ è:

$$\text{massimizzare} \longrightarrow \max(z) = \sum_{j=1}^n p_j x_j + \lambda(c - \sum_{j=1}^n w_j x_j)$$

$$\text{dove} \longrightarrow x_j \in \{0, 1\}, j = 1, \dots, n.$$

La funzione obiettivo può essere espressa come

$$z(L(KP, \lambda)) = \sum_{j=1}^n \tilde{p}_j x_j + \lambda c \quad (1.7)$$

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

$$\tilde{x}_j = \begin{cases} 1 & \text{se } \tilde{p}_j > 0 \\ 0 & \text{se } \tilde{p}_j < 0 \end{cases} \quad (1.8)$$

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

Quindi, definendo $J(\lambda) = \left\{ j : \frac{\tilde{p}_j}{w_j} > \lambda \right\}$, il valore della soluzione $L(KP, \lambda)$ sarà:

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

Per ogni $\lambda \geq 0$ c'è un upper bound in $z(KP)$ che, comunque, non potrà mai essere migliore della soluzione fornita da Dantzig; l'equazione (1.8) ci fornisce la soluzione del rilassamento continuo (o lagrangiano) di $L(KP, \lambda)$ nel modo seguente:

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

Il valore di λ che produce il valore minimo di $z(L(KP, \lambda))$ è $\lambda^* = \frac{p_s}{w_s}$. Con questo valore, in realtà, abbiamo che $\tilde{p}_j \geq 0$ per $j = 1, \dots, s-1$ mentre $\tilde{p}_j \leq 0$ per $j = s, \dots, n$. In questo modo $J(\lambda^*) \subseteq \{1, \dots, s-1\}$.

Da qui $\tilde{x}_j = \bar{x}_j$, per $j \in N \setminus \{s\}$ (dove \bar{x}_j è definita nel teorema seguente) e grazie alle (1.7) e (1.8), si evince che

$$z(L(KP, \lambda^*)) = \sum_{j=1}^{s-1} (p_j - \lambda^* w_j) + \lambda^* c = z(C(KP))$$

Infine si può notare che, per $\lambda = \lambda^*$, \tilde{p}_j diventa $p_j^* = p_j - w_j \frac{p_s}{w_s}$; il modulo $|p_j^*|$ è il decremento che si ottiene in $z(L(KP, \lambda^*))$ ponendo $\tilde{x}_j = 1 - x_j$ e da qui un lower bound sul corrispondente decremento della soluzione continua (da quando l'ottimo λ cambia imponendo le condizioni sopra elencate). Altri rilassamenti lagrangiani del problema KP sono stati studiati da Maculan (1983) e da Fisher (1981).

Teorema

La soluzione ottima \bar{x} del problema C(KP) è:

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

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

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

dove:

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

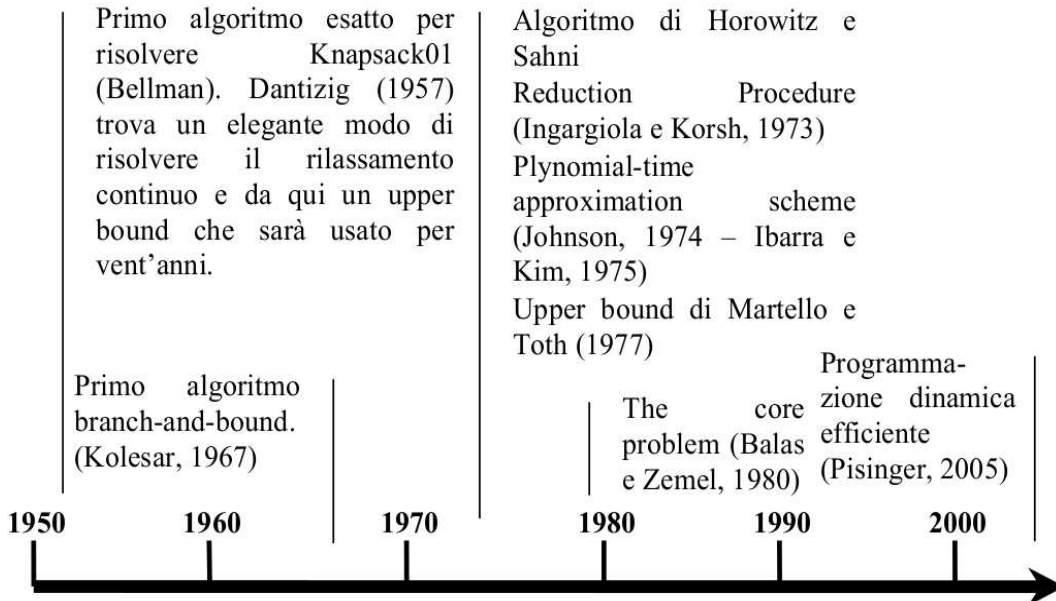


Figura 1.3: scoperte e applicazioni più importanti legate ai modelli knapsack problem in ordine cronologico

1.2.2 Bin Packing Problem

Il bin-packing problem (BPP) è un altro tipo di problema di natura combinatoria e consiste nel raggruppare oggetti di volumi diversi in un numero finito di compartimenti di capacità V , in maniera tale da minimizzare il numero di tali compartimenti da utilizzare.

Ovviamente, come per tutti i problemi di questo tipo, anche il BPP ha un campo di applicazione molto vasto, dall'industria ai trasporti, dalla logistica all'organizzazione aziendale, dall'informatica alle telecomunicazioni. Usando la terminologia e la formalizzazione tipiche dei problemi *knapsack*, indichiamo con n il numero di oggetti/items e di contenitori (*bins*) mentre indicheremo

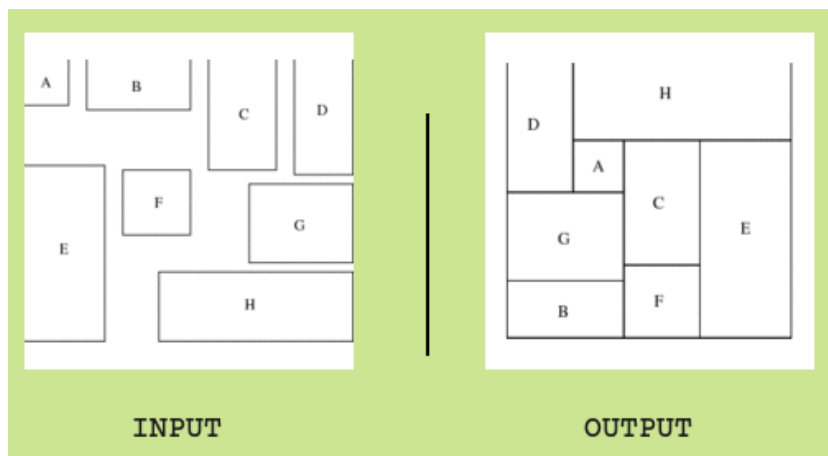


Figura 1.4: Bin Packing Problem

con w_j il peso dell'*item* j -esimo e con c la capacità di ogni *bin*.

Lo scopo del problema è di assegnare ogni item ad un bin in modo che il peso totale degli oggetti in ogni contenitore non ecceda la capacità c del contenitore stesso, e che il numero dei bins impiegati sia minimo.

Dal punto di vista matematico, il problema si pone nel modo seguente:

$$\text{minimizzare} \longrightarrow \min(z) = \sum_{i=1}^n y_i$$

$$\text{soddisfacendo} \longrightarrow \sum_{j=1}^n w_j x_{ij} \leq c y_i, i \in N = \{1, \dots, n\},$$

$$\sum_{i=1}^n x_{ij} = 1, j \in N,$$

$$y_j \in \{0, 1\}, i \in N,$$

$$x_{ij} \in \{0, 1\}, i, j \in N$$

Si definiscono qui di seguito i valori delle variabili:

$$y_i \begin{cases} 1 & \text{se il bin } i \text{ è usato} \\ 0 & \text{altrimenti} \end{cases} \quad (1.9)$$

$$x_{ij} \begin{cases} 1 & \text{se l'item } j \text{ è assegnato al contenitore } i \\ 0 & \text{altrimenti} \end{cases} \quad (1.10)$$

c è un intero positivo

$$w_j \geq c \text{ per } j \in N \quad (1.11)$$

Se l'assunzione (1.11) risulta violata e i w_j sono tutti interi, c può essere sostituito con $\lfloor c \rfloor$; se un item viola l'assunzione (??) l'istanza risulta inaccettabile. Tuttavia non esiste un modo semplice per trasformare una istanza in modo da poter trattare pesi negativi. Per semplicità assumiamo che, per ogni soluzione ammissibile, vengano usati i *bins* di indice più basso (ad esempio $y_i \leq y_{i+1}$ per $i = 1, \dots, n-1$).

L'approccio mediante algoritmi BPP viene trattato in letteratura basandosi sulla approssimazione degli algoritmi stessi e sulle loro performance. In questo campo rivestono particolare importanza gli studi 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 oggetto/item è assegnato al bin 1. Gli items $2, \dots, n$ sono poi considerati al crescere dell'indice, ossia ogni item è assegnato al bin corrente se adatto, altrimenti è assegnato a un nuovo bin che diventa quello corrente. Si deduce che la complessità dell'algoritmo è $O(n)^7$. E' facile infatti provare che, per ogni istanza I del BPP, la soluzione $NF(I)$ fornita dall'algoritmo soddisfa il bound

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

dove $z(I)$ indica la soluzione ottima. Esistono, inoltre, istanze per cui il rapporto $\frac{NF(I)}{z(I)}$ tra le due soluzioni è arbitrariamente fermato a 2: cioè la peggior performance che si possa avere nel rapporto di NF è che $r(NF) = 2$. Si nota che, per un problema di minimo, il peggior caso di un rapporto tra un generico algoritmo A approssimato è definito come il più piccolo numero

⁷in ambito computazionale, in relazione a una teoria della complessità, la notazione detta *big O* è usata per descrivere quanto la dimensione, il peso dell'input influisce sulle risorse impiegate nell'elaborazione di un algoritmo (di solito sui tempi di throughput o dell'occupazione di memoria)

$r(A) \in R$ tale che

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

dove $A(I)$ indica la soluzione ottenuta dall'algoritmo 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 siano $w_1 \leq w_2 \leq \dots \leq w_n$, e che in seguito siano applicati gli algoritmi NF, oppure FF, oppure BF. Gli algoritmi risultanti sono chiamati rispettivamente **Next-Fit Decreasing** (NFD), **First-Fit Decreasing** (FFD) e **Best-Fit Decreasing** (BFD).

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).

Nella tabella riassuntiva di fig. 1.5 sono illustrati i criteri di valutazione dei vari algoritmi, in cui r^∞ è l'*asymptotic worst case performance ratio*⁸; per un generico algoritmo approssimato A , esso è definito come il minor numero reale $r^\infty(A)$ tale che, per un qualche valore intero positivo K , sia $\frac{A(I)}{z(I)} \leq r^\infty(A)$ per ogni istanza I che soddisfi $z(I) \geq K$:

⁸stima asintotica della prestazione nel caso peggiore; il simbolo di infinito sta ad indicare il numero imprecisato di volte per cui viene ripetuta la stima

Algoritmo	Complessità	r^α
<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

Figura 1.5: confronto tra i vari algoritmi e loro performance

Rilassamento lineare

Enunciamo qui di seguito la versione del *continuous problem* riferita al rilassamento lineare nel caso del bin packing. Per il modello in esame il C(BPP) è il seguente:

$$\begin{aligned}
 &\text{minimizzare} \longrightarrow \min(z) = \sum_{i=1}^n y_i \\
 &\text{soddisfacendo} \longrightarrow \sum_{j=1}^n w_j x_{ij} \leq c y_i, i \in N = \{1, \dots, n\}, \\
 &\sum_{i=1}^n x_{ij} = 1, j \in N, \\
 &0 \leq y_i \leq 1, i \in N, \\
 &0 \leq x_{ij} \leq 1, i, j \in N
 \end{aligned}$$

Tale rilassamento può essere facilmente risolto ponendo $x_{ij} = 1$, $x_{ij} = 0$ con ($j \neq j$) e ponendo $y_i = \frac{w_i}{c}, \forall i \in N$. Da qui otteniamo

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

1.2.3 Two Dimensional Bin-packing Problem

Finora si sono classificati i problemi di packing a seconda del numero di dimensioni. In problemi BPP l'attenzione era focalizzata sull'obiettivo di trovare un modo per raggruppare insieme un certo numero di oggetti, impiegando il minor numero di contenitori possibile.

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* per cui 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 del packing si presenta spesso con vincoli lievemente differenti; molte varianti del problema sono state considerate in letteratura.

Contestualmente, quando si ha a che fare con problemi multi-dimensionali, si tende a raggrupparli nel cosiddetto insieme di problemi NP-hard⁹ ossia una particolare classe di problemi non risolvibili mediante algoritmi polinomiali. Ovviamente, più aumentano la complessità delle specifiche e più aumentano i vincoli da soddisfare:

- **Orientamento**

Si assume di solito che ogni rettangolo abbia una certa rotazione fissa, oppure che possa essere ruotato di 90°. 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).

- **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

⁹abbreviazione di *Non-deterministic Polynomial time*; sono problemi decisionali la cui soluzione non può essere determinata mediante l'utilizzo di software e computazioni usuali, ma mediante approcci euristici

contenitore. Ciò è spesso imposto da limitazioni tecniche di macchine da taglio automatiche (ad esempio seghe verticali o circolari) o di alcuni materiali. in figura 1.6 sono illustrati due esempi di piazzamenti di items con o senza il vincolo dei tagli a ghigliottina.

- **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. In figura 1.7 se ne presenta un dettaglio illustrativo.

- **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° , dopo i tagli fatti in una prima fase. Le strisce ottenute nella prima fase sono tutte tagliate nella seconda fase producendo gli items desiderati (illustrazione in figura 1.8). 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.

- **Packing Problem Unconstrained, Constrained o Doubly constrained**

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

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

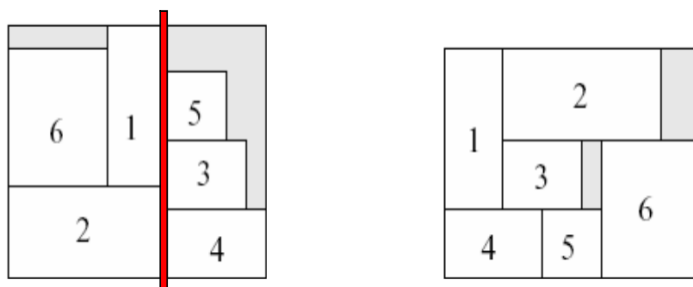


Figura 1.6: taglio a ghigliottina: nel caso a sinistra il taglio, in quello a destra la disposizione originale

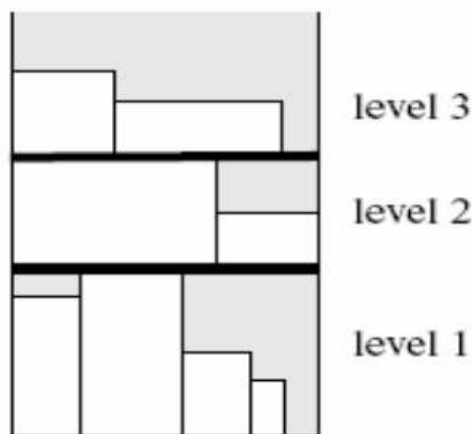


Figura 1.7: level packing

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-line¹⁰ 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 da un'al-

¹⁰Sugli 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

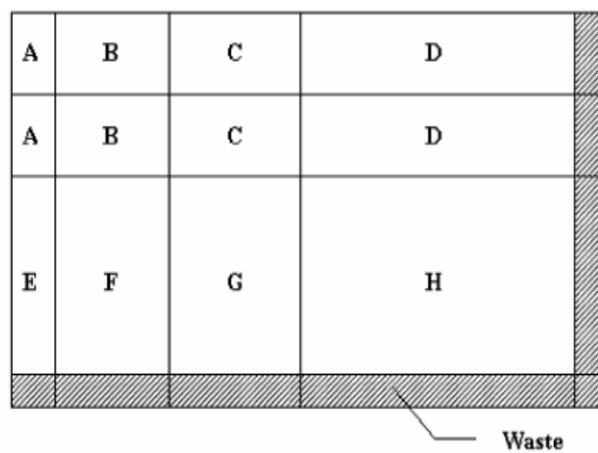


Figura 1.8: checkerboard pattern

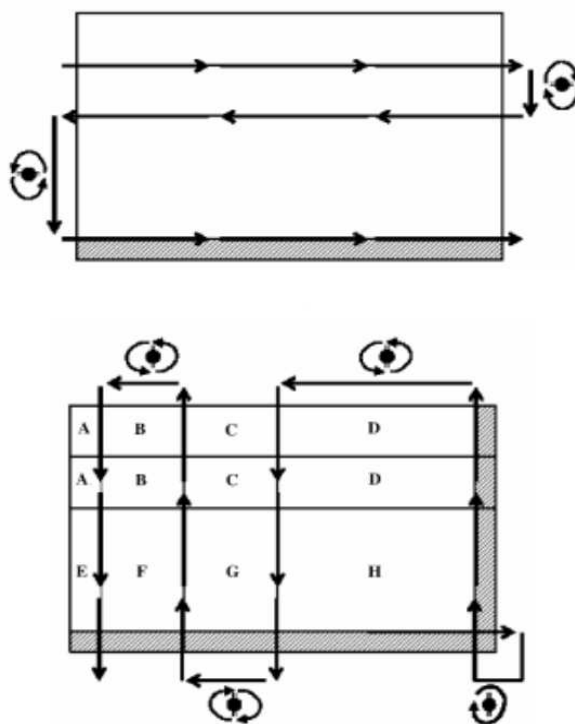


Figura 1.9: esempio di tagli a scacchi; prima e seconda fase

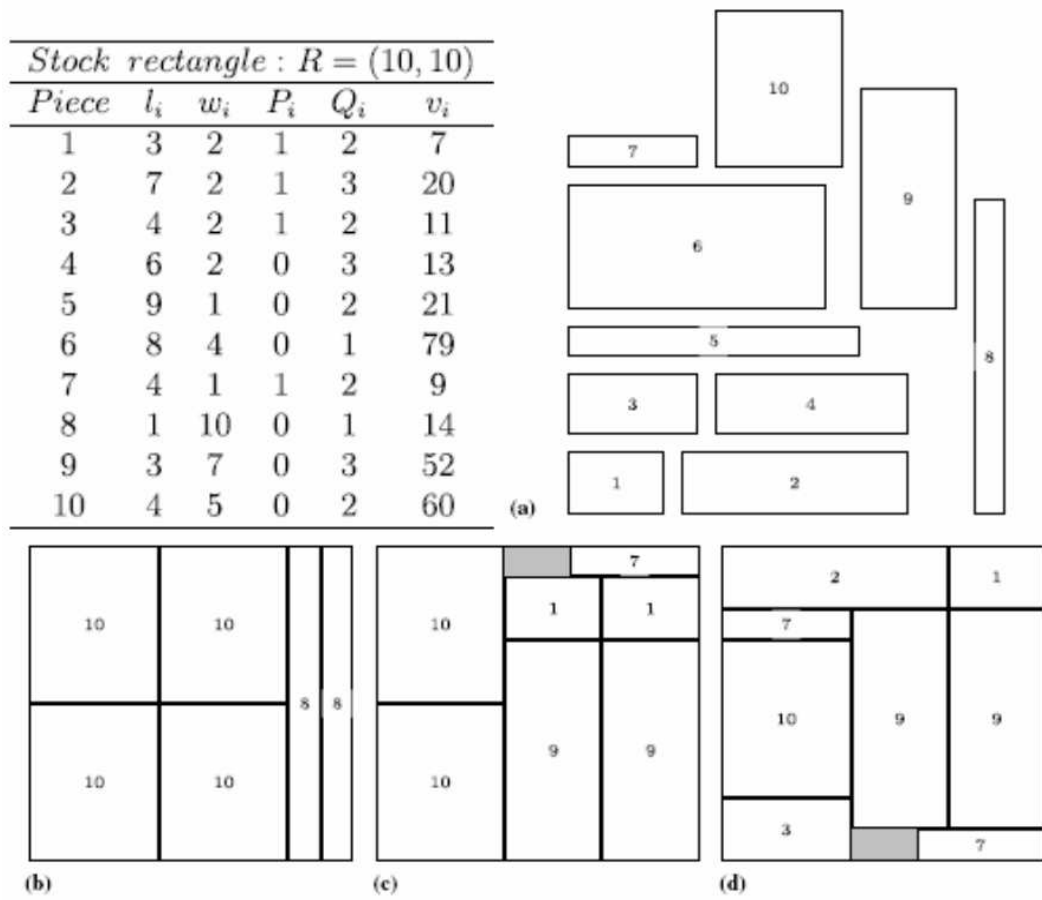


Figura 1.10: esempio di un stock di rettangoli di misura 10×10 in cui collocare i pezzi sopranumerati da 1 a 10. La figura mostra le soluzioni ottime per i problemi unconstrained (b), constrained (c) e doubly constrained (d)

tezza h_j , e un numero illimitato di contenitori rettangolari (*bins*) aventi tutti uguale larghezza W e altezza H .

L'obiettivo, come sempre, è quello di impaccare tutti gli oggetti minimizzando il numero di contenitori. E' richiesto di sistemare gli oggetti ortogonalmente senza sovrapposizioni (il lato w_j dell'item j deve essere parallelo al lato W del container). Si assumerà inoltre, senza perdita di generalità, che i dati di input siano interi positivi, e che $w_j \leq W$, $h_j \leq H$, con $j = (1, \dots, n)$. Questo problema non è altro che una estensione del BPP *one dimensional*, in quanto quest'ultimo è un caso speciale del BPP a due dimensioni, che compare quando $h_j = H$, $\forall j \in N$.

Il primo tentativo di costruire un modello del packing problem a due dimensioni è stato fatto da Gilmore e Gomory (1965), attraverso un' estensione del loro approccio al packing a una dimensione. 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:

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

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

LEVEL PACKING

Molti degli algoritmi approssimati per il 2BPP suppongono di impaccare gli oggetti *a livelli*, in cui il primo livello è costituito dal fondo del contenitore e gli oggetti sono via via impaccati appoggiando la loro base su di esso. Il livello superiore è determinato da una linea orizzontale “disegnata” sull'estremità superiore dell'oggetto più alto del livello sottostante (figura 1.7). Tre classiche strategie per l'impaccamento a livelli sono state rilevate da alcuni famosi algoritmi e in ognuno dei casi 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 possono essere illustrate come segue:

- **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 viene 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

Nella figura 1.11 è possibile vedere le tre strategie.

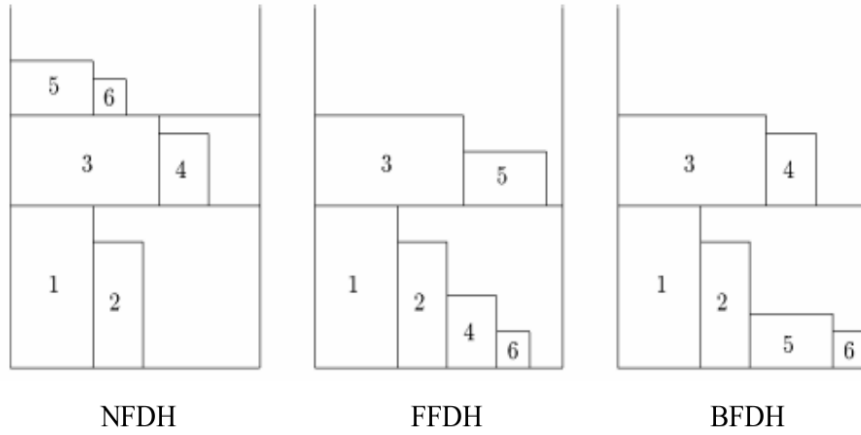


Figura 1.11: problemi e strategie di level packing: da sinistra a destra, NFDH, FFDH e BFDH

TWO DIMENSIONAL LEVEL BIN PACKING PROBLEM

Indichiamo con 2LBP il problema a due dimensioni ristretto al tipo di impaccamento a livelli, supponendo, senza perdere in generalità, che:

1. in ogni livello, l'oggetto a sinistra sia il più alto
2. in ogni contenitore, il livello più basso abbia maggiore altezza degli altri
3. gli oggetti siano disposti e rinumerati per valori decrescenti di h_j (altezza dell'item j -esimo)

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. 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:

$$\min \sum_{k=1}^n q_k \quad (1.12)$$

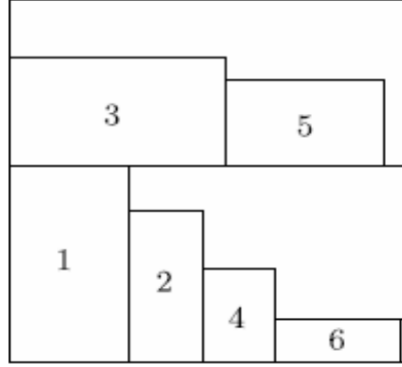


Figura 1.12: Level packing normalizzato secondo il 2LBP

$$\text{subject to } \sum_{i=1}^{j-1} x_{ij} + y_j = 1, \quad j = (1, \dots, n) \quad (1.13)$$

$$\sum_{j=i+1}^n w_j x_{ij} \leq (W - w_i) y_i, \quad i = (1, \dots, n-1) \quad (1.14)$$

$$\sum_{k=1}^{i-1} z_{ki} + q_i = y_i, \quad i = (1, \dots, n) \quad (1.15)$$

$$\sum_{i=k+1}^n h_i z_{ki} \leq (H - h_k) q_k, \quad k = (1, \dots, n-1) \quad (1.16)$$

$$y_i, x_{ij}, q_k, z_{ki} \in \{0, 1\} \quad \forall i, j, k \quad (1.17)$$

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 viene impaccato a livello i (rispettivamente il livello i viene allocato al contenitore k), e valore 0 altrimenti. Inoltre:

- le restrizioni $j > i$ e $i > k$ seguono dalle assunzioni 1 e 3
- le equazioni (1.13) e (1.15) impongono, rispettivamente, che ogni item sia impaccato esattamente una volta, e che ogni livello usato sia assegnato a un unico contenitore

- le equazioni (1.14) e (1.16) 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.

NON-LEVEL ALGORITHMS

Oltre all'impaccamento a livelli, nella Ricerca Operativa ci sono stati altri approcci al problema di packing a due dimensioni. 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., Jackobs (1996) e Liu e Teng (1999). Di questi ultimi l'algoritmo che ha conseguito risultati migliori è stato quello di Baker. 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 viene poi impaccato nella posizione più bassa di ogni contenitore inizializzato, giustificato a sinistra; se nessun contenitore può collocare l'item, ne viene inizializzato uno nuovo. Lodi et al. (1999) hanno proposto l'**alternate directions** (AD), algoritmo che consiste nell'inizializzare 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 figura 1.13.

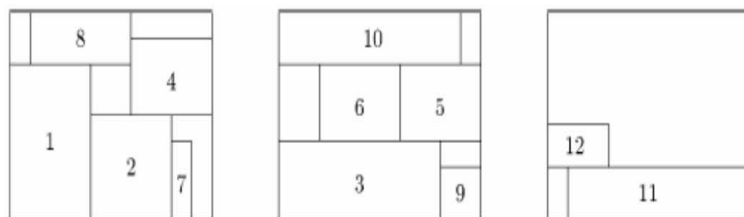


Figura 1.13: Algoritmo AD, Alternate Directions

Beasley (1985) ha considerato il *two dimensional cutting problem* nel quale ad ogni oggetto è associato un profitto, con l'obiettivo di impaccare il sottoinsieme di oggetti aventi profitto massimo in un unico contenitore (*cutting stock problem*). Esso ha una formulazione ILP¹¹ basata sulla rappresentazione discreta dello spazio geometrico e l'uso delle coordinate nelle quali gli oggetti possono essere collocati, vale a dire:

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

Il discorso vale 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, invece, 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. Se prendiamo in considerazione $G_w = (V, E_w)$ (rispettivamente $G_h = (V, E_h)$) come 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'

¹¹Integer Linear Programming, programmazione lineare intera: sono tutti quei problemi lineari che presentano al loro interno solo variabili intere, cioè variabili che possono assumere solo i valori contenuti all'interno del loro dominio di esistenza

stato dimostrato appunto da Fekete e Schepers che se il packing è accettabile allora:

- per ogni insieme S di solidi di G_w (rispettivamente di G_h) si ha che $E_{v_i \in S} w_i \leq W$ (rispettivamente $E_{v_i \in S} h_i \leq H$)
- $E_w \cap E_h = \{\}$

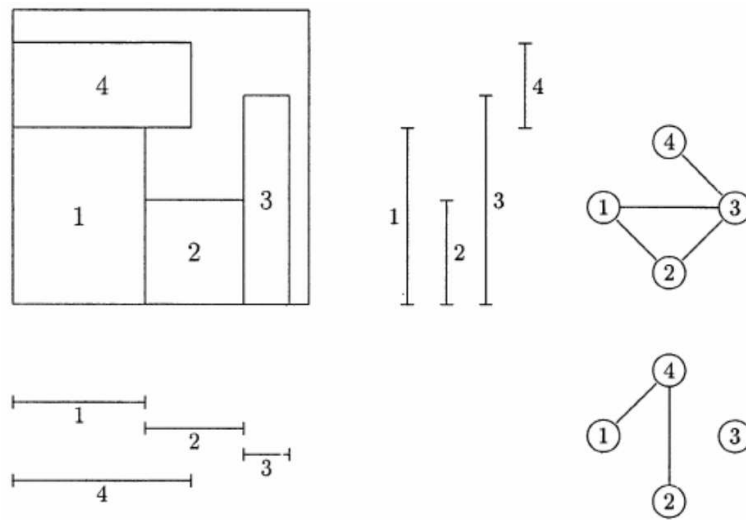


Figura 1.14: Algoritmo di Fekete e Schepers

ALGORITMI METAEURISTICI

L'**euristica**¹² è un metodo di approccio alla soluzione dei problemi che non segue un chiaro percorso, ma si affida all'intuito e allo stato temporaneo delle circostanze, al fine di generare nuova conoscenza. È opposto al procedimento algoritmico e quindi, in ambito informatico, si definisce euristico il lavoro di un software che non opera meccanicamente, ma utilizza invece una tecnica virtualmente creativa, cioè non si limita ad analizzare i dati secondo confronto di dati noti, ma prova a simularne il comportamento¹³.

¹²di derivazione greca, rappresenta una parte dell'epistemologia e del metodo scientifico

¹³tra i più diffusi software di questo tipo, troviamo gli antivirus e le applicazioni contro il *phishing* per cui è fondamentale affidarsi a tecniche non consuete, come invece lo è l'analisi per confronto

Nel settore della Ricerca Operativa, quindi più in generale dove si fondono informatica e matematica, si utilizzano algoritmi euristici come particolari tipi di procedimenti le cui soluzioni non rispecchiano la soluzione ottima per quel dato problema. L'euristica è un approccio di risoluzione dei problemi molto diffuso nella *simulazione* per vari possibili motivi:

- la risoluzione del problema ottimo può essere impossibile
- la risoluzione del problema ottimo può essere troppo costoso in termini di tempo o di capacità di elaborazione

Le tecniche metaeuristiche sono oggi un frequente strumento usato per trovare la soluzione approssimata dei problemi di ottimizzazione combinatoria. Dowsland (1993) presentò uno dei primi approcci metaeuristici al 2BPP. Il suo algoritmo *simulated annealing*¹⁴ esplora sia le soluzioni accettabili che le soluzioni nelle quali gli items si sovrappongono. Durante la ricerca, la funzione obiettivo consiste perciò nell'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 viene trovata, l'upper bound è fissato alla sua altezza.

Un'estensione dell'approccio di Dowsland al 2BP è stato proposto da Faero 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 risulta essere 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

¹⁴processo che mira a trovare un minimo globale quando si è in presenza di più minimi locali. deriva dalla scienza dei metalli, dov'è usato per descrivere il processo di eliminazione di difetti reticolari dai cristalli tramite una procedura di riscaldamento seguita da un lento raffreddamento (molto usato in Microelettronica). Nel nostro caso un difetto reticolare corrisponde ad una combinazione errata di due oggetti.

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:

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

(α rappresenta un peso positivo pre-stabilito). Tale formula fornisce 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.

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*:

- *outer branch-decision tree*: a ogni nodo decisionale, un item è assegnato a un contenitore senza specificare la sua posizione attuale
- *inner brach-decision tree*: è determinato un packing accettabile (se esiste) 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 viene 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, ad 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 viene 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 viene 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*.

ULTIME VERSIONI E APPLICAZIONI

I packing problems a due dimensioni continuano a essere studiati. Ultimamente riveste particolare importanza l'articolo riguardante 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 compare su Operations Research Letters) in cui 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) 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 altre due o più fasi richieste 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 striscie, e ogni taglio su un blocco produce una di queste striscie. L'algoritmo presentato ricorre a una programmazione dinamica per determinare il layout delle striscie per ogni blocco, risolve un problema di tipo 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* (2DVSBPP) 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*.

1.2.4 Two Dimensional Strip-packing Problem

Questo tipo di problema ha come scopo quello di allocare ortogonalmente tutti gli item su di una striscia (assolutamente senza alcuna sovrapposizione) minimizzando l'altezza totale occupata dal packing. Il problema in questione è di tipo NP-hard e quindi la ricerca ha improntato gran parte delle soluzioni basandosi su algoritmi approssimati, i quali si avvicinano molto alla soluzione ottima, ma non garantiscono l'ottimizzazione completa su tutti gli insiemi di dati proposti. Il *two dimensional strip-packing problem* (2SPP) può essere visto come la trasformazione del problema di *bin packing* a una dimensione, dove quest'ultimo ha soluzione "facile". Partendo infatti da una qualsivoglia istanza del 1BPP si ottiene facilmente un'istanza del 2SPP aggiungendo al problema una variabile $h_j = 1$ per $j \in J$: il valore della soluzione ottima di tale istanza è ottima anche per l'istanza del 1BPP.

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.

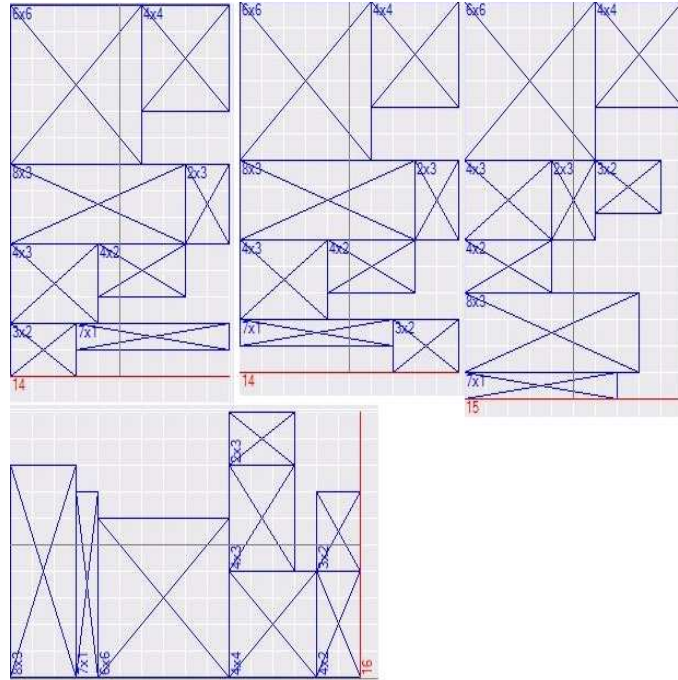


Figura 1.15: alcune soluzioni al 2SPP basate sull'euristica two dimensional variable sized bin packing problem (2DVSBP)

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 $w_j \leq W (j = 1, \dots, n)$. Per determinare la soluzione esatta del 2SPP, sono stati inseriti 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. È 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 $O(|I| \log |I|)$: Lo schema branch-and-bound è stato poi migliorato in due modi: attraver-

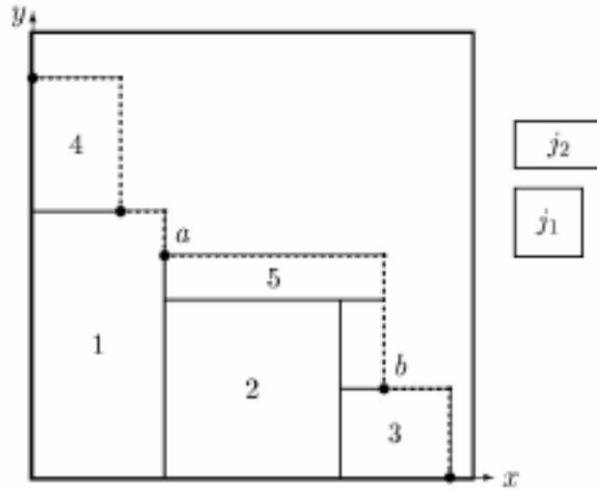


Figura 1.16: involucro associato agli items di $I = \{1, 2, 3, 4, 5\}$ dove i corner points sono indicati con un puntino nero.

so 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 di tipo (corner point, item): (a, j_1) , (a, j_2) , (b, j_1) , (b, j_2) , dove la coppia indica 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 $\{j\} \cup (I \setminus \{i\})$, $\forall i \in I$. 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)...).

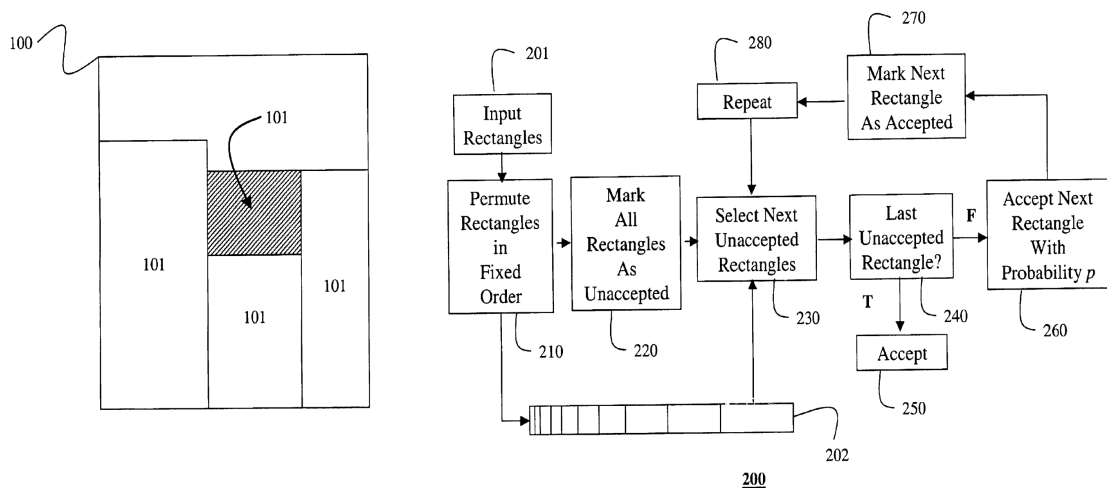


Figura 1.17: illustrazione schematica e molto sommaria del processo strip-packing

ALGORITMI METAEURISTICI

Un approccio metaeuristico interessante è quello presentato da Iori, Martello e Monaci (2003). Il 2SPP viene risolto attraverso una combinazione tra un algoritmo *tabu search* e un algoritmo di tipo genetico¹⁵. L'algoritmo di

¹⁵metodo euristico di ricerca ed ottimizzazione, ispirato al principio della selezione naturale di Charles Darwin che regola l'evoluzione biologica. Gli algoritmi genetici sono applicabili alla risoluzione di un'ampia varietà di problemi d'ottimizzazione non indicati

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 BUILD¹⁶ e TP_{2SP} ¹⁷, 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. È 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 metaeuristici 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 1.1 (al termine del capitolo) è riportato un elenco di alcuni di altri recenti sviluppi metaeuristici per il 2SPP. In particolare in essa sono stati distinti:

- quattro sottotipi di *strip packing problem*:

per gli algoritmi classici, compresi quelli in cui la funzione obiettivo è discontinua, non derivabile, stocastica, o fortemente non lineare.

¹⁶algoritmo 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*

¹⁷inizializza 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 propone: *Iori, Martello, Monaci: Metaheuristic Algorithms for the Strip Packing Problem*

- RF \rightarrow i pezzi possono essere ruotati di 90° ma non sono richiesti tagli a ghigliottina
 - RG \rightarrow i pezzi possono essere ruotati di 90° e sono richiesti tagli a ghigliottina
 - OF \rightarrow l'orientamento dei pezzi è fisso ma non sono richiesti tagli a ghigliottina
 - OG \rightarrow l'orientamento dei pezzi è fisso e sono richiesti tagli a ghigliottina
- tre gruppi di approccio alla soluzione dei metaeuristici (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 viene 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 algoritmi metaeuristici:
 - GA \rightarrow genetic algorithms
 - SA \rightarrow simulated annealing
 - TS \rightarrow tabu search

LEVEL PACKING

Anche per il *two dimensional strip packing problem* come per il BPP esiste un packing a livelli, proposto da Lodi, Martello e Monaci (2002). Tale alternativa consiste nel modificare la funzione obiettivo eliminando tutti i vincoli e le variabili relativi al bin packing problem visto precedentemente, in modo da ottenere il modello seguente:

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

1.2.5 Two Dimensional Knapsack Problem

Consideriamo innanzitutto un problema knapsack di tipo rettangolare¹⁸, ossia un problema a due dimensioni avente come *items* degli oggetti rettangolari.

Sono dati: un insieme di $N = (1, \dots, n)$ piccoli rettangoli (gli items), in cui ognuno ha una larghezza w_j , un'altezza h_j e un profitto p_j con $j \in N$ e un contenitore di larghezza W e altezza H . L'obiettivo consiste, analogamente agli altri casi precedenti, nell'inserire nel container (knapsack) un sottoinsieme di items che massimizzi il profitto; il vincolo è che gli oggetti non si sovrappongano e che ognuno di essi abbia il lato h_j parallelo al lato H del contenitore¹⁹. Si noti che 2KP è una generalizzazione del già citato *1-dimensional Knapsack problem*, visto come caso particolare avente $h_j = H \quad \forall j \in N$.

Anche il *two-dimensional Knapsack Problem* (2KP) è un problema NP-hard e in letteratura sono stati studiati molti metodi per risolverlo, a partire dal

¹⁸d'ora in poi chiameremo tale problema un *rectangle packing problem*

¹⁹si definisce con questi vincoli un tipo di problema denominato *orthogonal packing problem without rotation*

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. Procediamo illustrando un naturale rilassamento del 2KP:

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

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:

$$r(U_{KP}) \doteq \inf \left\{ \frac{OPT(I)}{U_{KP}(I)} \right\}$$

Il worst-case ratio trovato finora è $r(U_{KP}) = \frac{1}{3}$. Tale rilassamento, approfondito da Caprara e Monaci (2003) costituisce una buona base di partenza per quattro nuovi algoritmi per risolvere il 2KP: essi rappresentano l'approccio esatto al problema più recente.

Oltre a Caprara e Monaci (2003), altri studi sono stati condotti a riguardo:

Beasley

Basley (2003) ha presentato un algoritmo euristico di tipo genetico per la risoluzione del problema; 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:

1. si genera una popolazione iniziale

2. si valuta il fitness e l'unfitness degli individui (repeat)
3. si selezionano due individui genitori della popolazione (random)
4. si ricombinano i genitori producendo un solo figlio, attraverso il crossover
5. si muta il figlio; uno ogni dieci figli viene mutato
6. si valuta il fitness e l'unfitness del figlio e miglioralo usando uno schema euristico di miglioramento
7. si sostituisce 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 lo si scarta
8. ci si ferma dopo aver riportato la miglior soluzione incontrata

Hadjiconstantinou and Christofides

Hadjiconstantinou e Christofides (1995) hanno presentato una procedura esatta *tree-search*²⁰ 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 del *tree* usando un bound derivante da un rilassamento lagrangiano della formulazione binaria del problema. Un'ottimizzazione del subgradiente viene 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.Parrenho, J.M.Tamarit

Tali ricercatori presentano nel 2006 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

²⁰ricerca ad albero

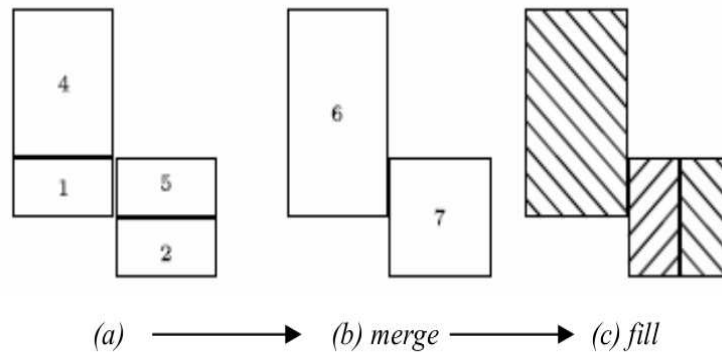


Figura 1.18: strategia Merge and Fill

particolare per i problemi *constrained* e *doubly constrained*.

Hadjiconstantinou, Iori

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.

1.3 Facility layout problem

Il *facility*²¹ *layout problem* (FLP) si riferisce all'allocazione di un certo spazio disponibile a una varietà di attività che hanno diverse relazioni tra di loro. Si tratta di un problema di ottimizzazione combinatoria che si presenta in una moltitudine 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 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.

Lo scopo principale di un algoritmo che risolva le “funzionalità” consiste nell’ “arrangiare” in maniera ottima gli elementi coinvolti nella progettazione del layout (elementi che naturalmente variano in base al contesto) per trovare la via più efficiente di disposizione degli oggetti. La ragione principale, come già detto precedentemente, risiede nel fatto che i costi di *material handling* coprono una buona percentuale dei costi totali di un'impresa, ed è quindi conveniente limitarne lo sviluppo e l'aumento²². Un layout dei macchinari efficiente può arrivare a ridurre questi costi dal 10% al 30% l'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. È per questo che l'impatto strategico della progettazione del layout non dovrebbe essere ignorato.

Negli ultimi anni, molte ricerche hanno proposto una varietà di procedure

²¹ci si riferisce alla funzionalità nel layout di un certo ambito progettuale

²²alcune ricerche hanno stimato che l'8% del prodotto lordo nazionale degli USA è stato speso e continua ad esserlo attualmente per nuovi impianti e attrezzature, e per la manutenzione e la modifica di quelli già esistenti. Altri studi dimostrano invece che i costi relativi agli spostamenti dei materiali coprono il 20-30% dei costi di un'impresa

per risolvere il problema. In particolare si è operato cercando di 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.

1.3.1 Modelli per il facility layout problem

MODELLO QAP

Il *quadratic assignment problem* (QAP) è oggi la formulazione più utilizzata nelle risoluzioni focalizzate sulle funzionalità, pertanto su questo modello verrà posta maggiore attenzione. Tale formulazione 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:

$$\min \sum_{i=1}^n \sum_{j=1}^n \sum_{s=1}^n \sum_{t=1}^n A_{ijst} x_{ij} x_{st} \quad (1.18)$$

$$\text{condizione1} \sum_{i=1}^n x_{ij} = 1, \forall j \quad (1.19)$$

$$\text{condizione2} \sum_{i=1}^n x_{ij} = 1, \forall i \quad (1.20)$$

$$\text{variabili} \rightarrow x_{ij} \in \{0, 1\}, \forall i, j \quad (1.21)$$

dove

$$A_{ijst} \begin{cases} w_{is} d_{jt} & \text{se } i \neq s \text{ oppure } j \neq t \\ F_{ij} & \text{se } i = s \text{ oppure } j = t \\ \infty & \text{se } i = s \text{ oppure } j \neq t \end{cases} \quad (1.22)$$

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

$$x_{ij} \begin{cases} 1 & \text{se l'impianto } i \text{ è allocato alla posizione } j \\ 0 & \text{altrimenti} \end{cases} \quad (1.23)$$

Il vincolo (1.19) è la restrizione che solo un impianto può essere piazzato in una determinata posizione, mentre il vincolo (1.20) 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:

$$\min \sum_{i=1}^n \sum_{j=1}^n \sum_{s=1}^n \sum_{t=1}^n A_{ijst} x_{ij} x_{st} + \sum_{i=1}^n \sum_{s=1}^n c_{is} x_{is} \quad (1.24)$$

Il QAP è un problema NP-hard (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 ha 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

$$y_{ijpq} = x_{ij}x_{pq} \quad (1.25)$$

Christofides et al. (1980) hanno proposto una tecnica di bound fortemente non lineare. 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 ng 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.

Per ulteriori dettagli rimandiamo al survey recente di Burkard, Dell'Amico e Martello [6].

MODELLO GRAPH THEORY

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. Sono state pubblicate numerose relazioni su questo argomento, in cui sono stati analizzati diversi modelli e algoritmi.

MODELLO MIP

Anche metodi basati sul *Mixed Integer Programming* sono stati presi in considerazione per risolvere il FLP. Sin dalla prima formulazione, che ha un

obiettivo basato sulla distanza ed è usato nella rappresentazione di un layout che è una estensione del discreto QAP, successivamente altri studi hanno sviluppato un caso speciale di questo MIP e hanno proposto un algoritmo two-step per risolvere il FLP assumendo variabile l'area che può risolvere il *dynamic facility layout* e rendere rettangolare l'area di tutti i dipartimenti. Inoltre si è 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. Altri studi 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. È 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 ci sono già alternative che propongono un approccio di programmazione matematica per il problema del layout generalizzato.

Un MIP dettagliato, sebbene il suo approccio riservi molte promesse, fornisce al momento soluzioni risolubili solo per FLP di dimensione sei o minori. L'obiettivo è basato sulla distanza rettilinea di flusso e di tempo tra il baricentro e i dipartimenti.

1.3.2 Metodologie di risoluzione

Varie metodologie di risoluzione sono disponibili per risolvere il facility layout problem. Qui di seguito ne sono presentate alcune²³ (una breve panoramica), tra cui le metodologie ad approccio esatto, euristico e metaeuristico, con uno sguardo alle proposte future.

METODI ESATTI

Sono stati usati metodi branch-and-bound per trovare una soluzione ottima del FLP formulato come assegnamento quadratico poiché il QAP coinvolge solo variabili binarie. In letteratura sono però riportate soluzioni ottime per problemi di dimensione fino a 30. Per $n > 30$ diventa impossibile per il

²³sono state tralasciate altre metodologie di risoluzione per il FLP, tra cui menzioniamo gli approcci a *rete neurale* e a *fuzzy logic*

computer risolvere il problema e, di conseguenza, anche un computer molto potente non può affrontare un'istanza di grandi dimensioni.

ALGORITMI EURISTICI

Gli algoritmi euristici possono essere classificati come algoritmi di tipo costruttivo poiché una soluzione viene costruita da uno schizzo, ma anche di tipo migliorativo poiché 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 nella tabella a fine capitolo. 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:

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

dove x_{ij} vale 1 se il dipartimento i è adiacente al dipartimento j , viceversa vale 0.

Il principio di base che sta dietro a questa funzione obiettivo è che il costo 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:

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

dove

X_{ij} è 1 se l'impianto i è assegnato alla posizione j , 0 altrimenti

C_{ik} è il costo per posizionare l'impianto i nella posizione k

D_{jl} è la distanza tra la posizione j e la posizione 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.

ALGORITMI METAEURISTICI

Alcune alternative con approccio metaeuristico, come ad esempio SA e GA²⁴, 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. Studi piuttosto recenti hanno studiato un algoritmo SA per il QAP. Una raccolta molto recente degli articoli basati sugli algoritmi SA è riportata nella tabella di figura 1.19.

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

Figura 1.19: raccolta di articoli sugli algoritmi SA per la risoluzione dei FLP

²⁴Simulated Annealing e Genetic Algorithms

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'adeguatezza degli algoritmi genetici per risolvere il FLP. La tabella di figura 1.20 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

Figura 1.20: raccolta di articoli sugli algoritmi GA per la risoluzione dei FLP

OBIETTIVI FUTURI E NUOVI SCENARI D'AZIONE

Dall'analisi delle molteplici alternative riportate in questo paragrafo, si evince che la ricerca sul FLP non è convergente, ma in qualche modo divergente. Al momento, l'approccio AI²⁵ può essere usato 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. È 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.

²⁵Artificial Intelligence

Authors, source	SPP subtype	Approaches	Metaheuristics
Jacobs (1996)	RF	1	GA
Dagli and Poshyanonda (1996, 2004)	RF	1	GA
Liu and Teng (1999)	RF	1	GA
Hopper and Turton (2000)	RF	1	GA
Hopper and Turton (2000)	RF	1	SA
Hopper and Turton (2000)	RF	1	NE★
Mumford, Valenzuela et al. (2003)	RG	1	GA
Kröger (1993, 1995)	RG	2	Parallel GA
Schnecker (1996)	RG	2	Parallel GA
Ratanapan and Dagli (1997, 1998)	RF	3	GA
Iori et al. (2002) and Monaci (2001)	OF	Hybrid	Hybrid GA+TS

Tabella 1.1: Algoritmi metaeuristici per il 2SPP con pezzi rettangolari; ★ indica NaiveEvolution, ovvero un GA con mutazione ma senza crossover

Capitolo 2

Ottimizzazione dei layout fieristici

Prendendo spunto dalle numerose fonti disponibili sul web o dai documenti a riguardo, si può riassumere brevemente lo scopo primario di una buona progettazione del layout, sia dal punto di vista aziendale che organizzativo (per eventi, fiere, manifestazioni, ecc.).

L'organizzazione vera e propria dello spazio di vendita (layout) prevede il disegno dei percorsi e della circolazione, in cui la sequenza delle aggregazioni merceologiche deve risultare naturale e di facile lettura per la propria clientela-obiettivo. Generalmente si crea un percorso principale, che stimoli il cliente a seguirlo, e spazi di viabilità secondaria, all'interno delle diverse aree del punto vendita. Nel fare ciò ci si deve assicurare che i percorsi siano sufficientemente ampi da garantire il passaggio agevole della clientela e dei commessi, eliminandone i punti morti. La presenza di un'illuminazione studiata e di una segnaletica efficace ma non invasiva deve, inoltre, guidare l'attenzione dei clienti verso i punti di maggiore attrattiva del locale.

In particolare, nel corso del presente capitolo, verrà focalizzata l'attenzione sull'analisi di casi di studio noti, quali le fiere di Fortaleza in Brasile e di Romont in Svizzera. Tali informazioni e dati sono infatti molto utili come modello per lo sviluppo e l'applicazione degli algoritmi visti nel primo capitolo all'ambito fieristico.

2.1 Specifiche e contesti per il progetto

L'obiettivo principale del progetto di un layout fieristico consiste nel posizionare il massimo numero di stand nell'area adibita all'esposizione, soddisfacendo un certo numero di requisiti operazionali (*constraints*). Data una superficie S adibita all'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. Un layout si dice "fattibile" se soddisfa un certo numero di *constraints* di base (tipici di un contesto fieristico) più altri requisiti specifici forniti dall'organizzazione nel caso specifico. Si possono riassumere le condizioni di "fattibilità" del layout nei seguenti punti comuni ad ogni organizzazione fieristica:

- gli stands non possono sovrapporsi
- gli stands devono essere completamente contenuti nella superficie S
- i visitatori (ma anche gli organizzatori) devono poter facilmente accedere agli stands

Oltre a tali requisiti generici, ogni organizzazione ha anche altre necessità proprie, per le quali è opportuno considerare altri constraints da inserire nel progetto. 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 fieristici sembra più vicino a quello del *packing*, in particolare il *two dimensional packing problem*¹. Tuttavia si tratta di un task di grande interesse nel campo dell'ottimizzazione combinatoria, essendo un campo di applicazione piuttosto nuovo; come ulteriore complicazione, rispetto ai casi generali di packing NP-hard, per un contesto fieristico nasce

¹appunto perché si ha a che fare con un problema di packing, allora si inserisce tale problema nell'insieme degli NP-hard (considerando il caso più generale in cui ogni stand può essere ruotato in ogni direzione possibile)

il problema legato alla irregolarità della superficie espositiva. Un esempio di superficie irregolare è quello di figura 2.1:

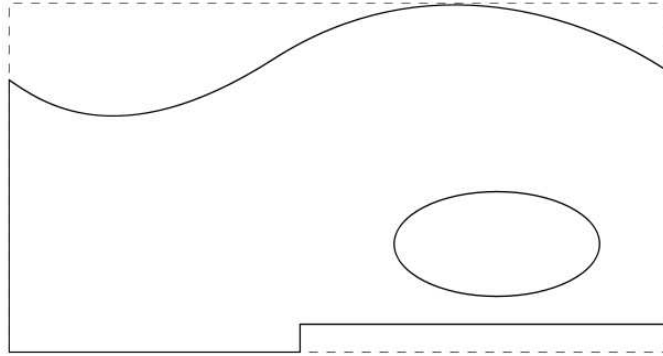


Figura 2.1: tipico esempio di superficie irregolare su cui progettare il layout

Accanto a questo importante e determinante *constraint* le specifiche, piuttosto comuni nella maggioranza dei casi, impongono anche di considerare gli stands orientati tutti nello stesso modo e allineati tutti nella stessa direzione lungo strisce verticali; ovviamente sarà opportuno riservare uno spazio minimo tra tali strisce per permettere un accesso facile alla clientela. Nel progettare un layout siffatto è bene tenere in considerazione tre obiettivi fondamentali, associati ad altrettante funzioni obiettivo:

- l'utilizzo ottimo dello spazio a disposizione
- lo scopo di attirare la maggiore clientela possibile
- la convenienza dal lato cliente

Ovviamente, come già detto in precedenza, si tratta di un tipico problema di 2BPP e di FLP, analizzati nel capitolo precedente. In particolare è opportuno considerare 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.

Riassumendo i principali punti salienti del problema, possiamo descrivere come operare seguendo una serie di accorgimenti e metodi risolutivi che indichiamo come *Fair Layout Optimization Problem* (FLOP). Si considerino le seguenti assunzioni:

assunzione 1 : l'area su cui progettare è una superficie² irregolare, contenuta in un rettangolo di altezza H e larghezza W che la incapsula in modo preciso (vedere le linee tratteggiate di figura 2.1);

assunzione 2 : gli stands sono rettangoli tutti identici e hanno altezza h e larghezza w

assunzione 3 : non sono ammesse rotazioni degli stands (tutti devono essere orientati allo stesso modo)

assunzione 4 : l'allocazione deve essere fatta ortogonalmente su striscie (*strips*) verticali parallele ai lati verticali del rettangolo, garantendo l'accesso da uno dei lati liberi per ciascuno stand

A questo proposito si procederà secondo una duplice variante del FLOP cercando di impaccare il massimo numero di stand nella superficie a disposizione in modo da non violare tutte le considerazioni fatte in precedenza e soddisfacendo le assunzioni appena fatte.

- FLOP 1: viene richiesto che ogni stand (ogni strip) possa essere acceduto da ambo i lati (striscie singole)
- FLOP 2: è possibile disporre gli stand in striscie accoppiate senza spazi tra di esse, in modo che l'accesso agli stand avvenga dal solo lato libero

In figura 2.2 sono illustrate le due varianti FLOP1 e FLOP2 rispettivamente. Senza perdere di generalità, assumeremo che la superficie sia accessibile da tutte le parti. Se si considera l'area espositiva come una matrice binaria M con H righe e W colonne, corrispondente al rettangolo sopracitato, numerando righe e colonne dall'angolo in basso a sinistra della matrice (si considerino

²da qui deriva il fatto che si progetta su due dimensioni

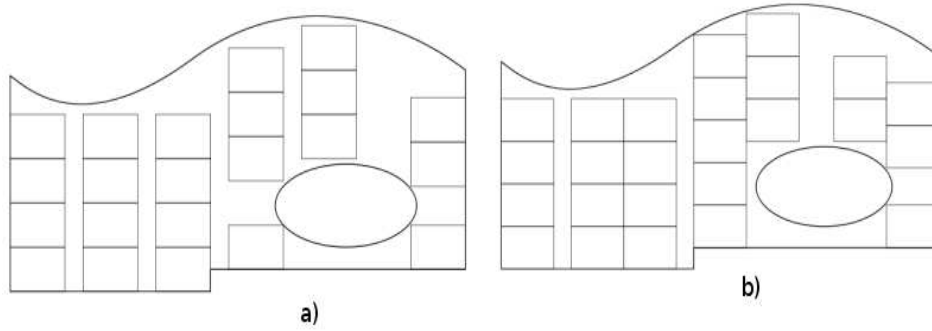


Figura 2.2: varianti del progetto: a) = FLOP1 (a sinistra) e b) = FLOP2 (a destra)

le due dimensioni della superficie come su di un piano cartesiano), si definisce la matrice come:

$$M_{i,j} = \begin{cases} 1 & \text{se lo spazio } \delta \times \delta \text{ alle coordinate } (i,j) \text{ viene allocato per lo stand} \\ 0 & \text{altrimenti} \end{cases}$$

dove, per semplicità si considerano tutte le variabili (altezze, larghezze, distanze, ecc.) come valori interi, espressi secondo una unità minima di lunghezza che indichiamo con δ , mentre $i = 1, \dots, H$ e $j = 1, \dots, W$. Pertanto

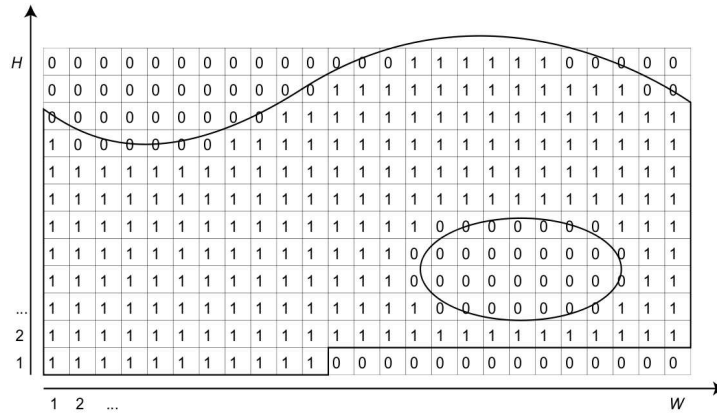


Figura 2.3: matrice M associata all'area di esposizione fieristica di figura 2.1

si seleziona una colonna j per il layout nel caso in cui una striscia di stands (*strip*) sia posizionata sulla colonna stessa col lato sinistro: ad esempio in figura 2.1 e 2.2 la prima colonna viene selezionata come prima strip (dove $w = 3$, $h = 2$). Supponiamo che ciascuna colonna j e ciascun intervallo

massimo di righe $[i_a, i_b]$ di M formino una sottomatrice avente tutti 1 come suoi elementi: tali elementi della sottomatrice ottenuta saranno rettangoli di larghezza w e altezza $(i_b - i_a + 1)$ col proprio vertice in basso a sinistra alla coordinata (i_a, j) . Di conseguenza, il massimo numero di stands che possono essere piazzati all'interno di tale sottomatrice è $\lfloor \frac{i_b - i_a}{h} \rfloor$. In figura 2.3 ad esempio, si contano per $j = W - 3$, due intervalli massimi di righe di larghezza 3: uno alto 2 e l'altro alto 5; la striscia corrispondente può quindi disporre di tre stands in totale.

Per ogni colonna $j \leq W - w + 1$, il massimo numero di stands v_j che possono essere posizionati col loro lato sinistro nella colonna j è ottenuto in maniera iterativa, determinando le successive colonne che soddisfino le condizioni specificate. Una schietta implementazione di questo metodo richiederebbe, per ogni colonna, un numero di iterazioni pari a $O(H)$ di complessità temporale pari a $O(w)$; quindi in totale un tempo $O(WHw)$.

Una procedura più veloce che richiede un tempo $O(WH)$ pari a quello impiegato per “caricare” la matrice M sarà illustrata di seguito; tale procedura calcola le stesse informazioni $v_i (j = 1, \dots, W - w + 1)$.

Algorithm 1 Computation of the v_j values.

Procedure Alloc.Stands

```

for  $i := 1$  to  $H$  do
     $M_{i,W+1} := 0$ ;
    if  $M_{i,1} = 1$  then  $p(i) := \min\{k : M_{i,k} = 0\} - 1$  else  $p(i) = 0$ 
end for;
for  $j := 1$  to  $W - w + 1$  do
     $v_j := k := 0$ ;
    for  $i := 1$  to  $H$  do
        if  $p(i) - j + 1 \geq w$  then  $k := k + 1$ 
        else
             $v_j := v_j + \lfloor k/h \rfloor$ ;
             $k := 0$ ;
            if  $(p(i) < j \text{ and } M_{i,j+1} = 1)$  then  $p(i) := \min\{k > j : M_{i,k} = 0\} - 1$ 
            end if
        end for;
         $v_j := v_j + \lfloor k/h \rfloor$ 
    end for
end

```

Si analizzi l'algoritmo: definiamo, per ogni riga i che sia $M_{i,j} = 1$, un puntatore $p(i)$ all'ultima colonna \hat{j} in modo che sia $M_{i,j} = M_{i,j+1} = \dots = M_{i,\hat{j}} = 1$. Se invece $M_{i,j} = 0$ per la colonna corrente j , allora $p(i)$ avrà un qualsivoglia valore minore di j ; in questo modo tutti gli intervalli massimi di riga possono essere determinati esaminando ogni elemento di M un numero fisso e costante di volte. La variabile contatore k memorizza, per la colonna corrente j e per l'attuale intervallo di riga, il numero di righe che soddisfano le specifiche e che quindi definiamo "idonee".

Il nostro problema di ottimizzazione combinatoria si focalizza perciò nella determinazione delle colonne j che possono essere inserite nell'impaccamento delle *strips* di stands.

2.1.1 Modelli matematici

Nel presente paragrafo si analizzeranno i due modelli matematici corrispondenti alle due varianti di *fair layout optimization problem* introdotte in precedenza. La prima parte, dedicata al FLOP1, non prevede alcuna richiesta riguardante la distanza tra due *strips* diverse di stands; inoltre si fa riferimento alle assunzioni 1-4 viste precedentemente. Consideriamo una variabile binaria

$$x_j = \begin{cases} 1 & \text{se una strip singola di stands ha il proprio lato sinistro sulla colonna } j \\ 0 & \text{altrimenti} \end{cases} \quad (2.1)$$

$$\forall j = 1, \dots, W - w + 1.$$

Il primo problema **FLOP1** può quindi essere modellato come segue:

$$(\text{FLOP1}) \max \sum_{j=1}^{W-w+1} v_j x_j \quad (2.2)$$

$$\text{subject to } \sum_{j=k-a-w+1}^k x_j \leq 1, \quad (k = w + a, \dots, W - w + 1) \quad (2.3)$$

$$x_j \in \{0, 1\}, \quad (j = 1, \dots, W - w + 1) \quad (2.4)$$

dove a rappresenta la larghezza minima del passaggio pedonale tra strips³. Dalla espressione (2.3) si nota che i vincoli impongono che gli stands non si

³che in inglese si indica con *aisle*=*navata* come si può vedere in figura 2.5

sovrappongano tra loro e che ci sia spazio sufficiente per le *aisles*: se una colonna k viene selezionata allora le colonne $(k-a-w+1, \dots, k-1)$ non possono essere selezionate a loro volta, come è possibile constatare dalla figura 2.4. È inoltre possibile notare che le constraints (2.3) per $(k = 1, \dots, w+a-1)$ non sono imposte, bensì derivano dal primo vincolo ($k = w+a$).

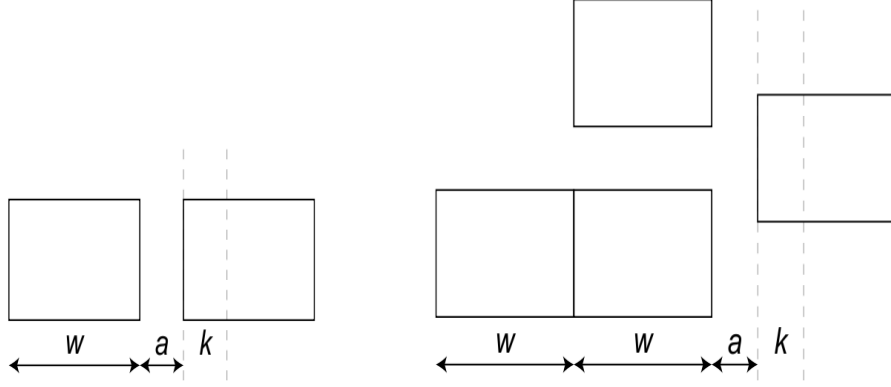


Figura 2.4: constraints per il FLOP1 (sinistra) e per il FLOP2 (destra)

Il secondo problema FLOP2 può essere modellato analogamente all'algoritmo precedente, introducendo un secondo tipo di variabile binaria per ogni colonna j :

$$\xi_j = \begin{cases} 1 & \text{se una strip doppia di stands ha il proprio lato sinistro sulla colonna } j \\ 0 & \text{altrimenti} \end{cases} \quad (2.5)$$

$$\forall j = 1, \dots, W - 2w + 1.$$

Osserviamo che il massimo numero di stands idonei ad essere compresi nella *double-strip* è $v_j + v_{j+w}$.

Ne deriva quindi il modello **FLOP2** seguente:

$$(\text{FLOP2}) \max \sum_{j=1}^{W-2w+1} (v_j + v_{j+w}) \xi_j \quad (2.6)$$

$$\text{subject to} \quad \sum_{j=k-a-w+1}^k x_j + \sum_{j=\max(1, k-a-2w+1)}^{\min(W-2w+1, k)} \xi_j \leq 1, \quad (k = w+a, \dots, W-w+1) \quad (2.7)$$

$$x_j \in \{0, 1\} \quad , \quad (j = 1, \dots, W - w + 1) \quad (2.8)$$

$$\xi_j \in \{0, 1\} \quad , \quad (j = 1, \dots, W - 2w + 1) \quad (2.9)$$

Analogamente al caso del FLOP1, qui i vincoli (2.7) impongono che le striscie (single e doppie) di stands non si sovrappongano e che ci sia, come al solito, spazio sufficiente per consentire il flusso dei clienti: se una colonna k viene selezionata (sia se singola sia se doppia), allora le colonne $k - a - w + 1, \dots, k - 1$ non possono esserlo altrettanto (nel caso di una *single-strip*), mentre le colonne $k - a - 2w + 1, \dots, k - 1$ non possono esserlo altrettanto (nel caso di una *double-strip*). Si può notare che gli indici *max* e *min* nella seconda sommatoria della equazione (2.7) escludono l'indice j poichè è fuori dal range. Nella parte destra di figura 2.4 si può osservare l'insieme di constraint utilizzate per il FLOP2, mentre nelle figure 2.5 e 2.6 è possibile vedere un esempio di layout che tiene conto di entrambe le varianti FLOP1 e FLOP2.

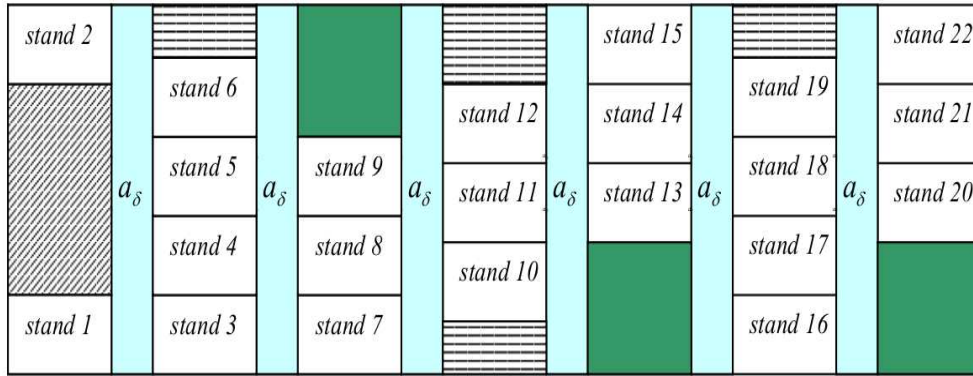


Figura 2.5: esempio di layout che tiene conto delle distanze tra le aisles per i pedoni e tra strips (a_δ). Gli spazi colorati indicano zone in cui non risulta possibile posizionare stand.

Infine, analizzando le due alternative presentate, si deduce una fondamentale proprietà:

le matrici dei vincoli di tutte e due le varianti sono totalmente
unimodulari;

dove per “matrice unimodulare” si intende una matrice A che soddisfi i seguenti requisiti:

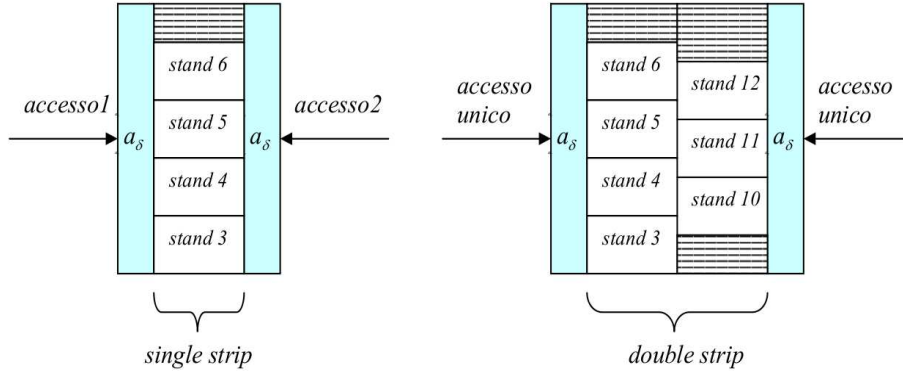


Figura 2.6: differenze tra FLOP1 e FLOP2: nel primo caso ogni stand ha due accessi e vi sono strip singole; nel secondo caso ogni stand ha un solo lato di accesso libero ed è disposto su strisce doppie.

- $a_{ij} \in \{-1, 0, 1\}$
- ciascuna colonna non deve avere più di due elementi non nulli
- 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 ad A_2 se e solo se sono concordi in segno

Se chiamiamo A la matrice associata al vincolo (2.3) si può facilmente constatare che A è proprio *unimodulare*: i coefficienti di ogni riga i (sempre riferendosi ai constraint (2.3)) sono tutti zero tranne un insieme di $w + a$ valori unitari consecutivi che partono proprio da i . Questo insieme di elementi unitari è un gruppo di elementi w_δ , ossia di larghezza w e unità di misura $\delta \approx 20cm$ (in generale). Ora sia a'_i il vettore associato alla riga i di A , per $i = 1, \dots, W - w + 1$, e si modifichi A ponga $a'_i = a'_i - a'_{i+1}$, per $i = 1, \dots, W - w + 1$. In questo modo, partendo da A si ottiene la matrice \tilde{A} di Figura 2.7. E' facile notare che la matrice A soddisfa le condizioni sufficienti per essere una matrice totalmente unimodulare. In particolare la colonna avente più di due elementi non nulli può essere divisa in $A1 = A$ e $A2 = \emptyset$. Da ciò segue che entrambi i problemi FLOP1 e FLOP2 possono essere risolti in un *polynomial time*⁴ sostituendo i vincoli (2.5) (corrispondenti ai vincoli

⁴si fa riferimento a un problema risolubile in tempi computazionali, mediante l'associazione della complessità del problema a una funzione polinomiale

caratteristiche tipiche del contesto, in modo da poter far fronte ad un largo insieme di specifiche, requisiti e vincoli, al fine di creare un progetto completo e adeguato ad ogni evenienza.

2.2.1 Caso 1: Fortaleza handcraft fair

Ogni giorno circa seicento artigiani si recano all'altezza della spiaggia di Meireles e assemblano e smontano i loro stand per dare vita a una famosa fiera di artigianato che da anni è presente sul lungomare di Fortaleza (Brasile).

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 consiste quindi nel 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.

In figura 2.8 è disponibile una piantina che ritrae la superficie di esposizione della fiera di Fortaleza.

L'algoritmo corrispondente al problema di programmazione lineare della fiera è stato implementato utilizzando un software free dotato di interfaccia grafica semplice e intuitiva che potesse permettere di facilitare le operazioni

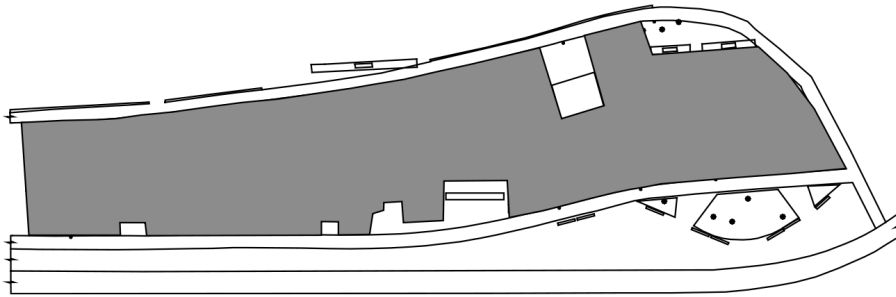


Figura 2.8: vista dall'alto della zona dedicata all'esposizione fieristica (indicata in grigio) in località Beira Mar, a Fortaleza

di disegno del layout⁵.

Le dimensioni dell'area che ospita il layout sono circa di 200 metri di larghezza (che indichiamo, coerentemente con quanto visto nelle sezioni precedenti con W) e 50 metri di altezza (che indichiamo con quindi con H). L'analisi, risalente al 2006, conta 617 stands, posizionati in 26 strisce doppie, con arrangiamento piuttosto simile a quello pensato per il FLOP2, mentre per quanto riguarda l'altezza e la larghezza degli stand, vale per tutti $h = 2$ e $w = 2$ metri. E' importante sottolineare che l'area è di tipo non convesso, ossia c'è la presenza di alberi, piccoli negozietti, lampioni, monumenti, per cui alcune aree non sono disponibili per l'installazione di stands. Nel caso in questione, definendo $\delta = 5cm$ si ottengono $w = h = 40$, $W = 4090$ e $H = 1084$; il valore indicativo fornito per determinare la larghezza del passaggio pedonale (*aisle*) è di $3.5m$, ovvero $a = 70$.

I passi effettuati per testare l'efficienza degli algoritmi FLOP1 e FLOP2 consistono in operazioni piuttosto complicate e laboriose, soprattutto nel passare dalla mappatura reale a quella virtuale (per lo scopo sono stati utilizzati strumenti di CAD) e per convertire il tutto in formato SHP⁶ contenente le "georeferenze" e le coordinate della superficie espositiva. A sua volta il file SHP è stato convertito in una matrice binaria M di dimensione 1084×4090

⁵tutte le simulazioni e i test sono stati eseguiti su macchine con processore Pentium 4 a 1.8GHz dotate di 2GB di RAM.

⁶shapefile, formato vettoriale per sistemi informativi geografici che traduce le informazioni grafiche in dati geometrici ricchi di attributi di corredo alle forme poligonali che determinano le elaborazioni

(comune ai due modelli), mediante un programma chiamato Delphi Object Pascal⁷; questa conversione ha richiesto un tempo computazionale molto lungo, circa 1000 secondi di elaborazione da parte della CPU.

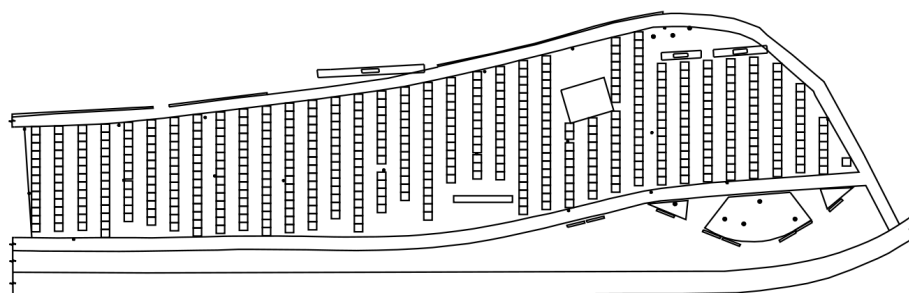


Figura 2.9: output relativo alla soluzione mediante algoritmo FLOP1.

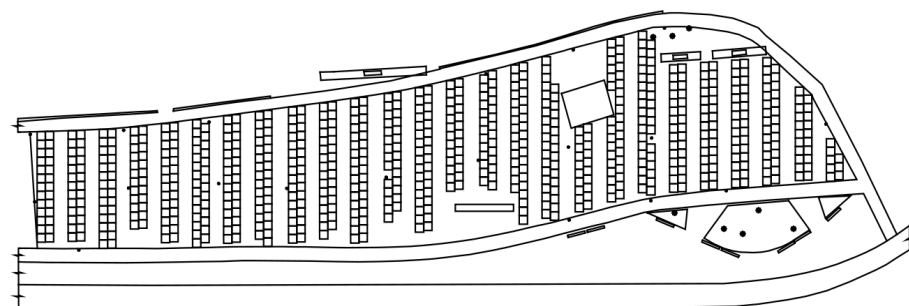


Figura 2.10: output relativo alla soluzione mediante algoritmo FLOP2.

Il modello FLOP1, la cui soluzione è rappresentata in figura 2.9, è stato elaborato e risolto in 24 secondi e la soluzione ottimizzata ha fornito come output un totale di 505 stands disposti su 36 file singole (le *strips*) con meno del 20% di decremento rispetto al caso di partenza (a striscie doppie). Il modello FLOP2, invece, è stato risolto dal software in 62 secondi e la soluzione, in figura 2.10, ha calcolato 742 stands arrangiati in 26 *double strips*, incrementando del 20% la soluzione di partenza.

Stando ad esigenze ulteriormente stringenti, alcuni organizzatori hanno richiesto anche di sfruttare la zona in alto a destra (di forma grosso modo triangolare) e si sono ottenute altre soluzioni, impiegando 759 stands sempre

⁷della software house “Borland”

su 26 striscie doppie (quindi sempre riferendosi al FLOP2). Nessuna delle soluzioni proposte dal FLOP2 ha avuto modo di combinarsi con quelle proposte dal FLOP1 e cioè non vi è modo di combinare striscie singole e doppie, probabilmente a causa del fatto che le *aisles* pedonali sono relativamente piuttosto larghe, e non conviene uscire dalle alternative trovate.

Una ulteriore possibilità per ottimizzare ancora di più le prestazioni computazionali consiste nel cambiare i valori di δ , che non ottimizzano le questioni di spazio, ma di tempo. Ad esempio per $\delta = 10cm$ l'algoritmo ha bisogno di 246 secondi per costruire la matrice M , di 4 secondi per risolvere il FLOP1 e di 10 secondi per quanto riguarda il FLOP2. Inoltre, nel primo caso si contano 503 stands invece che 505, mentre per il FLOP2 si hanno gli stessi 742 stands che in precedenza. Per $\delta = 20cm$ si hanno 64 secondi per costruire la matrice, e 2 secondi per FLOP1 e 1 secondo per FLOP2 con un ulteriore decremento degli stands, rispettivamente a 499 e 737. Per valori sempre più piccoli di δ , fino a $\delta = 5cm$, si sono ottenuti tempi di calcolo sempre più elevati, senza però alcun beneficio in ottimizzazione.

Il campo che interessa i layout fieristici ha assunto notevole importanza soprattutto negli ultimi anni, ed è diventato di fondamentale importanza in materie come la Ricerca Operativa e in applicazioni matematiche e informatiche. Tuttavia è un problema in piena evoluzione e in continua fase di studio per l'ottimizzazione combinatoria, perciò come tale richiede continui adattamenti e revisioni in modo da poter soddisfare un numero sempre più ampio di requisiti e coprire in larga scala tutte le eventualità. A titolo di esempio si consideri proprio il caso di Fortaleza, in cui gli organizzatori hanno proposto nuove alternative e introdotto nuove *constraints* al problema: nel caso appena analizzato, una delle specifiche consisteva nell'arrangiare le strips in modo perpendicolare alla spiaggia (la base della piantina); ma è possibile "comporre" gli stands in altro modo, producendo soluzioni migliori o adatte alle varie esigenze che cambiano di volta in volta. Considerando un approccio tipico dei layout generici, non sarebbe insensato disporre gli stands a gruppi di 4, come rappresentato in figura 2.11.

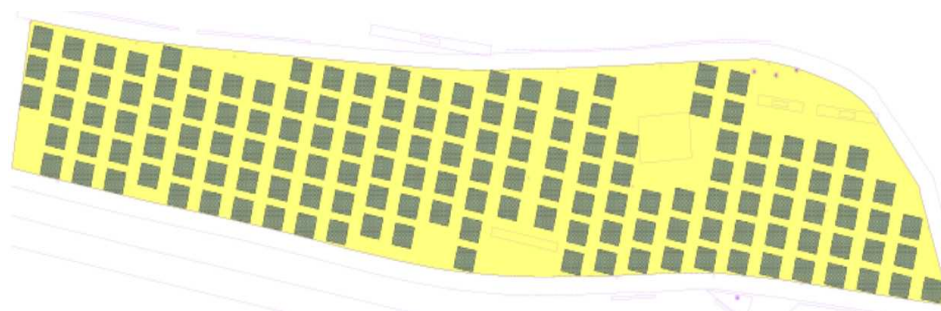


Figura 2.11: bozza alternativa alle varianti previste per la soluzione; si dispongono gli stands a gruppetti di quattro.

2.2.2 Caso 2: Regional fair in Romont

La fiera regionale di Romont ha luogo ogni due anni in un paese di 4 mila abitanti circa, a metà strada tra Ginevra e Berna, nella Svizzera francofona. La zona adibita alle esposizioni è di circa 5200 mq (le dimensioni precise sono di $W = 100$ metri di lunghezza e $H = 52$ metri di larghezza del rettangolo) e al suo interno si presentano merci e servizi di almeno un centinaio di espositori. Nel 1998 si sono registrati circa 50 mila visitatori durante gli 8 giorni dedicati alla manifestazione. Sotto quest'ottica gli organizzatori hanno necessitato di provvedimenti per ottimizzare il layout e far fronte ad eventuali aumenti dei visitatori. Di solito, l'approccio utilizzato dall'organizzazione consiste nel partire da una lista degli espositori e delle loro richieste in termini di spazi, per poi adeguare il layout soddisfacendo un cliente alla volta; un approccio alternativo, invece, prevede di partire da un layout predefinito e di assegnare a ciascun espositore una superficie che ne soddisfi le esigenze.

Si può suddividere il processo di sviluppo delle soluzioni in due branchie: allocare in primis lo spazio per gli stands, in maniera tale da non sovrapporre gli spazi espositivi; in secondo luogo allocare gli spazi per il flusso dei visitatori (ciò che di solito indichiamo con *aisles*). Vediamo in dettaglio questi due punti.

- Poiché una delle caratteristiche portanti della fiera di Romont è che non si mantiene tale e quale alle edizioni precedenti, i progettisti del layout devono ogni volta adattarsi alle nuove disposizioni ed esigenze

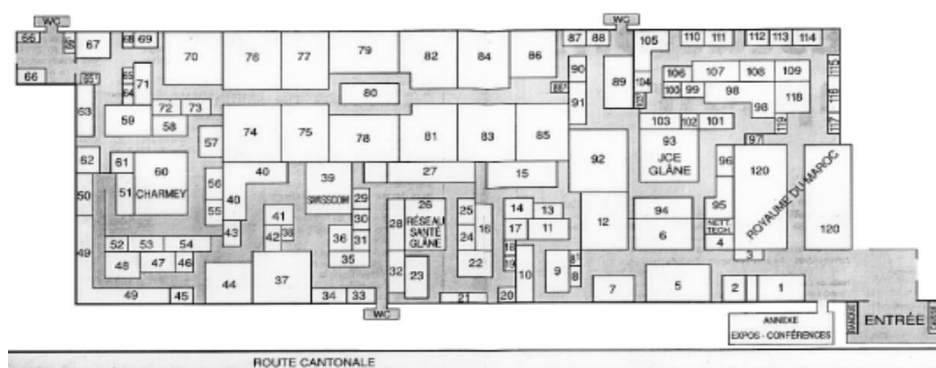


Figura 2.12: disposizione degli stands nell'edizione del 1998 della fiera.

e quindi riproporre ogni anno soluzioni diverse: il tutto si traduce in una raccolta di requisiti che comprende la prima parte del progetto. In particolare si raccolgono informazioni su:

- la forma e le dimensioni degli stands (se rettangolari o ad angolo⁸, con le relative misure)
- i lati di accesso agli stands
- Il flusso di persone che può transitare all'interno degli spazi adibiti può essere elevato ed è quindi necessaria un'efficace affluenza nel percorso riservato alla clientela. La peculiarità che si riscontra nella fiera di Romont sta nel fatto che il cliente deve seguire un percorso guidato e obbligato tra i vari stand, in modo da poterli visitare tutti⁹. Nel caso in esame lo spazio dei corridoi va da un minimo di 2 metri ad un massimo di 3

In aggiunta a queste caratteristiche vanno aggiunti altri vincoli che sono rappresentati da ostacoli all'interno dello spazio, ossia presenza di bagni e toilettes, di muri, colonne e barriere architettoniche, di uscite d'emergenza, ecc., seppur comunque il carico di lavoro da affrontare si basa su tre fattori determinanti:

⁸che possono essere piazzati in un angolo dell'area espositiva

⁹questa strategia è tipica anche in altri ambiti, quali ad esempio nei discount o in supermercati (vedi IKEA), ma è un caso piuttosto singolare quando si tratta di fiere regionali

utilizzo degli spazi : è l'aspetto più importante, dal quale derivano gli altri due. Si tratta dell'ottimizzazione di tutti gli spazi possibili, nel senso che ogni spazio inutilizzato (*dead space*), va comunque “reso utile” e adibito agli stands

valore attrattivo : rappresenta la “capacità” della fiera di attirare il maggior numero possibile di clienti e visitatori. Non dipende solamente da fattori quali marketing o promozioni, ma anche, e soprattutto, da un progetto accorto. Ciascuno stand deve avere una propria locazione il più attraente possibile: il caso ideale sarebbe di disporli in modo che l'*open side* di ciascuno di essi si affacci sul percorso dei visitatori

agevolazioni ai visitatori : naturalmente una qualsivoglia manifestazione, mostra o esposizione fieristica deve essere comoda da visitare e piacevole da vedere. Deve prevedere accorgimenti mirati alla comodità del cliente in modo da invogliarne e stimolarne la curiosità; bisogna quindi tenere conto dei percorsi (magari intervallando con aree di sosta e minimizzando i cambiamenti di direzione, per non affaticare la visita), degli spazi e del contesto in cui ci si trova

È ovvio che risulta impossibile soddisfare appieno tutte e tre le specifiche, perciò occorre cercare un trade-off tra di esse in modo da adeguare il progetto alle esigenze di organizzatori, espositori e visitatori. Generalmente tale compromesso dipende dall'abilità e dall'esperienza del progettista.

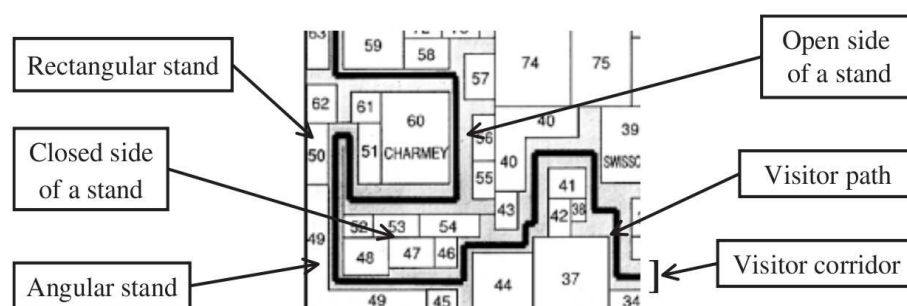


Figura 2.13: descrizione degli items utilizzati nel progetto e legenda.

Il modello cui ci si è attenuti nel progettare il layout è l'*adjacency model*, che

permette una rappresentazione basata sull'adiacenza del percorso dei visitatori ai lati aperti degli stand espositivi; viene usata una discretizzazione degli elementi coinvolti nel processo di layout e questo significa che la superficie e gli spazi richiesti sono espressi come valori interi e multipli dell'unità base dell'area, mentre la superficie espositiva è associata ad un rettangolo suddiviso in alcune unità di misura di base, come si può vedere dalle figure seguenti. In particolare la lunghezza del rettangolo è di L unità, mentre la altezza è B unità. Ogni punto all'interno del rettangolo può essere identificato mediante le coordinate sugli assi cartesiani come illustrato in figura 2.14.

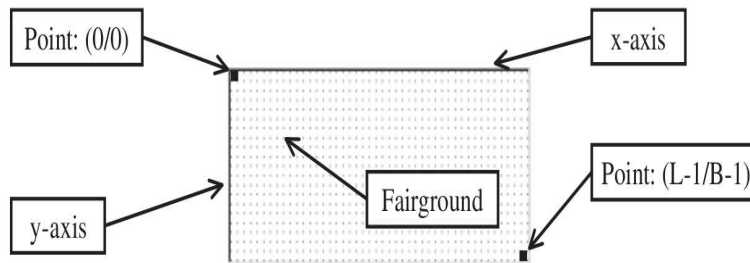


Figura 2.14: fairground relativo allo spazio espositivo, associato ad un rettangolo su un piano cartesiano.

Per quanto riguarda gli stands, essi sono rappresentati come aree rettangolari o angolari (ad angolo) e le diverse tipologie di stand sono illustrate in figura 2.15; tali spazi espositivi possono essere posizionati verticalmente o

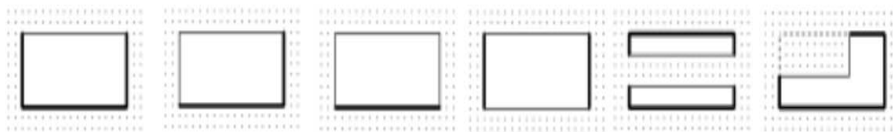


Figura 2.15: tipologie di stands/moduli impiegati nella fiera.

orizzontalmente, a seconda delle esigenze, avendo perciò fino a 4 possibili orientamenti dello stand¹⁰. In figura 2.16 è possibile vedere come viene posizionato uno stand ad angolo all'interno della superficie, assieme ad una lista di valori che esprimono dimensioni e coordinate per il posizionamento.

Si ha invece, per quanto concerne il percorso riservato ai visitatori, che il path

¹⁰supponiamo che le rotazioni possano avvenire ogni 90°

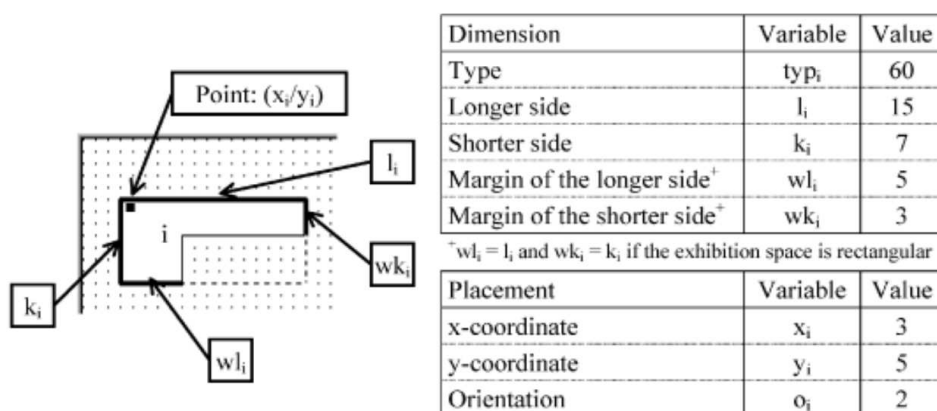


Figura 2.16: posizionamento di uno stand angolare i sul fairground.

non presenta alcun tratto curvilineo ed è quindi composto da tratti orizzontali e verticali (solo rettilinei) che devono essere paralleli agli assi cartesiani del rettangolo. Come si può vedere dalla figura 2.17, il quadrato nero indica un cambio di direzione del percorso, uno snodo che si può facilmente referenziare nell'algoritmo, perché corrisponde esattamente ad una unità base di misura.

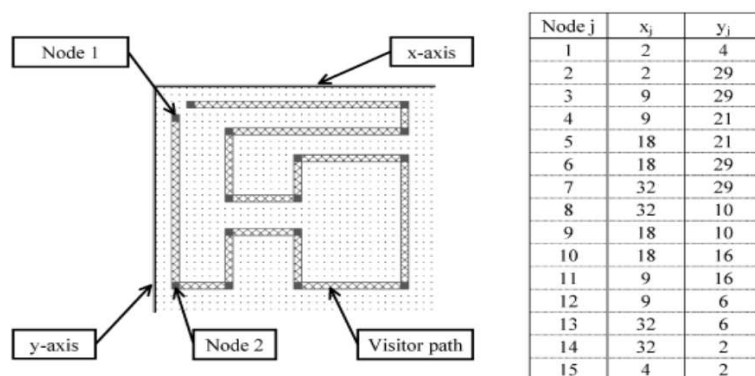


Figura 2.17: percorso per i visitatori e relative coordinate.

L'*adjacency model* implica la formulazione di tre criteri quantitativi per misurare la qualità di un layout progettato bene; tali criteri derivano dai tre fattori visti in precedenza:

utilizzo degli spazi : definiamo il coefficiente r che esprime il rapporto tra la superficie rettangolare (il fairground) e la somma delle aree occupate

dai padiglioni,

$$r = \frac{\sum_{i=1}^N ((k_i \times l_i) - [(k_i - wk_i) \times (l_i - wl_i)])}{L \times B},$$

dove N rappresenta il numero di stands, mentre L e B sono rispettivamente la lunghezza e la larghezza del fairground.

Notiamo che per padiglioni di forma rettangolare si ha $k_i = wk_i$ per cui il secondo addendo della sottrazione si annulla e quindi per questo caso la superficie viene calcolata solo tramite $k_i \times l_i$

valore attrattivo : tale requisito si misura grazie al calcolo dell'adiacenza $a_i = \frac{b_i}{q_i}$, dove il numeratore rappresenta la lunghezza del padiglione i -esimo riferito al percorso della clientela e il denominatore indica la lunghezza dell'open space dello stand i -esimo. Due esempi autoesplicativi del calcolo delle adiacenze sono illustrati in figura 2.18. Indipendentemente dal fatto che la forma dell'item sia rettangolare o angolare, l'adiacenza viene calcolata considerando valori tra 0 e 1, che indicano la percentuale con cui la lunghezza dell'open side combacia col percorso. Se l'indice di adiacenza a_i è alto, il valore attrattivo diventa importante, questo perché si alzano le probabilità che i visitatori si interessino al padiglione i -esimo

agevolazioni ai visitatori : questo fattore è piuttosto difficile da quantificare, perciò è stato adottato un criterio che consiste nel considerare due stands che si affacciano sullo stesso tratto di percorso. Introduciamo un coefficiente f tale che

$$f = \frac{\sum_{i=1}^N b_i}{2 \times g},$$

dove g indica la lunghezza del path dei visitatori.

Per varie ragioni il valore del coefficiente f si preferisce che sia alto per quanto riguarda i visitatori

Applicando queste soluzioni ad un caso reale (l'edizione della fiera del 2000) si sono riscontrati risultati soddisfacenti e il layout ottenuto è stato calcolato in base a questi tre criteri di progetto. In particolare si evidenziano i fattori $f = 80\%$ e $\bar{a}_i = 84\%$ dove quest'ultimo rappresenta l'indice di adiacenza



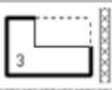

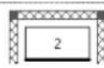

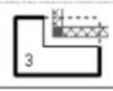
	Exhibition space 1 ($k_1=l_1=4$)	Exhibition space 2 ($k_2=4, l_2=8$)		Exhibition space 3 ($k_3=6, l_3=9, w_{k_3}=w_{l_3}=3$)
Visitor path 1	 $a_1 = \frac{b_1}{q_1} = \frac{4}{12} = 0.33$	 $a_2 = \frac{b_2}{q_2} = \frac{8}{16} = 0.5$	Visitor path 1	 $a_3 = \frac{b_3}{q_3} = \frac{3}{9} = 0.33$
Visitor path 2	 $a_1 = \frac{b_1}{q_1} = \frac{12}{12} = 1$	 $a_2 = \frac{b_2}{q_2} = \frac{16}{16} = 1$	Visitor path 2	 $a_3 = \frac{b_3}{q_3} = \frac{6}{9} = 0.66$
			Visitor path 3	 $a_3 = \frac{b_3}{q_3} = \frac{7}{9} = 0.77$

Figura 2.18: calcolo delle adiacenze: nel primo caso sono calcolate quattro combinazioni in base all'entità degli stand e del path, mentre nel secondo ne sono calcolate tre, con un solo tipo di stand ma con tre tipologie di percorso.

medio riscontrato. Come è illustrato in figura 2.19, gli stands più chiari (la maggioranza) corrispondono ad un *adjacency index* tra 0 e 0.5, mentre le zone più scure vanno da 0.5 a 1.

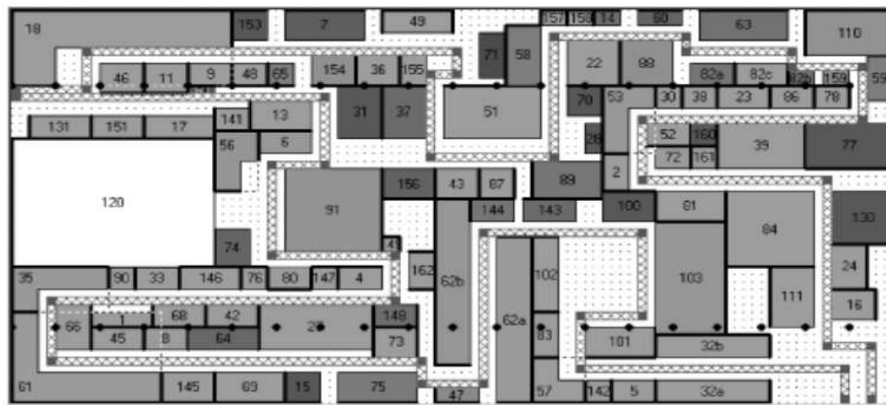


Figura 2.19: risultato finale dell'*adjacency model* applicato all'esposizione del 2000.

Basandosi perciò sull'*adjacency model* possiamo costruire un modello di layout partendo da una configurazione iniziale che consiste nel posizionare uno spazio di seguito all'altro accanto al percorso prestabilito per i visitatori, dove quest'ultimo viene già fornito all'inizio assieme ad alcuni stand che sono

già disposti per motivi molteplici. A questo proposito sono stati sviluppati tre algoritmi per procedere al piazzamento degli spazi espositivi della fiera di Romont.

I dati di input degli algoritmi sono costituiti da dati concernenti le dimensioni degli stands e il layout iniziale. In particolare intendiamo per layout iniziale la configurazione che include le dimensioni $L \times B$ del fairground, le coordinate (x_i, y_i) dei nodi del percorso e i dati sul posizionamento degli stand rettangolari e angolari (o_i, x_i, y_i) .

In figura 2.20 è raffigurato un layout iniziale della fiera di Romont, in cui sono dati il path e gli stand pre-posizionati e in cui vanno installati tutti i rimanenti spazi espositivi, raggruppati nella parte sinistra dell'immagine. Come dati iniziali vengono quindi forniti $g = 508$ e $M = 35$, ovvero la lunghezza del path in unità di area e il numero di nodi, rispettivamente. L'algoritmo non fa altro che scegliere gli items rettangolari, coerentemente coi criteri adottati, e piazzarli nel fairground finchè non si raggiunge un layout finale. L'output dell'algoritmo sarà costituito perciò dai dati relativi agli stands posizionati, quindi (o_i, x_i, y_i) , e dal numero di stands che non è stato possibile includere.

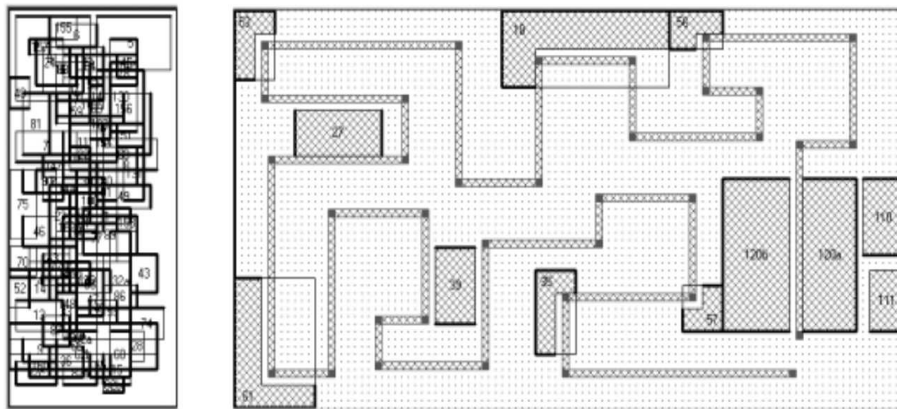


Figura 2.20: layout di partenza della fiera di Romont dove si vedono il path e alcuni stand preposizionati, mentre a sinistra sono raggruppati gli altri stand da includere nel progetto.

Diamo adesso alcune definizioni che ci permettono di proseguire con l'analisi:

- definiamo *path section* una porzione di percorso delimitata da due nodi

e quindi in maniera più semplice un segmento di percorso; se consideriamo segmenti orizzontali si hanno due direzioni che sono est e ovest, mentre per quelli verticali si parlerà di nord e sud; un cambio di direzione del percorso si ha quando si è all'inizio o alla fine di ciascun segmento, e in questo caso si può girare a destra ($d_j = R$) o a sinistra ($d_j = L$)

- definiamo *corridoi laterali* quegli spazi, ortogonali al percorso, necessari a garantire che uno stand sia accessibile attraverso entrate alternative, che non si affacciano necessariamente al path
- i *candidate points* sono le coordinate di ogni vertice di uno stand che può essere piazzato interamente nel layout grazie al fatto che tutte le coordinate sono interne all'area selezionata per il posizionamento. Inoltre sono *candidate points* anche quei punti in cui un nodo del percorso non influisce sullo stand, ad esempio andando a tagliare internamente lo stand

A questo punto proseguiamo con l'illustrare il primo algoritmo, il quale ha come scopo quello di testare la possibilità di piazzamento di uno spazio nel fairground, definendo tra i vari *constraints* anche la larghezza minima dei corridoi laterali e l'adiacenza dello stand al percorso. In particolare l'algoritmo determina se uno stand è idoneo o meno ad entrare nel rettangolo riservato all'esposizione: in caso positivo significa che lo stand non si sovrappone a nessun altro stand già presente, che non invade lo spazio riservato al path o ad altri elementi, che risiede interamente all'interno del rettangolo e che, infine, fornisce una larghezza minima del corridoio laterale idonea a quanto previsto.

Utilizzando però questa soluzione si ottiene un layout che esclude alcuni spazi espositivi e dunque è necessaria una riformulazione dell'algoritmo che consideri non più una ottimizzazione riferita ad un solo lato del segmento del path adiacente allo stand, bensì tutti i lati a contatto con esso. Tale aggiornamento sarà identificato come secondo algoritmo: questo approccio rivela che alcune aree del percorso dei visitatori non sono usate appieno e che le aree disponibili che si incontrano all'inizio del percorso sono più occupate

rispetto a quelle che si trovano verso la fine. Questo accade perché l'algoritmo2 valuta le posizioni partendo proprio dal nodo iniziale del path.

Un'ulteriore tecnica per “raffinare” la procedura consiste nel distribuire in maniera più uniforme gli stands lungo il percorso usufruendo al massimo degli spazi inutilizzati. L'algoritmo3 combina perciò le qualità positive delle due versioni precedenti e rappresenta una versione definitiva e ultimata del layout, perché permette di realizzare un progetto ottimo che soddisfa i requisiti e le specifiche e che riesce ad includere tutti gli stands nel fairground. In figura 2.21 sono rappresentate le tre versioni dell'algoritmo usato nell'edizione della fiera di Romont del 2000; si riconduce a fine capitolo per quanto riguarda la formuazione degli algoritmi secondo uno pseudo-linguaggio.

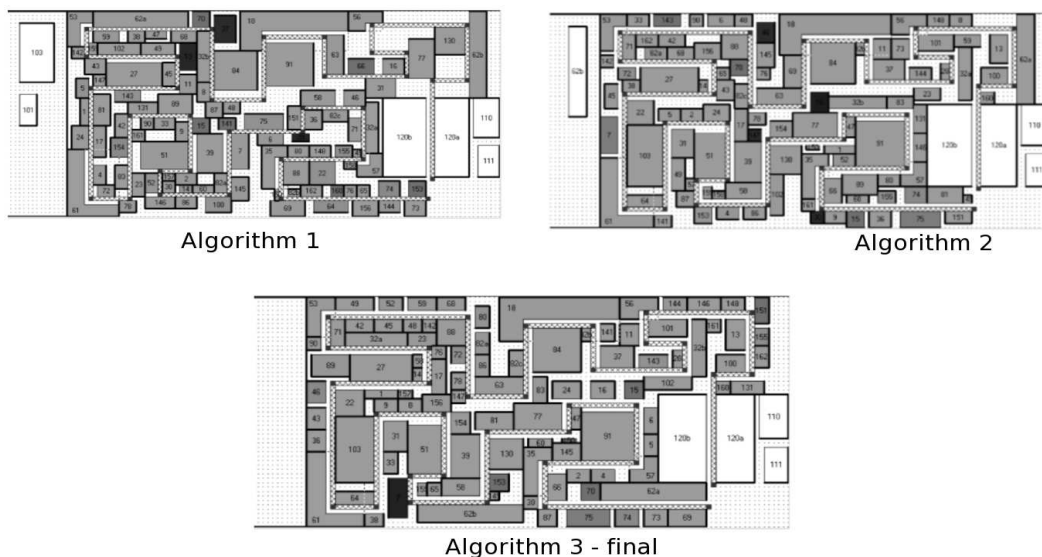


Figura 2.21: tre versioni dell'algoritmo impiegato per il layout ottimo dell'esposizione del 2000.

I risultati ottenuti applicando l'algoritmo nella sua versione finale mostrano che la procedura di costruzione genera risultati promettenti, basandosi sul modello dell'adiacenza. Inoltre i tre criteri precedentemente analizzati (utilizzo degli spazi, valore attrattivo e agevolazioni ai clienti visitatori) sono confinati in intervalli di accettabilità e i valori dei coefficienti f sono più alti di quelli riscontrati usando un layout “manuale”. Il coefficiente $r = 0.64$ che

misura il buono sfruttamento dello spazio è il migliore riscontrato.

Tutti i calcoli e le elaborazioni sono stati effettuati su una macchina con processore Pentium a 200MHz con gli algoritmi implementati mediante il linguaggio Visual Basic 6.0, per un tempo impiegato dal calcolatore che va da un minimo di 9.78 e 15.42 secondi. L'algoritmo 3 impiega normalmente un secondo in più rispetto all'algoritmo 2, che prevede minor complessità.

2.2.3 Considerazioni finali

La disposizione degli stand in ambito fieristico non è ancora particolarmente soggetta a studi scientifici, tuttavia lo studio precedentemente eseguito ha qualche parallelismo con il calcolo del layout aziendale; di solito un layout fieristico tiene conto di più criteri, invece in un layout aziendale il criterio fondamentale è quello di minimizzare i costi. Per quanto riguarda la fiera di Romont si sono utilizzati diversi criteri per avvicinarsi ad una soluzione accettabile. La procedura su cui si basavano i tre algoritmi visti precedentemente era improntata sulla collocazione degli stand dato un certo percorso prefissato e dati dei layout iniziali; proprio questi ultimi tendono ad influenzare maggiormente il raggiungimento di un possibile layout finale, il quale dipende da:

- dimensioni della superficie espositiva
- lunghezza del percorso (path)
- numero di cambi di direzione nel percorso (nodi)
- posizione (in coordinate cartesiane) degli angoli degli percorsi
- posizione obbligatoria e irremovibile di alcuni stands

Negli algoritmi precedentemente studiati gli stand venivano collocati uno alla volta; si potrebbe invece velocizzare il procedimento usando un algoritmo di packing in grado di collocare diversi stand nello stesso momento. La creazione del percorso iniziale può essere considerata come un importante problema anche se è solo l'inizializzazione del layout; l'individuazione di un percorso ideale faciliterebbe il raggiungimento di una soluzione accettabile

finale.

Per ciò che riguarda la fiera di Fortaleza, invece, si può concludere affermando che l'implementazione dei due tipi di FLOP ci ha portati all'ottenimento del nostro traguardo, potendo appunto affermare che il numero di stand che si possono collocare all'interno dell'area espositiva di Fortaleza è maggiore di quello del layout precedente.

	Read the input data. Define e_i and zwb . Let UP be the set of exhibition spaces that have to be placed, and $E = \emptyset$.
Step 1:	Select at random an exhibition space i in UP .
Step 2:	Examine the candidate points for the exhibition space i in the left- and right-hand side areas of the path sections (in regard to the direction of the visitor path): If there exist a feasible placement of the exhibition space i , then place i and set $UP = UP - \{i\}$, else set $E = E \cup \{i\}$.
Step 3:	If $UP \neq \emptyset$, then go to Step 1.
Step 4:	The obtained layout solution is feasible if $E = \emptyset$.

Figura 2.22: Algoritmo 1.

	<p>Read the input data. Define e_i and zwb. Define h, mb, dbL_j and dbR_j for all path sections $(j, j+1)$. Let UP be the set of exhibition spaces that have to be placed, and $E = \emptyset$.</p>
Step 1:	<p>Select in UP the exhibition space i of Type 3 that has the largest surface. Examine the candidate points in the U-shapes of the visitor path for the exhibition space i: If there exist a feasible placement of the exhibition space i, then place i, else set $E = E \cup \{i\}$. Set $UP = UP - \{i\}$. If UP contains exhibition spaces of Type 3, then go to Step 1, else set $UP = UP \cup E$ and $E = \emptyset$.</p>
Step 2:	<p>Select in UP the exhibition space i of Type 20 or 21 that has the largest surface. Examine the candidate points in the corners of the visitor path for the exhibition space i: If there exist a feasible placement of the exhibition space i, then place i, else set $E = E \cup \{i\}$. Set $UP = UP - \{i\}$. If UP contains exhibition spaces of Type 20 or 21, then go to Step 2, else set $UP = UP \cup E$ and $E = \emptyset$.</p>
Step 3:	<p>Go to Step 1 of Algorithm 1.</p>

Figura 2.23: Algoritmo 2.

	<p>Read the input data. Define e_i and zwb, h, mb, dbL_j and dbR_j. Define pa, the number of repetitions of Steps 5 to 7. Let UP be the set of exhibition spaces that have to be placed, and $E = \emptyset$. Set $zpa = 1$.</p>
Step 2:	Execute Step 1 of Algorithm 2.
Step 3:	Execute Step 2 of Algorithm 2.
Step 4:	<p>Order the exhibition space in UP by the short side k_i in descending order, and let UPO be the ordered set. Select in UPO the first exhibition space i (maximal k_i).</p>
Step 5:	Select at random a path section $(j, j+1)$ and choose a left- or right-hand side area at random.
Step 6:	<p>Examine the placement of the exhibition space i in the selected placement area of the path section $(j, j+1)$: If there exist a feasible placement of the exhibition space i, then place i, set $UPO = UPO - \{i\}$, select in UPO the next exhibition space i and go to Step 6, else set $zpa = zpa + 1$.</p>
Step 7:	If $zpa \leq pa$, then go to Step 5.
Step 8:	Select in UPO the first exhibition space i (maximal k_i).
Step 9:	<p>Examine the candidate points for the placement of the exhibition space i in the left- and right-hand side area of the path section $(j, j+1)$ regarding to the direction of the visitor path and set $UPO = UPO - \{i\}$: If there exist a feasible placement of the exhibition space i, then place i, else set $E = E \cup \{i\}$.</p>
Step 10:	If $UPO \neq \emptyset$, then go to Step 8.
Step 11:	The obtained layout solution is feasible if $E = \emptyset$.

Figura 2.24: Algoritmo 3.

Capitolo 3

Casi di studio e specifiche di progetto

Il presente capitolo è dedicato all'analisi e all'applicazione degli algoritmi esaminati nel primo capitolo, affiancati dal riferimento alle soluzioni viste nel secondo capitolo, in maniera tale da adattare tali conoscenze ai casi di studio presentati qui di seguito.

In particolare, tra i dati di interesse fondamentale per l'applicazione dei modelli, si concentrerà l'attenzione sulle dimensioni degli stand (moduli) e sulle distanze tra di essi, sulla larghezza dei corridoi e dei passaggi (*aisles*) secondo il modello di *Fair Layout Optimization Problem* già analizzato, e illustrato nelle figure seguenti:

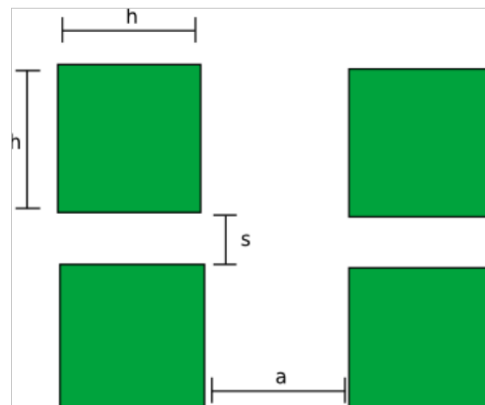


Figura 3.1: *Flop1*, caratteristiche e specifiche di packaging.

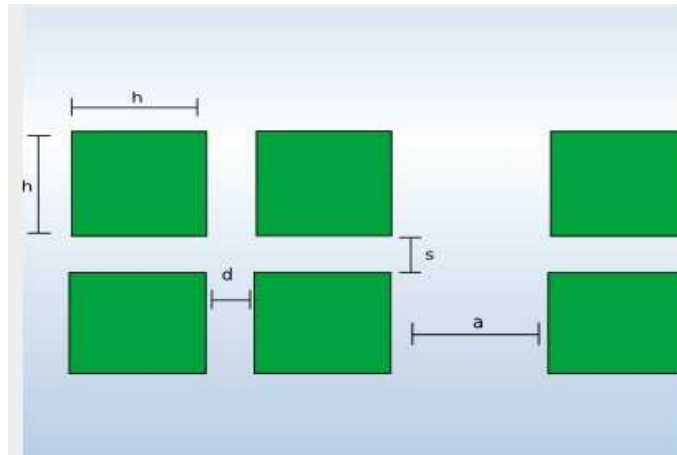


Figura 3.2: Flop2, caratteristiche e specifiche di packaging.

3.1 Presentazione dei casi di studio

I layout fieristici presi in considerazione nel progetto di tesi sono tre:

- la **Fiera del Levante** di Bari, importantissima e celebre fiera campionaria internazionale che si svolge annualmente nel capoluogo pugliese dal 13 al 21 settembre 2008
- la Fiera di Bolzano in occasione della **fiera mondiale dei veicoli alimentati a gas metano e ad idrogeno**, che si avrà luogo dal 9 al 12 Giugno 2008 contemporaneamente al Congresso ENGVA¹
- l'**Expo Ferroviaria 2008**, fiera dedicata all'industria e alla tecnologia ferroviarie, dal 20 al 22 Maggio 2008 presso il Lingotto Fiere di Torino

I modelli studiati nelle sezioni precedenti sono generalizzabili a vari contesti, per cui il fatto che si analizzino tre differenti contesti non è altro che una semplice e diretta conferma al concetto di adattare le soluzioni alle varie tipologie di layout che si presentano.

¹European Natural Gas Vehicle Association, associazione europea dei veicoli a gas naturali, come ad esempio metano e idrogeno

3.1.1 Fiera del Levante

La prima edizione della Fiera del Levante - Campionaria Internazionale - si è svolta nel 1930, mentre nel 1929 l'Ente Autonomo Fiera del Levante nasceva per iniziativa del Comune di Bari, dell'Amministrazione provinciale e della Camera di Commercio di Bari. La rassegna ha continuato a svolgersi puntualmente a settembre di ogni anno, con la sola interruzione durante gli anni della Seconda Guerra Mondiale, dal 1940 al 1946. Il quartiere fieristico si espande su di una superficie complessiva di circa 300 mila metri quadrati, che ospitano nel corso dell'anno una trentina di manifestazioni alcune delle quali hanno il riconoscimento di internazionalità.



Figura 3.3: vista panoramica dell'ingresso principale (Ingresso Monumentale) alla Fiera del Levante di Bari.

Complessivamente, gli espositori che partecipano annualmente alle manifestazioni in calendario sono oltre 5.000, nazionali ed esteri. A circa due milioni ammontano invece i visitatori. La manifestazione maggiore resta ancora la Campionaria internazionale di settembre, che può contare su oltre 2.000 espositori e più di 700.000 visitatori. La Fiera del Levante opera principalmente al servizio del grande mercato centromeridionale, pur allargando il suo campo operativo al sud est europeo ed all'area mediterranea.

Dal punto di vista dei dati tecnici la superficie espositiva, la cui planimetria è disponibile in figura 3.4, presenta:

- 44 padiglioni generali
- 33 padiglioni isolati
- 7 porte di accesso (ingressi alla fiera)
- una superficie espositiva complessiva di 300.000 mq
- una superficie espositiva coperta di 120.000 mq
- una superficie espositiva scoperta di 40.000 mq

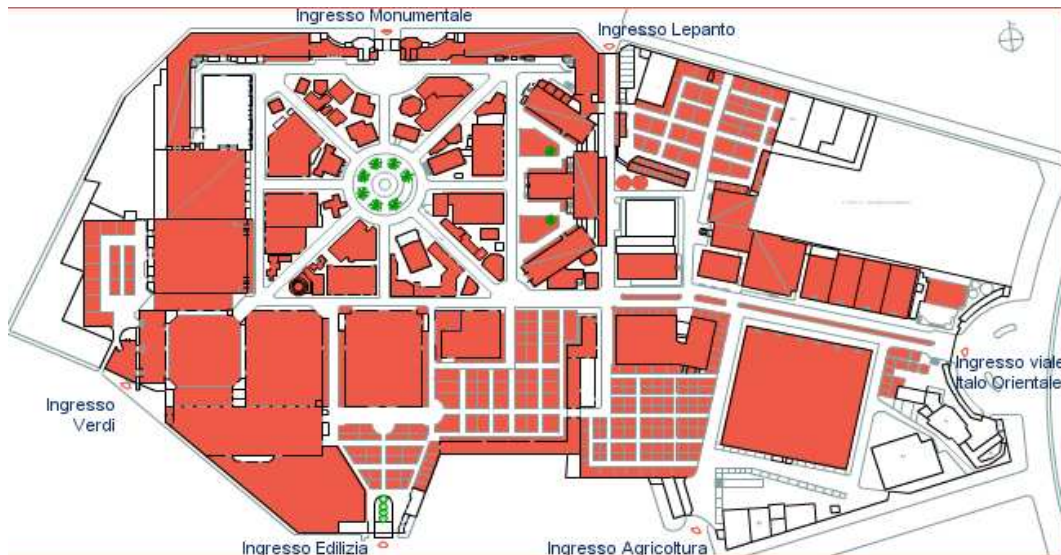


Figura 3.4: planimetria della Fiera del Levante di Bari.

I padiglioni considerati per l'applicazione degli algoritmi sono stati scelti in modo da avere un approccio il più eterogeneo possibile, al fine di coprire le più differenti casistiche: nella fattispecie sono stati presi in esame i padiglioni 11, 116 e 129. Mentre il primo presenta una superficie e una forma regolari, il secondo e il terzo sono molto più irregolari e presentano discontinuità nei contorni. La scelta di questi spazi espositivi è appunto giustificata dalle sostanziali differenze tra i tre padiglioni, sia per quanto riguarda la dimensione degli stand, sia per il loro arrangiamento negli spazi previsti. Dal punto di

vista delle dimensioni di interesse per l'applicazione degli algoritmi, di seguito sono elencate le principali caratteristiche di progetto (con riferimento a entrambi i flop e quindi alle figg. 3.1 e 3.2):

Dim. stand [h × w]	Largh. corridoi [a]	Dist. tra stand
<i>Padiglione 11</i>		
4×4 m	3 m (vert.); 4 m (orizz.)	nessuna
4×3 m	6.02 m (orizz.)	nessuna
4×3.50 m		nessuna
<i>Padiglione 116</i>		
4×4 m	5 m (vert.); 4 m (orizz.)	nessuna
6×6 m		nessuna
6×4 m		nessuna
<i>Padiglione 129</i>		
4×4 m	5 m (vert.); 4 m (orizz.)	nessuna
6×6 m		nessuna
6×4 m		nessuna

Tabella 3.1: Caratteristiche e dimensionamenti dei parametri di progetto

Ogni riga rappresenta i diversi dimensionamenti che si trovano all'interno di ciascun padiglione. In particolare non vi è una sola dimensione degli stand, né un'unica alternativa per quanto riguarda la larghezza dei corridoi. Questo è dovuto, oltre che ad esigenze di spazio, alla forma del padiglione (irregolare o meno che sia) e agli ostacoli strutturali (constraints) che si trovano all'interno degli spazi espositivi.

Nell'implementare gli algoritmi per questo contesto si terrà conto di una sola tipologia di dimensioni dei moduli ($[h \times w]$) e dei corridoi ($[a]$), fornendo varie un'unica alternativa progettuale in base ai dati di input standard, imposti dall'organizzazione. Nelle figura 3.5 sono mostrate le varie planimetrie dei padiglioni considerati, con un arrangiamento degli stands già predisposto per l'edizione 2007.

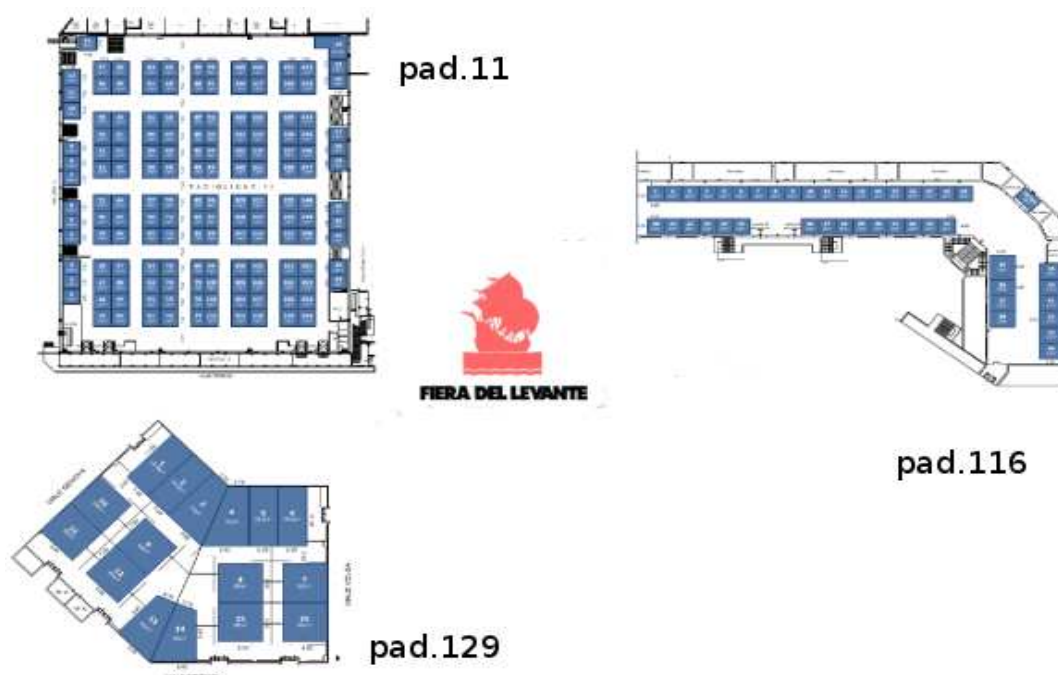


Figura 3.5: padiglioni della Fiera del Levante considerati nel progetto.

3.1.2 Fiera di Bolzano

Da sempre la città di Bolzano, nodo fondamentale tra nord e sud Europa, è stata sede di fiere e riunioni commerciali e mercantili. Tuttavia la prima fiera campionaria moderna risale al 1948, passando per gli anni '70 in cui si registrò un notevole sviluppo del quartiere fieristico, e giungendo ai giorni nostri in cui la Fiera di Bolzano è diventata una Società per Azioni, e ospita nel suo calendario più di 12 manifestazioni annuali di vario genere; si registrano complessivamente circa 171.000 visitatori l'anno.

“Fiera Bolzano” si trova al centro della Zona Produttiva Bolzano Sud, solo 2,5 km la separano dal centro città, per cui il complesso fieristico è facilmente raggiungibile grazie alla posizione alla periferia. Misura 40.000 mq tra superficie espositiva e annesso centro servizi; se poi si considera anche la struttura polifunzionale adiacente, il “Palaonda”, si hanno altri 5000 mq di spazio.

Procedendo con l'analisi dei dati salienti, il quartiere fieristico è costituito da 4 padiglioni espositivi, un padiglione polifunzionale (il Palaonda appunto), un Centro Servizi e un Centro Congressi e infine da un Hotel all'interno della

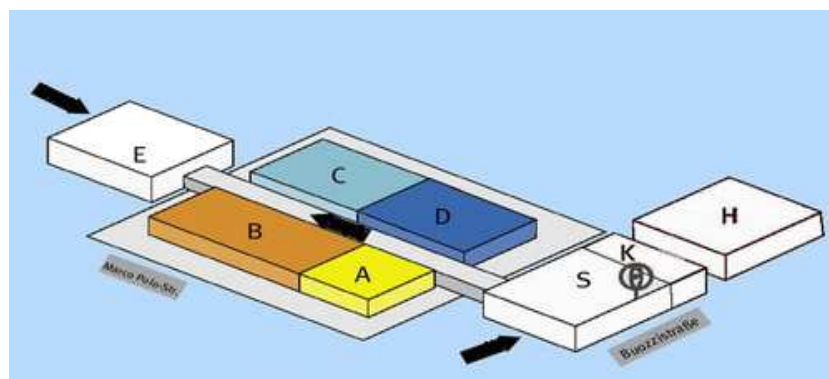


Figura 3.6: organizzazione del quartiere fieristico Fiera di Bolzano - Messe Bozen.

fiera; il tutto affiancato da ampi e comodi parcheggi sia interrati che superficiali e rialzati. L'omogeneità delle superfici degli stand consente a tutti gli espositori una presentazione efficace dei propri prodotti e servizi. In particolare, come si può confrontare anche in figura 3.6, la planimetria del quartiere è organizzata nel modo seguente:

- padiglione A - espositivo
2500 mq
- padiglione B - espositivo
4700 mq
- padiglione C - espositivo
6060 mq
- padiglione D - espositivo
7600 mq
- padiglione E - polifunzionale
5000 mq
- padiglione S - centro servizi
- padiglione K - centro congressi
- padiglione H - Hotel Fiera

Dal 9 al 12 giugno Fiera Bolzano S.p.A. ospiterà la fiera mondiale dei veicoli alimentati a gas metano e ad idrogeno. Contemporaneamente si terrà l'undicesimo Congresso europeo di ENGVA, associazione europea dei veicoli a metano e idrogeno. La fiera viene organizzata dalla rivista specializzata the Gas Vehicles Report con il sostegno della Provincia Autonoma di Bolzano. Al proprio interno il quartiere fieristico possiede un'ampia area stradale riservata agli espositori, ma direttamente confinante con l'intera costruzione



Figura 3.7: immagine a 360 gradi (espansa) del quartiere fiersistico Fiera di Bolzano - Messe Bozen.

di mostra; i test drive all'interno di tale struttura costituiscono una prova ulteriore per gli *exhibitors* di mostrare i loro prodotti in una zona che non richiede agli ospiti di lasciare il centro di mostra.

La manifestazione ENGVA ha luogo all'interno del padiglione D, la cui struttura è rappresentata in figura 3.9; tale struttura presenta vincoli e *constraints* tipiche, quali aree dedite ai servizi di ristorazione, uscite di emergenza, ingressi e uscite principali, strutture di sostegno (pilastri, ecc.). Nella implementazione dei layout andrà tenuto conto di tutte queste caratteristiche.

Concludendo, si ha quindi che il padiglione D, di dimensioni pari a 7600mq, ospiterà stand di dimensioni standard (*basic stand*) pari a 5×5 metri (fig. 3.8), mentre per i corridoi sarà prevista una larghezza minima di 4 metri. Inoltre saranno distinti due casi, nel primo dei quali la distanza minima tra stand è nulla, mentre nel secondo è di 2 metri (sia orizzontale che verticale).

Dim. stand [$h \times w$]	Largh. corridoi [a]	Dist. tra stand
5×5 m	4 m	nessuna
5×5 m	4 m	2 m

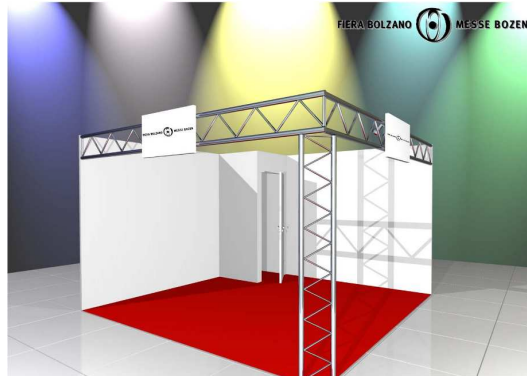


Figura 3.8: dimensioni standard di uno stand basic della Fiera di Bolzano - Messe Bozen.

3.1.3 Expo Ferroviaria 2008

La manifestazione torinese è il punto di incontro per l'industria ferroviaria internazionale, con espositori provenienti da 18 Paesi e specializzati nei più svariati campi dell'industria e della tecnologia che hanno a che vedere col mondo delle ferrovie. L'unica fiera ferroviaria che ha luogo regolarmente in Italia si svolge a Torino presso il Lingotto Fiere dal 20 al 22 maggio 2008. Strutturalmente l'Expo ospitata al Lingotto Fiere si suddivide in due padiglioni, i quali hanno le seguenti dimensioni:

<u>Padiglione1</u>	<u>Padiglione2</u>
dimensioni: 141×54 metri	dimensioni: 187×96 metri
totale: 7620 mq;	totale: 17960 mq;
altezza max: 10,5 metri;	altezza max: 14,2 metri;

Per quanto riguarda i dati inerenti gli spazi espositivi, considerando lo stand di figura 3.10, le dimensioni standard sono di $[h \times w] = 4 \times 4$ metri; per ciò che concerne le larghezze dei corridoi, invece, le dimensioni del parametro $[a]$ sono di 3 metri; si è scelto, per evitare complicazioni al modello e perché non indispensabili, di non considerare rilevanti (e quindi nulle) le distanze tra i singoli moduli.

In figura 3.13 è riportata la planimetria della manifestazione di Torino.

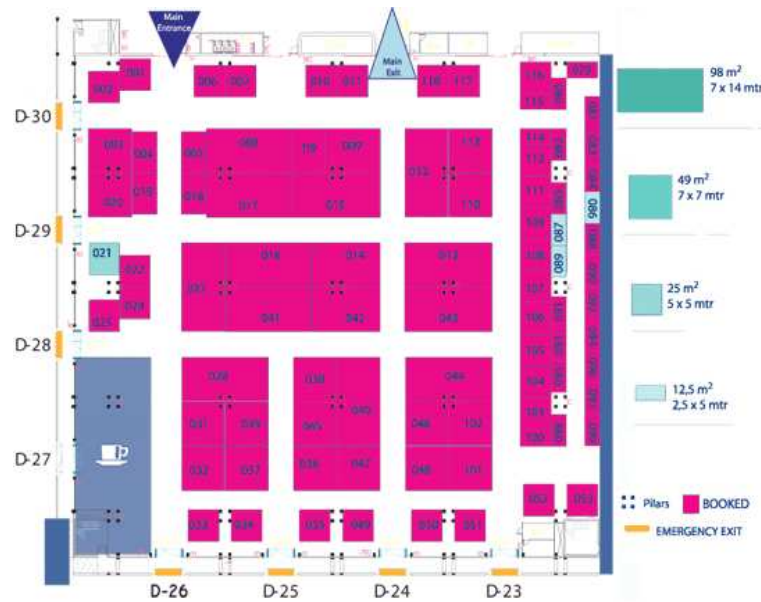


Figura 3.9: Padiglione D della Fiera di Bolzano - Messe Bozen.

Dim. stand [$h \times w$]	Largh. corridoi [a]	Dist. tra stand
4×4 m	3 m	nessuna

3.2 Criterio per la definizione e la scrittura delle matrici

Riprendiamo un argomento già trattato, ma comunque di fondamentale rilievo per l'implementazione degli algoritmi e dei modelli appena visti.

Si crei una matrice Φ , composta da quadrati di dimensione $[\delta \times \delta]$ che copra totalmente il rettangolo di dimensioni $[H \times W]$ contenente la superficie di esposizione. Usando un approccio granulare, si esprimeranno tutte le dimensioni (quelle esterne e degli stand) come multipli di questi (piccoli) valori. In particolare è definita la seguente dimensione:

$$X_{\delta} = \frac{X}{\delta}$$

dove $X = \{H, W, h, w\}$. Tale matrice ha perciò dimensioni $[W_{\delta} \times H_{\delta}]$. Un elemento della matrice assume valore 1 se il corrispondente quadratino $[\delta \times \delta]$

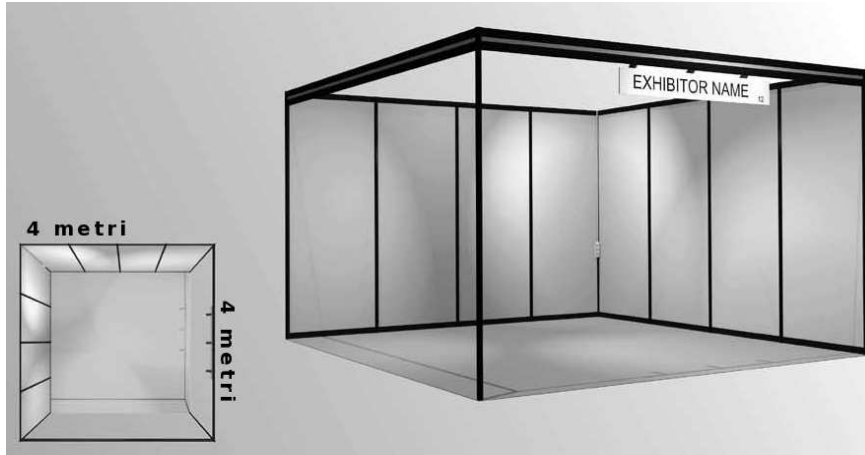


Figura 3.10: dimensioni dello stand standard considerato per l'Expo Ferroviaria 2008.

può essere usato completamente per l'impaccamento degli stand:

$$\Phi_{i,j} = \begin{cases} 1 & \text{se il quadrato } (i,j) \text{ può essere interamente usato per il package} \\ 0 & \text{altrimenti} \end{cases}$$

questo $\forall j = 1, \dots, W_\delta$ e $\forall i = 1, \dots, H_\delta$.

Si noti che per $\delta \ll W$, o ancora meglio per $\delta \ll w$, l'approssimazione risulta molto bassa.

Ora si numerino le colonne da 1 a W_δ . 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. A causa del criterio utilizzato e per evitare di dover costruire matrici in input di notevoli dimensioni è stata usata l'approssimazione al metro (ogni quadratino corrisponde ad un mq). Di questa approssimazione ne risentono ad esempio le dimensioni delle porte che a volte possono raggiungere la larghezza di 2 metri. Si ricordi che, dato un modello, 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 unimodulare² (TUM).

²una matrice totalmente unimodulare è una matrice (non necessariamente quadrata) per la quale anche ogni minore non singolare è unimodulare. Ne consegue che ogni suo elemento vale 0, +1 o -1. Un programma intero nel quale la matrice dei vincoli è totalmente unimodulare può essere risolto efficientemente, in quanto il suo rilassamento LP porta a soluzioni intere.



Figura 3.11: panoramica del quartiere Lingotto Fiere di Torino sede dell'Expo Ferroviaria 2008.

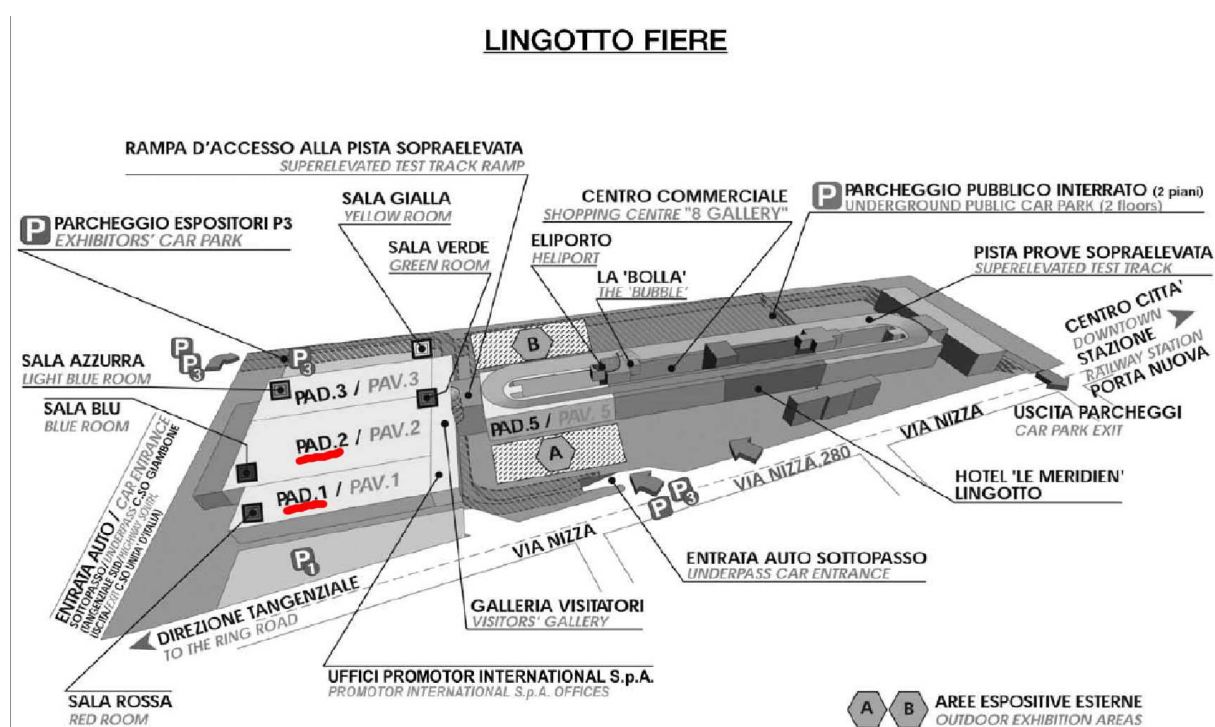


Figura 3.12: planimetria del quartiere Lingotto Fiere di Torino. In rosso sono sottolineati i padiglioni sede dell'Expo Ferroviaria 2008.

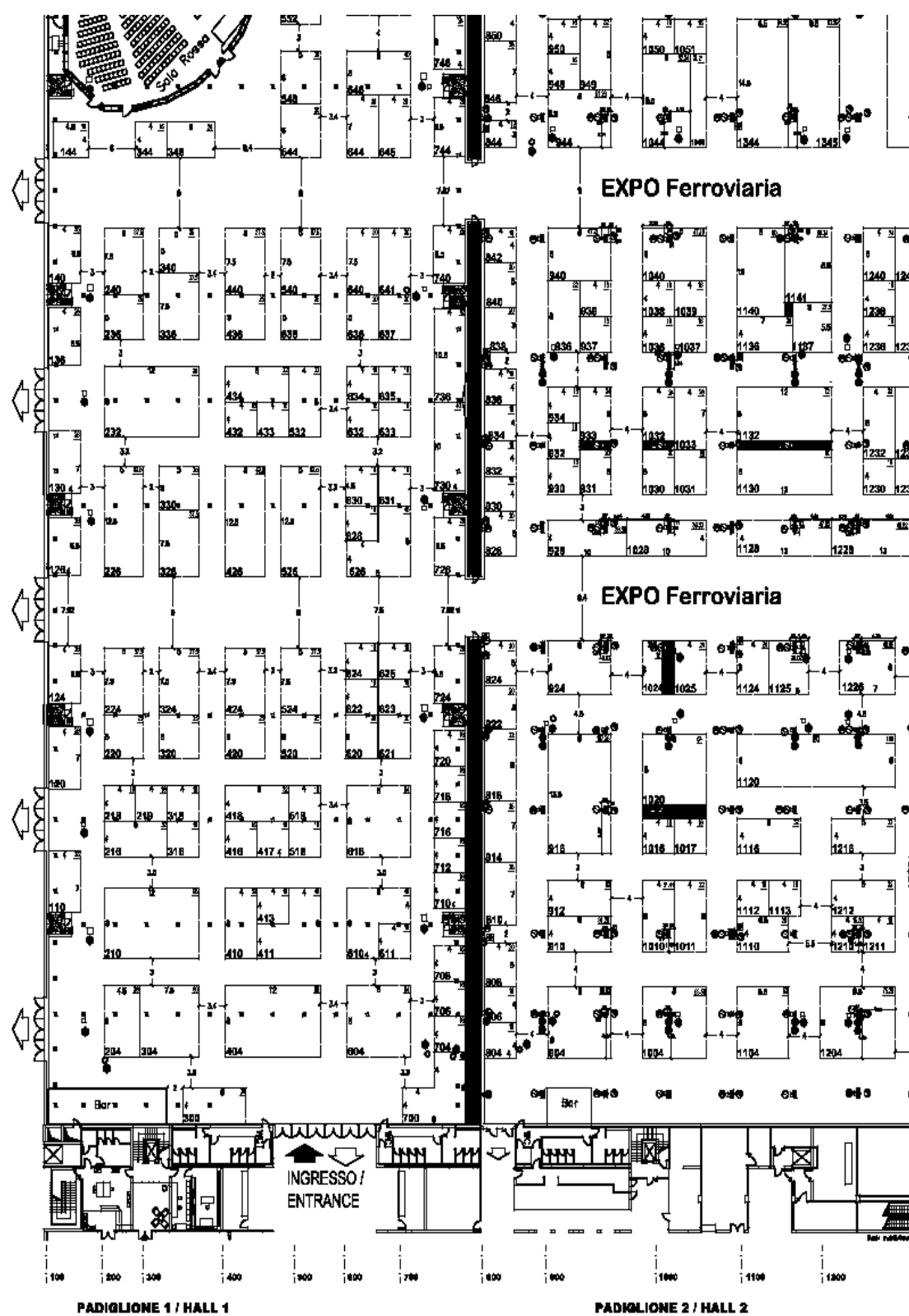


Figura 3.13: padiglioni della Expo Ferroviaria 2008. I due padiglioni sono separati dai muri portanti (in nero). Nella parte alta del secondo padiglione è stata tagliata la zona di ristorazione, comunque inutile al layout. La figura comprende solamente le zone idonee al contenimento degli stand.

Capitolo 4

Implementazione degli algoritmi e risultati computazionali

Questo capitolo è dedicato all'applicazione specifica degli algoritmi presentati nel capitolo 1 ai casi di studio presentati nel capitolo precedente. Partendo dalle planimetrie delle fiere di Bari, Bolzano e Torino, sono stati implementati gli algoritmi tramite un software sviluppato su piattaforma Java dall'Ing. Muritiba, presso il DEIS¹ di Bologna.

Tale applicativo è in grado di fornire soluzioni per l'impaccamento del maggior numero possibile di stand. Il software lavora costruendo le piantine dei padiglioni attraverso delle coordinate che vengono fornite come input; il concetto base è quello di creare un poligono esterno con la forma del padiglione e successivamente creare dei piccoli poligoni interni che evidenzino le aree in cui non è possibile impaccare gli stand (a causa di porte, bar, servizi, ecc.). Per creare questi poligoni è necessario fornire le coordinate $X - Y$ di ogni angolo partendo da un punto, ed elencare successivamente tutti i restanti punti in senso orario o in senso antiorario. I poligoni sono divisi in due tipi:

- polygon unlock - è il poligono che rappresenta la forma del padiglione o di ciò che voglio rappresentare; può essere uno solo
- polygon lock - rappresenta ogni singola area in cui non è possibile im-

¹Dipartimento di Elettronica, Informatica e Sistemistica

paccare alcuno stand ed è evidente che all'interno di un'area espositiva ne troveremo di diversi tipi

Come si vedrà poi meglio nello studio dei singoli casi reali, la forma degli spazi espositivi e dei corridoi è rappresentata in bianco mentre tutti i vincoli (polygon lock) sono rappresentati da spazi grigi. Una volta introdotti questi dati in input si è poi provveduto alle implementazioni di due diversi FLOP² per calcolare quanti stand poter inserire all'interno dei nostri padiglioni. Per ogni planimetria sono stati eseguiti entrambi i FLOP, le cui caratteristiche sono richiamate brevemente di seguito:

Flop1 : si tratta di singoli strip di stand di forma quadrata dove vengono forniti in input i valori di h (lato dello stand), s (piccoli corridoi di separazione tra gli stand) ed a (distanza tra gli strip di stand). Specificiamo però che s è sempre stato considerato nullo in entrambi i FLOP per ciascuna soluzione

Flop2 : si tratta di doppi strip di stand dove vengono forniti in input gli stessi dati del FLOP1; in più compare un dato d che però negli studi presentati di seguito sarà sempre considerato nullo

Con riferimento ai casi di studio proposti, verranno presentate di seguito le varie soluzioni relative all'implementazione degli algoritmi e dei *flop* ai vari padiglioni.

Si sottolinea che la procedura di implementazione degli algoritmi, per ciascuna soluzione, è stata effettuata su macchine costituite da processore Intel Core 2 Duo con 1GB di memoria RAM e sistema operativo Linux Ubuntu su cui è installata la Java Virtual Machine.

4.1 Fiera del Levante: soluzione

Con riferimento al padiglione 11 e alle dimensioni rappresentate nella tabella 3.1, presentiamo le soluzioni relative, prima impiegando il FLOP1 e quindi il FLOP2:

²acronimo per indicare il Fair Layout Optimization Problem

CPU-time FLOP1	CPU-time FLOP2	Stands FLOP1	Stands FLOP2
215 ms	680 ms	89	114

Ricordiamo che nelle ipotesi abbiamo considerato i moduli di dimensione standard 4×4 metri e i corridoi di larghezza 4 metri in linea orizzontale e 3 metri in verticale, tralasciando gli altri casi poiché la soluzione attuale rappresenta il miglior compromesso tra spazi occupati e numero di stand ottimo. In figura 4.1 sono rappresentate le soluzioni relative al Padiglione 11.

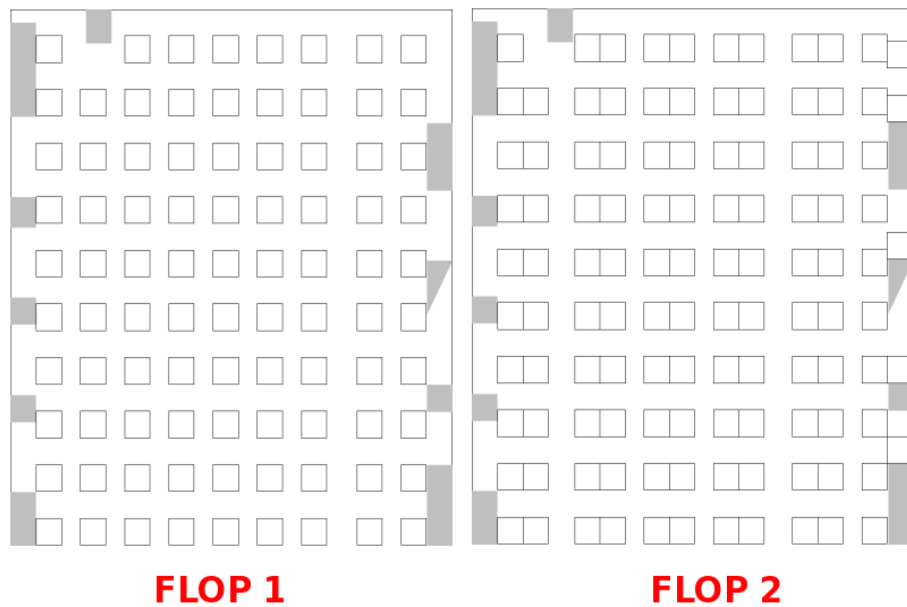


Figura 4.1: soluzioni proposte per il padiglione 11 con FLOP1 e FLOP2.

Per quanto riguarda il padiglione 116, sempre con riferimento alla tabella 3.1, si riportano entrambe le soluzioni adottate:

CPU-time FLOP1	CPU-time FLOP2	Stands FLOP1	Stands FLOP2
158 ms	351 ms	27	31

In figura 4.2 sono riportate le rispettive grafiche dei FLOP.

In questo padiglione le dimensioni relative alla dimensione dei moduli sono

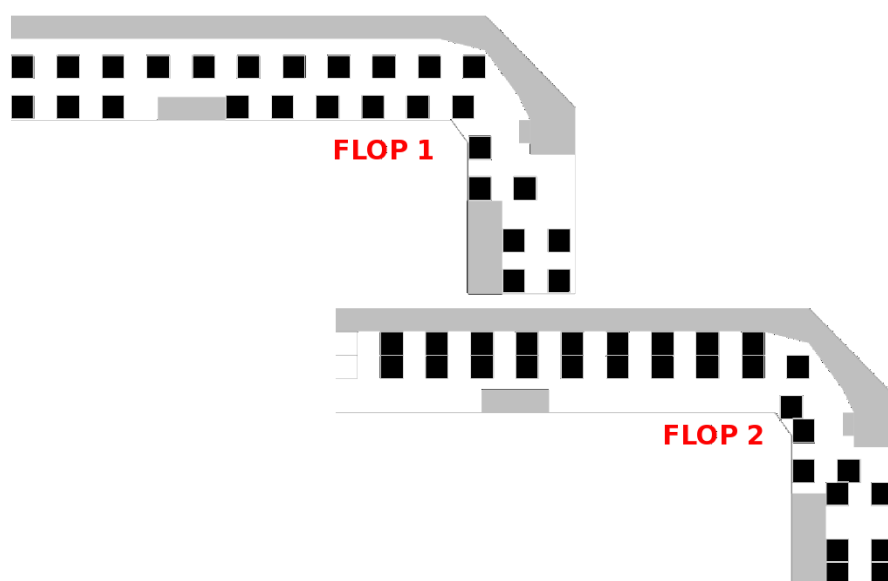


Figura 4.2: soluzioni proposte per il padiglione 116 con FLOP1 e FLOP2. In questo caso gli stand sono rappresentati in nero e i vincoli sempre in grigio.

sempre di 4×4 metri, mentre i corridoi hanno dimensione di 5 (verticale) \times 4 (orizzontale) metri.

Prendiamo infine in considerazione infine il Padiglione 129, di forma piuttosto irregolare analogamente al caso precedente, e riportiamo le soluzioni nella tabella seguente e nella figura 4.3:

CPU-time FLOP1	CPU-time FLOP2	Stands FLOP1	Stands FLOP2
106 ms	272 ms	24	31

dove, come per i precedenti padiglioni, la dimensione degli stand è di 4×4 metri e per i corridoi è di 5 (verticale) \times 4 (orizzontale) metri.

Nel listato seguente sono elencati i file di input da fornire al software Java per costruire il layout di ciascun padiglione (rispettivamente, i padiglioni 11, 116 e 129), tenendo in considerazione i vincoli e gli ostacoli imposti dal contesto:

```

polygon; unlock; 0 0; 0 80; 70 80; 70 0
lock1; lock; 0 0; 0 8; 4 8; 4 0
lock2; lock; 0 18.5; 0 22.5; 4 22.5; 4 18.5

```

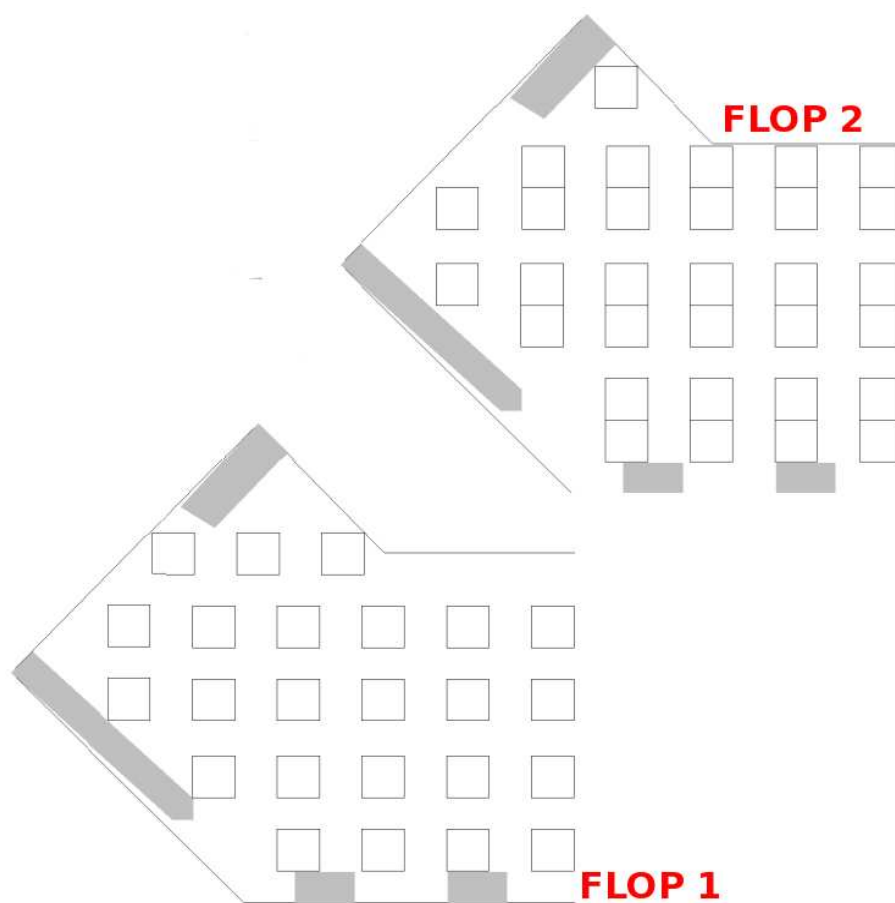


Figura 4.3: soluzioni proposte per il padiglione 129 con FLOP1 e FLOP2. In questo caso gli stand sono rappresentati in nero e i vincoli sempre in grigio.

lock3; lock; 0 33; 0 37; 4 37; 4 33

lock4; lock; 0 47.5; 0 52; 4 52; 4 47.5

lock5; lock; 0 64; 0 78; 4 78; 4 64

lock6; lock; 66 0; 66 12; 70 12; 70 0

lock7; lock; 66 20; 66 24; 70 24; 70 20

lock8; lock; 66 34.5; 66 42.5; 70 42.5; 66 34.5

lock10; lock; 66 53; 66 63; 70 63; 70 53

lock11; lock; 12 80; 12 75; 16 75; 16 80

polygon; unlock; 0 0; 0 84; 16 100; 48 100; 48 81; 22 81; 18 78; 18 0

lock1; lock; 0 0; 0 84; 16 100; 24 100; 24 92; 18 92; 14 90; 6 84; 4 76; 4 0

```
lock2; lock; 48 81; 32 81; 32 87; 48 87
lock3; lock; 18 26; 14 26; 14 38; 18 38
lock4; lock; 18 92; 22 92; 22 90; 18 90

polygon; unlock; 0 0; 0 31.2; 22 53.2; 46 30; 33.6 18; 33.6 0
lock1; lock; 0 6.4; 0 12; 3 12; 3 6.4
lock2; lock; 0 20.8; 0 26.4; 3 26.4; 3 20.8
lock3; lock; 8 38; 22 53.2; 24 51.2; 10 36; 8 36;
lock4; lock; 46 30; 43.2 27.2; 36 34; 38 37.2
```

4.2 Fiera di Bolzano: soluzione

Avendo precedentemente considerato la forma degli stand di 5×5 metri e la larghezza di tutti i corridoi di 4 metri (sia in orizzontale che in verticale), la soluzione presentata per il padiglione D della Fiera di Bolzano è riassunta nella tabella che segue:

	CPU-time FLOP1	CPU-time FLOP2	Stands FLOP1	Stands FLOP2
caso1	230 ms	710 ms	112	126
caso2	275 ms	800 ms	94	100

In questa rappresentazione sono stati prese in considerazione due varianti: nel primo caso (caso 1) la distanza imposta tra gli stands è nulla, mentre nel secondo (caso 2) è di 2 metri sia in orizzontale che in verticale. Per il resto tutti i parametri rimangono invariati.

Analogamente a quanto fatto per il contesto precedente, viene qui di seguito presentato il listato relativo al contesto di Bolzano coi rispettivi *lock* che rappresentano i vincoli e gli ostacoli all'interno del padiglione:

```
polygon; unlock; 0 0; 0 80; 85 80; 85 0
lock1; lock; 0 0; 0 30; 13 30; 13 0
lock2; lock; 14 0; 14 5; 19 5; 19 0
lock3; lock; 33 0; 33 5; 38 5; 38 0
lock4; lock; 52 0; 52 5; 57 5; 57 0
```

```

lock5; lock; 71 0; 71 5; 76 5; 76 0
lock6; lock; 0 30; 0 35; 5 35; 5 30
lock7; lock; 0 44; 0 49; 5 49; 5 44
lock8; lock; 0 63; 0 68; 5 68; 5 63
lock9; lock; 13 80; 20 80; 20 74; 13 74
lock10; lock; 51 80; 58 80; 58 74; 51 74
lock11; lock; 23 5; 23 7.5; 25.5 7.5; 25.5 5
lock12; lock; 41.5 5; 41.5 7.5; 44 7.5; 44 5
lock13; lock; 60 5; 60 7.5; 62.5 7.5; 62.5 5
lock14; lock 78.5 5; 78.5 7.5; 80 7.5; 80 5
lock15; lock; 78.5 24; 78.5 26.5; 80 26.5; 80 24
lock16; lock; 78.5 41.5; 78.5 44; 80 44; 80 41.5
lock17; lock; 78.5 59; 78.5 61.5; 80 61.5; 80 59

```

4.3 Expo Ferroviaria 2008: soluzione

Il calcolo degli elementi e degli spazi per i padiglioni dell'Expo Ferroviaria 2008 hanno portato ad una soluzione ottimizzata che viene riassunta nella tabella seguente:

CPU-time FLOP1	CPU-time FLOP2	Stands FLOP1	Stands FLOP2
174 ms	478 ms	264	374
371 ms	1046 ms	578	781

Tabella 4.1: Risultati dei calcoli per i padiglioni dell'Expo Ferroviaria 2008. In alto i dati riguardanti il pad.1 e in basso quelli per il pad.2.

mentre per quanto riguarda le immagini relative ad entrambi i padiglioni, esse sono rappresentate nelle figure 4.4 e 4.5. Ricordiamo che, per entrambi i padiglioni, le dimensioni dei moduli sono di 4×4 metri, mentre i corridoi hanno una larghezza di 3 metri.

Presentiamo infine i file di input che permettono la mappatura dei due padiglioni dell'Expo Ferroviaria:

```

polygon; unlock; 0 0; 0 153; 54 153; 54 0

```

```
lock1; lock; 0 0; 0 12.5; 54 12.5; 54 0
lock2; lock; 0 12; 0 16; 31 16; 31 12
lock3; lock; 0 20; 0 27.92; 4 27.92; 4 20
lock4; lock; 0 35.84; 0 38.34; 2.5 38.34; 2.5 35.84
lock5; lock; 0 45.34; 0 53.26; 4 53.26; 4 45.34
lock6; lock; 0 60.26; 0 62.76; 2.5 62.26; 2.5 60.26
lock7; lock; 0 69.26; 0 77.18; 54 77.18; 54 69.26
lock8; lock; 0 83.68; 0 86.18; 2.5 86.18; 2.5 83.68
lock9; lock; 0 93.18; 0 101.1; 4 101.1; 4 93.18
lock10; lock; 0 107.6; 0 110.1; 2.5 110.1; 2.5 107.6
lock11; lock; 0 116.6; 0 124.52; 54 124.52; 54 116.6
lock12; lock; 0 128.52; 0 153; 24.48 153; 24.48 128.52
lock13; lock; 51.5 153; 54 153; 54 150.5; 51.5 150.5
lock14; lock; 51.5 133.52; 54 133.52; 54 131.02; 51.5 131.02
lock15; lock; 51.5 110.1; 54 110.1; 54 107.6; 51.5 107.6
lock16; lock; 51.5 86.18; 54 86.18; 54 83.68; 51.5 83.6
lock17; lock; 51.5 62.76; 54 62.76; 54 60.26; 51.5 60.26
lock18; lock; 51.5 38.34; 54 38.34; 54 35.84; 51.5 35.84

polygon; unlock; 0 0; 0 199; 96 199; 96 0
lock1; lock; 0 0; 0 12.5; 96 12.5; 96 0
lock2; lock; 0 199; 96 199; 96 124.52; 60 124.52; 60 174.52; 0 174.52
lock3; lock; 0 116.6; 0 124.52; 96 124.52; 96 116.6
lock4; lock; 0 69.26; 0 77.18; 96 77.18; 96 69.26
```

4.4 Considerazioni finali

Rispetto all'arrangiamento degli spazi effettuato dagli organizzatori, i risultati ottenuti mediante calcoli e modelli matematici hanno portato ad un miglioramento e ad un'ottimizzazione in termini di stand impiegati e di vincoli rispettati, il tutto in accordo con le specifiche mostrate nel terzo capitolo. In particolare, per quanto riguarda le tre alternative e i tre layout analizzati, si sono riscontrati i risultati complessivi schematizzati in tabella 4.2

	Stands originali	Stands FLOP1	Stands FLOP2
Padiglione 11 Fiera del Levante	154	89	114
Padiglione 116 Fiera del Levante	45	27	31
Padiglione 129 Fiera del Levante	16	24	31
Padiglione D.1 Fiera di Bolzano	90	112	126
Padiglione D.2 Fiera di Bolzano	90	94	100
Padiglione 1 Expo Ferroviaria	114	274	374
Padiglione 2 Expo Ferroviaria	219	578	781

Tabella 4.2: Numero di stands per ogni layout analizzato, in relazione agli stand disposti dagli organizzatori nella configurazione originale di ciascun padiglione. Si noti che il padiglione D del contesto di Bolzano è stato suddiviso in due parti a seconda dei casi enunciati nel paragrafo 4.2

Oltre ai dati relativi al numero di stands “impacchettati” e alle elaborazioni computazionali, ricopre fondamentale importanza e rilevanza il dato relativo alla dimensione delle matrici associate ai vari FLOP.

In particolare, rimandando il lettore al paragrafo 3.2, le matrici create conseguentemente alle elaborazioni sono, per ciascuna delle manifestazioni fieristiche considerate, le seguenti:

Matrix Pad. 11	Matrix Pad. 116	Matrix Pad. 129
[799 x 699]	[999 x 479]	[531 x 459]

Tabella 4.3: Dimensione delle matrici per la Fiera del Levante

Matrix Pad. D.1	Matrix Pad. D.2
[749 x 639]	[819 x 669]

Tabella 4.4: Dimensione delle matrici per la Fiera di Bolzano - Messe Bozen con le due soluzioni in base alla distanza tra i moduli

Matrix Pad. 1	Matrix Pad. 2
[1529 x 539]	[1989 x 959]

Tabella 4.5: Dimensione delle matrici per l'Expo Ferroviaria 2008

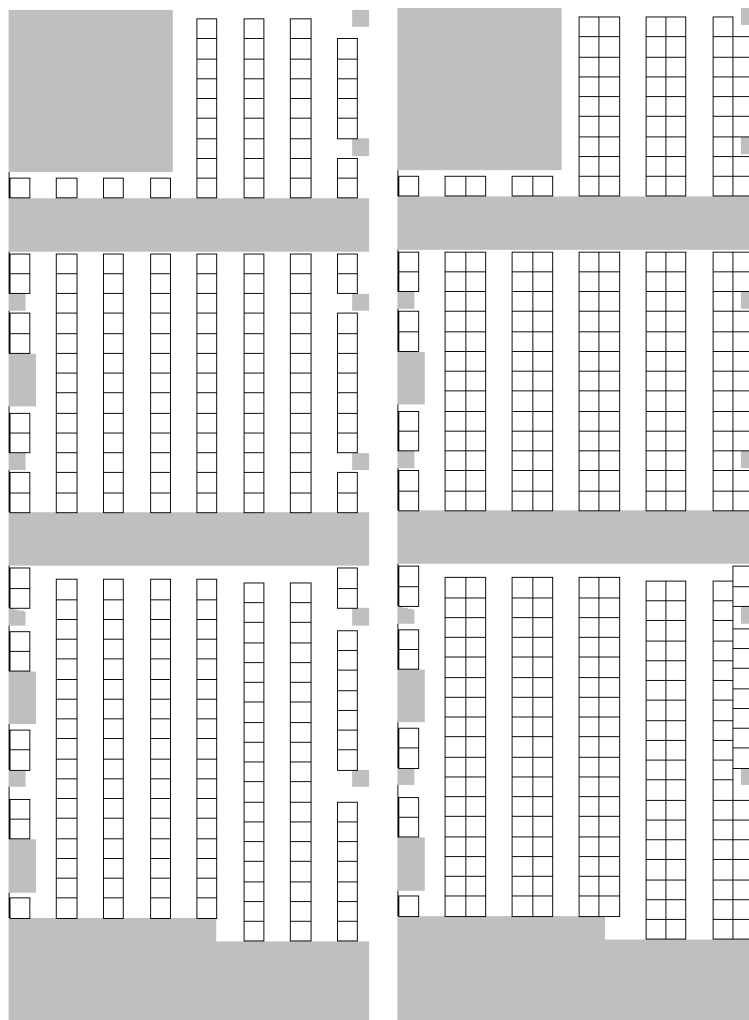


Figura 4.4: soluzioni proposte per il padiglione 1 dell'Expo Ferroviaria con FLOP1 e FLOP2 rispettivamente a sinistra e a destra.

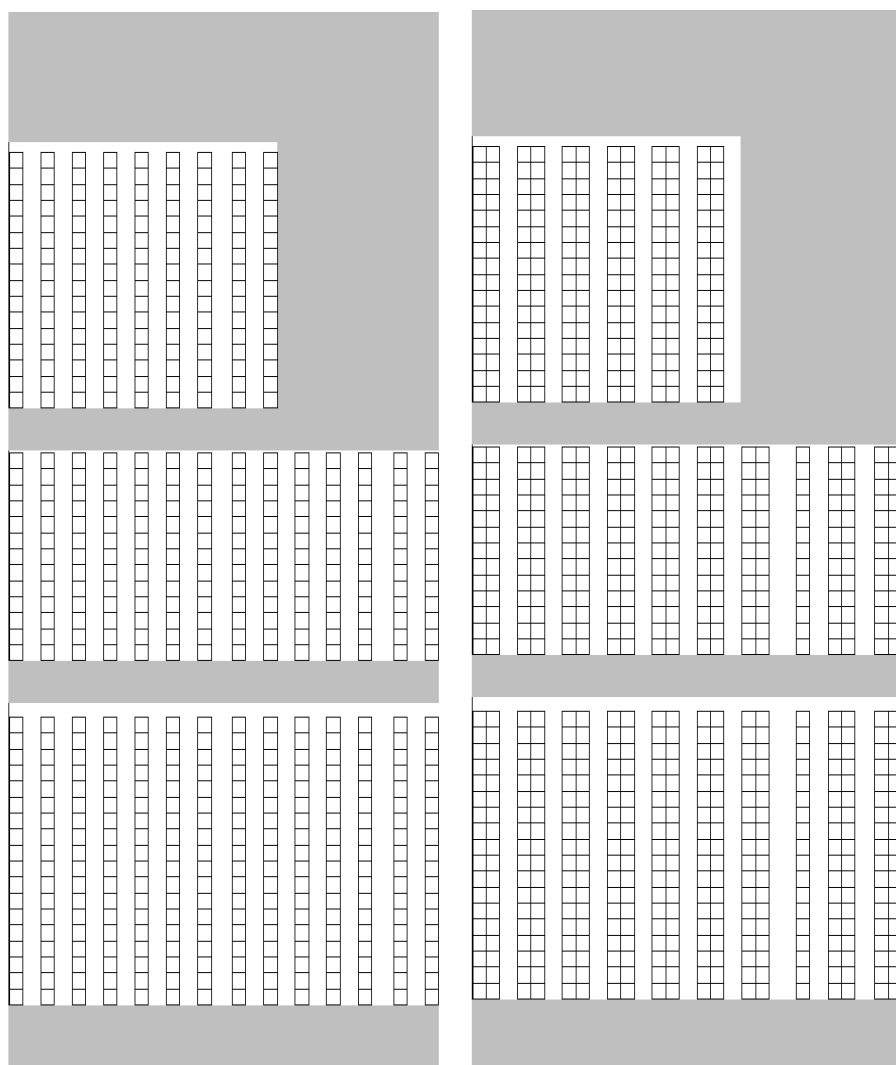


Figura 4.5: soluzioni proposte per il padiglione 2 dell'Expo Ferroviaria con FLOP1 e FLOP2 rispettivamente a sinistra e a destra.

Capitolo 5

Conclusioni

Prima di parlare degli obiettivi raggiunti, è bene ripercorrere velocemente tutta la strada fatta. Dopo aver individuato nel FLOP (Fair Layout Optimization 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 pochissimi testi 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 rivolti ad altri oggetti (cui possiamo ricondurre gli stands); un ampio capitolo della tesi è infatti 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 3. 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 prima analizzati nel capitolo 3 e poi inseriti nei contesti presentati nel capitolo 4. Infine, nel quinto capitolo sono stati descritti i risultati delle implementazioni sul software, tenendo ovviamente conto di vincoli e specifiche tipici di un layout di tipo fieristico. Proprio l'implementazione ha dimostrato la validità di tutti i modelli presentati nei capitoli addietro; 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
- tutti i modelli producono l'output istantaneamente o in breve tempo (al massimo viene richiesto qualche minuto per i modelli più laboriosi)
- tutti i modelli funzionano correttamente e restituiscono un output corretto e in linea con le richieste

A parte tutte queste considerazioni, ciò che fa di questa tesi uno scritto interessante consiste nella 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 questo è data dal fatto che i modelli siano stati correttamente usati in contesti diversi tra loro, con spazi espositivi e forme regolari e irregolari, con più o meno *constraints* da aggirare, e in spazi aperti o chiusi.

In sintesi, i punti di forza della tesi risiedono nell'innovazione dal punto di vista dello studio matematico col quale è affrontato l'argomento dei layout fieristici (è un argomento piuttosto nuovo e tuttora in pieno sviluppo) e la generalizzabilità dei modelli e degli algoritmi presentati.

Bibliografia

- [1] A.E. Fernandes Muritiba, Manuel Iori, Silvano Martello, M.J. Negreiros Gomes. *Models and algorithms for fair layout optimization problems*. Operations Research Letters, 2008 (in corso di pubblicazione)
- [2] P. Schneuwly, M. Widmer. *Layout modeling and construction procedure for the arrangement of exhibition spaces in a fair*. Blackwell, 2003
- [3] S. Martello, P. Toth. *Knapsack Problems. Algorithms and computer implementations*. John Wiley & Sons
- [4] A. Caprara, M. Monaci. *On the 2-dimensional knapsack problems*. Operational Research Letters, 2003
- [5] 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
- [6] R. Burkard, M. Dell'Amico, S. Martello. *Assignment Problems*. SIAM Monographs on Discrete Mathematics and Applications
- [7] G. Christofides, A. Mingozzi, P. Toth. *Contributions to the quadratic assignment problem*. European Journal of Operational Research, 1980
- [8] P. C. Gilmore, R. E. Gomory. *Multistage cutting problems of two and more dimensions*. Operations Research, 1965
- [9] E. L. Lawler. *The Quadratic Assignment Problem*. Management Science, 1963

-
- [10] A.Lodi, S.Martello, D.Vigo. *Approximation algorithms for the oriented two-dimensional bin packing problem*. European Journal of Operational Research, 1999
 - [11] 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
 - [12] E.Hadjiconstantinou, M.Iori. *An hybrid genetic algorithm for the two-dimensional single large object placement problem*. European Journal of Operational Research, 2006
 - [13] J.E.Beasley. *An exact two-dimensional non-guillotine cutting tree search procedure*. Operational Research, 1985
 - [14] R.Alvarez-Valdes, F.Parrenho, J.M.Tamarit. *A tabu search algorithm for a two-dimensional non-guillotine cutting problem*. European Journal of Operational Research, 2006
 - [15] D. Vigo. *Lezioni di Ricerca Operativa L-A*. Slides presentate a lezione, AA. 2007-2008
 - [16] Lisa Bianchini. *Tesi di Laurea sui layout fieristici*. Università degli Studi di Modena e Reggio Emilia
 - [17] Luca Rustichelli. *Tesi di Laurea sui layout fieristici*. Università degli Studi di Modena e Reggio Emilia
 - [18] Marc Baudoin. *Impara L^AT_EX! (...e mettilo da parte)*. École Nationale Supérieure de Techniques Avancées, Paris
 - [19] *Wikipedia*, www.wikipedia.org
 - [20] *DEIS, Università di Bologna*, www.deis.unibo.it
 - [21] *Operational Research at DEIS, Università di Bologna*, www.or.deis.unibo.it
 - [22] *Department of Computer Science, University of Liverpool*, www.csc.liv.ac.uk

-
- [23] *Institute of Electrical and Electronical Engineers*, www.ieee.org
 - [24] *Google*, www.google.it
 - [25] *Fiera di Bolzano*, www.fierabolzano.it
 - [26] *Fiera del Levante*, www.fieradellevante.it
 - [27] *Expo Ferroviaria 2008*, www.expoferroviaria.com
 - [28] *Science Direct*, www.sciencedirect.com
 - [29] *Associazione Italiana di Ricerca Operativa*, www.airo.org