Indice

1 Introduzione **2**
- 1.1 Definizione del problema . . . 2
- 1.2 Complessità . . . 3
- 1.3 Le classi di equivalenza di Dowsland . . . 5
- 1.4 Il teorema di Barnes . . . 8

2 Upper bounds **10**
- 2.1 Riduzione alle massime dimensioni utilizzabili . . . 11
- 2.2 Bound di Barnes . . . 12
- 2.3 Bound di Keber . . . 12
- 2.4 Bound LP . . . 12
- 2.5 Minimizzazione del bound sulle classi di equivalenza . . . 15
- 2.6 Altri vincoli strutturali: bound di Nelißen . . . 18
- 2.7 Un approccio LP 0-1 . . . 20
- 2.8 Valutazione dei bound . . . 24

3 Euristiche **27**
- 3.1 Le strutture k-block . . . 27
- 3.2 Euristica G4 . . . 29
- 3.3 Euristica di Morales e Morabito . . . 31
- 3.4 Generazione di strutture k-block . . . 32

4 Algoritmi esatti **36**
- 4.1 L'algoritmo di Dowsland . . . 37
- 4.2 Un nuovo algoritmo di tree search . . . 39
  - 4.2.1 Il metodo del contorno . . . 41
  - 4.2.2 Insieme ridotto dei raster points . . . 44
  - 4.2.3 Un criterio per la generazione dei pattern . . . 47
  - 4.2.4 Un nuovo bound strutturale per il cutting . . . 51
  - 4.2.5 Un nuovo criterio di dominanza . . . 54
  - 4.2.6 Sperimentazione numerica . . . 55

5 Conclusioni **60**

Capitolo 1

Introduzione

1.1 Definizione del problema

Il problema del pallet loading appartiene alla vasta famiglia dei problemi di *cutting & packing (C & P)*. Questo settore della ricerca operativa ha ampie applicazioni nell'industria manifatturiera e nei trasporti, ed è oggetto di studio approfondito a partire dagli anni 50 [13, 15, 27]. All'interno del C & P rientrano problemi molto diversi tra loro, ma che condividono una comune struttura: lo studio della disposizione geometrica di oggetti all'interno di domini più grandi, in assenza di sovrapposizioni, che garantisca un utilizzo ottimale delle risorse disponibili. Queste risorse possono essere il numero di contenitori usati, come nel bin packing, oppure lo spazio occupato all'interno di un contenitore, come nel container loading, o altro ancora; problemi ben noti come il knapsack, il multiprocessor scheduling, il memory allocation problem rientrano a pieno titolo nel C & P. Gli oggetti da inserire sono visti in "positivo", nel packing, o in "negativo", nel cutting.

Il Pallet Loading Problem (PLP) è noto anche sotto il nome di Manufacturer's Pallet Loading, in quanto concerne la sistemazione di scatole identiche, problema che sorge tipicamente nella fase di produzione di un bene [6]. Nel Pallet Loading Problem consideriamo una collezione illimitata di scatole identiche, e un pallet, di forma rettangolare. Si richiede di trovare una disposizione delle scatole all'interno del pallet che massimizzi il numero delle scatole inserite. L'impaccamento deve essere *ortogonale*: ogni scatola deve avere le facce parallele a quelle del pallet (osservazioni sul packing non ortogonale si trovano in [8]). Inoltre l'impaccamento deve avvenire per strati orizzontali; si hanno tre spessori possibili per ogni strato, corrispondenti alle tre dimensioni della scatola. Questi vincoli corrispondono a esigenze di praticità nel caricamento del pallet.
Il problema si scompone così in due sottoproblemi: a) trovare, per ogni spessore possibile, una disposizione all'interno dello strato che renda massimo il numero di scatole usate: si tratta in effetti di un problema bidimensionale,

il *rectangle packing problem* con rettangoli identici; b) trovare la sovrapposizione di strati ottimale: questo problema si esprime con un *integer knapsack problem*, con tre variabili corrispondenti ai tre possibili strati.
Senza perdita di generalità si assume che le dimensioni degli oggetti siano date da interi positivi. Il sottoproblema b) è il più semplice dei due, sia dal punto di vista pratico, sia dal punto di vista teorico; infatti un noto risultato di Lenstra [21] stabilisce che i problemi di programmazione lineare intera con numero fissato di variabili appartengono alla classe P.
In letteratura quindi con PLP ci si riferisce comunemente al sottoproblema a), che appartiene alla classe 2/B/O/C della tipologia di Dyckhoff [14]: si tratta di un problema bidimensionale (2), in cui si deve ottimizzare l'impaccamento di item in un (O) contenitore fissato (B), e gli item sono identici (C).

Per convenzione, verrà usato il termine *box* per indicare il rettangolo da impaccare, con dimensioni l e w, $l \geq w$; invece il termine *rettangolo* sarà riservato al contenitore, con dimensioni L e W, $L \geq W$. Un'istanza del PLP verrà rappresentata con (L, W, l, w).

Nelle sezioni che seguono viene discussa la complessità computazionale del PLP e viene introdotto un risultato fondamentale di K. A. Dowsland che consente di partizionare le istanze del PLP in classi di equivalenza.

1.2 Complessità

Un caso particolare di PLP è il *guillotine pallet loading*, in cui vengono presi in considerazione solo pattern a ghigliottina. Un pattern è detto a ghigliottina se è possibile suddividerlo ricorsivamente in parti fino al livello del singolo box usando tagli che vanno da lato a lato del rettangolo (rimanente); per una definizione più precisa si veda il capitolo 3. Questo tipo di pattern è richiesto per esempio quando vengono ricavati dei pezzi per mezzo di tagli effettuati su una lastra.
Tarnowski [35] ha dimostrato che il *guillotine pallet loading problem* è risolvibile in tempo polinomiale: l'algoritmo da lui ideato ha complessità temporale $O(\log_2 L \log_2^2 l)$. Tuttavia una frazione consistente delle istanze del PLP non ha soluzioni ottimali costituite da pattern a ghigliottina (circa il 20% [25]). La figura 1.1 mostra una soluzione ottimale per un'istanza del PLP priva di soluzioni a ghigliottina ottimali.

Per quanto riguarda il caso generale, in alcuni lavori si afferma che il PLP è NP-completo (vedi per esempio Dowsland [11]). Sembra tuttavia che questa affermazione sia dovuta ad un'errata interpretazione del risultato dimostrato in [18]; in quest'ultimo lavoro si stabilisce che il packing nel piano di quadrati identici è NP-completo. Si osservi che la descrizione del problema in [18] è abbastanza diversa dalla formulazione del PLP: i dati iniziali

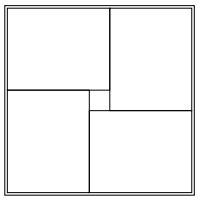

Figura 1.1: Pattern non a ghigliottina ottimale

sono costituiti dalla dimensione del quadrato e da un set di coordinate intere nel piano che costituiscono i possibili punti di allocazione. La dimostrazione che questo problema è in NP è piuttosto diretta; se per una certa istanza esiste un pattern che consente di dare una risposta affermativa al *decision problem* [19], un certificato computabile in tempo polinomiale è dato semplicemente dalla lista dei punti in cui viene inserito un box. Consideriamo il corrispondente problema di packing con rettangoli identici, anziché con quadrati, con la possibilità di avere due orientazioni, verticale e orizzontale, per i rettangoli; si vede che anche questa versione del problema è nella classe NP, usando una semplice estensione del certificato precedente per tenere conto anche delle orientazioni. Osserviamo però che se consideriamo un'istanza (L, W, l, w) del PLP, questo certificato *non* è polinomiale nella dimensione dell'input del problema, ossia non è polinomialmente limitato da $(\log L + \log W + \log l + \log w)$ (o equivalentemente da $\log L$). Non è quindi possibile dimostrare per questa via che il PLP è in NP.
D'altra parte il set "rettangolare" di punti di allocazione del PLP è una forte restrizione rispetto al caso considerato in [18].
In conclusione tra i due problemi non è possibile stabilire direttamente relazioni di dominanza.

In realtà, come osserva Nelißen [25], attualmente non ci sono prove dell'appartenenza del PLP alla classe NP; non si conoscono codifiche di dimensione polinomiale in $\log L$ per descrivere un pattern ammissibile.

Supponiamo di volere uno schema per generare tutti i possibili pattern con n box: in [25] è proposta la seguente procedura: per ogni box, si decida l'orientazione (verticale o orizzontale) e quindi lo si inserisca nel pattern in costruzione sistemandolo dapprima più in basso possibile, e quindi più a sinistra possibile; questa operazione di traslazione, che produce impaccamenti *normalizzati in basso a sinistra*, ovviamente esclude la generazione di pattern *non normalizzati*, ma si noti che per ogni pattern non normalizzato ne esiste uno equivalente normalizzato. Con questo procedimento vengono costruiti 2^n possibili pattern (alcuni equivalenti per simmetrie, o perché alcuni box non possono essere inseriti per mancanza di spazio), individuati dunque da una stringa di n bit in base alle orientazioni possibili.

Vogliamo far notare tuttavia che in questo modo non vengono generati tutti i possibili pattern normalizzati. La figura 1.1 è un esempio di pattern non ottenibile con la procedura descritta (si può verificare che il procedimento di Nelißen non consente di inserire il box in alto a sinistra).
È necessario precisare ulteriormente la disposizione dei box, indicando, oltre all'orientazione, anche *dove* va posto il box corrente. In questo modo ogni pattern normalizzato può essere generato. La stringa necessaria a descrivere un pattern risulta dominata da $O(n \log n)$; si noti che n può essere dell'ordine di LW/lw.

In questa discussione si è sottinteso che la dimensione del problema sia banalmente la dimensione dell'input. Tuttavia questa definizione non è la più naturale; è abbastanza chiaro infatti che la complessità di un'istanza PLP è collegata strettamente al numero massimo di box inseribili n_{OPT}, più che alla dimensione massima L del pallet. Il risultato di Dowsland, presentato nella prossima sezione, conferma questo punto di vista: esiste una divisione delle istanze del PLP in classi di equivalenza, e i membri di ogni classe, pur condividendo lo stesso insieme di pattern ammissibili, posseggono un range illimitato di valori per L. Dal punto di vista pratico perciò può essere sensato misurare la "dimensione" di un problema con n_{OPT}, o con la sua approssimazione $\lfloor LW/lw \rfloor$.

1.3 Le classi di equivalenza di Dowsland

Fissiamo un sistema di coordinate in modo che uno dei lati maggiore del rettangolo $L \times W$ sia sull'asse delle x, uno dei lati minori sia sull'asse delle y, e il rettangolo si trovi sul quadrante positivo.

Definiamo ora il concetto di normalizzazione: un pattern è detto *normalizzato in basso a sinistra*, o più brevemente *normalizzato*, se non è possibile "spingere" più in basso o più a sinistra nessun box del pattern. Per ogni pattern esiste un corrispondente pattern normalizzato, con lo stesso numero di box (per una dimostrazione di questo fatto si veda [10]); tale pattern normalizzato si può ottenere spingendo ricorsivamente tutti i box in basso e a sinistra. È sufficiente quindi limitarsi a considerare solo questo tipo di pattern.

In un pattern normalizzato l'angolo in basso a sinistra (x_{bs}, y_{bs}) di ogni box soddisfa

$$\begin{aligned} x_{bs} &= n_1 * l + m_1 * w \\ y_{bs} &= n_2 * l + m_2 * w \end{aligned}$$

per qualche coppia di interi non negativi (n_1, m_1) e (n_2, m_2). Ogni box infatti è appoggiato sia a sinistra che in basso su una "pila" di altri box la cui dimensione è costituita da una combinazione delle dimensioni dei box. Le coppie (n_1, m_1) e (n_2, m_2) possono essere usate come "coordinate" del box.

Sia S una delle dimensioni del pallet, e (n, m) una coppia ordinata di interi non negativi. La coppia (n, m) costituisce una *partizione ammissibile* di S se

$$n * l + m * w \leq S.$$

Una partizione ammissibile quindi è una combinazione di segmenti di lunghezza l e w che può essere sistemata in S. Se in più vale

$$0 \leq S - n * l - m * w < w,$$

allora (n, m) costituisce una *partizione efficiente* di S; in altre parole non può essere aggiunto nessun box a questa partizione. Ci sarà una partizione efficiente per ogni n che soddisfa

$$0 \leq n \leq \left\lfloor \frac{S}{l} \right\rfloor.$$

L'insieme delle partizioni ammissibili per L sarà denotato con $F(L, l, w)$; l'insieme delle partizioni efficienti per L sarà indicato con $E(L, l, w)$; per W si useranno rispettivamente le notazioni $F(W, l, w)$ e $E(W, l, w)$.
Si osservi che due istanze (L, W, l, w) e (L', W', l', w') hanno lo stesso insieme di partizioni efficienti

$$\begin{aligned} E(L, l, w) &= E(L', l', w') \\ E(W, l, w) &= E(W', l', w') \end{aligned}$$

se e solo se hanno lo stesso insieme di partizioni ammissibili

$$\begin{aligned} F(L, l, w) &= F(L', l', w') \\ F(W, l, w) &= F(W', l', w'). \end{aligned}$$

Diremo che queste due istanze sono equivalenti; l'insieme delle istanze del PLP risulta così partizionato in classi di equivalenza.

K. A. Dowsland [10] dimostra il seguente risultato:

Teorema 1.1 (Dowsland 1984) *Se due istanze I e I' del PLP hanno lo stesso insieme di partizioni efficienti, per ogni pattern P realizzabile in I esiste un corrispondente pattern P' realizzabile in I'.*
La corrispondenza è realizzata come segue: per ogni box in P il corrispondente box in P' mantiene la stessa orientazione; inoltre, se in P un box ha ascissa $x_{bs} = nl+mw$, in P' avrà $x'_{bs} = \max\{pl'+qw'; (p, q) \in F(L, l, w), pl+qw \leq x_{bs}\}$; formule simili valgono per la coordinata y'_{bs}.

Come conseguenza, due istanze equivalenti hanno lo stesso insieme di soluzioni. I membri di ogni classe di equivalenza perciò condividono lo stesso ottimo. Un upper bound sul numero di box trovato per un'istanza, può essere immediatamente esteso a tutta la classe.

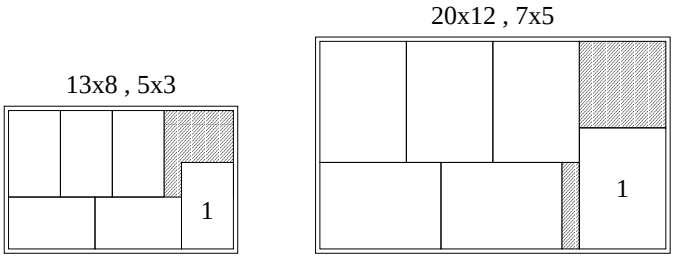

Figura 1.2: Controesempio per il teorema di Exeler

Si può vedere che due istanze (L, W, l, w) e (L', W', l', w') sono equivalenti se e solo se valgono le seguenti condizioni:

$$nl' + mw' \leq L' \quad \forall (n, m) \in E(L, l, w) \tag{1.1}$$

$$nl' + (m+1)w' > L' \quad \forall (n, m) \in E(L, l, w) \tag{1.2}$$

$$(\left\lfloor \frac{L}{l} \right\rfloor + 1)l' > L', \tag{1.3}$$

insieme con analoghe condizioni per W,W' e $E(W, l, w)$.

Se moltiplichiamo tutte le dimensioni di un'istanza per una costante reale positiva, otteniamo un problema equivalente. Perciò ogni istanza (L, W, l, w) può essere rappresentata dalla sua equivalente $(1, \frac{W}{L}, \frac{l}{L}, \frac{w}{L})$, che dipende solo da tre parametri, e quindi può essere vista come un punto nello spazio tridimensionale (chiaramente i parametri non saranno più interi). Consideriamo le disequazioni (1.1), (1.2), (1.3): fissando L, W, l, w esse definiscono una serie di piani nello spazio tridimensionale che limitano i valori possibili per $\frac{W'}{L'}$, $\frac{l'}{L'}$, $\frac{w'}{L'}$. Perciò ogni classe di equivalenza in questa rappresentazione è costituita dai punti di un poliedro semiaperto. In generale una classe di equivalenza conterrà infiniti membri o nessuno.
Uno studio di questa rappresentazione è effettuato in [10].

Exeler [16] dimostra una versione diversa del teorema 1.1; la tesi dimostrata è la seguente: se I e I' sono due istanze equivalenti, dato un pattern P di I, è possibile costruire un corrispondente pattern P' per I' nel seguente modo: se nel pattern P un box ha ascissa $x_{bs} = nl + mw$, in P' il corrispondente box avrà $x'_{bs} = nl' + mw'$, e analogamente per l'altra coordinata y'_{bs}; l'orientazione rimane invariata. In altre parole, il pattern P' corrispondente viene costruito usando le stesse partizioni ammissibili utilizzate in P. Benché questo risultato sia più plausibile di quello di Dowsland, in realtà è falso. La figura 1.2 illustra un controesempio. Le istanze $(13, 8, 5, 3)$ e $(20, 12, 7, 5)$ sono equivalenti; infatti $E(13, 5, 3) = \{(0, 4), (1, 2), (2, 1)\} = E(20, 7, 5)$, e $E(8, 5, 3) = \{(0, 2), (1, 1)\} = E(12, 7, 5)$. Tuttavia dalla figura si vede che

il box 1 ha ascissa $x_{bs} = 10$ corrispondente alla partizione $(2, 0)$ nel primo pattern; nel secondo pattern ha invece ascissa $x_{bs} = 15$, partizione $(0, 3)$. Nel secondo pattern non è possibile disporre il box 1 in una posizione individuata dalle stesse partizioni usate nel primo pattern.
Si noti che la partizione $(0, 3)$ ha lunghezza inferiore alla partizione $(2, 0)$ nel caso a sinistra, e superiore nel caso a destra.

1.4 Il teorema di Barnes

Nel 1973 Brualdi e Foregger [5] studiano l'impaccamento ortogonale ottimale di box identici in due e tre dimensioni, nel caso in cui le dimensioni del box siano una multipla dell'altra (box *armonici*).
Per semplicità, il risultato principale di Brualdi e Foregger per l'impaccamento nel caso bidimensionale verrà presentato nella versione data da Barnes [1].
Consideriamo d'ora in poi solo istanze in cui le dimensioni del box sono prime tra loro. Infatti se $d = \text{mcd}\{l, w\} \neq 1$, si può costruire un'istanza equivalente $(\lfloor \frac{L}{d} \rfloor, \lfloor \frac{W}{d} \rfloor, \frac{l}{d}, \frac{w}{d})$.

Teorema 1.2 (Brualdi e Foregger 1974) *Siano $L, W \geq n$ interi positivi, e*

$$a = L \bmod n$$
$$b = W \bmod n.$$

L'area inutilizzata nel miglior impaccamento possibile di box $1 \times n$ in un rettangolo $L \times W$ è:

$$A = \begin{cases} ab & se \quad a + b \leq n, \\ (n-a)(n-b) & se \quad a + b \geq n. \end{cases} \tag{1.4}$$

I pattern che realizzano l'impaccamento ottimale sono ottenuti con una divisione in blocchi del rettangolo; si veda la figura 1.3; ogni blocco viene riempito completamente con box orientati nella stessa direzione. I due schemi si riferiscono rispettivamente al caso $a + b \leq n$ (schema (i)), e al caso $a + b \geq n$ (schema (ii)). L'area tratteggiata costituisce lo spazio inutilizzato.

Sia $A = A(L, W)$ l'area inutilizzata nell'impaccamento ottimale di box $1 \times l$ in un rettangolo $L \times W$; similmente, sia $B = B(L, W)$ l'area inutilizzata nell'impaccamento ottimale di box $1 \times w$ in un rettangolo $L \times W$. l e w siano primi tra loro. Definiamo $R = R(L, W)$ come il più piccolo intero tale che

$$R \geq \max\{A, B\}, \qquad R \equiv A \pmod{l}, \qquad R \equiv B \pmod{w}.$$

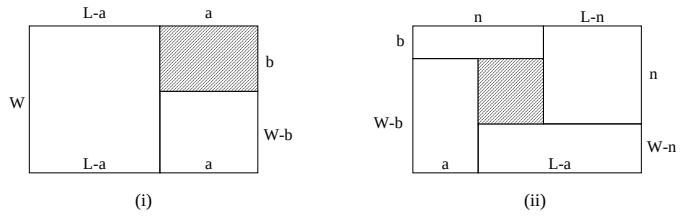

Figura 1.3: Schema a blocchi per i pattern ottimali del teorema di Brualdi e Foregger (le proporzioni non sono rispettate).

Si osservi che R costituisce un limite inferiore all'area inutilizzata nel miglior impaccamento possibile di box $l \times w$ in un rettangolo $L \times W$; infatti, poiché ogni box $l \times w$ può essere partizionato in strisce $1 \times l$ o $1 \times w$, l'area rimasta è almeno $\max\{A, B\}$. Ma quest'area deve essere anche congruente a $LW \pmod{l}$, e di conseguenza a $A \pmod{l}$, e allo stesso modo deve essere congruente a $B \pmod{w}$.
Ma Barnes [1] dimostra un risultato ancora più sorprendente:

Teorema 1.3 (Barnes 1978) *Se L e W sono sufficientemente grandi, l'area inutilizzata nel miglior impaccamento possibile di box $l \times w$ in un rettangolo $L \times W$ è precisamente $R(L, W)$*

In altre parole il bound $R(L, W)$ coincide con l'ottimo per dimensioni del pallet sufficientemente grandi, *una volta fissati l e w*.

Si noti l'esistenza di un risultato "simile" nel caso unidimensionale delle partizioni ammissibili: se l e w sono primi tra loro, per X sufficientemente grande (è sufficiente $X \geq lw$) esiste sempre una partizione $(n, m) \in F(X, l, w)$ tale che $nl + mw = X$. Come per il teorema di Barnes, quando viene superata una soglia critica nelle dimensioni del contenitore, le combinazioni possibili dei pezzi sono in numero sufficiente a garantire il raggiungimento del bound.

Capitolo 2

Upper bounds

Lo studio di upper bound sul numero massimo di box impaccabili in una determinata istanza del PLP è motivato soprattutto, oltre che da ragioni teoriche, dall'esigenza di rendere efficiente la ricerca negli algoritmi che costruiscono soluzioni esatte o approssimate del PLP, e di verificare l'accuratezza delle soluzioni approssimate.
Rispetto ad altri problemi di packing nel PLP si presenta una situazione estremamente favorevole, in cui upper bound relativamente semplici da calcolare corrispondono al valore ottimo in larga parte delle istanze; inoltre con qualsiasi upper bound si ha una stima dell'ottimo con un errore relativo che va asintoticamente a 0, al crescere delle dimensioni del pallet, una volta fissate le dimensioni del box. Nel caso particolare del bound di Barnes, si è visto che addirittura l'errore assoluto va a 0, al crescere delle dimensioni del pallet (si veda la sezione 1.4). Ciononostante, dimostreremo nella sezione finale di questo capitolo che l'errore assoluto risultante con il bound di Barnes non è limitato superiormente nell'insieme delle istanze del PLP.

In questo capitolo vengono presentati diversi upper bound proposti in letteratura, che in alcuni casi si basano su importanti risultati teorici, come nel caso del bound di Barnes o della minimizzazione sulle classi di equivalenza di Dowsland.
Nella sezione 2.7 discutiamo un approccio ILP a variabili 0-1, che costituisce una formulazione equivalente del PLP, e l'upper bound ricavato con esso.
Nella sezione 2.8 viene affrontato il problema dell'accuratezza di alcuni upper bound; i teoremi dimostrati in questa sezione costituiscono risultati originali.

In diversi lavori è stata valutata empiricamente l'efficacia dei vari upper bound. Il set di problemi generalmente preso in considerazione [9, 10, 16, 25] per queste valutazioni, è quello individuato dai seguenti vincoli:

$$6 \leq \frac{LW}{lw} \leq 51 \tag{2.1}$$

$$1 \leq \frac{l}{w} \leq 4 \tag{2.2}$$

$$1 \leq \quad \frac{L}{W} \quad \leq 2 \tag{2.3}$$

(L'intervallo (2.1) può essere leggermente diverso in alcuni lavori).
Le istanze di questo set saranno dette *di tipo I*. Si ritiene generalmente che questo insieme di istanze sia sufficientemente rappresentativo dei problemi incontrati nella pratica. Gli upper bound discussi sono stati testati in diversi lavori su campioni di istanze di tipo I.
Nei lavori più recenti [26, 24, 31], in seguito al miglioramento delle tecniche e degli algoritmi, si è potuto compiere test numerici sul seguente set più impegnativo:

$$51 \leq \quad \frac{LW}{lw} \quad \leq 100 \tag{2.4}$$

$$1 \leq \quad \frac{l}{w} \quad \leq 4 \tag{2.5}$$

$$1 \leq \quad \frac{L}{W} \quad \leq 2. \tag{2.6}$$

Istanze di questo set saranno dette *di tipo II*.

Un primo limite superiore elementare si ottiene dal vincolo che l'area occupata dai box deve essere inferiore all'area del rettangolo; il numero massimo di box contenuti in un pattern verifica perciò $n_{OPT} \leq \lfloor \frac{L*W}{l*w} \rfloor$. Il *bound dell'area* così definito è accurato solo nel 10% dei casi [9], considerando un campione di istanze di tipo I (salvo diversa precisazione, nel seguito del capitolo queste percentuali saranno riferite sempre a campioni di istanze di tipo I).

2.1 Riduzione alle massime dimensioni utilizzabili

Si ottiene un notevole miglioramento del bound quando si tiene conto che in molti casi i pattern normalizzati, descritti nella sezione 1.3, si limitano ad occupare un rettangolo più piccolo di quello dato.

Infatti in base alla definizione di pattern normalizzato la posizione di ogni angolo dei box risulta essere una combinazione lineare intera delle dimensioni l e w. La massima ascissa occupata dai box è quindi

$$L^* = \max\{nl + mw; (n, l) \in F(L, l, w)\},$$

e la massima ordinata è

$$W^* = \max\{nl + mw; (n, l) \in F(W, l, w)\},$$

dove $F(L, l, w)$ e $F(W, l, w)$ sono le partizioni ammissibili di L e W definite in sezione 1.3. Calcolando il bound dell'area con le nuove dimensioni ridotte (L^*, W^*), si ottiene un valore coincidente con l'ottimo nel 72% dei casi [9, 25].

2.2 Bound di Barnes

Come abbiamo visto nella sezione 1.4, il bound trovato da Barnes dà il valore corretto dell'ottimo quando le dimensioni del rettangolo sono abbastanza grandi, una volta fissate le dimensioni del box. Tuttavia per il set di problemi di tipo I il bound di Barnes dà un risultato migliore di quello dell'area solo nel 20% dei casi. Se però viene usato in congiunzione con la riduzione alle massime dimensioni usabili (sezione 2.1), il valore del bound coincide con l'ottimo nell'82% dei casi [25].

2.3 Bound di Keber

In alcuni casi l'upper bound può essere decrementato in base alla seguente considerazione dovuta a Keber (riportata da Nelißen in [25]): se n_{up} è un limite superiore per l'istanza (L, W, l, w), allora n_{up} è un limite superiore valido anche per l'istanza (L, W, l', w') con $l' \geq l$, $w' \geq w$. Questo fatto è utile perché può accadere che l'upper bound calcolato per il primo problema sia migliore rispetto a quello disponibile per il secondo problema. Un esempio è il problema $(24, 24, 6, 5)$, per il quale l'upper bound dell'area, calcolato usando l'area del rettangolo ridotto (sezione 2.1), vale $n_A = 19$; se però si considera il problema $(24, 24, 5, 5)$ e si usa anche in questo caso l'area del rettangolo ridotto si ottiene un bound $n_A = 16$. Si può quindi assegnare per l'istanza $(24, 24, 6, 5)$ un upper bound $n_A = 16$ (incidentalmente questo valore corrisponde all'ottimo).

Per un dato problema (L, W, l, w) Keber propone di esaminare tutte le istanze (L, W, i, j) con $l' \leq i \leq l$ e $w' \leq j \leq w$, dove l' e w' sono scelti arbitrariamente; si può fissare per esempio $l' = 0.7l$ e $w' = 0.7w$. Viene calcolato il minimo degli upper bound trovati per queste istanze. Un'implementazione di questo metodo in congiunzione col bound di Barnes (sezione 2.2) e l'uso del rettangolo ridotto (sezione 2.1) ha ottenuto il 95% di successi [25, 26].
Naujoks propone invece un'altra procedura: i quattro parametri (L, W, l, w) vengono divisi per una variabile intera n, che può assumere alcuni o tutti i valori nel range $[2, w]$, e si calcola il minimo degli upper bound trovati per tutti i problemi $(L/n, W/n, l/n, w/n)$. Anche questo metodo in congiunzione con l'uso del rettangolo ridotto e del bound di Barnes ha fornito il valore corretto dell'ottimo nel 95% dei casi [25, 26].

2.4 Bound LP

Un approccio LP fu introdotto da Isermann nel 1987 (riportato da Nelißen in [25]), e successivamente esteso al Rectangle Packing Problem da Schei-

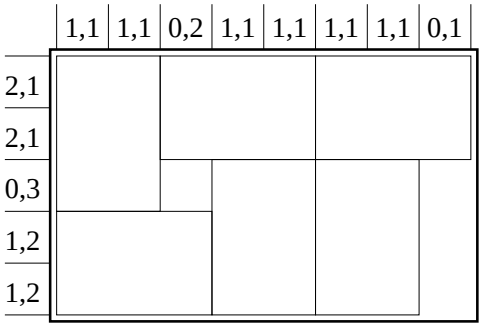

Figura 2.1: Decomposizione di un impaccamento nella formulazione LP

thauer [30]. Anche in questo caso viene sfruttata l'importante nozione di partizione ammissibile, vista nel capitolo precedente.

Supponiamo di avere un impaccamento e di "tagliare" il rettangolo $L \times W$ in strisce orizzontali di larghezza unitaria. Ogni striscia $L \times 1$ conterrà elementi $l \times 1$ e $w \times 1$, e quindi corrisponde ad una partizione ammissibile di L (si fa riferimento solo al numero degli elementi, e non alla loro posizione). La stessa procedura può essere ripetuta in senso verticale, con strisce $1 \times W$, ognuna delle quali corrisponderà ad una partizione ammissibile di W.

Ogni impaccamento perciò può essere visto come un insieme di partizioni orizzontali $(i, j) \in F(L, l, w)$ e partizioni verticali $(f, g) \in F(W, l, w)$. L'esempio in figura 2.1 mostra un pattern per l'istanza $(8, 5, 3, 2)$, che risulta essere composto dalle partizioni orizzontali $(2, 1)$, $(2, 1)$, $(0, 3)$, $(1, 2)$, $(1, 2)$, e dalle partizioni verticali $(1, 1)$, $(1, 1)$, $(0, 2)$, $(1, 1)$, $(1, 1)$, $(1, 1)$, $(1, 1)$, $(0, 1)$.

Sia $x_{i,j}$ il numero delle partizioni orizzontali $(i, j) \in F(L, l, w)$ presenti in un pattern; allo stesso modo, sia $y_{f,g}$ il numero delle partizioni verticali $(f, g) \in F(W, l, w)$ contenute nel pattern. Nell'esempio in figura $x_{2,1} = 2$, $x_{1,2} = 2$, $x_{0,3} = 1$; $y_{1,1} = 6$, $y_{0,2} = 1$, $y_{0,1} = 1$; per tutte le altre partizioni (i, j) e (f, g), $x_{i,j} = 0$ e $y_{f,g} = 0$.

Ogni pattern ammissibile per l'istanza (L, W, l, w) dovrà verificare i seguenti vincoli:

$$\sum_{(i,j)\in F(L,l,w)} i * l * x_{i,j} - \sum_{(f,g)\in F(W,l,w)} g * w * y_{f,g} = 0 \tag{2.7}$$

$$\sum_{(i,j)\in F(L,l,w)} j * w * x_{i,j} - \sum_{(f,g)\in F(W,l,w)} f * l * y_{f,g} = 0 \tag{2.8}$$

$$\sum_{(i,j)\in F(L,l,w)} x_{i,j} \leq W \tag{2.9}$$

$$\sum_{(f,g)\in F(W,l,w)} y_{f,g} \leq L \tag{2.10}$$

$$x_{i,j} \geq 0 \quad \forall (i, j) \in F(L, l, w) \tag{2.11}$$

$$y_{f,g} \geq 0 \quad \forall (f,g) \in F(W,l,w). \tag{2.12}$$

Il vincolo (2.9) richiede che il numero di strisce orizzontali non superi la larghezza del rettangolo. Naturalmente il segno $\leq$ può essere sostituito dal segno di uguaglianza, se includiamo anche la variabile $x_{0,0}$. Allo stesso modo, (2.10) chiede che il numero di strisce verticali non superi la lunghezza del rettangolo. Il vincolo (2.7) impone che l'area occupata dagli elementi orizzontali $l \times 1$ sia la stessa occupata dagli elementi verticali $1 \times w$, in quanto entrambi sono ottenuti "affettando" i box orizzontali del pattern. Per ogni striscia orizzontale corrispondente alla partizione (i,j), l'area occupata dagli elementi orizzontali $l \times 1$ è $i * l$; la prima sommatoria in (2.7) somma quest'area per tutte le strisce orizzontali. Allo stesso modo, per ogni striscia verticale corrispondente alla partizione (f,g), l'area occupata dagli elementi verticali $1 \times w$ è $g * w$; la seconda sommatoria in (2.7) somma quest'area per tutte le strisce verticali.
Il vincolo (2.8) impone la stessa cosa per i box verticali.
Si noti che i vincoli (2.7) - (2.12) con in più il vincolo di interezza delle variabili costituiscono condizioni necessarie ma non sufficienti per l'esistenza di un pattern ammissibile.

Resta da definire ora una funzione obiettivo. Poiché ogni box orizzontale (*H-box*) è costituito da w elementi orizzontali $l \times 1$, il numero di H-box in un pattern sarà:

$$H = \sum_{(i,j) \in F(L,l,w)} \frac{i * x_{i,j}}{w}. \tag{2.13}$$

Similmente, ogni box verticale (*V-box*) sarà costituito da w elementi verticali $1 \times l$. Quindi il numero dei V-box è

$$V = \sum_{(f,g) \in F(W,l,w)} \frac{f * y_{f,g}}{w}. \tag{2.14}$$

Consideriamo la seguente funzione obiettivo:

$$\max z = H + V = \sum_{(i,j) \in F(L,l,w)} \frac{i * x_{i,j}}{w} + \sum_{(f,g) \in F(W,l,w)} \frac{f * y_{f,g}}{w}. \tag{2.15}$$

Per ogni pattern ammissibile, z conta il numero di box presenti.

La parte intera della soluzione del problema (2.15), soggetto a (2.7)-(2.12), costituisce un upper bound per l'istanza del PLP data. È pratica comune risolvere la versione rilassata del problema, usando variabili $x_{i,j}$ e $y_{f,g}$ continue. Osserviamo tuttavia che il numero di variabili $x_{i,j}$ e $y_{f,g}$ è una funzione esponenziale dell'input del problema PLP.

Naujoks, come riportato in [25], propone un raffinamento del metodo: sia U l'upper bound ottenuto risolvendo il problema LP appena definito. Si

introduca questo vincolo ulteriore, in aggiunta ai sei già definiti,

$$H + V = \sum_{(i,j)\in F(L,l,w)} \frac{i * x_{i,j}}{w} + \sum_{(f,g)\in F(W,l,w)} \frac{f * y_{f,g}}{w} \geq U, \tag{2.16}$$

e si risolvano i problemi LP che si ottengono con queste funzioni obiettivo:

$$\max H = \sum_{(i,j)\in F(L,l,w)} \frac{i * x_{i,j}}{w} \tag{2.17}$$

e

$$\max V = \sum_{(f,g)\in F(W,l,w)} \frac{f * y_{f,g}}{w}. \tag{2.18}$$

In altre parole, massimizziamo indipendentemente il numero degli H-box e dei V-box, con la condizione che l'upper bound U sia effettivamente raggiunto. Se i valori ottimi trovati $\hat{H}$ e $\hat{V}$ sono tali che $\lfloor \hat{H} \rfloor + \lfloor \hat{V} \rfloor < U$, non possono esistere impaccamenti ammissibili con un numero di box pari a U e in tal caso l'upper bound U viene decrementato di 1.

2.5 Minimizzazione del bound sulle classi di equivalenza

Come si è visto nel capitolo precedente, le istanze del PLP contenute in una classe di equivalenza condividono lo stesso ottimo. Un upper bound trovato per un'istanza vale di conseguenza per tutta la classe. Sfruttando questo fatto, è possibile migliorare i bound visti in precedenza, individuando il valore minimo che assumono all'interno della classe di equivalenza a cui appartiene il problema originario.

In diversi lavori [10, 16, 25] è stato sviluppato un procedimento che permette di trovare il minimo del bound dell'area su una classe di equivalenza. In Dowsland [10] i dettagli del procedimento adottato sono piuttosto vaghi; tuttavia il nuovo metodo, applicato ad un campione di 5000 istanze, fornisce il valore corretto per l'ottimo nel 92% dei casi, mentre nel restante 8% l'accuratezza del bound rimane sconosciuta. Successivamente [12] viene verificato che il nuovo bound fornisce il valore corretto dell'ottimo nel 98% dei casi. La verifica è stata eseguita con un algoritmo esatto che verrà descritto nel capitolo 4. Dowsland inoltre ricava che le istanze di tipo I con il vincolo (2.1) sostituito da

$$1 \leq \frac{LW}{lw} < 51,$$

sono costituite da 8565 classi di equivalenza. Solo per 204 di queste (2.4%) l'upper bound non fornisce il valore esatto dell'ottimo.

Nelißen [25] adotta la seguente procedura, che costituisce una versione corretta di un metodo proposto da Exeler [16].

Sia (L, W, l, w) la nostra istanza. Poiché un riscalamento delle dimensioni del problema non cambia il valore del bound dell'area, ci riduciamo a considerare solo problemi equivalenti di tipo $(X, Y, x, 1)$, in cui la larghezza del box è unitaria (le rimanenti dimensioni X, Y e x non saranno più necessariamente intere). Un'istanza $(X, Y, x, 1)$ sarà equivalente a (L, W, l, w) se e solo se verifica (vedi le disequazioni (1.1)-(1.2))

$$nx + m \leq X \quad \forall (n, m) \in E(L, l, w) \tag{2.19}$$

$$nx + m + 1 > X \quad \forall (n, m) \in E(L, l, w) \tag{2.20}$$

$$\left(\max_{(n,m) \in E(L,l,w)} n + 1 \right) x > X, \tag{2.21}$$

$$px + q \leq Y \quad \forall (p, q) \in E(W, l, w) \tag{2.22}$$

$$px + q + 1 > Y \quad \forall (p, q) \in E(W, l, w) \tag{2.23}$$

$$\left(\max_{(p,q) \in E(W,l,w)} p + 1 \right) x > Y \tag{2.24}$$

(Il metodo di Exeler [16] tralascia le disuguaglianze (2.21) e (2.24), di conseguenza l'intero procedimento è invalidato, come nota Nelißen [25]. È utile osservare che l'errore procedurale di Exeler qui accennato non ha niente a che vedere con l'errore da noi corretto nel primo capitolo del presente lavoro.)

Si supponga che per un certo valore di x esistano istanze che verificano le (2.19)-(2.24); tra queste, l'istanza con area XY minima sarà quella corrispondente al rettangolo ridotto $(X^*, Y^*, x, 1)$, dove

$$\begin{aligned} X^* &= \max_{(n,m) \in E(L,l,w)} nx + m, \\ Y^* &= \max_{(p,q) \in E(W,l,w)} px + q. \end{aligned}$$

Per questa istanza il rapporto area rettangolo/area box si riduce ad essere una funzione di x. Il procedimento si riduce quindi a cercare il minimo di questa funzione sui valori di x permessi dalle (2.19)-(2.24). Questa trasformazione a un problema unidimensionale è dovuta a Naujoks (riportata da Nelißen in [25]).

Si assegni un indice ad ogni coppia di $E(L, l, w)$ e $E(W, l, w)$: sia $r = |E(L, l, w)|$, $s = |E(W, l, w)|$, $R = \{1, 2, \ldots, r\}$ e $S = \{1, 2, \ldots, s\}$. In base alle (2.19)-(2.24), per un certo valore di x esiste un'istanza equivalente a (L, W, l, w) se e solo se

$$n_i x + m_i + 1 > n_j x + m_j \quad \forall i \neq j \in R \tag{2.25}$$

$$\left(\max_{i \in R} n_i + 1 \right) x > n_j x + m_j \quad \forall j \in R \tag{2.26}$$

$$p_i x + q_i + 1 \quad > p_j x + q_j \quad \forall i \neq j \in S \tag{2.27}$$

$$\left(\max_{i \in S} p_i + 1\right) x \quad > p_j x + q_j \quad \forall j \in S. \tag{2.28}$$

Con qualche trasformazione si arriva alle seguenti condizioni equivalenti:

$$x > \quad \frac{m_j - m_i - 1}{n_i - n_j} \quad \forall i \neq j \in R : n_i > n_j \tag{2.29}$$

$$x < \quad \frac{m_j - m_i + 1}{n_i - n_j} \quad \forall i \neq j \in R : n_i > n_j \tag{2.30}$$

$$x > \quad \frac{m_j}{\max_{i \in R} n_i + 1 - n_j} \quad \forall j \in R \tag{2.31}$$

$$x > \quad \frac{q_j - q_i - 1}{p_i - p_j} \quad \forall i \neq j \in R : p_i > p_j \tag{2.32}$$

$$x < \quad \frac{q_j - q_i + 1}{p_i - p_j} \quad \forall i \neq j \in R : p_i > p_j \tag{2.33}$$

$$x > \quad \frac{q_j}{\max_{i \in R} p_i + 1 - p_j} \quad \forall j \in R. \tag{2.34}$$

Poiché tutte le disuguaglianze sono strette, esse definiscono per x un intervallo aperto (a, b), con $a, b \in \mathbf{Q}$; (rispetto alla trattazione di Nelißen aggiungiamo che, poiché per definizione $x \geq 1$, se $a < 1$ si deve considerare l'intervallo $[1, b)$). Per ogni x appartenente a questo intervallo, si costruisca l'istanza equivalente $(X(x), Y(x), x, 1)$, con $X(x) = \max_{(n,m) \in E(L,l,w)} nx + m$, e $Y(x) = \max_{(p,q) \in E(W,l,w)} px + q$. Le funzioni $X(x)$ e $Y(x)$, definite in (a, b), risultano essere poligonali.

Il metodo procede come segue: l'intervallo (a, b) viene partizionato in subintervalli individuati dai punti $x_1 = a < x_2 < \ldots < x_k = b$, dove $\{x_2, \ldots, x_{k-1}\}$ è l'insieme dei punti di discontinuità della derivata prima di $X(x)$ o della derivata prima di $Y(x)$.
In ogni subintervallo $(x_i, x_{i+1}]$ (o (x_i, x_{i+1}) se $x_{i+1} = b$) il rapporto tra le aree del rettangolo e del box è dato dalla funzione:

$$f(x) = \frac{(n_i x + m_i)(p_i x + q_i)}{x}, \tag{2.35}$$

dove (n_i, m_i) e (p_i, q_i) sono le partizioni che realizzano il massimo rispettivamente di $nx + m$ e di $px + q$ nel subintervallo dato (*partizioni dominanti*). La funzione $f(x)$ è razionale e convessa; il minimo si trova perciò o nel punto $x_0 = \sqrt{\frac{m_i q_i}{n_i p_i}}$ in cui si annulla la derivata prima, se x_0 è interno al subintervallo, o agli estremi del subintervallo .
Inoltre nei punti $\{x_2, \ldots, x_{k-1}\}$ la derivata destra di $f(x)$ è maggiore della derivata sinistra. Se ne conclude che su tutto l'intervallo (a, b) il rapporto tra l'area del rettangolo e del box è una funzione convessa di x. Ciò permette di usare la seguente procedura:

1. Sia $i = 1$.

2. Trova il subintervallo $(x_i, x_{i+1}]$ (o (x_i, x_{i+1}) se $x_{i+1} = b$), e le partizioni dominanti (n_i, m_i) e (p_i, q_i) in tale subintervallo.

3. Se $x_0 = \sqrt{\frac{m_i q_i}{n_i p_i}}$ è nell'intervallo $(x_i, x_{i+1}]$, o se $f(x_i) \leq f(x_{i+1})$, fine (a causa della convessità, il minimo globale è stato trovato).

4. Altrimenti:
se $x_{i+1} = b$, fine (il minimo si trova in b),
altrimenti incrementa i e vai a 2.

Il procedimento restituisce il minimo del rapporto area rettangolo/area box sulla classe di equivalenza: la parte intera di tale minimo costituisce il bound cercato. Un'istanza corrispondente che realizza il minimo sarà chiamata *problema equivalente minimale*. Gli estremi dell'intervallo (a, b) andrebbero esclusi dalla ricerca, poiché si tratta di un intervallo aperto; tuttavia, nel caso che il minimo si trovi in a (o in b), la sua parte intera è comunque un bound valido; infatti per $x = a + \epsilon$ ($x = b - \epsilon$) si hanno istanze ammissibili, e $\lfloor f(a) \rfloor = \lfloor f(a + \epsilon) \rfloor$ ($\lfloor f(b) \rfloor = \lfloor f(b - \epsilon) \rfloor$) pur di prendere ϵ abbastanza piccoli; si ottiene dunque lo stesso bound.

2.6 Altri vincoli strutturali: bound di Nelißen

Nel 1993 Nelißen [25, 26] introduce un nuovo metodo per il calcolo dell'upper bound, che combina l'approccio LP con l'uso di altri vincoli strutturali.

La formulazione LP vista nella sezione 2.4 costituisce un rilassamento del problema PLP, anche quando si mantiene il vincolo di interezza per le variabili; come conseguenza il bound LP è a volte poco accurato. L'approccio di Nelißen interviene proprio su questo aspetto, sottoponendo l'upper bound LP ad una verifica con ulteriori vincoli.

La tecnica è la seguente: si suppone che il bound LP sia accurato, in altre parole si suppone che corrisponda all'ottimo. Con questa ipotesi si ricavano quindi delle condizioni necessarie che devono essere verificate da una qualsiasi soluzione: se si verifica un'inconsistenza tra queste, se ne deduce che l'assunzione di partenza è sbagliata e si può decrementare il bound.

Nelißen applica il suo metodo non direttamente all'istanza data, ma ad un problema equivalente minimale. Ciò è motivato dalla constatazione empirica che il metodo presenta un numero maggiore di successi con questa scelta. D'altra parte in una soluzione ottima di un problema equivalente minimale il rapporto tra area residua e area del box è minimo (rispetto alle altre istanze della classe di equivalenza); nella generazione di pattern quindi si hanno a disposizione meno gradi di libertà e il numero di pattern ammissibili si riduce; il problema equivalente minimale risulta perciò più sensibile all'applicazione di ulteriori condizioni.

Sia U il valore del bound LP, ottenuto come visto nella sezione 2.4 (viene mantenuta qui la stessa notazione). Innanzitutto con i vincoli (2.7)-(2.12) e (2.16) vengono risolti i seguenti problemi LP:

$$\max H = \sum_{(i,j)\in F(L,l,w)} \frac{i * x_{i,j}}{w}, \tag{2.36}$$

per il numero di H-box,

$$\max V = \sum_{(f,g)\in F(W,l,w)} \frac{f * y_{f,g}}{w}, \tag{2.37}$$

per il numero di V-box;

$$\max z^L_{i,j} = x_{i,j} \tag{2.38}$$

$$\max z^W_{f,g} = y_{f,g}, \tag{2.39}$$

per tutte le L-partizioni efficienti $(i,j) \in E(L,l,w)$ e per tutte le W-partizioni efficienti $(f,g) \in E(W,l,w)$, rispettivamente. Per le L-partizioni e le W-partizioni non efficienti vengono risolti:

$$\max z^L_{ne} = \sum_{(i,j)\in F(L,l,w)/E(L,l,w)} x_{i,j} \tag{2.40}$$

$$\max z^W_{ne} = \sum_{(f,g)\in F(W,l,w)/E(W,l,w)} y_{f,g}. \tag{2.41}$$

Ognuno dei massimi trovati viene arrotondato all'intero inferiore. Inoltre per ciascuna funzione obiettivo sopraelencata, escluse H e V, viene calcolato allo stesso modo il minimo, di cui poi viene considerato l'arrotondamento all'intero superiore. I valori H_{min} e V_{min} si ottengono ponendo $H_{min} = U - V_{max}$ e $V_{min} = U - H_{max}$. La procedura fin qui descritta è stata introdotta da Naujoks (per riferimenti si veda [25]).
In questo modo si hanno a disposizione per il problema (L, W, l, w) le seguenti quantità:

1. Il massimo numero H_{max} e V_{max} di H-box e V-box.

2. Intervalli chiusi $\left[l_{x_{i,j}}, u_{x_{i,j}}\right]$ per tutte le partizioni efficienti $(i,j) \in E(L,l,w)$, e intervalli chiusi $\left[l_{y_{f,g}}, u_{y_{f,g}}\right]$ per tutte le partizioni efficienti $(f,g) \in E(W,l,w)$, che danno il range dei valori ammissibili per le variabili corrispondenti.

3. intervalli chiusi $[l_{x_{ne}}, u_{x_{ne}}]$ e $[l_{y_{ne}}, u_{y_{ne}}]$, che costituiscono il range dei valori ammissibili per la somma delle variabili corrispondenti alle L-partizioni non efficienti e alle W-partizioni non efficienti, rispettivamente.

A questo punto Nelißen dalle proprietà dei pattern del PLP ricava una serie di vincoli supplementari su H_{max} e V_{max}, e sui limiti superiori ed inferiori per le variabili $x_{i,j}$ e $y_{f,g}$. L'applicazione di questi vincoli può portare a restringere gli intervalli di ammissibilità ricavati sopra. Nel caso che un intervallo venga ridotto all'insieme vuoto, o che la somma $H_{max} + V_{max}$ scenda sotto il bound U, si arriva ad un'inconsistenza. Ciò permette di escludere l'esistenza di soluzioni con U box, e quindi il valore del bound viene decrementato di 1.
L'intero procedimento può essere ripetuto col nuovo valore del bound. L'applicazione del metodo ad un campione di istanze del tipo I (formule (2.1)-(2.3)) e del tipo II (formule (2.4)-(2.6)) ha fornito il valore esatto dell'ottimo rispettivamente nel 99.86% e nel 98.81% dei casi. La verifica dell'accuratezza del bound è stata effettuata con un algoritmo esatto. Il metodo viene applicato anche alle 204 classi di equivalenza, per cui il bound di Dowsland non aveva fornito un valore accurato (si veda la sezione 2.5); in 151 classi (74%) il nuovo approccio trova il valore corretto dell'ottimo. Attualmente questi risultati sono i migliori in letteratura per quanto riguarda l'accuratezza dell'upper bound.

2.7 Un approccio LP 0-1

Nelle sezioni precedenti è stato notato che nella formulazione ILP di Isermann i vincoli imposti costituiscono condizioni necessarie ma non sufficienti per l'esistenza di una soluzione.
In questa sezione presentiamo un approccio LP 0-1 che costituisce una formulazione *equivalente* del PLP. L'approccio è stato proposto in maniera indipendente da Morabito e Morales in [24]. Quest'approccio è stato introdotto da Beasley nel 1985 [2] per il Rectangle Packing Problem. Tuttavia è interessante derivare la stessa formulazione LP 0-1 per altra via. L'idea si basa sulla trasformazione del PLP in *Independent Set Problem*; questa stessa trasformazione è stata utilizzata da Dowsland [11] per costruire un algoritmo esatto per il PLP, come si vedrà nel capitolo 4; ma già Fowler ed altri [18] propongono la trasformazione di alcuni problemi di packing in Independent Set Problem.

Ricordiamo che, dato un grafo non orientato $G(V, E)$, un sottinsieme di vertici $V' \subseteq V$ è un *independent set* se nessuna coppia di vertici del sottinsieme è collegata da un arco di G. Il problema di ottimizzazione collegato chiede di trovare un independent set di cardinalità massima.

Dato un problema di PLP, costruiamo un'istanza dell'Independent Set Problem nel modo di seguito descritto. La disposizione di un singolo box nel pallet è individuata dalle coordinate del punto di inserimento (la posizione

di un box è data dalla posizione del suo angolo in basso a sinistra) e dall'orientazione del box, orizzontale o verticale; nel grafo $G(V, E)$ inseriamo un vertice per ogni disposizione possibile di un singolo box; ci sarà quindi un vertice per ogni combinazione possibile dei tre parametri che danno la posizione e l'orientazione del box. Colleghiamo con un arco le coppie di vertici per cui si ha sovrapposizione dei box corrispondenti. Un impaccamento risulta privo di sovrapposizioni tra box *se e solo se* nel corrispondente sottinsieme di vertici $V' \subseteq V$ non ci sono archi che collegano coppie di nodi. Con questa costruzione ad ogni pattern ammissibile del PLP corrisponde un independent set nel grafo, di pari cardinalità. Viceversa, ogni independent set del grafo individua un pattern per il PLP. Il problema di trovare un pattern con il massimo numero di box si riduce a quello di trovare un independent set di cardinalità massima.

Una semplice formulazione LP 0-1 per l'Independent Set Problem è la seguente:

$$\max \quad \sum_{i \in V} x_i \tag{2.42}$$

$$\text{s.t.} \quad x_i + x_j \leq 1 \quad \forall (i,j) \in E \tag{2.43}$$

$$x_i \in \{0,1\} \quad \forall i \in V, \tag{2.44}$$

dove x_i è una variabile 0-1 che vale 1 se il vertice $i \in V$ viene inserito nell'independent set, e 0 altrimenti; quindi x_i vale 1 se scegliamo un box nella posizione e con l'orientazione corrispondenti al vertice i. Il vincolo (2.43) impone che per ogni arco al più uno dei due vertici adiacenti sia inserito nell'independent set; ciò corrisponde a scegliere al massimo un box, per ogni coppia di box che si sovrappongono. La funzione obiettivo conta il numero di vertici (e quindi di box) scelti.
Questa formulazione si rivela poco interessante, se rilassiamo il problema al caso continuo. Infatti una soluzione ammissibile sempre presente nel problema rilassato è quella in cui tutte le variabili assumono il valore $1/2$; in generale questa soluzione può fornire un valore molto superiore all'ottimo dell'Independent Set Problem.
Si può ottenere un rafforzamento dei vincoli per il caso rilassato se consideriamo le sovrapposizioni multiple di box, che corrispondono a clique nel grafo. Consideriamo il rettangolo $L \times W$ scomposto in una griglia di quadrati 1×1; per ogni quadrato di coordinate $(j,k) \in [0, L-1] \times [0, W-1]$, indichiamo con S_{jk} l'insieme dei vertici corrispondenti ai box che si sovrappongono in quell'area. Otteniamo così questa seconda formulazione:

$$\max \quad \sum_{i \in V} x_i \tag{2.45}$$

$$\text{s.t.} \quad \sum_{i \in S_{jk}} x_i \leq 1, \quad j \in [w-1, L-w]$$

$$k \in [w-1, W-w] \tag{2.46}$$

$$x_i \in \{0,1\} \quad \forall i \in V. \tag{2.47}$$

Si noti che in (2.46) non vengono considerati tutti i quadrati della griglia; i quadrati esclusi non danno luogo a vincoli nuovi rispetto a quelli considerati (la (2.46) è corretta se $W > 2(w-1)$; se invece $W \leq 2(w-1)$, è necessario operare qualche semplice modifica sugli intervalli di j e k). Si osservi che la riduzione degli intervalli considerati, da noi introdotta, rappresenta un miglioramento rispetto al modello proposto da Morales e Morabito.

Una verifica empirica su alcune istanze del PLP sembra confermare che il bound ottenuto con la versione rilassata del problema (2.45)-(2.47) è più accurato di quello fornito dalla precedente formulazione LP (sezione 2.4). Un recente lavoro [22] conferma questa evidenza sperimentale. Tuttavia il prezzo pagato è un notevole aumento delle dimensioni del problema. Citiamo come esempio il test sull'istanza $(48, 48, 8, 7)$: con il metodo della sezione 2.4 si ottiene per l'upper bound il valore $U = 38$, mentre con la nuova formulazione risulta $U = 36$, che corrisponde al valore dell'ottimo; ma la risoluzione del problema con CPLEX richiede ben 6 minuti di tempo macchina su una SPARC station 10 (il metodo di Dowsland fornisce lo stesso risultato in meno di un centesimo di secondo).

La difficoltà è nel fatto che sia il numero delle variabili che il numero dei vincoli è dell'ordine di $L * W$. Si può pensare di ridurre il numero delle variabili considerando solo pattern normalizzati; in questo caso le coordinate possibili per un box corrispondono all'insieme delle partizioni ammissibili, come visto nella sezione 1.3.

Vogliamo dare ora una definizione formale di questo insieme ridotto di coordinate. L'insieme

$$S(L) = S(L, l, w) = \{r : r = nl + mw,\ r \leq L,\ n, m \in \mathbf{Z}_+\} \tag{2.48}$$

costituisce l'insieme dei *raster points* in direzione L. Esso corrisponde all'insieme delle lunghezze originate dalle partizioni ammissibili $(n, m) \in F(L, l, w)$. Sia $S(W) = S(W, l, w)$ l'insieme corrispondente in direzione W. Per quanto detto nella sezione 1.3, sarà sufficiente considerare solo i punti di allocazione con coordinate $x \in S(L, l, w)$ e $y \in S(W, l, w)$. Questo insieme ridotto di coordinate è stato definito e introdotto in origine per il Rectangle Packing Problem da Herz [20]; anche Beasley [2] utilizza questo stesso insieme. Con questa scelta viene ridotto anche il numero dei vincoli nella formulazione LP, poiché l'insieme di coordinate ammissibili partiziona il rettangolo $L \times W$ in zone rettangolari: all'interno di ciascuna zona si ha in tutti i punti la stessa sovrapposizione di box.
Si noti che ogni soluzione ammissibile per questa nuova formulazione LP è una soluzione ammissibile anche per il problema (2.45)-(2.47).

Tuttavia è possibile diminuire ulteriormente il numero dei punti di allocazione considerando l'*insieme ridotto dei raster points*, introdotto da Terno, Lindemann e Scheithauer nel 1987 e descritto in [29]. Si tratta di un sottinsieme dei raster points definito nel seguente modo. Usando

$$\langle s \rangle := \max\{r \in S(L) : r \leq s\}, \tag{2.49}$$

l'insieme ridotto $\tilde{S}(L)$ dei raster points è dato da

$$\tilde{S}(L) = \tilde{S}(L, l, w) = \{\langle L - r \rangle : r \in S(L)\}. \tag{2.50}$$

Si ha una definizione analoga per $\tilde{S}(W, l, w)$.
Riprendendo l'esempio con l'istanza $(48, 48, 8, 7)$, si ha in questo caso $\tilde{S}(L) = \tilde{S}(W) = \{0, 8, 16, 24, 32, 40, 48\}$, che costituisce una notevole riduzione dell'insieme delle coordinate possibili. L'uso dell'insieme ridotto dei raster point è proposto anche nel lavoro di Morales e Morabito. Nel capitolo 4 vedremo perché ci si può limitare a questo insieme di coordinate. Con la nuova formulazione LP, l'esempio $(48, 48, 8, 7)$ viene risolto da CPLEX in pochi centesimi di secondo. Si noti che in generale l'insieme ridotto dei raster points è vantaggioso soprattutto per piccole dimensioni del pallet; se invece $L, W \gg l*w$, non si hanno differenze significative tra le cardinalità degli insiemi ridotti e quelle degli insiemi originali.

Per quanto detto, questo metodo per il calcolo dell'upper bound presenta una crescita eccessiva delle dimensioni del problema LP con l'aumento delle dimensioni dell'istanza PLP. Tuttavia, rispetto agli altri metodi, questa formulazione è caratterizzata da una maggiore generalità, che si rivela utile quando si considerano alcuni problemi vincolati:

1. *Aree proibite*: In alcuni casi pratici l'area rettangolare del pallet può non essere completamente utilizzabile, per la presenza di altri oggetti o di dispositivi per il trasporto del pallet.

2. *Pattern parziali*: Una variante del PLP è l'impaccamento in pallet parzialmente carichi. In questo caso si deve completare un impaccamento parziale già presente.

La formulazione LP 0-1 descritta in questa sezione consente molto facilmente di trattare questi casi, semplicemente ponendo a 0 o a 1 le variabili corrispondenti alle posizioni proibite o imposte; questa operazione è possibile anche quando le aree vincolate hanno forma irregolare. Gli algoritmi esatti che presenteremo nel capitolo 4 non sono in grado di affrontare queste particolari situazioni vincolate, poiché sfruttano la forma rettangolare dell'area del pallet per aumentare l'efficienza della ricerca. Lo stesso limite vale per le euristiche del capitolo 4.

2.8 Valutazione dei bound

In un recente lavoro di Letchford e Amaral [22] alcuni bound del PLP sono stati studiati dal punto di vista teorico. In particolare gli autori dimostrano che sia il bound di Barnes sia il bound LP di Isermann dominano il bound dell'area; in altre parole il bound dell'area non può mai fornire un risultato migliore di questi due bound. Entrambi però sono a loro volta dominati dal bound LP 0-1 (chiamato packing bound in [22]). Inoltre viene mostrato con degli esempi che non si hanno relazioni di dominanza tra il bound di Barnes e il bound LP di Isermann, né tra il bound LP 0-1 e il bound di Nelißen.

Resta da capire tuttavia quanto questi upper bound si avvicinino all'ottimo, almeno asintoticamente. Dal momento che il bound di Barnes è esatto per rettangoli sufficientemente grandi (quando si fissino le dimensioni del box), ci si può chiedere se comunque abbia un buon comportamento per tutte le istanze del PLP. La risposta è negativa: in questa sezione vedremo che il bound dell'area, il bound di Barnes e il bound LP di sezione 2.4 possono discostarsi dall'ottimo per valori arbitrariamente grandi, anche considerando la riduzione alle massime dimensioni usabili di sezione 2.1.

Indichiamo con $OPT(I)$ il numero di box contenuti in una soluzione ottima dell'istanza I.

Proposizione 2.1 *Siano $U_A(I)$ e $U_B(I)$ gli upper bound ottenuti rispettivamente dal bound dell'area e da quello di Barnes. Si ha allora:*

$$\sup_{I \in PLP} \{U_A(I) - U_B(I)\} = +\infty.$$

Dim.: Si noti che vale sempre $U_A(I) \geq U_B(I)$.
Consideriamo un'istanza $(L = W = 3n,\ l = 2n,\ w = 1)$, dove n è un intero positivo. Un upper bound sul numero di box impaccabili corrisponde ad un lower bound sull'area inutilizzata; con il bound di Barnes otteniamo un limite inferiore per l'area inutilizzata pari a n^2. Dal bound dell'area ricaviamo invece un limite inferiore di $n^2 \bmod (2n)$. Essendo $2n$ l'area di un box, risulta:

$$U_A - U_B = \frac{n^2 - n^2 \bmod (2n)}{2n} = \left\lfloor \frac{n}{2} \right\rfloor.$$

La tesi segue dal fatto che n può essere scelto arbitrariamente grande. □

Proposizione 2.2 *L'upper bound di Barnes U_B verifica:*

$$\sup_{I \in PLP} \{U_B(I) - OPT(I)\} = +\infty.$$

Dim.: Consideriamo l'istanza $(L = W = n^2 - 1,\ l = n + 1,\ w = n)$, dove n è un intero positivo. Notiamo che L e W rappresentano già le massime dimensioni usabili (si veda la sezione 2.1). Il bound di Barnes ci dà per l'area

inutilizzata il valore $R = n+1$, che corrisponde ad un bound $U_B = n^2-n-1$. Usando il procedimento di Keber di sezione 2.3, consideriamo una seconda istanza ($L = W = n^2 - 1, l = n, w = n$). Riduciamo L e W alle massime dimensioni usabili del pallet: $L^* = W^* = n^2 - n$. Con il bound dell'area ricaviamo $U = \left\lfloor \frac{L^* * W^*}{n^2} \right\rfloor = n^2 - 2n + 1$, che costituisce un upper bound anche per la prima istanza (si tratta in effetti dell'ottimo). Dunque

$$U_B - OPT \geq U_B - U = n - 2,$$

valore che può essere reso arbitrariamente grande, da cui la tesi. □

Proposizione 2.3 *Sia U_{LP} l'upper bound LP descritto nella sezione 2.4. Si ha:*

$$\sup_{I \in PLP} \{U_{LP}(I) - OPT(I)\} = +\infty.$$

Dim.: Consideriamo ancora l'istanza ($L = W = n^2 - 1, l = n + 1, w = n$), con n intero positivo. Proveremo la tesi usando un'opportuna soluzione ammissibile del problema LP. Utilizzando le stesse notazioni della sezione 2.4, poniamo

$$x_{n-1,0} = y_{n-1,0} = \frac{(n^2-1)n}{2n+1} \tag{2.51}$$

$$x_{0,n-1} = y_{0,n-1} = \frac{(n^2-1)(n+1)}{2n+1}, \tag{2.52}$$

e le restanti variabili vengono poste a 0 (si tenga presente che nel bound LP di Isermann vengono considerate variabili continue). Le variabili $x_{n-1,0}$ e $y_{n-1,0}$ corrispondono alle partizioni efficienti col massimo numero di elementi $l \times 1$ e $1 \times l$, rispettivamente; mentre le variabili $x_{0,n-1}$ e $y_{0,n-1}$ corrispondono alle partizioni efficienti col massimo numero di elementi $w \times 1$ e $1 \times w$, rispettivamente. L'assegnamento (2.51)-(2.52) costituisce una soluzione ammissibile rispetto ai vincoli (2.7)-(2.12), perché

$$x_{n-1,0} + x_{0,n-1} = y_{n-1,0} + y_{0,n-1} = n^2 - 1,$$

e

$$A_{l\times 1} = A_{1\times w} = A_{1\times l} = A_{w\times 1} = \frac{(n^2-1)^2 n}{2n+1},$$

dove $A_{l\times 1}$ è l'area occupata dagli elementi $l\times 1$, e gli altri termini sono definiti in maniera analoga. L'area totale occupata dai box in questa soluzione è $A = \frac{2(n^2-1)^2 n}{2n+1}$, che corrisponde ad un numero di box N pari a

$$N = \left\lfloor \frac{A}{A_{box}} \right\rfloor = \left\lfloor n^2 - \frac{3}{2}n - \frac{1}{4} + \frac{9}{4(2n+1)} \right\rfloor.$$

Usiamo nuovamente l'upper bound U definito nella proposizione 2.1; poiché $U_{LP} \geq N$, si ha:

$$\begin{aligned} U_{LP} - OPT &\geq N - U \\ &= \left\lfloor n^2 - \frac{3}{2}n - \frac{1}{4} + \frac{9}{4(2n+1)} \right\rfloor - n^2 + 2n - 1 \\ &= \left\lfloor \frac{1}{2}n - \frac{5}{4} + \frac{9}{4(2n+1)} \right\rfloor . \end{aligned}$$

Anche in questo caso otteniamo una differenza non limitata superiormente. Il miglioramento al bound LP proposto da Naujoks (sezione 2.4) non cambia la situazione. Infatti nella soluzione ammissibile data, la somma del numero degli H-box e dei V-box è

$$\lfloor H \rfloor + \lfloor V \rfloor = 2 \left\lfloor \frac{n^2}{2} - \frac{3}{4}n - \frac{1}{8} + \frac{9}{8(2n+1)} \right\rfloor .$$

Questa quantità è uguale a N se n è multiplo di 4. Per tali valori di n l'applicazione del metodo di Naujoks fornisce un upper bound non inferiore a $\lfloor H \rfloor + \lfloor V \rfloor = N$; anche in tal caso allora la differenza tra upper bound e valore ottimo non è limitata superiormente. □

Come abbiamo visto, esistono istanze del PLP per cui il bound di Barnes e il bound LP si rivelano poco accurati (forse l'aspetto più curioso è il fatto che le istanze "difficili" usate nelle dimostrazioni hanno un aspetto piuttosto normale; soddisfano tra l'altro i vincoli (2.2)-(2.3)).
Per quanto riguarda la minimizzazione sulle classi di equivalenza di Dowsland, il bound LP 0-1 e l'upper bound di Nelißen, le valutazioni basate su esperimenti numerici, limitate alle istanze di tipo I e II, suggeriscono che la differenza tra bound e ottimo non sia superiore a 1 o 2 box (vedi anche [22]). Rimane aperto in ogni caso il problema di una valutazione esatta dell'accuratezza di questi metodi.

Capitolo 3

Euristiche

La letteratura sul PLP è molto ricca di euristiche; una rassegna esauriente è presentata in [25]. Come nel caso degli upper bound, ci troviamo nella situazione fortunata in cui anche euristiche molto semplici riescono ad avvicinarsi all'ottimo per gran parte delle istanze (si veda ad esempio [34, 4, 3]. Negli ultimi anni, a partire da Nelißen, sono state proposte euristiche più complesse, di tipo ricorsivo; in questi algoritmi gli impaccamenti vengono costruiti utilizzando particolari strutture nidificate.

In questo capitolo presentiamo in particolare le due euristiche più recenti dovute a Scheithauer e Terno [31] e a Morales e Morabito [24]. Le prestazioni raggiunte da queste euristiche sono notevoli: solo una frazione molto piccola dei problemi test standard non è risolta all'ottimo. Tuttavia riteniamo che l'aspetto più interessante di queste euristiche è l'introduzione delle strutture ricorsive k-block [31], che apre un nuovo ambito teorico all'interno del PLP. Nella sezione finale del capitolo proponiamo un metodo per la generazione sistematica di strutture k-block, con cui è possibile produrre euristiche che lavorano sulla ricerca di pattern k-block massimali.

3.1 Le strutture k-block

L'euristica G4 introdotta da Scheithauer e Terno [31] si basa su una struttura ricorsiva di pattern che generalizza la nozione di pattern a ghigliottina.
Diamo ora alcune definizioni preliminari. Indichiamo con P l'insieme dei box di un generico pattern.
Un sottinsieme $S \subseteq P$ è detto un *block pattern* se esiste un rettangolo $R = \{(x, y) \in \mathbf{R}, \underline{x} \leq x \leq \overline{x}, \underline{y} \leq y \leq \overline{y}\}$ che contiene i box di questo sottinsieme, e tale che $\text{int}(R)$ non interseca box che appartengono a $P \setminus S$.
La lunghezza e la larghezza del blocco sono definite come le dimensioni del più piccolo rettangolo R contenente il blocco. Il pattern P è banalmente un block pattern. Un block pattern è *omogeneo* se tutti i box contenuti hanno la stessa orientazione.

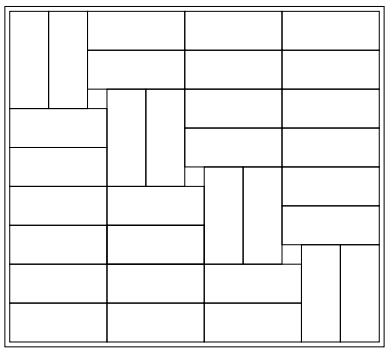

Figura 3.1: Esempio di struttura 4-block.

Definizione 3.1 *Un pattern P è con* struttura a ghigliottina *(*G-structure*) se $|P| \leq 3$ oppure se esiste una partizione di P in due block pattern S_1 e S_2 dotati entrambi di struttura a ghigliottina.*

Definizione 3.2 *Diciamo che un pattern è dotato di struttura 1-block se è omogeneo. Per $k \geq 2$, diciamo che un pattern P ha struttura k-block se esiste una partizione $\{P_i\}$, $i = 1, \ldots, q$ con $q \leq k$, di P tale che ogni P_i è un block pattern a struttura p_i-block con $p_i \leq k$.*

Secondo questa definizione l'insieme dei pattern con struttura k-block comprende tutti i pattern con struttura p-block con $p \leq k$.
Alcune euristiche precedenti [4, 25, 34] utilizzano pattern a blocchi, tuttavia la differenza fondamentale tra questi pattern e le strutture k-block sta nel fatto che queste ultime sono ricorsive. Un esempio di struttura 4-block è in figura 3.1. Si noti che l'insieme dei pattern a struttura 2-block coincide con quello dei pattern a struttura 3-block. Inoltre si ha che:
Un pattern ha struttura 3-block se e solo se ha struttura a ghigliottina.
Restano aperte diverse questioni: un pattern con n box è sicuramente a struttura n-block. Ma dato un pattern P, qual è il più piccolo k per cui è possibile affermare che P ha una struttura k-block? Inoltre, data un'istanza (L, W, l, w), qual è il più piccolo k per cui è possibile trovare una soluzione ottimale con struttura k-block? E infine, esiste un k tale che per ogni istanza del PLP è possibile trovare un pattern ottimale con struttura k-block?
A quest'ultima domanda si può dare una risposta parziale grazie al teorema di Barnes. Nella dimostrazione del teorema viene costruita una soluzione ottima (per L e W *abbastanza grandi*) che fa uso di pattern a struttura 7-block. Vale perciò il seguente corollario:

Proposizione 3.1 (Corollario al teorema di Barnes) *Dati l e w, per L e W sufficientemente grandi esiste un pattern ottimale con struttura 7-block.*

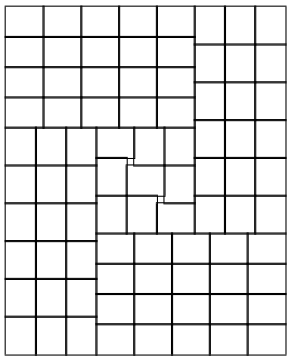

Figura 3.2: Impaccamento ottimale per l'istanza $(46, 37, 5, 4)$. Per quest'istanza non esistono soluzioni ottime G4.

In questo quadro teorico Scheithauer e Terno puntano l'attenzione sulla struttura 4-block:

Definizione 3.3 *Diciamo che un pattern P ha una struttura* G4 *se ha struttura k-block con $k \leq 4$.*

(G va interpretata come *guillotine 4-block* o *generalized 4-block*)
In [31] viene confrontata opportunamente la struttura G4 con quelle usata nelle euristiche precedenti, per evidenziarne la generalità, e si arriva a stabilire che tutte le euristiche analizzate sono casi particolari della G4. Osserviamo che esistono istanze che non ammettono soluzioni ottime G4. Un esempio da noi costruito a tavolino è l'istanza $(46, 37, 5, 4)$ (in figura 3.2 è mostrato un impaccamento ottimale). Usando estesamente l'upper bound di Barnes abbiamo verificato l'assenza di soluzioni ottime di tipo G4 per questa istanza.

3.2 Euristica G4

In base alla definizione, una struttura G4 può essere costituita da un pattern omogeneo, oppure da due pattern G4 (in presenza di un taglio a ghigliottina) oppure da 4 pattern G4. Allo scopo di calcolare pattern G4 massimali, è necessario disporre di pattern G4 massimali per rettangoli più piccoli.
Si prenda come riferimento la figura 3.3. Denotiamo con $n(L', W')$ il numero di box contenuti in un pattern G4 massimale per un pallet di lunghezza L' e larghezza W', con $0 \leq L' \leq L$ e $0 \leq W' \leq W$. Sia inoltre

$$\begin{aligned} n_V(L', W', a) &= n(a, W') + n(L' - a, W'), \\ n_H(L', W', b) &= n(L', b) + n(L', W' - b). \end{aligned} \tag{3.1}$$

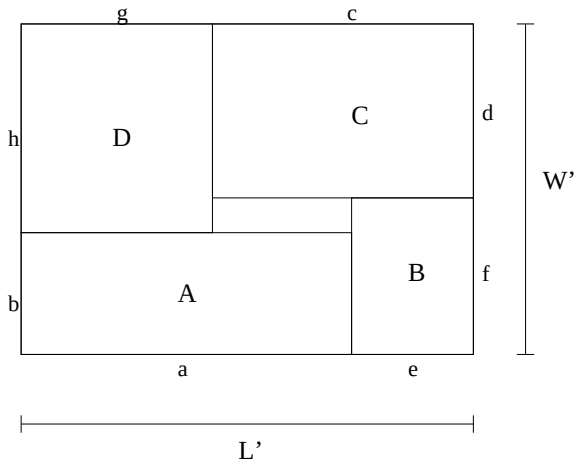

Figura 3.3: Notazioni per la formula di ricorrenza

Si ha la seguente formula ricorsiva:

$$n(L',W') = \max\left\{ \max_{1\le a\le L'/2} n_V(L',W',a), \max_{1\le b\le W'/2} n_H(L',W',b) \right.$$
$$\left. \max_{1\le a\le L'} \max_{1\le b\le W'} \max_{L'-a\le c\le L'} \max_{1\le d\le W'-b} \{n(a,b)+n(e,f)+n(c,d)+n(g,h)\} \right\},$$
$$1 \le W' \le W, \qquad 1 \le L' \le L.$$
$$n(0,b) = 0, \quad 0 \le b \le W, \quad n(a,0) = 0, \quad 1 \le a \le L$$
$$n(l,w) = n(w,l) = 1. \tag{3.2}$$

Scheithauer e Terno apportano un miglioramento alla formula ricorsiva di base con l'uso dei *raster points* (vedi capitoli precedenti).

L'algoritmo realizzato in [31] utilizza la programmazione dinamica. Vengono usati alcuni upper bound presenti in letteratura per verificare il raggiungimento dell'ottimalità, e precisamente: U_A— il bound dell'area (capitolo 2); U_B— il bound di Barnes (sezione 2.2); U_D— la minimizzazione sulle classi di equivalenza del bound dell'area (sezione 2.5); U_L— il bound LP di Isermann.
All'inizio viene inizializzato l'array $n(L',W')$ ponendo

$$n(w,W') := \left\lfloor W'/l \right\rfloor, W' \in S(W), \qquad n(L',w) := \left\lfloor L'/l \right\rfloor L' \in S(L).$$

Il calcolo di $n(L',W')$ viene effettuato quando sono già stati calcolati i valori $n(p',q')$ per tutte le coppie (p',q') con $p' \le L'-1, q' \le W'$, o con $p' \le L', q' \le W'-1$.
Per ogni valore di L',W' il calcolo procede nello stesso ordine presente nella formula ricorsiva 3.2; vengono valutati prima gli impaccamenti omogenei,

quindi i pattern ottenibili dalla scomposizione in due blocchi del rettangolo $L' \times W'$, infine i pattern ottenibili dalla scomposizione in quattro blocchi. Queste valutazioni vengono opportunamente accompagnate dal calcolo di upper bound per verificare l'eventuale raggiungimento dell'ottimalità.
L'algoritmo è stato implementato in Pascal su un PC Pentium 90. Scheithauer e Terno dimostrano che per ogni istanza di tipo I (formule 2.1-2.3), l'euristica raggiunge l'ottimalità. Inoltre considerano un insieme di 206 istanze "hard" di tipo II (formule 2.1-2.3), per cui le euristiche usate da Nelißen [25] non fornivano soluzioni ottimali. In 167 di queste istanze l'euristica G4 arriva ad una soluzione ottima.

3.3 Euristica di Morales e Morabito

Il successo dell'euristica G4 suggerisce che la struttura 4-block sia già sufficiente a fornire pattern ottimali per larga parte delle istanze di interesse pratico. Grazie all'efficienza dell'algoritmo è stato possibile trattare facilmente istanze con un numero di box $n > 50$ [31], anche se in questo caso l'euristica non raggiungeva sempre l'ottimo. È naturale perciò migliorare questo aspetto estendendo l'algoritmo con l'uso della struttura ricorsiva a 5 blocchi, utilizzando in altre parole anche il blocco centrale del pattern di figura 3.3. Si pone immediatamente una difficoltà: il numero di subpattern da calcolare o controllare può subire un aumento notevole, che può rendere di fatto inutilizzabile il nuovo metodo. Nell'algoritmo proposto da Morales e Morabito [24] si è intervenuti in due modi per rimediare a questa difficoltà:
1) Si passa da un approccio che fa uso della programmazione dinamica [31], in cui vengono risolte prima tutte le istanze più piccole di quella attuale, ad un approccio tree-search, che permette di sondare determinati subpattern solo quando è necessario. Il numero di sottoproblemi che superano i vincoli posti dagli upper bound può essere infatti una piccola frazione del numero totale.
2) Si limita il numero di livelli della ricorsione.
3) A differenza del G4, inoltre, nella fase iniziale l'algoritmo calcola un lower bound accurato usando un'implementazione dell'euristica di Bischoff e Dowsland [4].
L'algoritmo è stato implementato in Pascal su un Pentium 100Mhz. Morales e Morabito sperimentano diverse versioni del loro algoritmo; le due versioni che presentano i maggiori benefici in rapporto al costo computazionale sono gli algoritmi 3a e 3b; il primo usa l'insieme dei raster point, il secondo utilizza il corrispondente insieme ridotto. In entrambi la ricorsione è limitata a 3 livelli. Questa scelta è dovuta al fatto che nei problemi test utilizzati un numero maggiore di livelli non aumenta il numero di istanze risolte all'ottimo. Nella figura 3.3 vediamo una delle istanze non risolte all'ottimo dall'euristica (l'esempio è tratto da Morales e Morabito [24]).

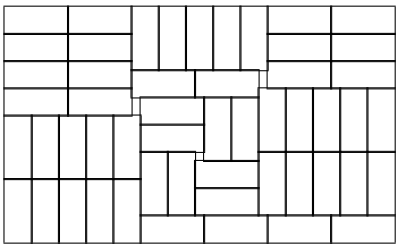

Figura 3.4: Impaccamento ottimale per l'istanza $(43, 26, 7, 3)$. Per quest'istanza non esistono soluzioni ottime ottenibili con l'euristica a 5 blocchi.

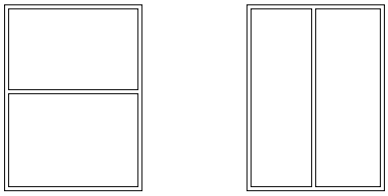

Figura 3.5: Pattern per la generazione di strutture 2-block.

Nel prossimo capitolo verranno confrontati i risultati computazionali delle euristiche di Morales e Morabito e G4 con quelli dell'algoritmo esatto da noi proposto nella sezione 4.2.

3.4 Generazione di strutture k-block

Un problema fondamentale nella realizzazione di euristiche che lavorino su strutture k-block, è quello di individuare tutti i possibili pattern dotati della struttura prescelta.
La definizione ricorsiva di struttura k-block richiede che ad ogni livello della ricorsione venga usato un pattern di rettangoli, all'interno del quale nidificare altri pattern. Di qui la necessità di disporre di una collezione completa di pattern, da usarsi sistematicamente ad ogni livello della ricorsione, per garantire la generazione di tutte le possibili strutture k-block.
Ai fini della realizzazione di un algoritmo diventa utile classificare questi pattern in tipi, a seconda della posizione relativa dei rettangoli del patten. Nel caso delle strutture 2-block, per esempio, si hanno i due tipi di configurazioni di figura 3.5. Una volta individuato un tipo di pattern, diventa poi semplice compiere una ricerca sistematica che lavori solo sulle dimensioni dei rettangoli, e non sulla loro posizione relativa.
Vogliamo definire meglio il concetto di *tipo* di pattern. Considereremo perciò

pattern di rettangoli di diverse dimensioni, all'interno di un contenitore rettangolare $L \times W$. Useremo le seguenti notazioni. La posizione di un singolo rettangolo nel pattern è individuata dalle coordinate dell'angolo in basso a sinistra del rettangolo. Sia (x_i, y_i) la posizione dell'i-esimo rettangolo, e siano $(\overline{x}_i, \overline{y}_i)$ le coordinate del suo angolo in alto a destra.

Definizione 3.4 *Un pattern di n rettangoli* $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i = 1, \ldots, n\}$ *è detto* riducibile *se esiste un sottinsieme* $S \subset P$, $|S| \neq 1$, $|S| \neq n$, *che costituisce un* block pattern *(vedi sezione 3.1).*

Un pattern è riducibile se e solo se è una struttura k-block con $k < n$.

Definizione 3.5 *Due pattern di n rettangoli* $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i = 1, \ldots, n\}$ *e* $P' = \{(x'_i, y'_i, \overline{x}'_i, \overline{y}'_i),\ i = 1, \ldots, n\}$ *sono detti* dello stesso tipo *se esiste un opportuno riordinamento degli indici dei rettangoli in* P' *per cui si verifica*

$$\left.\begin{array}{l} x_i < x_j \iff x'_i < x'_j \\ \overline{x}_i < x_j \iff \overline{x}'_i < x'_j \\ \overline{x}_i < \overline{x}_j \iff \overline{x}'_i < \overline{x}'_j \end{array}\right\} \quad \forall i, j \quad t.c. \quad 1 \leq i \leq n,\ 1 \leq j \leq n.$$

Dalla definizione risulta che due pattern dello stesso tipo o sono entrambi riducibili o sono entrambi non riducibili.

Ci proponiamo di generare algoritmicamente i tipi possibili di pattern irriducibili normalizzati con n rettangoli. È stato preso in considerazione un algoritmo di tipo tree-search con le seguenti caratteristiche:

1. Viene usato il metodo del contorno per la generazione dei pattern.

2. Branching: sia $I_k = \{1, 2, \ldots, k\}$, e sia $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i \in I_k\}$, il pattern parziale di k rettangoli corrispondente al nodo corrente dell'albero di ricerca. Inoltre, sia $\{x^{(j)}\}_{j=1,\ldots,r}$ l'insieme ordinato delle ascisse corrispondenti ai lati verticali dei rettangoli di P:

$$\forall i \in I_k \quad \exists j \quad \text{t.c.} \quad x_i = x^{(j)} \tag{3.3}$$

$$\forall i \in I_k \quad \exists j \quad \text{t.c.} \quad \overline{x}_i = x^{(j)} \tag{3.4}$$

$$x^{(1)} < x^{(2)} < \ldots < x^{(r)} \qquad r \leq 2k, \tag{3.5}$$

$$x^{(1)} = \min_{i \in I_k}\{x_i\}, \qquad x^{(r)} = \max_{i \in I_k}\{\overline{x}_i\} \tag{3.6}$$

Definiamo in maniera analoga l'insieme $\{y^{(j)}\}_{j=1,\ldots,s}$. Poniamo inoltre $x^{(r+1)} = y^{(s+1)} = +\infty$. Nel branching viene scelto:
a) il punto di allocazione $(x, y) \in \{(\eta_l, \xi_l)\}$ del rettangolo $k+1$-esimo.
b) le coordinate $(\overline{x}, \overline{y})$ dell'angolo superiore destro. Tenendo conto della definizione 3.5, è sufficiente scegliere l'intervallo aperto $(x^{(j)}, x^{(j+1)})$, con $x^{(j)} \geq x$, in cui si verrà a trovare $\overline{x}$, e l'intervallo aperto $(y^{(j)}, y^{(j+1)})$, con $y^{(j)} \geq y$, in cui si verrà a trovare $\overline{y}$. Per ridurre le dimensioni dell'albero di ricerca, non vengono considerati i casi in cui $\overline{x} = x^{(j)}$ o $\overline{y} = y^{(j)}$ per qualche j: ogni pattern irriducibile di questo tipo può essere visto come caso limite di uno dei pattern generati.

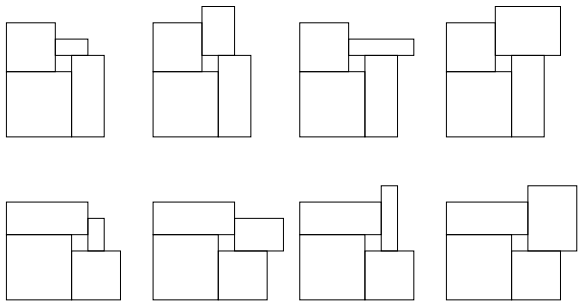

Figura 3.6: Pattern irriducibili normalizzati con 4 rettangoli.

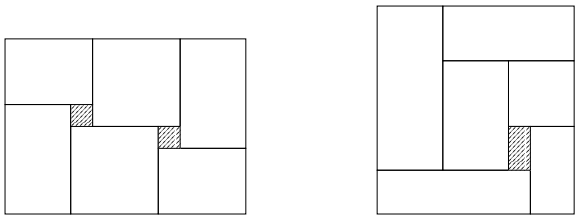

Figura 3.7: Alcuni pattern irriducibili con 6 rettangoli.

3. In ogni nodo dell'albero di ricerca viene controllato se esiste un *block pattern* nell'area del contorno $U(A)$. In caso affermativo viene eseguito il backtracking, poiché il sottoalbero corrispondente al nodo genererà solo pattern riducibili.

È possibile un miglioramento dell'algoritmo se nel branching si considerano solo le coordinate $x^{(j)}$ e $y^{(j)}$ "rilevanti", e se si tiene conto delle simmetrie. Utilizzando questa procedura per generare i pattern irriducibili con 4 rettangoli, abbiamo ottenuto le configurazioni di figura 3.6. Le configurazioni della prima riga possono essere considerate come casi particolari del pattern di figura 3.3. Le configurazioni della seconda riga non sono nient'altro che riflessioni delle configurazioni della prima riga. Si conclude perciò che i pattern utilizzati nell'euristica G4 sono sufficienti per ottenere la migliore soluzione a struttura 4-block.
Allo stesso modo, abbiamo verificato per la prima volta che tutti i pattern irriducibili di 5 rettangoli possono essere compresi nel pattern di figura 3.3 (a meno di riflessioni). Possiamo concludere quindi che l'euristica di Morales e Morabito trova in effetti impaccamenti a struttura 5-block massimali. Per i pattern irriducibili di 6 rettangoli si possono avere invece più configurazioni non trasformabili una nell'altra. Nell'esempio in figura 3.7, tratto da [33], vediamo due di queste configurazioni. Si noti che entrambe le configurazioni possono essere modificate in modo sistematico per la generazione

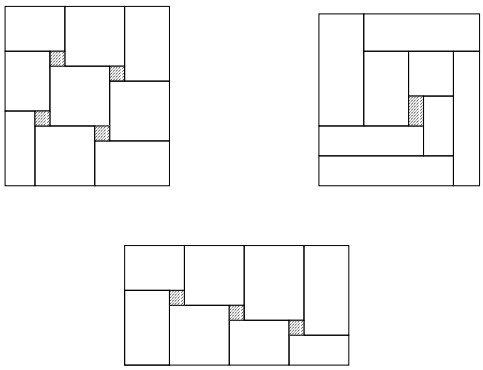

Figura 3.8: Altri pattern irriducibili ottenuti con modifiche sistematiche ai pattern di figura 3.7.

di pattern irriducibili a $6, 8, \ldots, 2k$ rettangoli (vedi figura 3.8). Altre configurazioni irriducibili si ottengono collocando rettangoli nelle aree centrali inutilizzate rappresentate con l'ombreggiatura (figure 3.7 e 3.8). Osserviamo tuttavia che con l'algoritmo presentato in questa sezione è possibile generare sistematicamente *tutte* le configurazioni a struttura k-block, per k qualsiasi.

Capitolo 4

Algoritmi esatti

Il numero di impaccamenti ammissibili per istanze generiche del PLP è molto elevato; tipicamente il numero di pattern costruibili con n box dipende esponenzialmente da n, nel caso peggiore. Di conseguenza l'elaborazione di un algoritmo esatto richiede uno studio approfondito dei metodi di branching e cutting, che sfrutti le proprietà peculiari del PLP. In effetti rispetto ad altri problemi combinatorici il PLP possiede una struttura "forte", ricca di proprietà che tuttavia non sono facilmente individuabili, come già si è visto nei precedenti capitoli. Per questo motivo in letteratura sono presenti pochi lavori su algoritmi esatti, mentre la ricerca ha finora privilegiato lo studio di euristiche più o meno complesse.

L'approccio più spesso adottato per la risoluzione esatta di problemi di packing generici è quello della costruzione progressiva di impaccamenti; si genera un albero di ricerca in cui ogni nodo corrisponde ad un impaccamento parziale; i nodi figli si ottengono considerando le posizioni possibili per l'inserimento di un nuovo pezzo. E' evidente che un impaccamento di n elementi può essere ottenuto con l'aggiunta di un pezzo a partire da diversi impaccamenti di $n-1$ elementi. Risulta importante perciò elaborare un criterio di branching che eviti la generazione ripetuta dello stesso pattern in diversi punti dell'albero di ricerca. Nello stesso tempo è necessario garantire che tutti i pattern vengano generati. In effetti quest'ultima condizione può essere molto indebolita, richiedendo semplicemente che per ogni impaccamento ne venga generato almeno uno equivalente o dominante, ossia con un numero uguale o maggiore di elementi inseriti.

Il lavoro più noto nel campo dei metodi esatti per il PLP è quello di Dowsland [11, 12]. Purtroppo in questi lavori non vengono affrontate le questioni sopracitate. Più recentemente, anche Nelißen ha utilizzato un algoritmo esatto, creato allo scopo di verificare l'accuratezza dell'upper bound da lui introdotto [26].

Nella sezione 4.1 viene presentato l'algoritmo sviluppato da Dowsland in una serie di lavori [11, 12]. Nelle sezioni successive presentiamo un algoritmo

esatto di tipo tree search, in cui sono stati introdotti una serie di nuovi metodi che consentono una riduzione notevole dell'albero di ricerca.

4.1 L'algoritmo di Dowsland

L'approccio di Dowsland [11, 12] viene derivato dalla teoria dei grafi, in particolare viene utilizzato l'Independent Set Problem. Tuttavia la versione finale dell'algoritmo abbandona il punto di vista della teoria dei grafi, per sfruttare più efficientemente le caratteristiche strutturali del PLP. Nei due lavori citati la descrizione dell'algoritmo è piuttosto generica e priva di dettagli su alcuni aspetti fondamentali. In particolare non viene esplicitato chiaramente il metodo di generazione dei pattern, né viene chiarito se vengono introdotti criteri per evitare la generazione ripetuta di pattern uguali o equivalenti.

Nella sezione 2.7 è stata illustrata la corrispondenza tra PLP e Independent Set Problem (ISP): ad ogni istanza del PLP viene associato un grafo $G(V, E)$; il problema di trovare un impaccamento ottimale si riduce a quello di trovare un independent set $V' \subseteq V$ di cardinalità massima.

Indichiamo con (x, y, o) una possibile disposizione di un *singolo* box all'interno del pallet, dove x e y sono le coordinate del punto di allocazione, e $o \in \{O, V\}$ indica l'orientazione del box, che può essere orizzontale o verticale. Per ogni valore possibile di questa terna viene creato un vertice del grafo. Inoltre, per ogni coppia di disposizioni (x, y, o) e (x', y', o') che danno luogo a sovrapposizione dei box corrispondenti, viene creato un arco tra i vertici relativi. Per un'istanza (L, W, l, w) il numero dei vertici generato è

$$|V| = (L + 1 - l)(W + 1 - w) + (L + 1 + w)(W + 1 - l),$$

dove il primo prodotto corrisponde alle possibili posizioni dei box orizzontali e il secondo alle possibili posizioni dei box verticali. Per problemi tipici del PLP il numero di vertici può essere dell'ordine di 10^6; poiché gli algoritmi noti per la soluzione dell'ISP si rivelano praticabili solo per grafi densi con non più di qualche centinaio di vertici, Dowsland conclude che è necessario operare una qualche forma di riduzione delle dimensioni del grafo. Vengono applicati consecutivamente due metodi di riduzione; nel primo, viene considerata la classe di equivalenza a cui appartiene il problema originale (ricordiamo che le istanze all'interno di una classe di equivalenza condividono lo stesso ottimo e gli stessi pattern; una soluzione per un'istanza è facilmente trasformabile in una soluzione per un'istanza equivalente). In questa classe viene cercata l'istanza con il più piccolo valore per w, o perlomeno con w sufficientemente piccolo: si presuppone che questa scelta risulti la più efficace ai fini della riduzione delle dimensioni del grafo. Nell'ultima versione dell'algoritmo [13], viene ritenuto più opportuno usare invece l'istanza equivalente che minimizza il rapporto area pallet/area box.
Il secondo metodo di riduzione utilizza l'insieme dei raster points, definito

nella sezione 2.7 (formula (2.48)). Infatti, come è stato notato nella sezione 2.7, con pattern normalizzati è sufficiente limitarsi a considerare solo quelle posizioni dei box con coordinate che costituiscono una combinazione lineare intera di l e w. Dowsland individua un'ulteriore possibilità di riduzione notando che è possibile normalizzare un pattern simmetricamente nelle quattro direzioni: ogni box viene spinto verso il bordo più vicino sia nella direzione verticale, sia nella direzione orizzontale, ricorsivamente. In questo modo è sufficiente considerare le coordinate ottenute come combinazioni lineari di l e w dal bordo *più vicino* del pallet, in direzione verticale e in direzione orizzontale.
Applicando questi due metodi di riduzione ad un campione di istanze del seguente tipo

$$1 \leq \frac{LW}{lw} < 51 \tag{4.1}$$

$$1 \leq \frac{l}{w} \leq 4 \tag{4.2}$$

$$1 \leq \frac{L}{W} \leq 2 \tag{4.3}$$

(rispetto alle istanze di tipo I, formule (2.1)-(2.3), si ha un range diverso per il rapporto area pallet/area box, formula (4.1)), si ottengono problemi con non più di 600 vertici.

A questo punto viene calcolato un upper bound applicando il bound di Barnes al problema equivalente minimale (sezioni 2.2 e 2.5). In [11] inizialmente per la soluzione dell'Independent Set Problem viene scelto un algoritmo di Loukakis e Tsouros. Si tratta di un algoritmo di tipo tree search, in cui ad ogni passo viene aggiunto o tolto un vertice all'independent set corrente. In ogni nodo dell'albero di ricerca sono definiti tre insiemi: S— l'independent set corrente, S^+— l'insieme dei vertici candidati a estendere S, e S^-— l'insieme dei vertici scartati. Indicando con M la cardinalità della migliore soluzione trovata, se nel nodo corrente $|S| + |S^+| \leq M$, viene eseguito il backtracking, non essendoci possibilità di trovare soluzioni migliori. Dowsland modifica questo test confrontando $|S| + |S^+|$ con l'upper bound calcolato inizialmente: in altre parole viene cercata una soluzione con un numero di box corrispondente all'upper bound, e viene eseguito il backtracking se non è possibile raggiungere questo valore. Questa modifica consente una maggiore efficienza, dal momento che il valore fornito dall'upper bound corrisponde all'ottimo nel 98% dei casi; inoltre nei casi in cui l'upper bound non corrisponde all'ottimo, la differenza tra i due valori è dell'ordine di qualche box. Nel caso che non vengano trovate soluzioni, viene decrementato l'upper bound e la procedura riparte col nuovo valore. Tuttavia l'algoritmo di Loukakis e Tsouros si rivela non abbastanza efficiente per i grafi originati da tipici problemi PLP, grafi di grosse dimensioni ma caratterizzati da una relativa sparsità (tipicamente 8% - 13% di densità).

Per questo motivo Dowsland abbandona l'approccio derivato dall'Independent Set Problem, in favore di un approccio diretto che lavora sulla

costruzione di pattern. Si ha ancora un algoritmo di tipo tree search, in cui ad ogni passo viene aggiunto (o tolto) un box al pattern parziale costruito. Inizialmente viene determinata l'area inutilizzata $A = L * W - U * l * w$ presente in una soluzione con U box, dove U è l'upper bound per il numero di box. Per ogni nodo dell'albero, corrispondente ad un pattern parziale, viene calcolata l'area inutilizzata creata dal pattern che inevitabilmente rimarrà fino a che non viene eseguito il backtracking. Se quest'area è maggiore di A, l'upper bound non può essere raggiunto nel corrispondente sottoalbero e l'algoritmo esegue il backtracking. Se l'algoritmo non trova soluzioni, la ricerca viene ripetuta con il valore dell'upper bound decrementato.
Viene introdotto un ulteriore miglioramento in base alla seguente considerazione: per ogni pattern in cui la maggior parte dell'area inutilizzata si trova nella metà superiore del pallet, esiste un pattern simmetrico in cui la maggior parte dell'area sprecata è nella metà inferiore. Perciò mentre si lavora nella metà superiore del pallet il test per il backtracking può essere rafforzato confrontando l'area sprecata con $A/2$ anziché con A.

Le verifiche numeriche con quest'algoritmo finale si sono rivelate soddisfacenti. Dowsland testa l'algoritmo su un campione di 1000 istanze che verificano (4.1)-(4.3); i rapporti L/W, l/w, e LW/lw sono campionati con distribuzione uniforme negli intervalli dati. Il 95% delle istanze è risultato solvibile in meno di 1 secondo di tempo macchina utilizzando un VAX 780, e meno del 2% ha richiesto più di 20 secondi. Altri insiemi di dati hanno fornito risultati simili.
Nel successivo lavoro [13] Dowsland realizza un approccio combinato per il PLP, algoritmico e di database, allo scopo di ottenere un pacchetto software per PC IBM utilizzabile nella pratica. Dapprima viene stabilito che l'insieme di istanze (4.1)-(4.3) è costituito da 8565 classi di equivalenza; quindi per ogni classe viene individuato il problema equivalente minimale. Tutte le 8565 istanze così ottenute vengono risolte con l'algoritmo esatto precedentemente descritto; 456 di esse richiedono un tempo macchina superiore ad un certo tempo limite stabilito (l'equivalente di tre minuti di PC IBM(1987)); le soluzioni di queste 456 istanze vengono quindi memorizzate in un file. In questo modo ogni istanza dell'insieme (4.1)-(4.3) può essere risolta entro un limite di tempo stabilito: data un'istanza, si individua il problema equivalente minimale; se questo è contenuto nel database, ne viene estratta la soluzione; altrimenti, il problema viene risolto eseguendo direttamente l'algoritmo esatto, che termina nel limite di tempo stabilito.

4.2 Un nuovo algoritmo di tree search

Presentiamo in questa sezione un algoritmo esatto di tipo tree-search, che utilizza diverse tecniche, di cui alcune già sviluppate per problemi di packing

diversi dal PLP.
Lo schema per la generazione dei pattern usa il *metodo del contorno* sviluppato da Scheithauer e Terno, e applicato al Rectangle Packing Problem [29] e al Container Loading Problem [28].
Nei lavori di Herz [20], Christofides e Whitlock [7], Beasley [2], viene introdotto l'*insieme dei raster points* nella soluzione del Rectangle Packing Problem, che consente di ridurre il numero delle possibili coordinate per la disposizione dei rettangoli (i primi due lavori considerano solo pattern a ghigliottina); si tratta in effetti di limitarsi a considerare solo quelle coordinate ottenibili come combinazioni delle dimensioni dei pezzi.
In [29] Scheithauer descrive l'*insieme ridotto dei raster points*, che costituisce un ulteriore raffinamento nella procedura di generazione dei pattern. Quest'insieme, definito nella sezione 2.7, viene usato nell'algoritmo che presenteremo.
Alcuni upper bound descritti nel capitolo 2 verranno usati: più precisamente abbiamo utilizzato la minimizzazione sulla classe di equivalenza di Dowsland (sezione 2.5), i bound di Barnes (sezione 2.2) e LP di Isermann (sezione 2.4).

Nelle sezioni successive introdurremo dei nuovi criteri basati su proprietà strutturali del PLP, che consentono una efficace riduzione dell'albero di ricerca. Un primo criterio si basa su una valutazione delle aree inutilizzate ed impiega gli *insiemi rappresentativi*, introdotti da Brualdi e Foregger [5]. Gli altri test introdotti si basano su proprietà di dominanza o equivalenza tra pattern. La sezione finale è dedicata ai test numerici. Per quanto riguarda le sezioni 4.2.1, 4.2.2, 4.2.3, si è scelto di svolgere la teoria in modo più generale, cosicché risulti applicabile anche al Rectangle Packing Problem.

Descriveremo ora una versione base dell'algoritmo. Il procedimento generale è il seguente: inizialmente viene individuata l'istanza equivalente minimale (sezione 2.5), e sarà questa ad essere risolta. Viene calcolato un upper bound accurato N_{max} sul numero massimo di box impaccabili. Quindi viene generato ed esplorato un albero di ricerca (percorso secondo una depth first search), i cui nodi corrispondono a pattern parziali. Per ogni nodo il branching viene effettuato scegliendo la disposizione del prossimo box che verrà inserito. Il test per il backtracking viene effettuato verificando se il pattern parziale corrente permette di avere soluzioni con N_{max} box. Se la ricerca termina senza aver trovato soluzioni, viene decrementato l'upper bound e viene eseguita una nuova ricerca.
Un'alternativa a questo procedimento è quella di salvare durante la ricerca la miglior soluzione provvisoriamente trovata, e di eseguire il backtracking solo se il nodo corrente non consente una soluzione migliore di questo lower bound. Analogamente a Dowsland, si è scelto il primo procedimento in base al fatto che l'upper bound usato è molto accurato; inoltre come vedremo il backtracking viene eseguito relativamente presto grazie all'uso dell'insieme ridotto dei raster points ed ad un insieme di nuovi test di backtracking.
Nell'algoritmo è stato introdotto un criterio per evitare la generazione ripetu-

ta di uno stesso pattern. Con ciò si evita una crescita eccessiva dell'albero di ricerca originata dalle possibili permutazioni nell'ordine di inserimento dei box.

4.2.1 Il metodo del contorno

Per i nostri scopi è utile presentare il metodo del contorno per il caso più generale del Rectangle Packing Problem, che riguarda l'impaccamento di rettangoli non identici in un rettangolo più grande. In questa sezione vengono introdotte le notazioni che useremo nel seguito. Rispetto al lavoro di Scheithauer [29], alcune definizioni sono state leggermente semplificate e altre sono state aggiunte.

La posizione di un singolo rettangolo nel pattern è individuata dalle coordinate dell'angolo in basso a sinistra del rettangolo. Sia (x_i, y_i) la posizione dell'i-esimo rettangolo, e siano $(\overline{x}_i, \overline{y}_i)$ le coordinate del suo angolo in alto a destra. Usiamo la notazione:

$$\text{box}(x_i, y_i, \overline{x}_i, \overline{y}_i) := \{(x, y) \in \mathbf{R}^2 : x_i < x \leq \overline{x}_i,\ y_i < y \leq \overline{y}_i\}.$$

Un pattern $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i = 1, \ldots, n\}$ è costituito dall'insieme di punti

$$A = \bigcup_{i=1}^{n} \text{box}(x_i, y_i, \overline{x}_i, \overline{y}_i).$$

L'insieme

$$U(A) := \{(x, y) \in \mathbf{R}^2_+ : \exists i, 1 \leq i \leq n, \text{ tale che } x \leq \overline{x}_i,\ y \leq \overline{y}_i\}$$

costituisce la *regione coperta dal pattern* A. Ovviamente vale $A \subseteq U(A)$. Si supponga di avere un impaccamento parziale A; con il metodo del contorno solo l'area non occupata da $U(A)$ è disponibile per ulteriori inserimenti di box. Quindi l'insieme $U(A)/A$ deve essere considerato area definitivamente "sprecata" dall'impaccamento A; quest'area viene chiamata anche *sfrido*.
Il *contorno* del pattern A (vedi figura 4.1) è definito dall'insieme poligonale

$$K(A) := \overline{U(A) \cap \{(x, y) : xy > 0\} \setminus \text{int } U(A)},$$

dove le notazioni $\overline{S}$ e int S indicano rispettivamente la chiusura e la parte interna dell'insieme S.

Viceversa, dato un contorno K, la regione coperta U è definita come segue:

$$U = U(K) := \{(x, y) \in \mathbf{R}^2_+ : \exists (r, s) \in K \text{ t.c. } x \leq r, y \leq s\}. \tag{4.4}$$

Con riferimento alla figura 4.1, un contorno $K(A)$ può essere rappresentato dalla sequenza

$$\{(\eta_i, \xi_i) : i = 1, \ldots, m\}$$

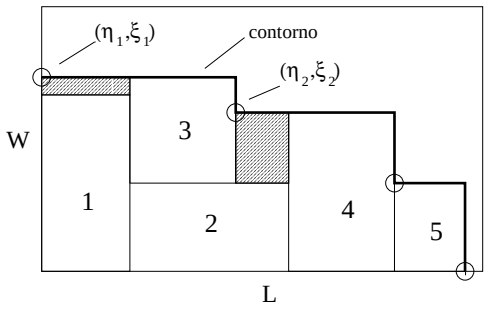

Figura 4.1: Contorno di un pattern. I cerchietti indicano i possibili punti di inserimento di un nuovo box, nel caso di pattern normalizzati

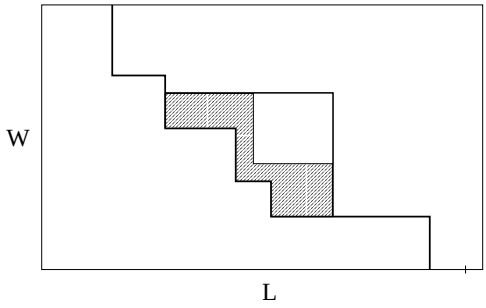

Figura 4.2: Inserimento di un nuovo rettangolo. In base al metodo del contorno l'area tratteggiata non è più utilizzabile e viene detta area di sfrido.

di m punti, caratterizzati da

$$\eta_1 < \eta_2 < \ldots < \eta_m, \quad \xi_1 > \xi_2 > \ldots > \xi_m.$$

L'aggiunta di un rettangolo al pattern modifica il contorno come illustrato nella figura 4.2. È chiaro che se *normalizziamo* in basso a sinistra il rettangolo inserito, l'insieme dei possibili punti di inserimento può essere ridotto ai punti $\{(\eta_i, \xi_i)\}$.

Vogliamo dimostrare ora che ogni pattern (normalizzato o no) può essere costruito col metodo del contorno; in altre parole è possibile costruire qualunque pattern con l'inserimento successivo di rettangoli, con il vincolo che ogni rettangolo può essere inserito solo nell'area consentita dal *contorno* determinato dai rettangoli già sistemati (in [28, 29] il metodo del contorno è presentato ma non giustificato).

Definizione 4.1 *Dato un pattern* $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i = 1, \ldots, n\}$*, diremo*

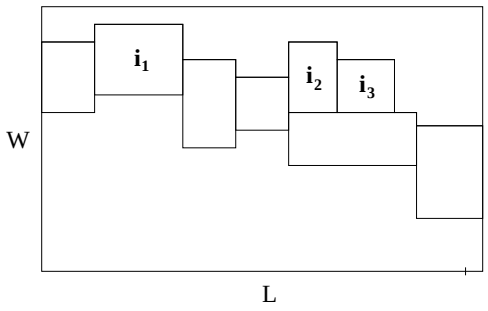

Figura 4.3: Schema per la dimostrazione del lemma 4.1.

che il rettangolo j domina *il rettangolo* i *se*

$$x_i < \overline{x}_j, \quad y_i < \overline{y}_j.$$

Se il rettangolo j domina il rettangolo i, in base al metodo del contorno necessariamente nella costruzione dell'impaccamento l'inserimento di i non può seguire l'inserimento di j, ma deve invece precederlo.

Lemma 4.1 *Dato un pattern* $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i), i = 1, \ldots, n\}$. *esiste almeno un rettangolo del pattern che non è dominato da altri rettangoli.*

Dim.: Diamo una dimostrazione costruttiva del lemma.
Sia i_1 l'indice del rettangolo con il massimo valore di $\overline{y}$ (in caso di parità si scelga quello con x maggiore). A partire da i_1, generiamo per ricorrenza una successione di indici $\{i_j\}, j = 1, 2, \ldots$ nel seguente modo: sia i_j l'ultimo elemento della successione generato; se esistono rettangoli di indice i con $x_i \geq \overline{x}_{i_j}$ e $\overline{y}_i > y_{i_j}$, consideriamo tra questi quello con $\overline{y}_i$ maggiore (in caso di parità si scelga quello con x_i maggiore); sia i_{j+1} il suo indice. Altrimenti, se non esistono rettangoli di questo tipo, la successione si conclude con i_j (vedi figura 4.3).

Per ogni indice j vale la seguente proprietà: non ci sono rettangoli del pattern che intersecano la regione $R_{i_j} := \{(x, y) \in \mathbf{R}^2_+ : x > x_{i_j},\ y > \overline{y}_{i_j}\}$. Dimostriamo la proprietà per induzione, supponendola vera fino all'indice j della successione (il caso i_1 è ovvio). Per assurdo, supponiamo che esista un rettangolo $\text{box}(x, y, \overline{x}, \overline{y})$ che interseca la regione $R_{i_{j+1}}$. Per costruzione si ha $\overline{y} > \overline{y}_{i_{j+1}} > y_{i_j}$. Inoltre $x < \overline{x}_{i_j}$ (altrimenti non sarebbe corretta la scelta dell'elemento $j+1$-esimo della successione). Ma si ha anche $\overline{x} > x_{i_{j+1}} \geq \overline{x}_{i_j}$. Dunque il rettangolo interseca o l'elemento i_j, e quindi il pattern contiene sovrapposizioni, o la regione R_{i_j}, contro l'ipotesi induttiva. Si arriva quindi ad una contraddizione.
La successione $\{i_j\}$ termina, poiché $\{x_{i_j}\}$ è una successione di interi crescente e limitata da L. Sia i_k l'ultimo elemento della successione. L'elemento i_k non è dominato da nessun rettangolo, e quindi è l'elemento cercato: infatti, per la

proprietà dimostrata sopra, un eventuale rettangolo $(x, y, \overline{x}, \overline{y})$ che domini i_k deve avere $x > \overline{x}_{i_k}$, e $\overline{y} > y_{i_k}$. Ma allora ci sarebbero candidati per l'elemento i_{k+1}, e la successione continuerebbe. □

Benché il risultato del lemma sia intuitivo, la sua dimostrazione contiene in effetti tutta la difficoltà della derivazione del seguente teorema.

Teorema 4.1 *Ogni pattern* $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i \in I\}$ *di rettangoli può essere costruito col metodo del contorno.*

Dim.: Per induzione sul numero di rettangoli del pattern. Il caso $n = 1$ è ovvio. Consideriamo un pattern di $n + 1$ rettangoli $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i \in I\}$. Per il lemma dimostrato, esiste un rettangolo k del pattern che non è dominato da altri rettangoli. Per definizione quindi k non interseca la regione $U(A)$, dove A è l'insieme ottenuto eliminando il rettangolo k:

$$A = \bigcup_{i \in I, i \neq k} \mathrm{box}(x_i, y_i, \overline{x}_i, \overline{y}_i).$$

Perciò k può essere aggiunto a questo pattern rispettando il metodo del contorno. La tesi segue dal fatto che il pattern parziale A è costituito da n rettangoli, e per l'ipotesi induttiva può essere costruito col metodo del contorno. □

Per evidenti ragioni di dominanza tra pattern, dato un impaccamento parziale un algoritmo esatto che usa il metodo del contorno può limitarsi a esaminare solo i punti di inserimento dati dall'insieme $\{(\eta_i, \xi_i)\}$. Ciò corrisponde a considerare solo pattern normalizzati, in base alla definizione della sezione 1.3.

4.2.2 Insieme ridotto dei raster points

Nella sezione 2.7 sono stati definiti i raster point e gli insiemi ridotti dei raster points per un'istanza (L, W, l, w) del PLP. Le definizioni vengono facilmente estese nel caso più generale del Rectangle Packing Problem: si consideri un pallet $L \times W$ e un insieme di indici $i = 1, 2, \ldots, k$ che identificano i tipi di rettangoli a disposizione, di dimensioni $l_i \times w_i$ (prendiamo in considerazione il caso in cui non sono permesse rotazioni di 90 gradi); l'insieme dei raster point sarà definito da

$$S(L) := \left\{ r : r = \sum_{i=1}^{k} n_i l_i,\ r \leq L, n_i \in \mathbf{Z}_+, i = 1, 2, \ldots, k \right\}, \tag{4.5}$$

e si ha una formula analoga per $S(W)$.
L' insieme ridotto dei raster point è dato allora dalle seguenti definizioni, identiche a quelle della sezione 2.7:

$$\tilde{S}(L) = \{\langle L - r \rangle : r \in S(L)\}, \tag{4.6}$$

dove

$$\langle s \rangle := \max\{r \in S(L) : r \leq s\}. \tag{4.7}$$

Formule analoghe per $\tilde{S}(W)$.

Come osservato in [29], nella costruzione di un impaccamento è sufficiente limitarsi ad usare come punti di inserimento dei rettangoli le coordinate date dall'insieme $\tilde{S}(L) \times \tilde{S}(W)$:

Asserzione 4.1 (Scheithauer, 1998) *Per ogni impaccamento del Rectangle Packing Problem ne esiste uno equivalente in cui le coordinate dei punti di inserimento appartengono agli insiemi ridotti dei raster point.*

Per completezza diamo una dimostrazione dell'asserzione, di cui nel lavoro citato viene data solo un'idea.

Dim.: Dato un impaccamento, per il teorema 4.1 esso è generabile col metodo del contorno. Assumiamo che il contorno determinato dai primi $k-1$ rettangoli della sequenza sia descritto dai punti $\{(\eta_i, \xi_i)\}$, con $\eta_i \in \tilde{S}(L)$ e $\xi_i \in \tilde{S}(W)$ per ogni i; assumiamo inoltre che il punto di inserimento (x_k, y_k) del rettangolo k-esimo corrisponda a uno di questi punti. Se normalizziamo a destra i rettangoli di indice $j > k$ (operazione che può essere eseguita senza superare il contorno), la distanza dei lati verticali di ognuno di questi box dal lato destro del pallet apparterrà a $S(L)$; nessun rettangolo di indice $j > k$ interseca l'area $A = \{(x, y) \in \mathbf{R}^2_+ : x < L - \langle L - \overline{x}_k \rangle, y < \overline{y}_k\}$, che può essere considerata quindi non più utilizzabile. Inoltre si ha $(\overline{x}_k, \overline{y}_k) \in S(L) \times S(W)$, Risulta perciò possibile "far avanzare" il contorno operando per la coordinata $\overline{x}_k$ la seguente sostituzione:

$$\overline{x}_k \to \overline{x}'_k := \langle L - \langle L - \overline{x}_k \rangle \rangle \,,$$

Dalla definizione (2.50) risulta $\overline{x}'_k \in \tilde{S}(L)$. Con un ragionamento analogo, normalizzando verso l'alto i rettangoli di indice $j > k$, è possibile sostituire $\overline{y}_k$:

$$\overline{y}_k \to \overline{y}'_k := \langle W - \langle W - \overline{y}_k \rangle \rangle \,.$$

Con questa operazione i punti di inserimento del contorno determinato dai primi k rettangoli appartengono a loro volta all'insieme $\tilde{S}(L) \times \tilde{S}(W)$. A questo punto, normalizzando in basso a sinistra i rettangoli di indice $j > k$, si può iterare il procedimento al passo $k+1$-esimo. Si noti che non vengono spostati i primi k rettangoli, perciò il nuovo impaccamento è generabile secondo il metodo del contorno con una sequenza invariata per quanto riguarda i primi k passi.

Osserviamo che anche il punto di inserimento iniziale di coordinate $(0, 0)$ appartiene a $\tilde{S}(L) \times \tilde{S}(W)$. Ripetendo induttivamente le operazioni descritte, si ottiene una successione di impaccamenti equivalenti, l'ultimo dei quali verifica la proprietà cercata. □

Nella precedente dimostrazione si può osservare che una sostituzione più conveniente per le coordinate $(\overline{x}_k, \overline{y}_k)$ è la seguente:

$$\begin{aligned} \overline{x}_k &\rightarrow \overline{x}'_k := L - \langle L - \overline{x}_k \rangle, \\ \overline{y}_k &\rightarrow \overline{y}'_k := W - \langle W - \overline{y}_k \rangle. \end{aligned} \tag{4.8}$$

In questo modo viene assegnata allo sfrido un'area maggiore [29]. Un impaccamento costruito iterativamente con le sostituzioni 4.8 verifica una diversa proprietà, derivata dal lemma seguente.

Lemma 4.2 (Proprietà dell'insieme ridotto dei raster point.) *Se $x \in \tilde{S}(L)$, allora $\tilde{S}(x) \in \tilde{S}(L)$.*

Dim.: Sia $y \in \tilde{S}(x)$. Per definizione esisterà $r_2 \in S(L)$ tale che $y = \langle x - r_2 \rangle$. Inoltre poiché $x \in \tilde{S}(L)$, esisterà $r_1 \in S(L)$ tale che $x = \langle L - r_1 \rangle$. Vogliamo mostrare che $y = \langle L - (r_1 + r_2) \rangle \in \tilde{S}(L)$. Consideriamo allora un generico $z \in S(L)$, tale che $z \leq L - (r_1 + r_2)$. Si ha $z + r_2 \leq L - r_1$, e poiché $z + r_2 \in S(L)$, si ottiene $z + r_2 \leq \langle L - r_1 \rangle = x$. Perciò $z \leq x - r_2$, da cui $z \leq \langle x - r_2 \rangle = y$. Dunque y è il massimo raster point non maggiore di $L - (r_1 + r_2)$, da cui la tesi. □

Teorema 4.2 *Per ogni impaccamento del Rectangle Packing Problem ne esiste uno equivalente in cui ogni punto di inserimento (x, y) verifica $L - x \in \tilde{S}(L)$, $W - y \in \tilde{S}(W)$*

Dim.: Si procede come nella dimostrazione dell'asserzione 4.1. Si assume che i punti $\{(\eta_i, \xi_i)\}$ che descrivono il contorno determinato dai primi $k-1$ rettangoli inseriti verifichino la proprietà: $L - \eta_i \in \tilde{S}(L)$ e $W - \xi_i \in \tilde{S}(W)$ per ogni i; inoltre il rettangolo k-esimo sia inserito in uno di questi punti. Anche in questo caso si verifica che normalizzando in alto a destra i rettangoli di indice $j > k$, è possibile spostare il contorno operando le sostituzioni 4.8. Per costruzione, si ha che $L - \overline{x}'_k \in \tilde{S}(L - x_k)$ e $W - \overline{y}'_k \in \tilde{S}(W - y_k)$. Applicando il lemma 4.2, si conclude che $L - \overline{x}'_k \in \tilde{S}(L)$ e $W - \overline{y}'_k \in \tilde{S}(W)$. In tal modo anche i punti che descrivono il contorno determinato dai primi k rettangoli verificano la proprietà richiesta. Alla fine del procedimento i rettangoli di indice $j > k$ vengono normalizzati in basso a sinistra. Come per l'asserzione 4.1, si itera il procedimento, costruendo una successione di impaccamenti equivalenti, di cui l'ultimo ha la proprietà richiesta. Per quanto riguarda il passo $k = 1$, si può porre senza perdita di generalità $L \in \tilde{S}(L)$ e $W \in \tilde{S}(W)$ (si veda la sezione 2.1). □

L'uso dell'insieme ridotto dei raster point richiede un maggior lavoro computazionale rispetto alla semplice costruzione di pattern normalizzati. Supponendo di lavorare con insiemi di numeri interi limitati, possiamo d'ora in poi considerare costanti i tempi necessari per le operazioni aritmetiche elementari e per l'accesso agli elementi di un vettore. La soluzione adottata nel

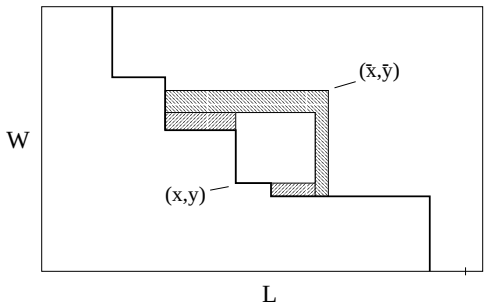

Figura 4.4: Area inutilizzata determinata dall'inserimento di un box. È stato usato l'insieme ridotto dei raster points

presente lavoro consiste nel calcolare nella fase preliminare dell'algoritmo gli insiemi ridotti dei raster point, costruendo poi un vettore che consenta ad ogni passo iterativo di accedere in un tempo costante ai valori necessari per determinare il nuovo contorno.

4.2.3 Un criterio per la generazione dei pattern

Il metodo del contorno di per sé non assicura che ogni pattern sia generato una sola volta. Diverse sequenze di inserimento dei rettangoli possono portare allo stesso impaccamento. Questa situazione viene definita da Scheithauer [29] P-equivalenza (*Permutation*-equivalence), e può portare ad un aumento esplosivo del numero di nodi visitati nell'albero di ricerca. Per evitare questa ridondanza Scheithauer propone un criterio algoritmico per il backtracking di complessità $O(k)$, dove k è il numero di rettangoli dell'impaccamento corrente. L'idea su cui si basa il criterio è la seguente: viene definito un ordinamento lessicografico per le sequenze di inserimento dei rettangoli che generano impaccamenti identici. Per ogni nuovo rettangolo inserito, l'algoritmo effettua un test per controllare se la sequenza di inserimento attuale sia lessicograficamente la minore tra quelle che generano lo stesso impaccamento: in caso contrario viene eseguito il backtracking.
Nel presente lavoro abbiamo introdotto invece il seguente criterio, che si rivela notevolmente vantaggioso rispetto al metodo di Scheithauer.

Criterio 4.1 *Sia* $P = \{(x_i, y_i, \overline{x}_i, \overline{y}_i),\ i = 1, \ldots, n\}$ *un impaccamento parziale, dove la successione* $i = 1, \ldots, n$ *corrisponde alla sequenza di inserimento dei rettangoli; il branching si effettua considerando per il rettangolo* $n+1$*-esimo solo le disposizioni per cui si verifica la condizione* $\overline{x}_{n+1} > x_n$.

Teorema 4.3 *Usando il criterio 4.1, ogni pattern costruibile con il metodo del contorno è generato* al più *una volta.*

Dim.: Consideriamo due diverse sequenze di inserimento dei rettangoli, s_a e s_b, che generano lo stesso pattern. Assumiamo per assurdo che le due sequenze possano essere generate entrambe con il criterio 4.1. Supponiamo che le sequenze coincidano fino al passo $j-1$-esimo, mentre i rettangoli inseriti al passo j-esimo, R_a e R_b, risultano diversi; inoltre si abbia $x_{R_b} \leq x_{R_a}$, dove x_{R_a} e x_{R_b} sono le le ascisse dei punti di inserimento di R_a e R_b. Consideriamo la sequenza generatrice s_a: il rettangolo R_b in questa sequenza è inserito successivamente al rettangolo R_a; sarà quindi il k-esimo elemento della sequenza, con $k > j$. Mostriamo che esiste una sottosequenza di s_a, individuata dagli indici $i_0 = j < i_1 < \cdots < i_M = k$, tale che ogni rettangolo della sottosequenza, a parte i_0, domina il precedente (la relazione di dominanza tra rettangoli è stata definita nella sezione 4.2.1). Generiamo la sottosequenza induttivamente, nel seguente modo: supponiamo di aver generato gli indici $i_0, i_1, \ldots, i_{m-1}$; i_m viene scelto allora come il più piccolo indice di s_a che verifica: A) $i_m > i_{m-1}$, B) $x_k \leq x_{i_m} \leq x_{i_{m-1}}$. Notiamo che necessariamente il rettangolo i_m-esimo domina il rettangolo i_{m-1}-esimo. Infatti, poiché $x_{i_m} \leq x_{i_{m-1}}$, si ha $y_{i_m} \geq \overline{y}_{i_{m-1}}$ in base al metodo del contorno, e quindi $\overline{y}_{i_m} > y_{i_{m-1}}$. Inoltre $\overline{x}_{i_m} > x_{i_{m-1}}$, in base al criterio 4.1; infatti, poiché il criterio impone che $\overline{x}_{i_m} > x_{i_m-1}$, se per assurdo $\overline{x}_{i_m} \leq x_{i_{m-1}}$, si avrebbe che $x_{i_m-1} < x_{i_{m-1}}$; l'indice $i_m - 1$ allora verificherebbe le proprietà A) e B), in contrasto con la richiesta che i_m sia il più piccolo indice che le verifica.
Dunque ogni rettangolo della sottosequenza domina il precedente; in base al metodo del contorno ogni rettangolo della sottosequenza può essere inserito solamente dopo che il precedente è già stato inserito. Si conclude quindi che l'inserimento di R_a deve necessariamente precedere quello di R_b, e quindi s_b non può essere generata. □

Resta da dimostrare che ogni pattern viene generato *almeno* una volta. In effetti il criterio 4.1 non soddisfa questo requisito, tuttavia verifica una proprietà più utile.

Definizione 4.2 *Dati due contorni K e K', diciamo che K'* è dominato da *K se $K' \subseteq U(K)$ ($U(K)$ è la regione coperta da K, come definito nella sezione 4.2.1).*

Teorema 4.4 *Per ogni pattern P generabile con i metodi delle sezioni 4.2.1 (metodo del contorno con normalizzazione) e 4.2.2 (uso dell'insieme ridotto dei raster point), esiste un pattern equivalente P' generabile con gli stessi metodi e l'uso del criterio 4.1, e tale che il contorno di P' è dominato dal contorno di P.*

Dim.: Diamo una dimostrazione costruttiva. Per induzione: si assuma che gli impaccamenti di k rettangoli generabili con i metodi delle sezioni 4.2.1

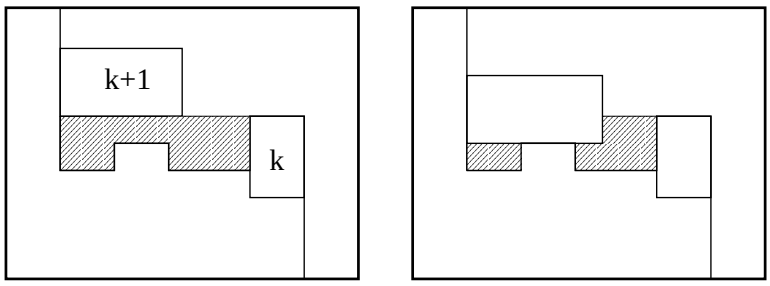

Figura 4.5: Il criterio 4.1 non consente la generazione del pattern (i). Tale pattern però può essere tralasciato in favore del pattern (ii), per ragioni di dominanza.

e 4.2.2 verifichino la tesi. Sia dato un impaccamento P di $k+1$ rettangoli, generabile con i metodi delle sezioni 4.2.1 e 4.2.2, e sia R_{k+1} il rettangolo $k+1$-esimo della sequenza generatrice. Usiamo la notazione P_k per indicare il pattern parziale costituito dai primi k rettangoli della sequenza generatrice di P; per l'ipotesi induttiva esiste un pattern P''_k equivalente a P_k, costituito dagli stessi rettangoli, generabile con i metodi sopracitati e l'uso del criterio 4.1. P''_k inoltre non crea sovrapposizioni con R_{k+1}, per la proprietà di dominanza dei contorni. Consideriamo ora il pattern costituito da P''_k e da R_{k+1}. Normalizziamo R_{k+1} in basso a sinistra, a partire dalla sua posizione corrente (trasliamo R_{k+1} verticalmente verso il basso e quindi orizzontalmente verso sinistra, in modo che la posizione finale risulti normalizzata rispetto al contorno di P''_k); siano $(\eta, \xi, \overline{\eta}, \overline{\xi})$ le coordinate che individuano la nuova disposizione di R_{k+1}. Detto P'' il nuovo impaccamento di $k+1$ rettangoli, per costruzione il contorno di P'' è dominato dal contorno di P. Si osservi a questo punto che P'' è generabile coi metodi delle sezioni 4.2.1 e 4.2.2, ma non necessariamente rispetta il criterio 4.1 (in particolare è possibile che il rettangolo R_{k+1} non verifichi il criterio; un esempio è il pattern (i) della figura 4.5). Sia $m = \max\{i \in \mathbf{Z}_+; i \leq k, x_i < \overline{\eta}\}$; notiamo che m è sempre definito per $k \geq 2$. Osservazione fondamentale: poiché nel contorno di P''_k esiste un punto di inserimento di ascissa η, anche nel contorno di P''_m esiste necessariamente un punto di inserimento di ascissa η (l'ordinata può essere diversa nei due casi).
Generiamo ora un impaccamento P' nel seguente modo: i primi m rettangoli della sequenza coincidano con quelli dell'impaccamento P''; si abbiano cioè identici punti di inserimento e orientazioni. Al passo $m+1$-esimo si inserisca il rettangolo R_{k+1}, nel punto del contorno di P''_m di ascissa η, mantenendo invariata l'orientazione. Quindi a partire dal passo $m+2$-esimo si inseriscano i rimanenti rettangoli di P'' (con indice compreso tra $m+1$ e k), mantenendo invariata la sequenza, i punti di inserimento e le orientazioni.

Si verifica facilmente che la sequenza generatrice di P' è in accordo con i metodi delle sezioni 4.2.1 e 4.2.2; inoltre il contorno di P' è dominato dal contorno di P'' e quindi da quello di P. In più, la sequenza che genera P' è in accordo con il criterio 4.1; per costruzione, infatti, anche i rettangoli $m+1$-esimo e $m+2$-esimo verificano il criterio, così come lo verificano in modo ovvio i rimanenti rettangoli. Poiché P' è il pattern cercato, il passo induttivo della dimostrazione risulta provato. Osserviamo infine che la tesi è verificata banalmente nel caso $k = 2$, primo passo dell'induzione. □

Il criterio 4.1 dunque non consente di generare ogni pattern, ma garantisce che per ogni pattern ammissibile sia generato almeno un pattern equivalente (ovvero con lo stesso insieme di rettangoli). Si può osservare che l'utilizzo del criterio 4.1, oltre a impedire la ripetizione di impaccamenti, consente di evitare la generazione di alcuni pattern che per ragioni di dominanza possono essere tralasciati; per esempio il pattern (i) in figura 4.5 non può essere generato, ma può essere generato il pattern (ii), o una sua versione equivalente. Ciò si rivela notevolmente vantaggioso per la riduzione dell'albero di ricerca. Si noti infine che la valutazione del criterio richiede un tempo costante, il che rappresenta un notevole miglioramento rispetto al test di Scheithauer.

L'algoritmo base risultante esegue i seguenti passi:

1. Data l'istanza (L, W, l, w) da risolvere, inizialmente viene individuata l'istanza equivalente minimale, che sarà risolta al posto del problema originale. Su questa istanza vengono calcolati il bound di Barnes e il bound LP di sezione 2.4. Sia N_{max} il minimo di questi upper bound, e $A = L*W - N_{max}*l*w$ lo spazio sprecato (sfrido) in un impaccamento di N_{max} box.

2. Contemporaneamente, vengono calcolati il massimo numero di H-box H_{max} e il massimo numero di V-box V_{max} secondo quanto visto nella sezione 2.4.

3. L'albero di ricerca è generato ed esplorato secondo il metodo del contorno. Viene evitata la generazione di pattern con più di H_{max} H-box, e con più di V_{max} V-box (un accorgimento già adottato da Isermann).

4. Il branching è effettuato in accordo con il criterio 4.1. In pratica, dato l'insieme $\{(\eta_i, \xi_i)\}$ dei possibili punti di allocazione in un impaccamento parziale, si tratta di trovare l'indice i_H a partire dal quale si hanno punti di allocazione permessi per l'inserimento di un H-box, e analogamente viene individuato l'indice i_V a partire dal quale è possibile inserire V-box.

5. Ogni volta che un box è aggiunto (o tolto) al pattern, l'algoritmo aggiorna l'area sprecata inclusa nel contorno. Quest'area non è più utiliz-

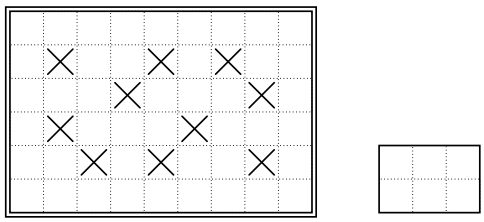

Figura 4.6: Insieme rappresentativo di cardinalità minima per l'istanza $(9, 6, 3, 2)$; non è verificata in questo caso la proprietà di max-min.

zabile fino a che non viene eseguito il backtracking dal nodo corrente. In questa fase viene inclusa l'area di sfrido supplementare determinata dall'uso dell'insieme ristretto dei raster point. Il backtracking viene eseguito quando l'area sprecata supera il valore A, area massima di sfrido, calcolato in precedenza.

6. Se la ricerca termina senza aver trovato soluzioni con un'area sprecata non superiore ad A, l'upper bound viene decrementato, vengono ridefiniti H_{max} e V_{max} e viene effettuata una nuova ricerca.

4.2.4 Un nuovo bound strutturale per il cutting

L'uso dell'insieme ridotto dei raster point si rivela molto efficace nel ridurre l'albero di ricerca, poiché consente di valutare in anticipo le aree del rettangolo che inevitabilmente andranno sprecate, a partire da un certo impaccamento parziale del rettangolo. In questo modo, utilizzando il bound dell'area, è possibile effettuare tempestivamente il backtracking nei sottoalberi in cui l'upper bound non può essere raggiunto.
Tuttavia il controllo dell'area disponibile è un criterio di backtracking debole, che non tiene conto dei vincoli strutturali del PLP. In molti casi, anche dopo aver inserito solo pochi box, nello spazio rimasto risulta impossibile completare l'impaccamento fino a raggiungere il valore teoricamente previsto dal bound dell'area. In questa sezione illustriamo un nuovo test per il backtracking, che effettua una valutazione più accurata dell'area di sfrido.

Nel 1973 Brualdi e Foregger [5] propongono un nuovo bound per il PLP. Data un'istanza (L, W, l, w), consideriamo l'insieme V delle LW celle unitarie contenute nel rettangolo; un sottinsieme $R \subseteq V$ è definito un *insieme rappresentativo* se ogni disposizione di un singolo box copre almeno una cella di R; in altre parole R è rappresentativo se non è possibile inserire un box nel rettangolo senza ricoprire almeno una cella di R. Nella figura 4.6 vediamo un insieme rappresentativo. Dalla definizione risulta che la cardinalità di un insieme rappresentativo è necessariamente maggiore o uguale a quella di un

impaccamento, dal momento che ogni box impaccato deve ricoprire almeno una cella rappresentativa.

Si pone la questione se possa valere un teorema di max-min, ovvero se la cardinalità di un insieme rappresentativo minimo sia sempre uguale a quella di un impaccamento massimo. La risposta è negativa [5], e il controesempio presentato è quello di figura 4.6: un impaccamento ottimo contiene 9 box, mentre Brualdi e Foregger dimostrano che la minima cardinalità per un insieme rappresentativo dell'istanza data è 10.

Tuttavia la proprietà di max-min è verificata per il caso speciale delle istanze del PLP con box armonici. Un box è detto *armonico* quando una delle sue dimensioni è multipla dell'altra. In questo caso è sempre possibile esibire un insieme rappresentativo con cardinalità uguale a quella dell'impaccamento ottimale.

L'insieme rappresentativo può essere usato per ottenere un bound anche sul numero di box che possono essere aggiunti ad un impaccamento parziale. Questo numero è infatti limitato superiormente dalla cardinalità di un qualsiasi set rappresentativo costruito sull'area disponibile. Perciò nel nostro algoritmo di tree-search per ogni nodo dell'albero di ricerca si può costruire euristicamente un set rappresentativo minimo o comunque che "approssima" il minimo, e usarlo come test per il backtracking. Nel presente lavoro tuttavia si è preferito non seguire questa strada, perché l'introduzione di un'euristica ad ogni nodo può appesantire molto l'algoritmo, senza essere compensata da una adeguata efficienza del criterio. Nella pratica infatti il bound fornito dal set rappresentativo minimo è spesso strettamente superiore al valore ottimo, e nel PLP questa differenza può rendere il criterio inutile.

Il metodo da noi proposto per superare questa difficoltà è il seguente: vengono presi in considerazione gli insiemi rappresentativi minimi per l'istanza $(L, W, l, 1)$. Come mostrato in [5], un insieme rappresentativo minimo per questa istanza è dato da $R = \{(x, y) \in \mathbf{Z}_+^2 : x + y = kl - 1,\ k \in \mathbf{Z}_+,\ 0 \leq x \leq L - 1,\ 0 \leq y \leq W - 1\}$, dove (x, y) sono le coordinate dell'angolo in basso a sinistra della generica cella dell'insieme. Si osservi che le celle di R sono disposte su rette diagonali parallele, tracciate ogni l celle. Ogni box $l \times 1$ disposto sul rettangolo coprirà esattamente una cella di R. La cardinalità di R, che per il teorema di max-min corrisponde a quella di un impaccamento ottimo, è data da $|R| = (LW - A)/l$, dove A è lo spazio inutilizzato in una soluzione ottima dell'istanza $(L, W, l, 1)$ (vedi equazione 1.4). Certamente sono minimi anche gli insiemi rappresentativi ottenuti da R per riflessione attorno ad uno degli assi di simmetria del rettangolo. Un esempio di uno di questi insiemi rappresentativi minimi per l'istanza $(8, 8, 5, 2)$ è illustrato in figura 4.7. Data un'istanza (L, W, l, w) e un insieme rappresentativo minimo R per l'istanza $(L, W, l, 1)$, si consideri un pattern di n box $l \times w$. Ogni box copre w celle di R, poiché può essere scomposto in w rettangoli $l \times 1$. Quindi il numero di celle di R non coperte da box è $|R| - nw$. Se in un impaccamen-

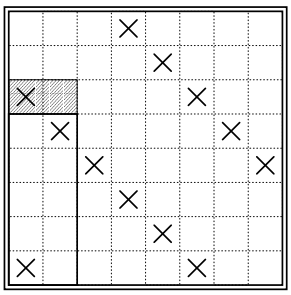

Figura 4.7: Un esempio di applicazione del bound per l'istanza $(8, 8, 5, 2)$. Le crocette individuano un insieme rappresentativo minimo per l'istanza $(8, 8, 5, 1)$.

to parziale il numero di celle contenute nello sfrido è maggiore di $|R| - nw$, non sarà possibile completare l'impaccamento fino ad avere n box. Questa proprietà è interessante quando n corrisponde al valore ottimo cercato: in tal caso l'algoritmo può eseguire un backtracking.
La figura 4.7 illustra un esempio di applicazione del bound: il bound dell'area e di Barnes per l'istanza $(8, 8, 5, 2)$ forniscono il valore 6 (l'ottimo è 5). Si supponga quindi di eseguire l'algoritmo cercando un impaccamento di cardinalità 6. Un qualsiasi insieme rappresentativo minimo R per l'istanza $(8, 8, 5, 1)$ ha cardinalità $|R| = 12$, quindi il numero di celle di R contenute nello sfrido, in un impaccamento di 6 box, è $12 - 6 * 2 = 0$. Dopo aver inserito un box nel rettangolo, si determina uno sfrido che copre una cella dell'insieme rappresentativo evidenziato. Ciò è sufficiente per eseguire il backtracking; infatti le 9 celle di R rimaste non sono sufficienti a coprire i 5 box 5×2 che devono essere ancora inseriti. Si noti che gli altri 3 insiemi rappresentativi minimi che otteniamo per riflessione non avrebbero causato il backtracking in questo caso. Può risultare opportuno perciò eseguire il test con tutti i 4 insiemi rappresentativi.
Nella discussione precedente si sono considerati solo gli insiemi rappresentativi per box di tipo $l \times 1$. Naturalmente si possono ripetere le stesse considerazioni per i box di tipo $w \times 1$, ma ci si può aspettare che usando una lunghezza più piccola i vantaggi del bound siano minori. Nella sperimentazione numerica in effetti si è osservato che l'uso della lunghezza massima del box era più vantaggioso. L'uso congiunto degli insiemi rappresentativi per le due lunghezze si è rivelato utile per alcune istanze difficili, ma troppo costoso in rapporto ai benefici per gli insiemi test standard. Il profitto maggiore si è ottenuto quindi usando contemporaneamente i 4 insiemi rappresentativi descritti.
Per quanto riguarda il costo computazionale, si noti che la struttura regolare degli insiemi rappresentativi considerati permette di contare le celle rapp-

resentative contenute in un rettangolo $X \times Y$ in un tempo $O((X+Y)/l)$ (numero di diagonali contenute nel rettangolo) anziché $O(XY)$; in questo modo il tempo dipende dai rapporti tra le lunghezze; si ha quindi un'invarianza di scala. Ogni box inserito determina uno sfrido che può essere scomposto in una serie di rettangoli, per ognuno dei quali va eseguito il conteggio. A priori il numero di rettangoli di sfrido è (molto) grossolanamente limitato da $O(k)$, dove k è il numero di box dell'impaccamento parziale. In effetti dal punto di vista pratico la situazione è molto migliore, poiché viene applicato prima il bound dell'area, che riduce molto la possibilità di sfridi estesi.
Il bound sviluppato può essere utilizzato anche nelle euristiche ricorsive per il PLP più recenti, che effettuano tree search piuttosto estese e necessitano di tecniche per limitare la ricerca (vedi capitolo 3); anche in questi algoritmi infatti vengono determinati degli sfridi durante la generazione dei pattern. Osserviamo infine che il bound dell'insieme rappresentativo, a differenza degli altri criteri discussi, non può essere applicato al Rectangle Packing Problem.

4.2.5 Un nuovo criterio di dominanza

Nell'algoritmo base descritto nella sezione 4.2.1 resta da definire l'ordine in cui vengono visitati i nodi che si diramano dal nodo corrente. Per ogni impaccamento parziale il branching si riduce a scegliere l'orientamento del nuovo box e il suo punto di inserimento sul contorno. Sono stati presi in considerazione due ordinamenti:

1. Si ordinano le disposizioni prima in base all'orientamento del box, quindi in base al punto di inserimento. I punti di inserimento sono considerati in ordine di ascissa crescente. Nell'implementazione finale si è scelto di considerare prima tutte le disposizioni con box verticale, quindi quelle con box orizzontale.

2. Si ordinano le disposizioni prima in base al punto di inserimento, quindi in base all'orientazione del box. Come in 1) i punti di inserimento sono considerati in ordine di ascissa crescente, e per ogni punto viene valutata prima la disposizione verticale, quindi la orizzontale.

In termini di efficienza computazionale la differenza tra i due metodi si è rivelata netta. Il secondo metodo (vedi sezione 4.2.6) riduce considerevolmente sia i valori medi sia i valori massimi dei tempi di calcolo. Nel caso che l'upper bound iniziale non sia accurato e l'algoritmo sia costretto a effettuare una ricerca completa sull'albero, prima di decrementare l'upper bound e ripartire, il numero di nodi visitati è naturalmente identico nei due metodi, e quindi non si hanno differenze significative nelle prestazioni.

Un vantaggio maggiore si ottiene con l'introduzione di un nuovo criterio di dominanza da noi proposto, che come vedremo è conseguenza del criterio 4.1. Il criterio è ricavato dal seguente teorema.

Teorema 4.5 *Sia* $\{(\eta_i, \xi_i) : i = 1, \ldots, m\}$ *l'insieme dei punti che descrive il contorno di un impaccamento parziale di* n *box per l'istanza* (L, W, l, w). *Sia* (η_k, ξ_k), $k < m$, *un punto del contorno tale che* $\eta_{k+1} - \eta_k \geq l$. *Allora non è possibile ottenere pattern ottimali completando l'impaccamento col metodo del contorno, l'uso dei raster point e il criterio 4.1, se il box* $n+1$*-esimo è inserito in un punto* (η_j, ξ_j), *con* $j > k$.

Dim.: Mostriamo che, con le ipotesi del teorema, se in un completamento il box $n+1$-esimo è inserito in (η_j, ξ_j), con $j > k$, allora l'area $A = \{(x, y) \in \mathbf{R}^2_+ : x < \eta_{k+1}\}$ non può intersecare nessuno dei box del completamento. Infatti, supponiamo per assurdo che qualcuno dei box del completamento intersechi l'area A; sia $\overline{n}$ l'indice del primo box della sequenza di completamento con questa proprietà. A causa del metodo del contorno, le ascisse dei punti descriventi il contorno che si trovano in A restano invariate fino all'inserimento del box $\overline{n}$. In tal caso per il box $\overline{n}$ si ha che $x_{\overline{n}} \leq \eta_k$, quindi $\overline{x}_{\overline{n}} \leq \eta_{k+1}$. Ma secondo il criterio di generazione 4.1 non può essere generata tale disposizione, da cui l'assurdo.
Non resta infine che osservare che l'area A contiene spazio inutilizzato in cui è sempre possibile aggiungere un box in posizione orizzontale. Ciò garantisce che i pattern considerati nell'ipotesi non sono ottimali. □

Dal teorema si deduce che se nel contorno esiste un punto di inserimento (η_k, ξ_k), con $\eta_{k+1} - \eta_k \geq l$, allora tutti i punti di inserimento di ascissa $\eta_i > \eta_k$ possono essere tralasciati nel branching. Ciò permette in moltissimi casi una riduzione consistente dell'albero di ricerca. L'applicazione del criterio ha un costo computazionale costante, di fatto trascurabile.
Il criterio può essere applicato anche al caso del Rectangle Packing Problem, se si considera al posto di l la più grande lunghezza orizzontale disponibile.

4.2.6 Sperimentazione numerica

L'algoritmo è stato scritto in C e fatto girare su un PC IBM Pentium II 300MHz, 128Mb di RAM, sistema operativo Linux 2.0.9. Si è utilizzato CPLEX 6.0 per la risoluzione dei problemi LP considerati nell'algoritmo. La versione base dell'algoritmo (A1) descritta nella sezione 4.2.3 implementa il metodo del contorno (sezione 4.2.1), l'uso dell'insieme ridotto dei raster point (nella seconda versione presentata nella sezione 4.2.2), e il criterio 4.1 per evitare la ripetizione di pattern (sezione 4.2.3). I pattern vengono generati usando l'ordinamento 1) nel branching, descritto nella sezione 4.2.5. Nella versione (A2) dell'algoritmo si introduce il bound dell'insieme rappresentativo nella forma più efficiente illustrata nella sezione 4.2.4. L'algoritmo (A3)

non è altro che (A2) con l'introduzione dell'ordinamento 2) nel branching, visto nella sezione 4.2.5; si è ritenuto interessante evidenziare la differenza di prestazioni introdotta da questa semplice modifica. L'algoritmo (A4) è ottenuto aggiungendo alla versione (A3) il criterio di dominanza descritto nella sezione 4.2.5.

I problemi test utilizzati corrispondono a quelli generalmente considerati in letteratura: SET1 è un insieme di 100 istanze che verificano i vincoli (2.1)-(2.3). Il campione è stato generato im modo random, usando una distribuzione uniforme per le quantità LW/lw, l/w e L/W. SET2 è un insieme di 100 istanze ottenuto allo stesso modo, ma con i vincoli (2.4)-(2.6). L'insieme SET3, che ci è stato gentilmente messo a disposizione da Schei-

Tabella 4.1: SET1 e SET2: risultati computazionali

Set	LW/lw	Alg	t_m	> 1	> 1000
SET1	$[6, 51]$	A1	0.37	3	0
		A2	0.50	1	0
		A3	0.02	0	0
		A4	0.02	0	0
SET2	$[51, 100]$	A1	254.65	47	23
		A2	225.79	41	19
		A3	20.33	9	1
		A4	1.79	7	0

thauer, è costituito da 206 istanze hard selezionate da Nelißen da un campione di circa 40.000 istanze che rappresentano tutte le classi di equivalenza presenti nell'insieme di problemi che soddisfano i vincoli (2.4),(2.5) e

$$51 \leq \frac{LW}{lw} \leq 101 \tag{4.9}$$

(si noti la piccola differenza tra il vincolo (2.6) e (4.9)). Nella tabella 4.1 sono stati comparati i risultati ottenuti dagli algoritmi su SET1 e SET2. Le colonne > 1 e > 1000 indicano il numero di istanze che ha richiesto tempi di calcolo superiori rispettivamente a 1 secondo e a 1000 secondi. Il tempo limite è stato fissato a 1000 sec. La colonna t_m indica il tempo di soluzione medio sull'insieme; in questa e nelle seguenti tabelle i tempi sono indicati in secondi. Nella media sono state incluse anche le istanze non risolte nel tempo limite. Come si vede, la versione A1 non riesce a risolvere 23 istanze di SET2 nel tempo limite fissato, mentre la versione A4 risolve tutte le istanze considerate entro 1000 s. Si osservi che i tempi medi indicati per A1 e A2 sarebbero molto più alti se non fosse stato fissato per l'algoritmo un tempo limite. Facciamo notare inoltre che la versione base A1 è comunque dotata del criterio per evitare le ripetizioni di pattern (*Permutation*-equivalence): senza tale criterio il numero di nodi visitati diventa talmente alto da rendere

impraticabile la sperimentazione numerica.
Si è verificato che in ciascuno dei due insiemi test è presente un'istanza per cui l'upper bound calcolato all'inizio non è accurato. Per ciascuna di queste due istanze l'algoritmo effettua una ricerca completa sull'albero, prima di decrementare il bound e trovare la soluzione ottima.
Si può osservare che l'introduzione del bound del set rappresentativo (algoritmo A2) ha sicuramente comportato un aumento del tempo di elaborazione ad ogni nodo, ma complessivamente si ha una riduzione dei tempi per le istanze più "difficili". È stato verificato che il tempo richiesto per la valutazione di un nodo raddoppia nel caso che venga usato il bound dell'insieme rappresentativo. La riduzione del numero di nodi visitati dovuta al bound è quindi più consistente di quello che i dati della tabella lasciano supporre.
Nella tabella 4.2 è confrontato il numero di nodi esplorati dagli algoritmi (A1) e (A2) per le 10 istanze più "difficili" dell'insieme SET1. La colonna n_{OP} indica il numero di box della soluzione ottima trovata, le colonne t rappresentano i tempi di calcolo.

Tabella 4.2: Istanze hard di SET1

		A1		A2	
Istanza	n_{OP}	t	nodi	t	nodi
(9545,4878,1938,503)	42	0.16	36248	0.27	35661
(14218,7570,3079,875)	38	0.36	81325	0.01	1016
(5475,3615,1289,410)	34	0.22	50351	0.21	31238
(1184,680,257,70)	43	2.28	537742	0.04	4036
(6846,5710,1597,588)	39	0.19	38264	0.02	595
(9798,7680,2209,770)	42	0.23	50693	0.08	6914
(3137,2522,729,214)	48	1.33	307038	0.28	29454
(2611,1666,607,192)	34	0.33	83009	0.37	56874
(5740,2911,1003,488)	27	0.15	41053	0.17	34517
(9386,5819,1600,841)	38	29.63	7777979	46.82	6183132

L'euristica di Morabito e Morales [24], descritta nel capitolo 3, è stata sperimentata su due set S1 e S2 di 100 istanze generate in modo random con lo stesso procedimento di SET1 e SET2; per S2 è stato però usato il vincolo (4.9) anziché il (2.6). Si tratta di un'euristica ricorsiva, implementata in Pascal su un Pentium 100Mhz. Per evitare una crescita eccessiva dell'albero di ricerca la ricorsione è stata limitata a 3 livelli; è stato scelto questo limite perchè consentiva di ottenere una soluzione ottima per tutte le istanze di S1 e S2 (prendiamo in considerazione la versione 3 del loro algoritmo). I tempi medi ottenuti, rapportati a quelli di un Pentium II 300Mhz, sono 0.2 per S1 e 3.0 per S2 nella versione dell'algoritmo che usava l'insieme ridotto dei raster point (3b), 0.1 per S1 e 2.4 per S2 nella versione che non lo usava (3a) (fattori di conversione ricavati dal sito www.specbench.org). Si confrontino

i tempi con quelli della tabella 4.1. SET3 è stato usato da Scheithauer e Terno [31] come test per l'euristica G4, descritta nel capitolo 3. In 167 delle 206 istanze l'euristica ha fornito il valore ottimo. Il tempo medio ottenuto per le 206 istanze è 0.06 secondi (tempo Pentium II 300Mhz-equivalente). La versione (A4) dell'algoritmo trova una soluzione ottima entro il tempo limite di 1000 secondi per 198 istanze. La media dei tempi per queste istanze è 6.87 sec.
In [31] l'euristica G4 viene sperimentata anche su un set di problemi con $LW/lw > 100$: SET4 è costituito dalle 6 istanze $(L, 200, 21, 19)$ dove L assume i valori 200, 250, 300, 350, 400, 450. Queste istanze hanno soluzioni ottime di tipo G4. L'insieme è stato usato da Scheuthauer e Terno per verificare l'efficacia dei bound utilizzati. Nella tabella 4.2.6 vengono confrontati i tempi di calcolo delle istanze di SET4 con quelli ottenuti da Scheitahuer.

Tabella 4.3: Risultati per le istanze SET4 ($W = 200, l = 21, w = 19$)

L	200	250	300	350	400	450
n_{OP}	100	125	149	175	200	225
A4	0.05	0.03	0.19	0.02	0.04	0.04
G4	0.06	0.07	0.29	0.56	0.86	5.21

SET5 è una raccolta di 12 esempi che Morabito e Morales [24] esaminano in dettaglio. Gli esempi D1-D2, N1-N5 e ST1-ST5 sono stati tratti rispettivamente da Dowsland, Nelißen e Scheithauer e Terno. L'istanza N2 è una delle 8 istanze di SET3 non risolte nel tempo limite. La tabella 4.2.6 confronta i tempi ottenuti da (A4) e dall'euristica 3a. Come si può notare, anche in questo caso la maggior parte delle istanze è stata risolte in un tempo confrontabile con quello dell'euristica. In più l'algoritmo esatto risolve in un tempo più che accettabile l'istanza N1, non risolta all'ottimo dall'euristica.
Riassumendo, si può notare che i tempi di calcolo dell'algoritmo esatto sono paragonabili a quelli ottenuti dalle euristiche G4 e 3a; in alcuni casi la ricerca esatta comporta tempi inaccettabili; si hanno casi d'altra parte dove l'algoritmo termina in tempi ragionevoli, dove le euristiche non arrivano all'ottimalità. L'efficacia dell'algoritmo esatto rispetto alle euristiche è in parte sorprendente, dal momento che le due euristiche considerate operano importanti riduzioni dell'albero di ricerca evitando di generare blocchi equivalenti in base a considerazioni di simmetria. D'altra parte si può osservare che queste euristiche non hanno modo di evitare la generazione ripetuta di un pattern, dal momento che questo in genere può essere scomposto in blocchi in più modi.

Tabella 4.4: Istanze SET5: risultati computazionali

Ist.	(L, W, l, w)	n_{OP}	t(A4)	t(3a)
D1	(22,16,5,3)	23	0.01	0.1
D2	(86,82,15,11)	42	0.01	0.1
N1	(43,26,7,3)	53	0.03	0.5 $*$
N2	(87,47,7,6)	97	7524.68	13.3
N3	(153,100,24,7)	90	5.77	0.1
N4	(42,39,9,4)	45	0.02	0.6
N5	(124,81,21,10)	47	0.02	1.6
ST1	(40,25,7,3)	47	0.47	0.6
ST2	(52,33,9,4)	47	0.02	0.9
ST3	(57,44,12,5)	41	0.02	0.3
ST4	(56,52,12,5)	48	0.05	0.6
ST5	(300,200,21,19)	149	0.19	0.1

$*$ soluzione dell'euristica non ottimale

Capitolo 5

Conclusioni

Negli ultimi vent'anni sono stati raggiunti dei risultati teorici notevoli nello studio del PLP: il teorema di Barnes, le classi di equivalenza di Dowsland e la dimostrazione di Tarnowski che il PLP a ghigliottina è in P sono tra i risultati più importanti. Nonostante ciò il PLP resta un problema dalle caratteristiche peculiari nel panorama dei problemi di packing e più in generale di ottimizzazione combinatorica, a tal punto che a tutt'oggi non si dispone ancora di una sua caratterizzazione dal punto di vista della complessità computazionale.
Negli anni più recenti la ricerca ha sondato alcuni aspetti strutturali del PLP, con il risultato di sviluppare metodi euristici che forniscono soluzioni ottime nella stragrande maggioranza delle istanze appartenenti alle classi standard usate in letteratura. Tra questi ricordiamo le strutture ricorsive k-block di Scheithauer e Terno, formulazione teorica che apre tutta una serie di nuove questioni sul PLP su alcune delle quali ci siamo soffermati nel presente lavoro. Dato il successo delle euristiche recenti, abbiamo ritenuto fattibile la costruzione di un algoritmo esatto di tipo tree search che sia nel contempo efficiente sui problemi test standard.
I principali risultati originali ottenuti nel presente lavoro sono i seguenti:
- nel capitolo 1 è stato fornito un controesempio al teorema di Exeler, versione alternativa del teorema di Dowsland;
- nel capitolo 2 è stato analizzato il comportamento asintotico di alcuni upper bound; in particolare è stato dimostrato che la differenza tra l'upper bound di Barnes e il valore ottimo, che per il teorema di Barnes si annulla al crescere delle dimensioni del pallet, è in effetti non limitata superiormente nell'insieme delle istanze del PLP.
- nel capitolo 3 viene presentato un metodo per la generazione sistematica di strutture k-block, con il quale è possibile ottenere un'insieme completo di pattern da utilizzare per la ricerca di impaccamenti a struttura k-block massimali. Con questo metodo è stato verificato che l'euristica di Morales e Morabito trova in effetti l'impaccamento 5-block massimale (nel caso in cui

non viene limitata la ricorsione), così come l'euristica G4 di Scheithauer e Terno trova l'impaccamento 4-block massimale.
- nel capitolo 4 viene elaborato un algoritmo esatto per il PLP; l'aspetto teorico fondamentale è l'individuazione di un insieme di criteri che permettano una pesante riduzione del numero di pattern analizzati, e nel contempo garantiscano la permanenza di soluzioni ottimali nell'albero di ricerca (si veda a questo proposito anche [17]). Il problema della generazione ripetuta di pattern (P-equivalenza) è stato risolto con un nuovo criterio, che si rivela nettamente vantaggioso rispetto al metodo presentato da Scheithauer. Si è inoltre individuato un upper bound per il backtracking che, basandosi su proprietà strutturali del PLP, si rivela decisivo soprattutto nelle istanze "hard". Infine è stato individuato un ulteriore semplice test per la riduzione dell'albero di ricerca, basato su una nuova proprietà di dominanza tra pattern, che si ottiene come conseguenza dell'introduzione del criterio per la P-equivalenza.
L'efficacia di questi criteri è stata valutata attraverso la sperimentazione numerica su insiemi di istanze test standard. L'algoritmo esatto proposto nel presente lavoro mostra che, sfruttando opportunamente le proprietà di cui la struttura del PLP è ricca, è possibile ottenere ottime prestazioni. Osserviamo comunque che alcuni dei nuovi criteri proposti sono stati studiati per il caso più generale del Rectangle Packing Problem.
I test numerici mostrano che le diverse tecniche adottate contribuiscono in maniera determinante alla riduzione dell'albero di ricerca. L'introduzione dei nuovi criteri ha consentito di abbassare i tempi di calcolo di alcuni ordini di grandezza, permettendo un confronto con le euristiche recenti ([31] e [24]), che hanno mostrato di raggiungere quasi sempre l'ottimalità nei problemi test standard. In gran parte dei casi l'algoritmo esatto ha prestazioni paragonabili a quelle delle euristiche. Nel contempo istanze hard non risolte all'ottimalità dalle euristiche sono state risolte in tempi accettabili.

Bibliografia

[1] F. W. Barnes, *Packing the maximum number of $m \times n$ tiles in a large $p \times q$ rectangle*, Discrete Math 26, 1979, 93-100.

[2] J. E. Beasley, *An exact two-dimensional non-guillotine cutting tree search procedure*, Oper. Res. 33, 1985, 49-64.

[3] A. Belvedere, *Su un problema di packing di rettangoli*, Tesi di Laurea, Corso di Laurea in Matematica, Padova 1996.

[4] E. Bischoff e W. B. Dowsland, *An application of the micro to product design and distribution*, J. Opl Res. Soc. 33, 1982, 271-280.

[5] R. A. Brualdi e T. H. Foregger, *Packing boxes with harmonic bricks*, J. Combin. Theory (B) 17, 1974, 81-114.

[6] H. Carpenter e W. B. Dowsland, *Practical considerations of the pallet loading problem*, J. Opl Res. Soc. 36, 1985, 489-497.

[7] N. Christofides e C. Whitlock, *An algorithm for two-dimensional cutting problems*, Oper. Res. 25, 1977, 30-44.

[8] P. De Cani, *A note on the two-dimensional rectangular cutting-stock problem*, J. Opl Res. Soc. 29, 1978, 703-706.

[9] K. A. Dowsland, *Determining an upper bound for a class of rectangular packing problems*, Comput. & Ops. Res. 12, 1985, 201-205.

[10] K. A. Dowsland, *The three-dimensional pallet chart: an analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems*, J. Opl Res. Soc. 35, 1984, 895-905.

[11] K. A. Dowsland, *An exact algorithm for the pallet loading problem*, Eur. J. Oper. Res. 31, 1987, 78-84.

[12] K. A. Dowsland, *A combined data-base and algorithmic approach to the pallet loading problem*, J. Opl Res. Soc. 38, 1987, 341-345.

[13] K. A. Dowsland, *Packing problems*, Europ. J. Oper. Res. 56, 1992, 2-14.

[14] H. Dyckhoff, *A typology of cutting and packing problems*, Eur. J. Oper. Res. 44, 1990, 145-159.

[15] H. Dyckhoff, G. Scheithauer e J. Terno, *Cutting and Packing*, in *Annotated Bibliographies in Combinatorial Optimization*, a cura di M. Dell'Amico, F. Maffioli e S. Martello, John Wiley & Sons, Chichester 1997, 393-413.

[16] H. Exeler, *Upper bounds for the homogeneous case of a two-dimensional packing problem*, ZOR - Methods and Models of Operations Research 35, 1991, 45-58.

[17] M. Fischetti e P. Toth, *A new dominance procedure for combinatorial optimization problems*, Operations Research Letters 4, 1988, 181-187.

[18] R. J. Fowler, M. S. Paterson e S. L. Tanimoto, *Optimal packing and covering in the plane are NP-complete*, Inf. Proc. Letters 12, 1981, 133-137.

[19] M. R. Garey e D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, Freeman, San Francisco 1979.

[20] J. C. Herz, *Recursive computational procedure for two-dimensional stock cutting*, IBM J. Res. Develop. 16, 1972, 462-469.

[21] H. W. Lenstra JR, *Integer programming with a fixed number of variables*, Math. Oper. Res. 8, 1983, 538-548.

[22] A. N. Letchford e A. Amaral, *Analysis of upper bounds for the pallet loading problem*, sottoposto a Eur. J. Opl Res., giugno 1999.

[23] S. Martello e D. Vigo, *Exact solution of the two-dimensional finite bin packing problem*, Management Science 44, 1998, 388-399.

[24] R. Morabito e S. Morales, *A simple and effective recursive procedure for the manufacturer's pallet loading problem*, J. Opl Res. Soc. 49, 1998, 819-828.

[25] J. Nelißen, *New approaches to the pallet loading problem*, Working Paper RWTH Aachen 1993.

[26] J. Nelißen, *How to use structural constraints to compute an upper bound for the pallet loading problem*, Europ. J. Oper. Res. 84, 1995, 662-680.

[27] F. Penzo, *Problema di cutting e packing*, Tesi di Laurea, Corso di Laurea in Scienze Statistiche ed Economiche, Padova 1997.

[28] G. Scheithauer, *Algorithms for the container loading problem*, in *Operations Research Proceedings 1991*, Springer-Verlag Berlin, Heidelberg 1992, 445-452.

[29] G. Scheithauer, *Equivalence and dominance for problems of optimal packing of rectangles*, Ricerca Operativa 83, 1997, 3-34.

[30] G. Scheithauer, *A new LP-bound for the rectangle packing problem*, PREPRINT, 1995.

[31] G. Scheithauer e J. Terno, *The G4-heuristic for the pallet loading problem*, J. Opl Res. Soc. 47, 1996, 511-522.

[32] G. Scheithauer e J. Terno, *A new heuristic for the pallet loading problem*, Operations Research Proceedings 1995, Springer-Verlag Berlin Heidelberg, 1996, 84-89.

[33] G. Scheithauer e U. Sommerweiß, *4-block heuristic for the rectangle packing problem*, Working Paper 1997.

[34] A. Smith e P. De Cani, *An algorithm to optimize the layout of boxes in pallets*, J. Opl Res. Soc. 31, 1980, 573-578.

[35] A. Tarnowski, J. Terno e G. Scheithauer, *A polynomial time algorithm for the guillotine pallet loading problem*, INFOR 32, 1994, 275-287.