UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA GESTIONALE

TESI DI LAUREA

APPLICAZIONE DI ALGORITMI DI NESTING ALLA ROBOTICA INDUSTRIALE

Relatore: Ch.mo Prof. GIOVANNI BOSCHETTI

Laureando: FABIO PASTORELLO

Anno Accademico: 2010/2011

INDICE

SOMMARIO

In questa tesi si esamina il problema del nesting, che consiste nel disporre un insieme di oggetti all'interno di uno o più contenitori di dimensioni rnaggiori, con obiettivi che non sempre si riducono a quello di utilizzare in modo ottimale il volume disponibile.

Si descrivono in dettaglio alcune tra le più significative procedure elaborate negli ultimi anni per la soluzione di tali problemi.

Si analizza in particolare un algoritmo creato sulla base dei principi di Bishoff & Dowsland.

Tale algoritmo, è utilizzato per studiare il problema del caricamento di un nastro trasportatore.

I risultati delle simulazioni vengono presentati e commentati nell'ultimo capitolo.

INTRODUZIONE

Con la denominazione di problemi di Cutting and Packing (C&P) si indica la varietà assai articolata e complessa di questioni legate all' ottimizzazione dello spazio.

A partire dalla fine degli anni '60 si è registrato un crescente interesse nei confronti dei problemi di Cutting & Packing, accompagnato dalla pubblicazione di un numero davvero notevole di saggi ed articoli, molti dei quali incentrati sulla presentazione di algoritmi per la soluzione di questioni specifiche, questo per le importanti ricadute economiche ed operative di eventuali soluzioni efficaci ed efficienti. Parallelamente ed inevitabilmente si è arricchita la relativa letteratura, al punto che risulta molto difficile, se non impossibile, elaborare un quadro sinottico dello stato dell'arte.

E' semplice constatare come il contesto industriale offra numerosi esempi di situazioni nelle quali tali problematiche emergono in modo assolutamente esplicito. E' sufficiente citare, ad esempio:

- il sezionamento di oggetti mono- e bi-dimensionali (tubi, fili, barre; lastre, fogli, pannelli);
- Il caricamento di veicoli, pallets, containers;
- il bilanciamento e lo scheduling di linee di produzione;
- la ripartizione di spazi di memoria per il contenimento di dati;
- la programmazione di spot pubblicitari di durata diversa utilizzando un numero minimo di interruzioni dei programmi.

Da uno studio della letteratura, in primo luogo si avverte una netta differenza qualitativa fra i problemi di C&P mono- e bi- dimensionali e quelli, ben più complessi ed articolati, che coinvolgono figure in tre dimensioni. Per questi ultimi, la valutazione delle attuali conoscenze è tutt'altro che univoca, anche se la tecnica più affermata e convincente per affrontare un problema di packing in tre dimensioni resta quella di caricare gli oggetti all'interno dei contenitori costruendo strati paralleli e trasformando quindi il problema tri-dimensionale in una combinazione di problemi a due dimensioni.

Il primo capitolo di questa tesi ha l'obiettivo di descrivere in modo più preciso la struttura logica dei problemi di C&P, riassumendo per sommi capi i concetti più importanti che permettano di avere un quadro generale.
Il secondo capitolo illustra i principi alla base delle varie tecniche risolutive e concentra l'attenzione sui più significativi procedimenti elaborati negli ultimi anni dai ricercatori: l' algoritmo di Bishoff & Dowsland e quello di Morabito & Morales.

Il terzo capitolo illustra un algoritmo che permette l' ottimizzazione di spazi rettangolari. Ne vengono mostrati i principi di funzionamento, le modalità di creazione dello stesso,le future possibili implementazioni al fine di rispondere alle esigenze concrete di una generica industria, relative al processo di caricamento ottimale di un nastro trasportatore.

CAPITOLO PRIMO
Struttura logica e classificazione dei problemi di Cutting & Packing

1.1 Struttura logica dei problemi di Cutting & Packing

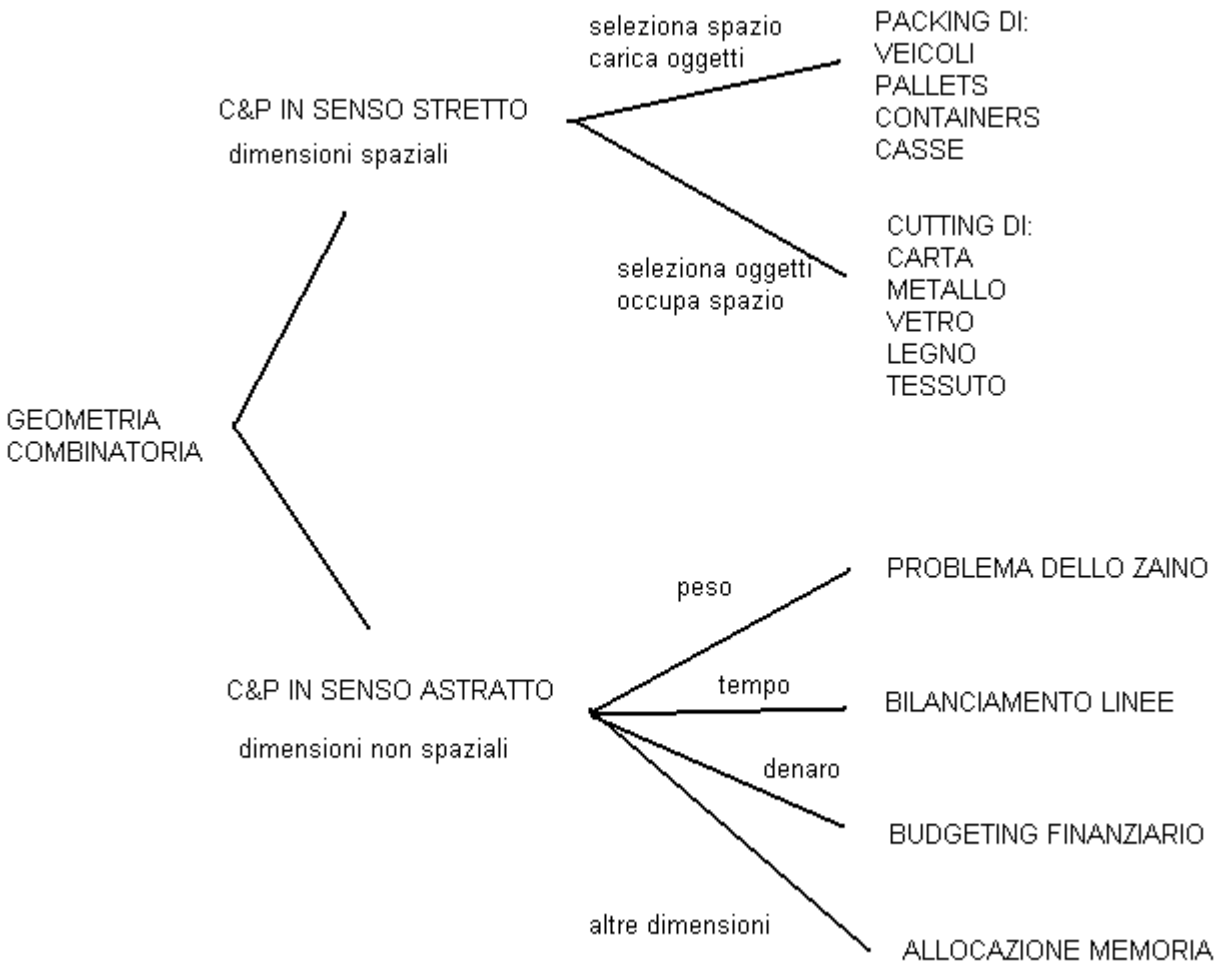
I problemi di C&P sono generalmente caratterizzati dalla presenza di due classi di dati di ingresso, i cui elementi definiscono entità geometriche di forma assegnata, o figure, in uno spazio ad una, due o tre dimensioni:

l'insieme degli oggetti "grandi" , che chiameremo objects

l'insieme degli oggetti "piccoli", o items

In generale si tratterà di realizzare un sezionamento (pattern) di uno o più objects mediante combinazioni geometriche di i items opportunamente orientati. I pezzi residui, ossia le figure del pattern che non appartengono all'insierne degli items, vengono considerati come scarto (o sfrido). Il sezionamento si prefigge di raggiungere scopi di volta in volta precisati al momento della formulazione del problema, i quali raramente si esauriscono nella richiesta di minimizzare lo scarto.

Nel 1990 Dychoff propose una classificazione dei problemi di C&P, la quale innanzitutto distingue tra C&P in senso stretto e C&P astratto. Nel primo caso si affrontano questioni in cui objects ed items sono caratterizzati da dimensioni spaziali (lunghezze, aree o volumi) ; diversamente, nei casi di C&P astratto si possono considerare dimensioni "generalizzate" quali peso, tempo, spazio di memoria.



Dalla definizione delle operazioni di cutting e packing emerge il profondo legame concettuale che esiste tra loro.

in una operazione di cutting si taglia un corpo solido, ovvero si riempie lo spazio da esso occupato con quello occupato da oggetti più piccoli

simmetricamente,

in una operazione di packing si seziona lo spazio vuoto di un contenitore, ovvero si collocano corpi solidi di piccole dimensioni secondo un pattern geometrico prestabilito.

Da tali definizioni, appare contata la distinzione tra problemi di cutting, in cui un oggetto solido di grandi dimensioni viene sezionato per ottenere pezzi più piccoli e problemi di packing o loading, in cui, di fatto, gli objects da sezionare sono gli spazi utilizzabili di veicoli, containers, palletts, casse e così via. In questo caso, il caricamento di items di piccole dimensioni nei contenitori può essere considerato alla stregua di una partizione dello spazio vuoto in zone che vengono occupate dagli items ed altre che restano vuote e quindi inutilizzate.

1.2 Classificazione dei problemi di Cutting & Packing

Per poter esporre la classificazione di Dyckoff, è necessario definire le diverse caratteristiche che possono presentare sia items object e patterns, sia il problema stesso.

1.2.1 Obiettivo

Un obiettivo è essenzialmente una condizione di massimo o di minimo posta su una o più grandezze significative. E' ovvio supporre che i problemi di C&P siano nati dall'esigenza di rendere minimo lo scarto, sia che si tratti di sfrido (nel cutting) che di spazio inutilizzato (nel packing). Un'analisi
più attenta dei numerosi casi pratici descritti in letteratura ci permette comunque di capire che questo non può essere l'unico criterio ad indirizzare la ricerca di soluzioni. Come semplice esempio, si consideri il taglio di lamiere di acciaio per ottenere pannelli

di dimensione assegnata. E' opportuno trovare un comprornesso tra la necessità di contenere la percentuale di scarto e quella di non complicare esageratamente la complessità delle operazioni di sezionamento: il pattern che minimizza la quantità di sfrido può essere infatti troppo costoso in termini di tempo richiesto all'operatore. Allo stesso modo, nel caso del caricamento di un container, si devono scartare piani che, pur minimizzando il volume dello spazio inutilizzato, richiedano agli addetti movimentazioni troppo complicate o portino a disposizioni del carico non stabili. Accettando comunque, come per altri aspetti già trattati, una descrizione di massima, si incontrano prevalentemente funzioni obiettivo riguardanti:

- le quantità (di items, objects o scarto);
- la geometria dei patterns;
- la successione, la combinazione o il numero di patterns utilizzati;

Non sono infine rare le situazioni in cui si considerano due o più obiettivi contemporaneamente.

1.2.2 Disponibilità

Praticamente tutti i casi pratici sono condizionati da vincoli sulle quantità di objects o di items disponibili, sull'ordine in cui essi devono essere utilizzati o, ancora, sulla data entro la quale objects o items specifici devono essere processati.

In particolare le quantità possono essere definite in modo rigido fin dall'inizio o ammettere invece un certo grado di tolleranza. Questa seconda eventualità semplifica di norma la ricerca di soluzioni.Potrebbero inoltre essere presenti vincoli di

assegnazione: nell'elaborazione e nella collocazione dei pattern si deve chiarire se si debbano impiegare tutti gli items (objects) disponibili, o una selezione di questi. Un'attenta analisi delle situazioni più comuni induce a considerare due soli casi:

- utilizzare un sottoinsieme di items per ottenere patterns che ricoprano l'intero insieme degli objects (tutti gli objects, una selezione di items);
- scegliere un sottoinsieme di objects all'interno dei quali allocare tutti gli items disponibili (tutti gli items, una selezione di objects).

1.2.3 Dimensionalità

La dimensionalità si riferisce più propriamente al problema, o al tipo di patterns utilizzati per la soluzione, che ai singoli objects o items considerati.

Può essere definita come il numero minimo di dimensioni (ciascuna espressa attraverso numeri reali) necessarie per descrivere la geometria dei patterns. In particolare, accanto ai tradizionali casi mono- , bi- e tri-dimensionali, si possono incontrare situazioni (nel cutting) in cui gli objects sono descritti da due dimensioni significative, una sola delle quali è però fissata (ad es. la larghezza di un coil di acciaio) ed altre (nel packing) in cui è necessario considerare, con le tre dimensioni spaziali, anche duella temporale (ad es. scatole che devono restare all'interno dei contenitori per un periodo di tempo fissato). Problemi siffatti si definiscono rispettivamente 1,5- e multi-dimensionali.

1.2.4 Restrizioni sui patterns

La combinazione di piccole figure che porta alla definizione di un pattern è spesso sottoposta a pesanti limitazioni, legate in prevalenza alle specifiche tecnologie di taglio o di caricamento utilizzabili, nonché alla natura dei materiali coinvolti. Sebbene la varietà di tali condizioni sia molto ampia, i casi più comuni riguardano:

- vincoli sulle distanze minima e massima tra items o tra tagli che sezionano objects (ad. es. nel taglio di materiali vetrosi o nel caricamento di containers);
- vincoli sull'orientazione relativa degli items (ad. es. nel taglio di materiali non isotropi o nel caricamento di prodotti fragili) ;
- vincoli sul numero complessivo e sulla varietà massima di ftoms che compongono un pattern;
- vincoli sui tipi di taglio consentiti.

1.2.5 Geometria delle figure

La figura di un object o di un item è la sua rappresentazione geometrica nello spazio delle dimensioni significative. Essa è univocamente determinata quando se ne conoscano la forma, la taglia e l'orientazione. La forma di una figura si dice rispettivamente regolare o irregolare a seconda che la si possa descrivere con un numero più o meno contenuto di parametri significativi. Due figure geometricamente simili hanno la stessa forma. Due figure di uguale forma potranno differire solo per taglia ed orientazione; se esse hanno anche la stessa taglia, saranno geometricamente congruenti e potranno essere distinte solo in base alla loro orientazione.

Consideriamo in proposito il caso semplice di un parallelepipedo rettangolo avente dimensioni x, y e z diverse tra loro. Se per esso sono consentite nel pattern tutte le orientazioni, di fatto lo si considera come un'unica figura. Diversamente, se sono consentite soltanto orientazioni che differiscano tra loro per una rotazione di 90°, come nel caso di scatole che debbano poggiarsi su una faccia stabilita a priori, il parallelepipedo individua due figure distinte. Se, infine, il pattern prescrive rigidamente l'orientazione del blocco, quest'ultimo rappresenta un insieme di sei figure distinte.

1.2.6 Assortimento

Ciascun problema è caratterizzato non soltanto dalla geometria delle figure coinvolte, ma anche dal loro assortimento, ossia dal numero di figure utilizzabili, tanto per gli oggetti grandi, quanto per quelli piccoli. In tal senso è opportuno distinguere le situazioni in cui objects ed items hanno forme diverse da quelle in cui essi differiscono invece solo per taglia o orientazione. Un'altra questione riguarda invece la possibilità, non sempre assicurata, di combinare comunque tra loro le figure consentite.

1.2.7 Classificazione di Dyckoff

Caratteristica	Sigla	Significato	Note
Dimcnsionalità			
	1	Problema uni-dimensionale	
	2	Problema bi-dimensionale	
	3	Problema tri-dimensionale	
	N	Problema N-dimensionale	N>3
Assegnazione			
	B	Tutti gli objects, una selezione di items	B sta per Beladeproblem
	V	Tutti gli ítems, una selezione di objects	V sta per VerladeProblem
Assortimento objects			
	O	Un solo object	
	I	Objects geometricamente congruenti	
	D	Objects diversi tra loro	
Assortimento items			
	F	Pochi items diversi tra loro	Si intende una dozzina
	M	Molti items di molte forme diverse	Migliaia di ítems
	R	Molti items di poche forme diverse	Poche decine di forme
	C	Items geometricamente congruenti	

La tabella offre la possibilità di 4 x 2 x 3 x 4 = 96 diverse combinazioni, ciascuna delle quali viene indicata con una quadrupla . Quasi tutte le quadruple rappresentano un caso reale.

CAPITOLO SECONDO
Tecniche risolutive e Algoritmi

2.1 Tecniche risolutive

E' opportuno cercare di classificare, se pur sommariamente, le tecniche e gli algoritmi descritti in letteratura. A questo proposito alcuni autori hanno evidenziato interessanti linee guida.

Hinxman distingue innanzitutto fra metodi esatti e metodi euristici. Un metodo esatto garantisce il raggiungimento della soluzione ottima; viceversa un metodo euristico non può fornire garanzie sull'ottimalità della soluzione e, nella maggioranza dei casi, si limita a produrre soluzioni ragionevolmente buone, sulla base di criteri fissati a priori.

Dagli formula acute osservazioni relativamente al cutting stock, che si possono estendere senza dubbio all'intera categoria delle questioni di C&P. Egli nota come ciascuna realtà operativa o produttiva imponga, in sede di formulazione del problema, richieste, vincoli e restrizioni legate alle tecnologie utilizzate, al tipo di prodotto, alle abitudini degli addetti e dei clienti. Sfortunatamente, non è sempre possibile dare a tutte queste specifiche un'adeguata interpretazione in termini matematici; quando poi ci si riesce, si rischia di complicare eccessivamente l'insieme di equazioni (e disequazioni) di partenza. In conclusione, tanto più complesso e peculiare è il contesto, tanto più preferibilmente si utilizzeranno metodi euristici per la ricerca di soluzioni.

2.1.1 Programmazione lineare (esatto)

Formulando l'ipotesi aggiuntiva che gli x siano sufficientemente grandi e che esista una tolleranza sugli ordinativi N , è possibile rilassare la condizione di integrità x€ Z+ accettando soluzioni frazionarie che vengono successivamente arrotondate:

- per eccesso, determinando una sovrapproduzione;
- per difetto, rendendo successivamente necessario affrontare la questione dei pezzi mancanti.

In entrambi i casi si utilizza il metodo del simplesso. Tuttavia, già considerando un numero relativamente basso di objects ed itetms differenti, la quantità di patterns accettabili, e quindi di incognite, "esplode" secondo il caratteristico comportamento dei problemi combinatori. Per questo motivo, l'attenzione dei ricercatori si è rapidamente orientata verso criteri payless che permettessero di tenere sotto controllo il processo di generazione dei patterns, selezionando opportunamente quelli più promettenti. Si è rivelato particolarmente incisivo, a tal fine, l'approccio di Gilmore e Gomory, che hanno utilizzato una versione modificata del simplesso, introducendo alla fine di ogni step un nuovo pattern, ricavato risolvendo un problema dello zaino ausiliario. L'algoritmo mostra in genere convergenza verso la soluzione e la sua esecuzione viene arrestata quando le successive iterazioni non producono più miglioramenti significativi in termini di riduzione della funzione obiettivo.

2.1.2 Branch and bound (esatto)

Ovviamente, la via più sicura per risolvere il problema sarebbe quella di elencare tutte le soluzioni possibili scegliendo quella corrispondente al minimo della funzione obiettivo (metodo esaustivo). La natura combinatoria della questione rende la lunghezza ed il peso computazionale di tale elenco assolutamente inaccettabile già in situazioni apparentemente semplici. In alternativa, esistono strategie enumerative, che consistono di norma nell'esplorazione di un albero di ricerca, condotta nel rispetto di regole fissate a priori. La radice di tale albero è associata al problema iniziale; da essa discendono nodi che rappresentano sottoproblemi ricavati da quello originario fissando il valore di una delle incognite (nel caso specifico scegliendo quante volte utilizzare un particolare pattern), e così via per i nodi di livello successivo. Per evitare che la costruzione dell'albero si riveli a sua volta un semplice elenco di tutte le soluzioni è necessario abbreviarne la visita, utilizzando algoritmi di tipo Branch and Bound (B&B). Questi permettono la semplificazione desiderata, restringendo l'esplorazione alle zone più prornettenti dell'albero e "potando" tempestivamente rami che conducano a soluzioni poco vantaggiose.

2.1.3 Riduzione del problema (euristico)

Si tratta di scomporre il problema iniziale in quesiti più semplici, eventualmente ripetendo l'operazione per ciascuno dei sottoproblemi generati e confrontando tra loro i risultati di scomposizioni diverse. L'unione delle soluzioni parziali così ottenute forma

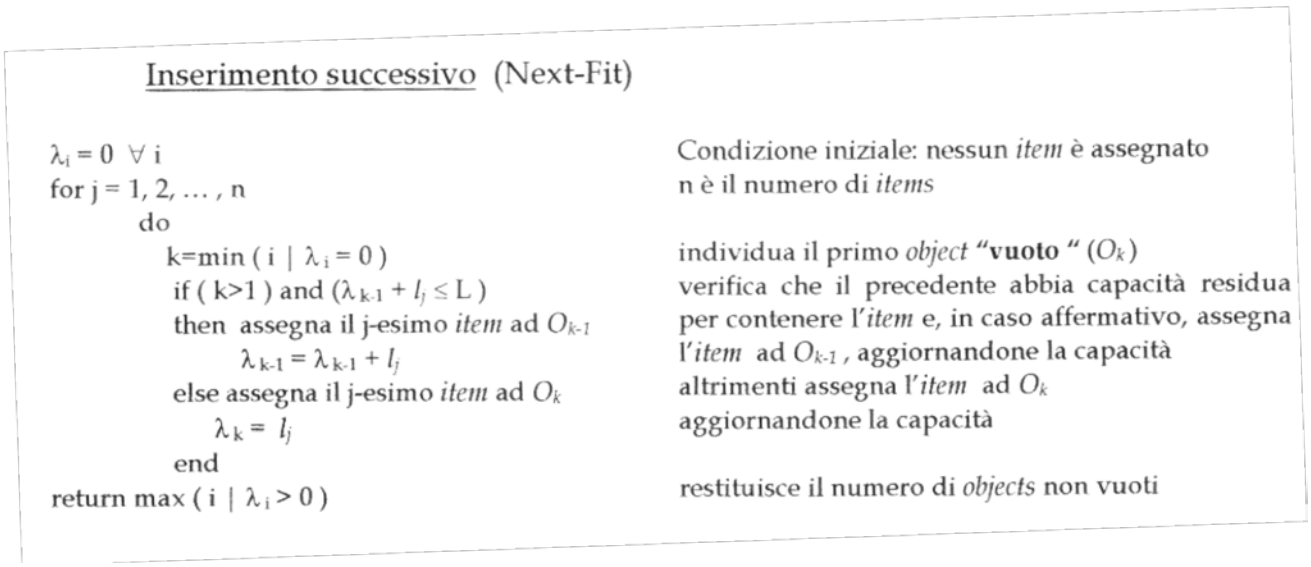
una soluzione dei problema originario: è tuttavia importante osservare che, da una combinazione di soluzioni ottime, non si ottiene necessariamente una soluzione ottima. Il limite più marcato di questa tecnica sta nel progressivo peggioramento della qualita della soluzione a mano a mano che l'insieme degli items si svuota.

2.1.4 Casi particolari

Limitatamente al caso di objects congruenti, sono stati proposti semplici algoritmi euristici, che permettono di trovare rapidamente una soluzione, con livelli di approssimazione crescenti e prestazioni accettabili. Tali algoritmi, posizionano gli items a mano a mano che essi vengono considerati, senza utilizzare alcuna informazione sulle dimensioni dei successivi.

2.1.4.1 Algoritmo di inserimento successivo (Next –Fit)

Colloca il primo item dentro al primo object. In seguito, piazza ciascuno degli item nell'ultimo contenitore utilizzato se possibile, oppure in uno nuovo.



Inserimento successivo (Next-Fit)

$\lambda_i = 0 \ \forall$ i — Condizione iniziale: nessun *item* è assegnato
for j = 1, 2, … , n — n è il numero di *items*
do
k=min (i | $\lambda_i = 0$) — individua il primo *object* **"vuoto "** (O_k)
if (k>1) and ($\lambda_{k-1} + l_j \leq L$) — verifica che il precedente abbia capacità residua
then assegna il j-esimo *item* ad O_{k-1} — per contenere l'*item* e, in caso affermativo, assegna
$\lambda_{k-1} = \lambda_{k-1} + l_j$ — l'*item* ad O_{k-1} , aggiornandone la capacità
else assegna il j-esimo *item* ad O_k — altrimenti assegna l'*item* ad O_k
$\lambda_k = l_j$ — aggiornandone la capacità
end
return max (i | $\lambda_i > 0$) — restituisce il numero di *objects* non vuoti

Esempio numerico

Siano assegnate le misure di 12 items, che devono essere allocati in contenitori uguali di lunghezza 100:

λ_i={50, 3, 48, 53, 53, 4, 3, 41, 23, 20, 52, 49 }

I.'algorifino di inserimento successivo fornirà la seguente soluzione:

Cl :	50	3	
C2:	48		
C3:	53		
C4:	53	4	3
C5:	41	23	20
C6:	52		
C7:	49		

2.1.4.2 Algoritmo di inserimento al principio (First -Fit)

Colloca ogni item nel primo contenitore sufficientemente capace. Se non ne trova alcuno, ne utilizza uno nuovo.

Inserimento al principio (First-Fit)

$\lambda_i = 0 \ \forall$ i — Condizione iniziale: nessun *item* è assegnato

for j = 1, 2, … , n — N è il numero di *items*

do

k=min (i | ($\lambda_i + l_j \leq L$) — Individua il primo *object* **con capacità r sufficiente a contenere l'***item*

assegna il j-esimo *item* ad O_k — Assegna l'*item* ad O_k

$\lambda_k = \lambda_k + l_j$ — Aggiornandone la capacità

end

return max (i | $\lambda_i > 0$) — Restituisce il numero di *objects* non vuoti

Con gli stessi dati del caso precedente si ottiene la seguente soluzione:

C1:	50	3	4 3	23
C2:	48	41		
C3:	53	20		
C4:	53			
C5 .	52			
C6:	49			

2.1.4.3 Algoritmo di inserimento ottimale (Best-Fit)

Posiziona ciascun item scegliendo, fra i contenitori già utilizzati con la necessaria capacità, quello che offre lo spazio maggiore

Inserimento otiinnale (Best-Fit). Sempre utilizzando gli stessi dati:

C1 :	50	3	41
C2:	48	4	3
C3:	53	23	
C4 :	53	20	

2.1.4.4 Algoritmo di inserimento al principio con ordinamento decrescente (First-Fit Decreasing).

La logica è la stessa dei precedenti, ma gli items vengono allocati dopo essere stati ordinati per lunghezze decrescenti.

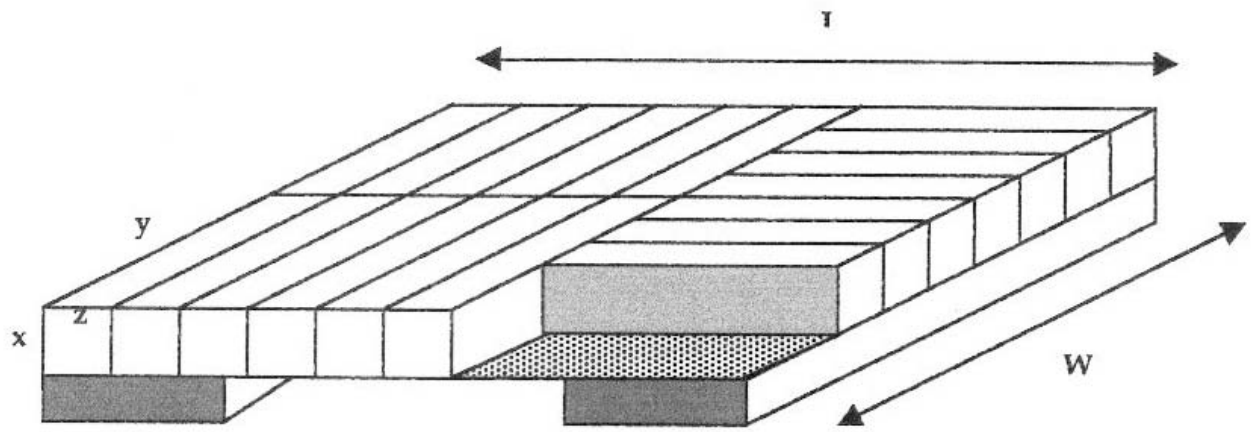
λ_i={53, 53, 52, 50, 49, 48, 41, 23, 20, 4, 3, 3}

Cl :	53	41	4	
C2:	53	23	20	3
C3:	52	48		
C4:	50	49		
C5:	3			

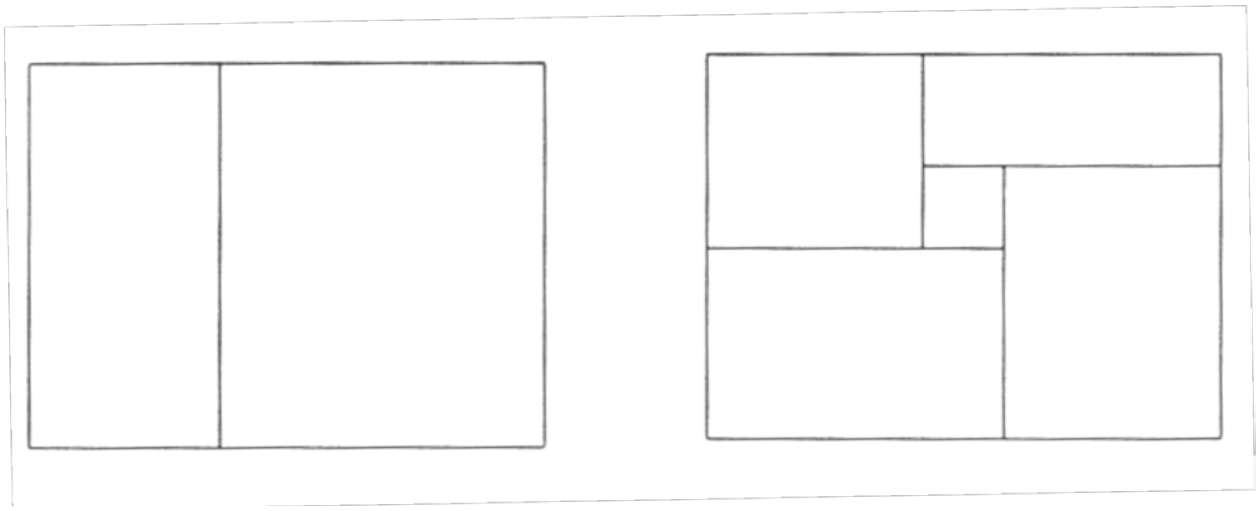
2.2 Algoritmi risolutivi

2.2.1 Algoritmo di Bischoff e Dowsland

Gli autori affrontano il problema della ricerca dei patterns bi-dimensionali che ricoprono in modo ottimale il piano di carico



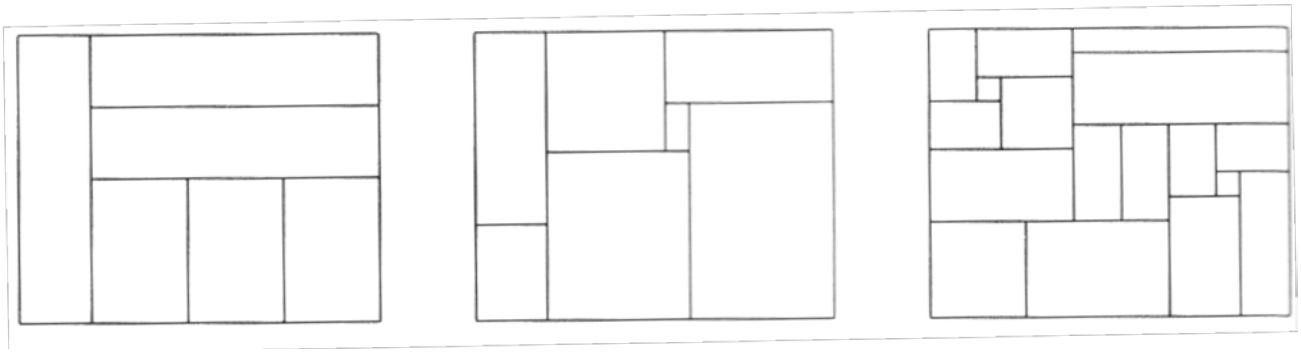
Il cuore dell'algoritmo è la sezione dedicata alla elaborazione dei patterns, della quale viene qui descritta sinteticamente la logica. E' necessario in primo luogo classificare i patterns in base al tipo di tagli, o sezionamenti, dai quali essi sono costituiti. Si distinguono quindi i tagli a ghigliottina da quelli non a ghigliottina. Un taglio a ghigliottina, applicato ad un rettangolo, lo divide a metà in due rettangoli più piccoli; altrimenti si parla di taglio non a ghigliottina. Questi ultimi vengono poi chiamati tagli non a ghigliottina dei primo ordine se, applicati ad un rettangolo, producono un insieme di cinque rettangoli più piccoli, senza che sia possibile individuarne una coppia separata da un taglio a ghigliottina.



TAGLIO A GHIGLIOTTINA TAGLIO NON A GHIGLIOTTINA

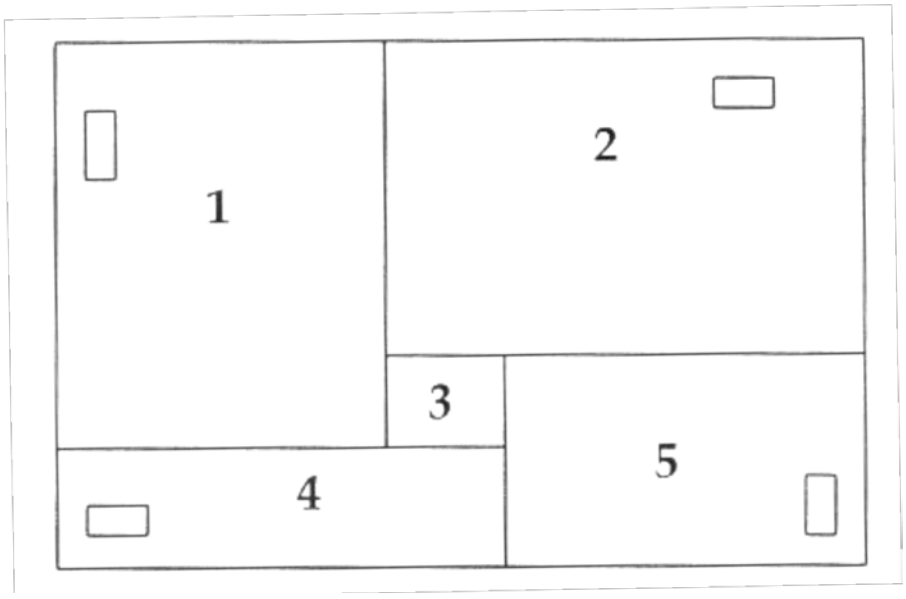
Utilizzando questa terminologia, si definiscono i seguenti tipi di pattern :

- A ghigliottina, se ottenuti applicando ripetutamente ed esclusivamente tagli a ghigliottina;
- Non a ghigliottina del primo ordine, formati mediante successivi tagli a ghigliottina e/o non a ghigliottina del primo ordine: ciò significa che un pattern a ghigliottina è un caso particolare di pattern non a ghigliottina del primo ordine;
- Non a ghigliottina di ordine superiore altrimenti.



L'algoritmo cerca la soluzione ottimale nell'insieme dei patterns non a ghigliottina, preferiti perché in grado di assicurare al carico maggiore stabilità, con il seguente procedimento:

- Si suddivide la base del pallet mediante un pattern non a ghigliottina del primo ordine che porta alla definizione di al massimo cinque rettangoli più piccoli o blocchi;
- All'interno di ciascun blocco le scatole sono disposte c on la stessa orientazione;
- L'orientazione comune degli items nei blocchi 1 e 5 è poi ortogonale a quella nei blocchi 2 e 4, mentre non viene fissata a priori quella delle scatole eventualmente collocate nel blocco 3.



Con cicli opportunamente strutturati, si fanno variare in modo indipendente tra loro le dimensioni dei blocchi 1 e 5. Le dimensioni degli altri blocchi restano fissate di conseguenza.

L'algoritmo controlla di volta in volta che il pattern elaborato non presenti

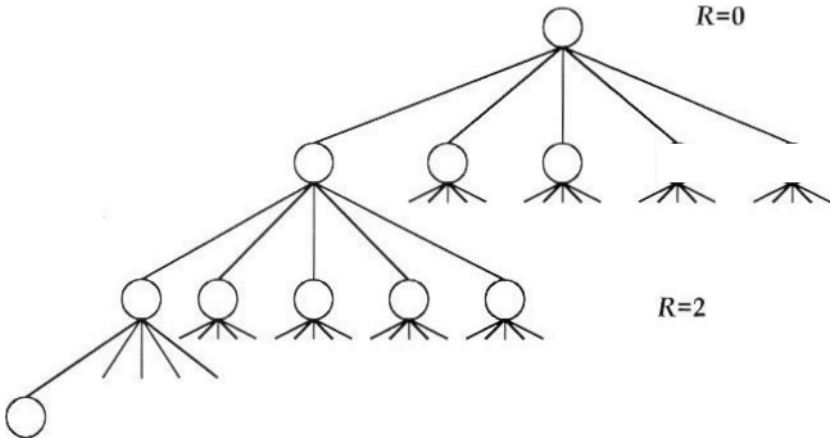
sovrapposizioni tra i blocchi, mentre un'ulteriore serie di istruzioni permette infine di evitare l'analisi di patterns simmetrici e quindi equivalenti dal punto di vista operativo. Per ogni pattern così elaborato si calcola il numero totale di scatole allocate, n : se esso coincide con un upper bound n^{ub} inizialmente fissato, la procedura si interrompe; in caso contrario, lo si confronta con la migliore soluzione n^* fino a quel punto ottenuta. Nel caso si registri un miglioramento, il pattern viene memorizzato ed il valore di n^* viene aggiornato.

2.2.2 Algoritmo di Morales e Morabito

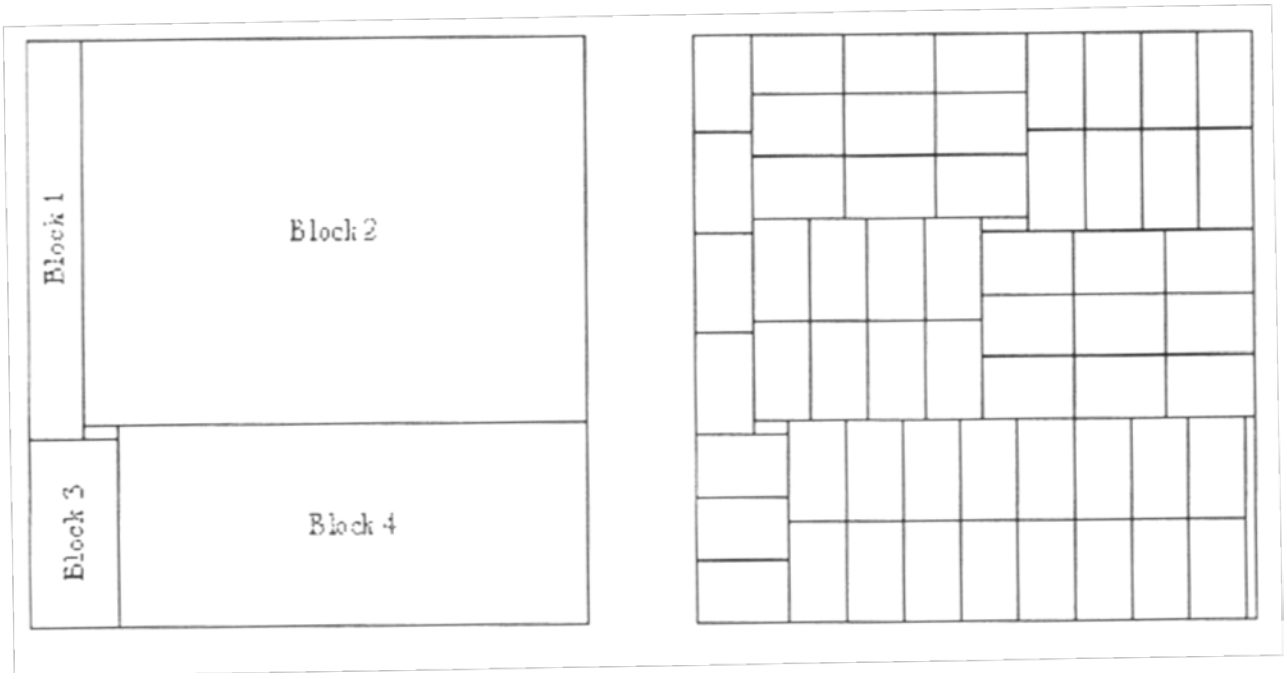
Il precedente algoritmo di B&D consiste in pratica nell'enumerazione implicita degli elementi dell'insieme (di solito grande) dei patterns non a ghigliottina del primo ordine applicati ad un rettangolo di dimensioni L,W. Morabito e Morales (M&M) propongono una efficace generalizzazione del metodo di B&D, che permette di esplorare lo spazio delle soluzioni in modo assai più aggressivo. In pratica:

- si suddivide il rettangolo (L,W) in cinque blocchi di dimensioni Li, bi , i = 1,..., 5, mediante 1'algoritnno di B&D
- lo stesso algoritmo viene poi applicato ricorsivamente a ciascuno dei blocchi, fino a quando questi non siano così piccoli da non poter essere ulteriormente decomposti.

Si costruisce così un albero binario di ricerca: ogni nodo dell'albero rappresenta un blocco, cui è possibile associare il numero di scatole in esso contenuto. Da ciascuno dei nodi partono al più cinque archi diversi, che corrispondono ai cinque biocchi ottenuti con un taglio non a ghigliottina del primo ordine. Il sezionamento di un blocco comporta quindi la discesa di un livello nell'albero, che viene visitato secondo la logica depth-first (di volta in volta si seziona per primo l'ultimo blocco generato). La profondità dell'albero di ricerca dipende dalla dimensione del pallet e delle scatole da caricare: per contenere in termini accettabili il volume dei calcoli e il conseguente inipiego di CPU, si stabilisce a priori il massimo numero di livelli (o grado di ricorsione), R.



I patterns più complessi sono in tal caso costituiti da 5R diversi sottoblocchi. il programma calcola il numero di scatole complessivamente assegnate a ciascuno dei patterns visitati, aggiornando il valore massimo ed interrompendo eventualmente la procedura quando esso coincida con quello di un upper bound fissato a priori.



Pattern ottenuto applicando l'algoritino con una sola chiamata ricorsiva.

CAPITOLO TERZO
Alpha

3.1 Alpha

Nell' industria del packaging è oggetto di uso intensivo il nastro trasportatore,che permette la movimentazione di oggetti lungo le varie stazioni.

Esso è formato generalmente da una serie di rulli posizionati uno accanto all' altro, o da un nastro,che genera la "pista" su cui viene fatto scorrere l' oggetto in questione.

Se noi ora quest oggetto lo associamo all' item, e le dimensioni della "pista" vengono identificate da un object avente lunghezza $L \to \infty$ e larghezza W finita, risulta evidente che, a parità di tempo e velocità di scorrimento del nastro, con un carico dalla disposizione ottimizzata, si è in grado di movimentare un maggior numero di itens.

Questa ottimizzazione permette sia di risparmiare tempo, sia di non dover

sostituire/modificare l' attrezzatura nel caso in cui si dovesse fronteggiare un improvviso aumento di produttività oraria.

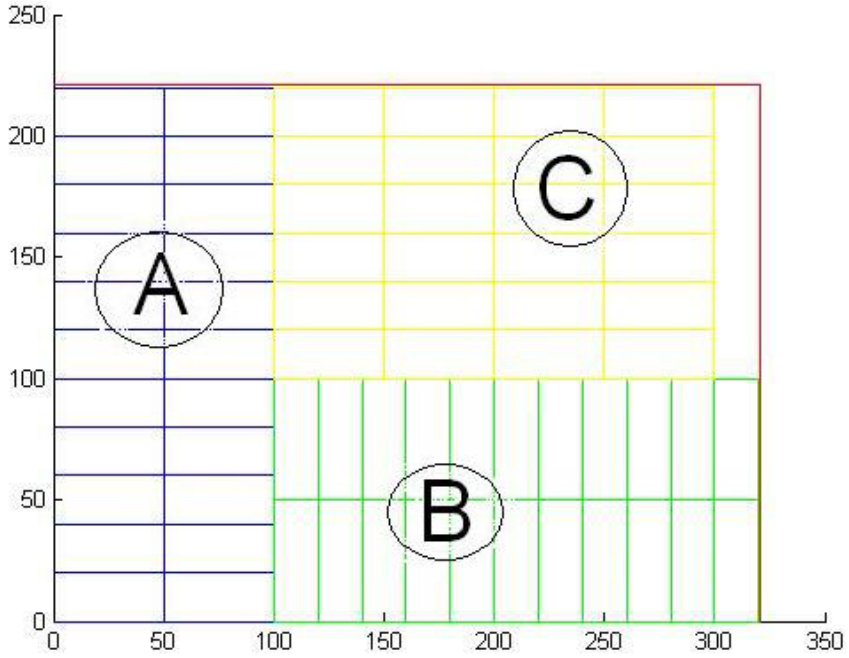
Attraverso l' implementazione dei principi di B&D, con il software MATLAB, è stato creato Alpha, un programma che permette di generare la miglior disposizione fisica di items rettangolari all' interno di un object di dimensioni maggiori e forma uguale.

3.2 Come opera Alpha

L' object (L,W) è diviso in al più 3 blocchi rettangolari A,B,C , all' interno dei quali si allocano gli items (l,w), con un'orientazione prefissata. Convenzionalmente, sia per object che per items, vale la condizione che la misura del lato (L o l) sia maggiore di quella dell' altro (W o w).

Fondamentalmente, l' algoritmo esamina in modo iterativo tutti i patterns a ghigliottina e non a ghigliottina, facendo variare le dimensioni dei blocchi A,B,C, restituendo alla fine il pattern che permette di allocare il maggior numero di items.

L' unico modo per poter fare ciò, è creare una serie di cicli for uno dentro l' altro,che terminano quando Alpha ha esaminato tutte le possibili combinazioni,le quali vengono generate secondo una precisa ratio:



- Items di tipo A: posizionati da sinistra verso destra,orizzontalmente (poggiano lungo il loro lato più lungo l);
- Items di tipo B: posizionati dal basso verso l' alto,verticalmente (pobbiano lungo il loro lato più corto w);
- Items di tipo C: posizionati orizzontalmente in modo da riempire tutto lo spazio rimanente.

Per n che va da 0 al massimo inseribile, inserisco di volta in volta, prima gli items di tipo C, i quali vanno ad occupare tutto lo spazio lasciato libero da A e B. i quali vengono posizionati in tutte le combinazioni possibili al fine di ricavare il pattern più efficiente.

3.3 Struttura di Alpha

1. Inserisco le dimensioni di object e items;
2. Visualizzo la figura dell' object attraverso il comando "line";
3. A parte, sono stati creati 3 sottoprogrammi: rect1, rect2, rect3, che permettono una visualizzazione chiara del diversi items, ognuno di un colore diverso dall' altro, a seconda del fatto che esso appatrenga ad A (blu), B (verde), C (giallo);
4. Attraverso l' uso dei cicli di for, il programma elabora e visualizza tutti i patterns, concludendo l' elaborazione con un elenco in cui si elenca il numero massimo di items contenuti da ogni singolo pattern;
5. Tramite l' elenco risalgo alla corrispondente figura e quindi alla corrispondente disposizione.

3.4 codici Matlab

3.4.1 Alpha

```
close all
clear

l=50;
w=20;

L=321;
W=221;

aa=0;

maxr1 = floor(L/l);

p = zeros(maxr1+1,1);

% line([0,0],[0,W]);
% line([0,L],[0,0]);
% line([0,L],[W,W]);
% line([L,L],[0,W]);

%rect(50,20,50,20)

for i=0:maxr1
    maxs2 = floor(W/l);
    t= zeros(maxs2+1,1);

    for q=0:maxs2

    figure
   % for j=0:
line([0,0],[0,W],'color','r');
line([0,L],[0,0],'color','r');
line([0,L],[W,W],'color','r');
line([L,L],[0,W],'color','r');

   s1 = floor(W/w);
   r1 = i;

   for a=0:s1-1
       for b=0:i-1
        rect(b*l,a*w,l,w)
       end
   end
                  s2 =q;
                  r2= floor((L-i*l)/w);
```

```
                        for c=0:r2-1
                            for d=0:q-1
                            rect2(i*l+c*w,d*l,w,l)
                            end
                        end

                              s3 = floor((W-(q*l))/w);
                              r3 = floor((L-(i*l))/l);

                                   for e=0:s3-1
                                        for f=0:r3-1


rect3(i*l+f*l,q*l+e*w,l,w)
                                        end
                                    end
   aa=aa+1
   p(aa) = r1*s1+r2*s2+r3*s3;

                   end
   % end
        end
p
```

3.4.2 Rect

3.4.2.1 Rect 1

```
function  rect( x,y,L,W)
%RECT Summary of this function goes here
%   Detailed explanation goes here


line([x,x],[y,y+W]);
line([x,x+L],[y,y]);
line([x,x+L],[y+W,y+W]);
line([x+L,x+L],[y,y+W]);


end
```

3.4.2.2 Rect 2

```
function  rect2( x,y,L,W)
%RECT Summary of this function goes here
%   Detailed explanation goes here


line([x,x],[y,y+W],'color','g');
line([x,x+L],[y,y],'color','g');
line([x,x+L],[y+W,y+W],'color','g');
line([x+L,x+L],[y,y+W],'color','g');


end
```

3.4.2.3 Rect 3

```
function  rect3( x,y,L,W)
%RECT Summary of this function goes here
%   Detailed explanation goes here


line([x,x],[y,y+W],'color','y');
line([x,x+L],[y,y],'color','y');
line([x,x+L],[y+W,y+W],'color','y');
line([x+L,x+L],[y,y+W],'color','y');

end
```

3.5 Ordine di posizionamento

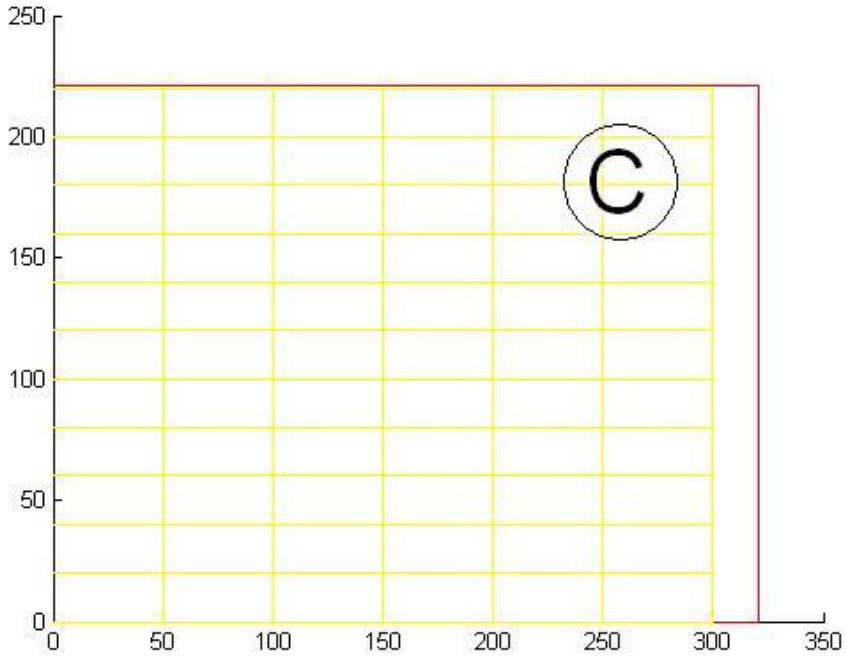
NOTA: n = numero max di items in verticale

m = numero max items in orizzontale

CASO 1

Numero file A = 0

Numero file B = 0

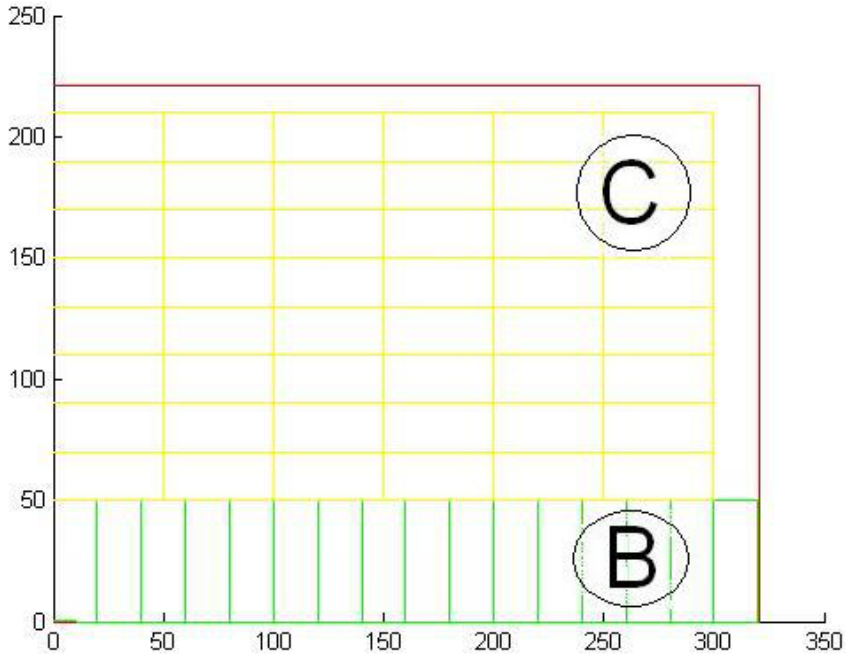


Nel primo caso ho solamente items di tipo C che riempiono tutta la figura.

CASO 2

Numero file A = 0

Numero file B = 1

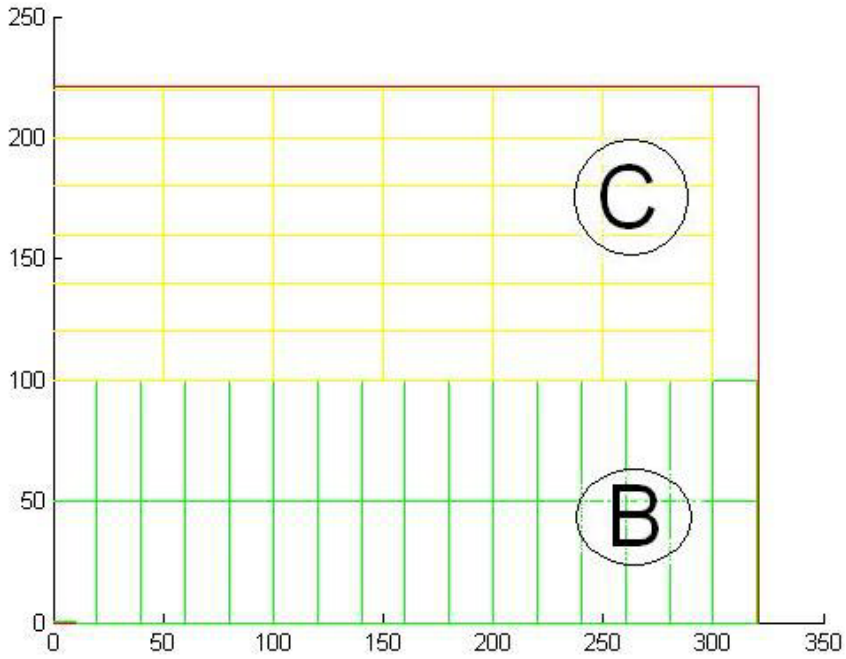


Nel secondo caso ho una fila di items B che occupano la parte bassa dell' object e gli items di C che riempiono tutto lo spazio rimanente.

CASO 3

Numero file A = 0

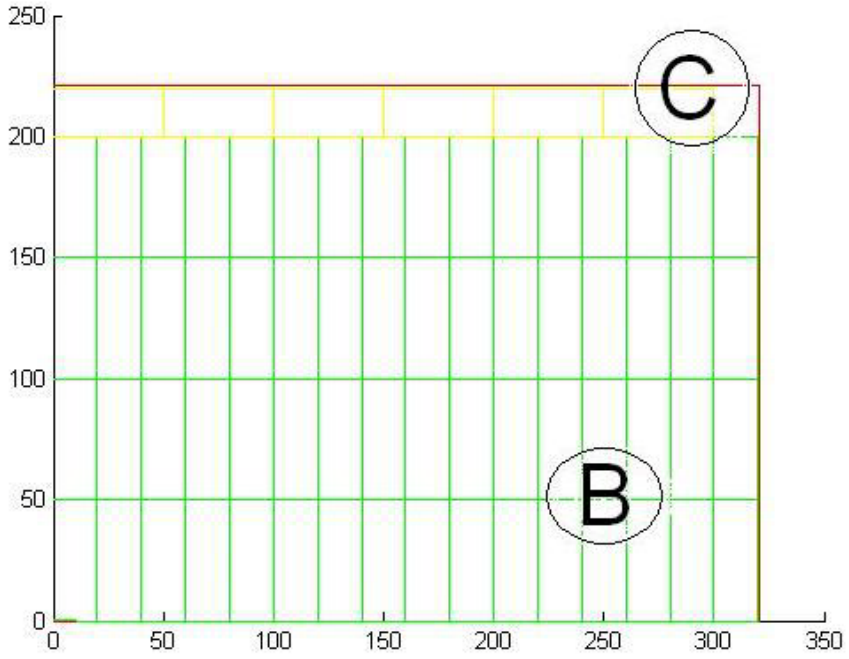
Numero file B = 2



CASO n

Numero file A = 0

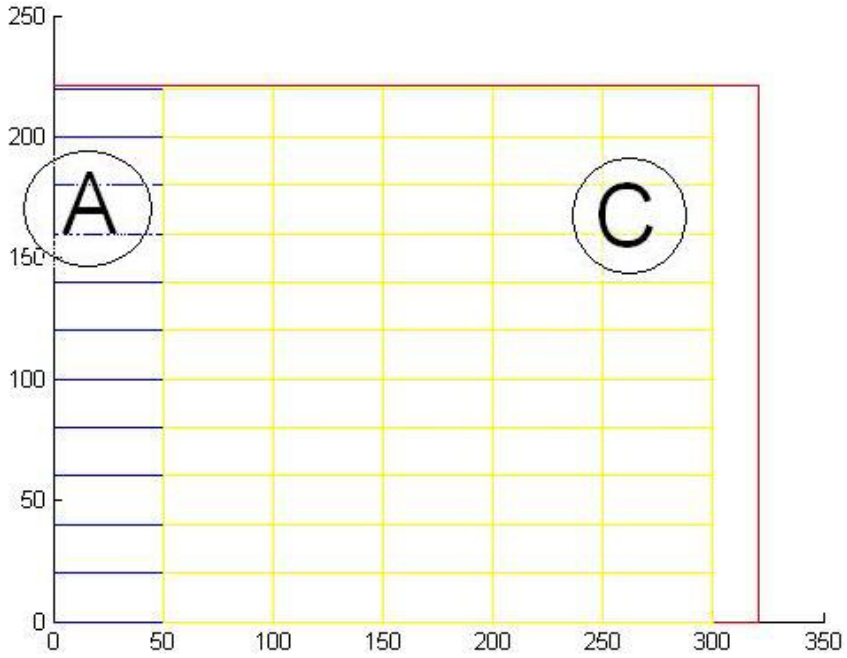
Numero file B = n



CASO n+1

Numero file A = 1

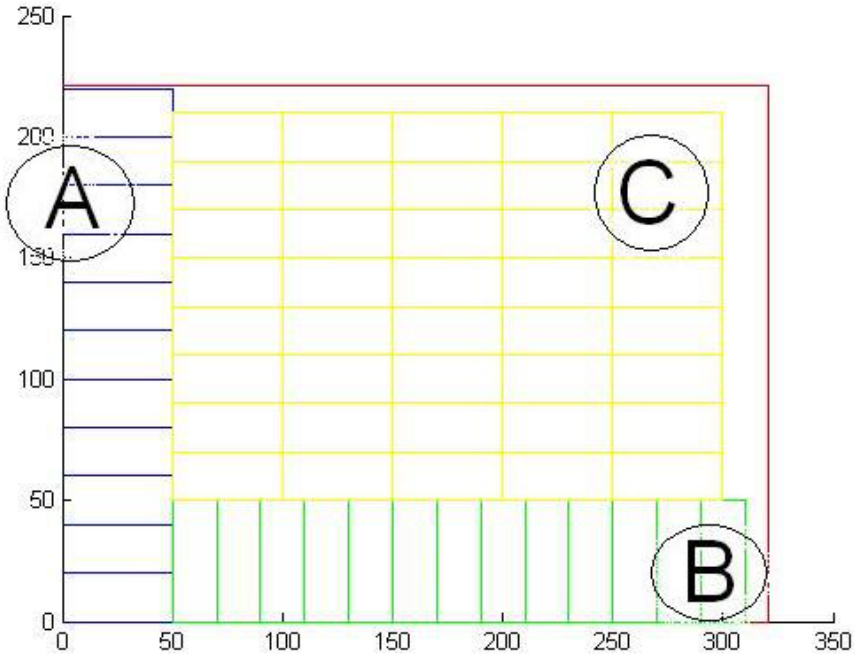
Numero file B = 0



CASO n+2

Numero file A = 1

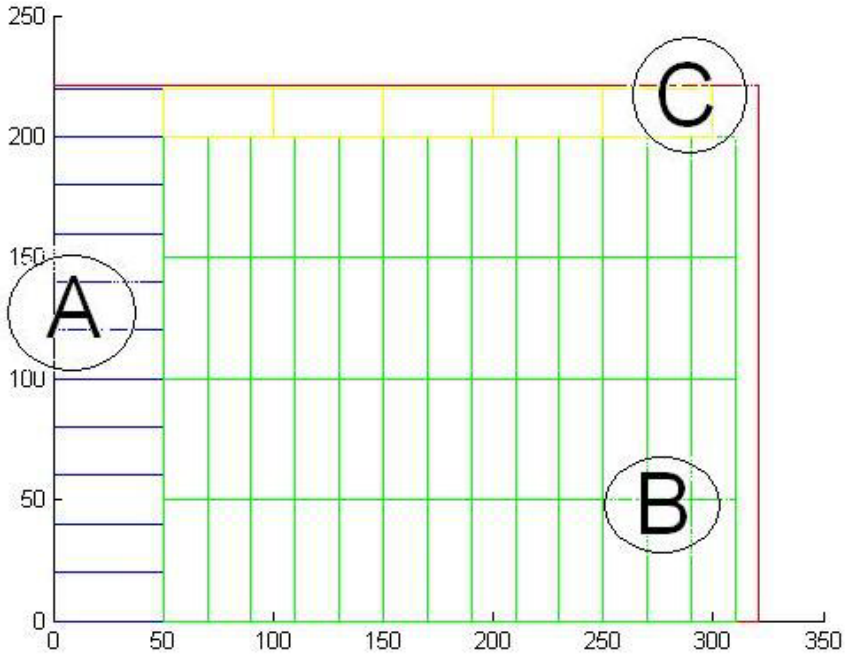
Numero file B = 1



CASO 2n

Numero file A = 1

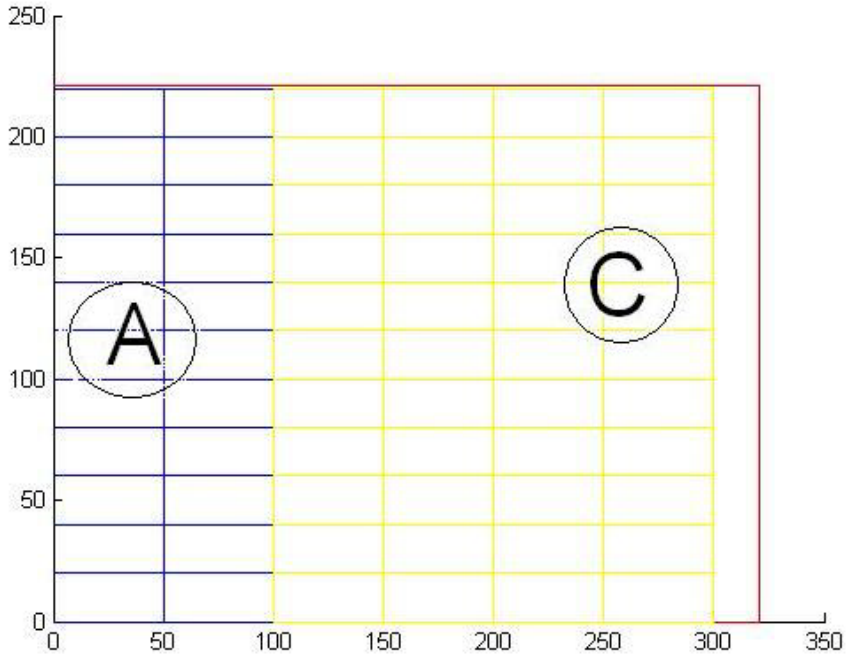
Numero file B = n



CASO 2n+1

Numero file A = 2

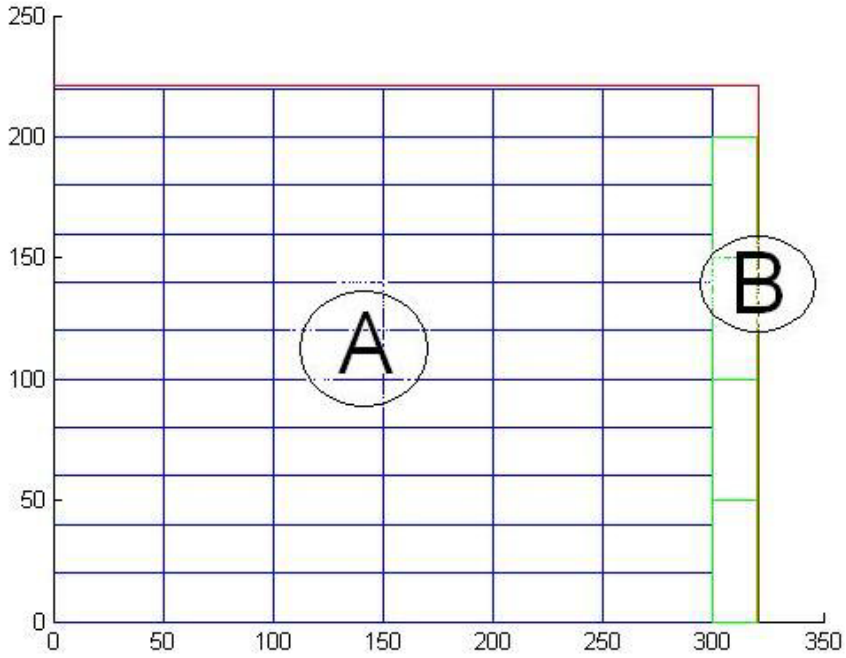
Numero file B = 0



CASO m*n

Numero file A = m-1

Numero file B = n



3.6 I predecessori: Alpha[1] e Alpha[2]

Alpha è stato generato per gradi, creando prima un programma che presentava una sola serie di blocchi A, a cui sono state aggiunte progressivamente le serie B e C.

Prima di Alpha, sono quindi stati creati 2 predecessori chiamati Alpha[1] e Alpha[2].

3.6.1 Codici Matlab

3.6.1.1 Alpha[1]

```
close all
clear

l=50;
w=20;

L=321;
W=221;

aa=0;


maxr1 = floor(L/l);

p = zeros(maxr1+1,1);

% line([0,0],[0,W]);
% line([0,L],[0,0]);
% line([0,L],[W,W]);
% line([L,L],[0,W]);

%rect(50,20,50,20)

for i=0:maxr1

    t= zeros(q+1,1);

    for q=0:floor(W/l)

    figure
   % for j=0:
line([0,0],[0,W],'color','r');
line([0,L],[0,0],'color','r');
```

```
line([0,L],[W,W],'color','r');
line([L,L],[0,W],'color','r');

   s1 = floor(W/w);
   r1 = i;                    %%%%%


   for a=0:s1-1
       for b=0:i-1
        rect(b*l,a*w,l,w)
       end
   end



   aa=aa+1
   p(aa) = r1*s1;

                   end
   % end
        end


p
```

3.6.1.2 Alpha²

```
close all
clear

l=50;
w=20;

L=321;
W=221;

aa=0;

maxr1 = floor(L/l);

p = zeros(maxr1+1,1);
```

```
% line([0,0],[0,W]);
% line([0,L],[0,0]);
% line([0,L],[W,W]);
% line([L,L],[0,W]);

%rect(50,20,50,20)

for i=0:maxr1
    maxs2 = floor(W/l);
    t= zeros(maxs2+1,1);

    for q=0:maxs2

    figure
   % for j=0:
line([0,0],[0,W],'color','r');
line([0,L],[0,0],'color','r');
line([0,L],[W,W],'color','r');
line([L,L],[0,W],'color','r');

   s1 = floor(W/w);
   r1 = i;

   for a=0:s1-1
       for b=0:i-1
        rect(b*l,a*w,l,w)
       end
   end
                  s2 =q;
                  r2= floor((L-i*l)/w);
                      for c=0:r2-1
                          for d=0:q-1
                          rect2(i*l+c*w,d*l,w,l)
                          end
                      end


   aa=aa+1
   p(aa) = r1*s1+r2*s2;

                  end
   % end
        end

p
```

3.6.2 Considerazioni sull' efficienza

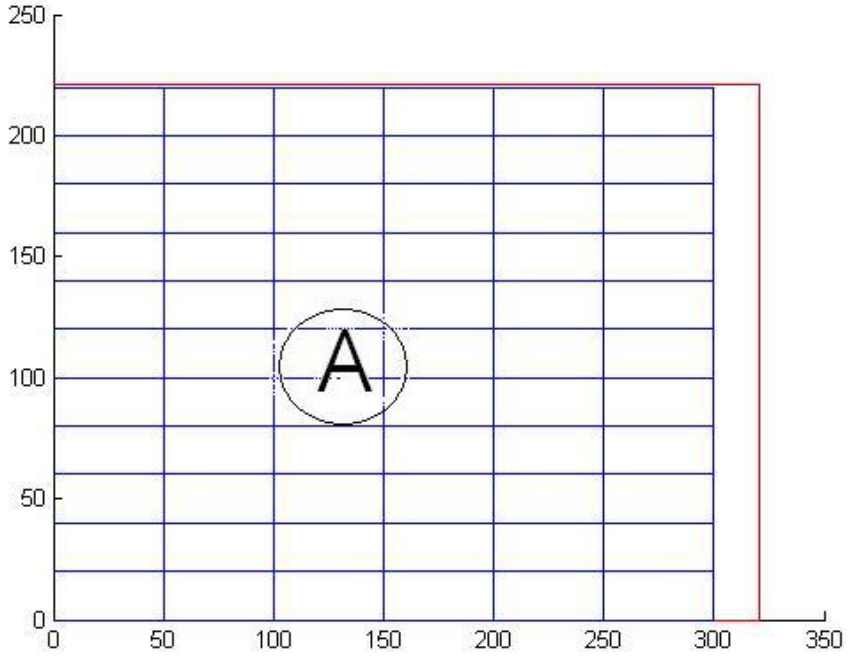
Mettendo a confronto $Alpha^1$ e $Alpha^2$, risulta evidente il progresso nel processo di riempimento:

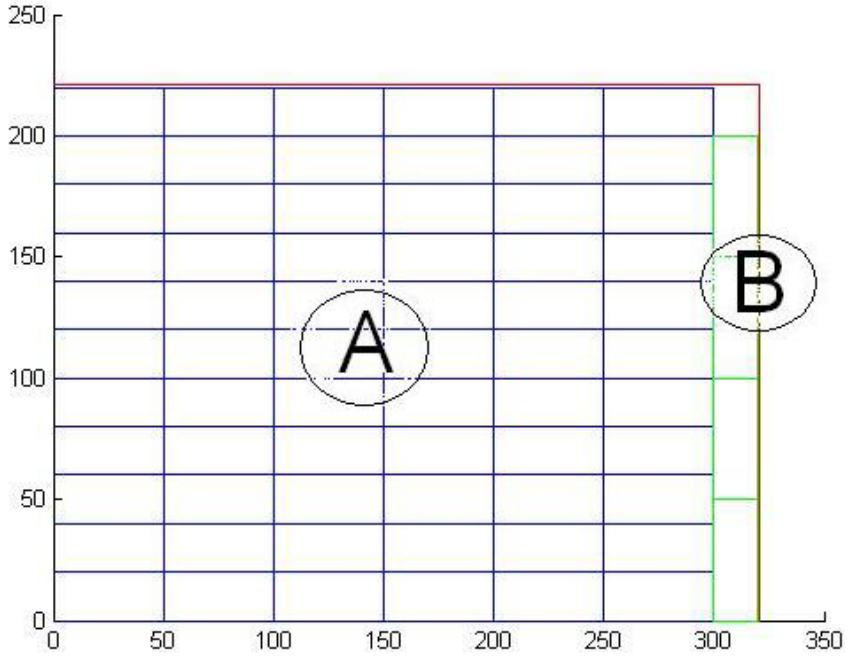
Essendo $Alpha^1$ un "sottoprogramma" di $Alpha^2$, si può affermare che la miglior soluzione ottima generabile da $Alpha^1$ sarà ≤ della miglior soluzione ottima generabile da $Alpha^2$.

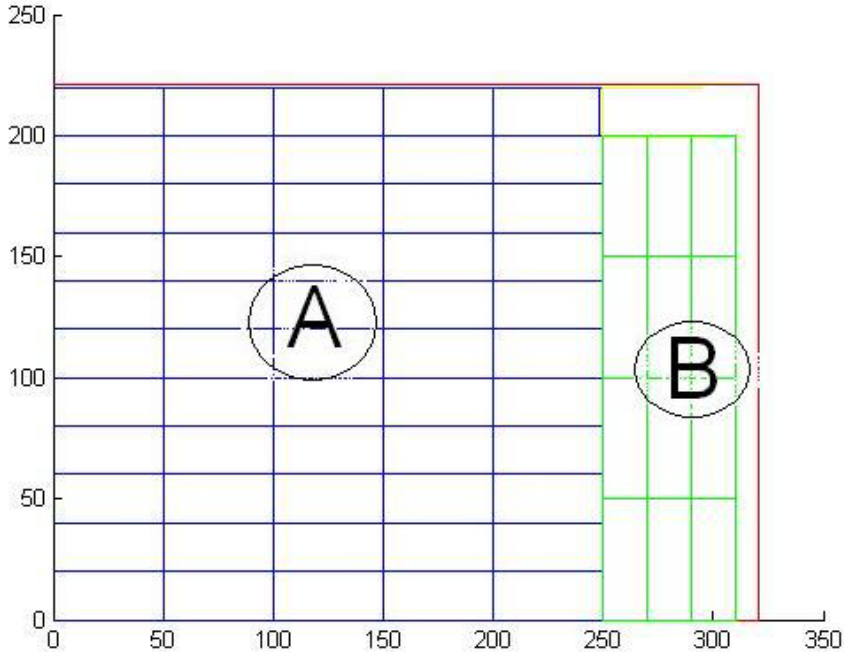
Allo stesso modo, confrontando $Alpha^2$ con Alpha

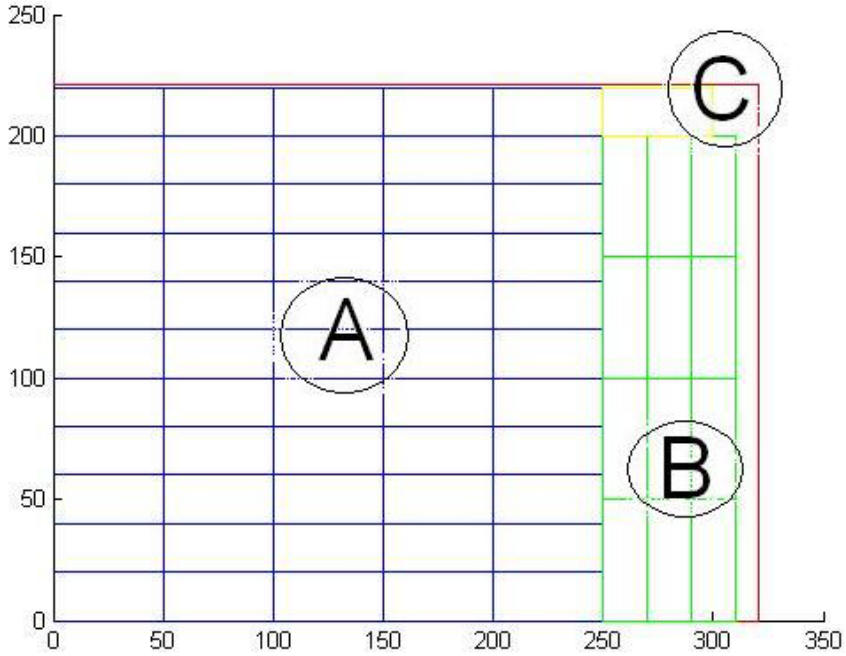
Anche in questo caso, la miglior soluzione ottima generabile da $Alpha^2$ sarà ≤ della miglior soluzione ottima generabile da Alpha.

Applicando la medesima ratio, si affermare che, inserendo un quarto blocco D che si interfacci con i preesistenti A,B,C, potrei ottenere una soluzione ottima che, nel suo caso migliore, sia ≥ della miglior soluzione ottima di Alpha.


250
200
150
100
50
0
0
50
100
150
200
250
300
350
A


250
200
150
100
50
0
0
50
100
150
200
250
300
350
A
B


250
200
150
100
50
0
0
50
100
150
200
250
300
350
A
B


250
200
150
100
50
0
0
50
100
150
200
250
300
350
C
A
B

CONCLUSIONI

In questa tesi, l' attenzione è stata focalizzata sul tema del nesting in 2 dimensioni, ovvero sulla ricerca di modelli efficaci ed efficienti per organizzare il posizionamento di oggetti.

Successivamente ad una panoramica generale che ha permesso di inquadrare l' argomento, sono stati presi in considerazione ali algoritmi elaborati da Bishoff & Dowsland e da Morabito & Morales.

Un' applicazione dei principi di Bishoff & Dowsland,ha permesso la realizzazione di Alpha, programma in linguaggio MATLAB, che permette di ottimizzare la disposizione di items rettangolari all' interno di object della stessa forma.

In prospettiva , grazie alla sua versatilità, Alpha potrebbe adattarsi ad ogni tipo di problema con items e objects rettangolari, divenendo cosi' un prezioso strumento nelle delicate fasi di packaging,cutting & loading.

BIBLIOGRAFIA

1. E. Bishoff, W. Dowsland, (1982), " An application of the micro to product design and distribution", Journal of Operational Research Society,vol. 33, pp. 271-280

2. R. Morabito, S. Morales, (1998), "A simple and effective recursive procedure for the manufacturer's pallet loading problem", Journal of Operational Research Society, vol. 49, pp. 819-828

3. K.A. Dowsland, (1990), "Efficient automated pallet loading", European Journal of Operational Research, vol. 44, pp.232-246

4. G. Scheitauer, (1997), "Equivalence and Dominance for problems of optimal packing of rectangles", Ricerca Operativa, vol. 83, pp. 1-33

5. R.W. Haessler, (1971), "A heuristic programming solution to a non linear cutting stock problcm", Management Science, 17B, pp. 793-802

6. E.G. Coffinan, P.W. Shor, (1993),"Packing in two dimensions: Asymptotic average-case analysis of algorithins", Algoritmica,vol. 9 pp. 253-277

7. J. S. Ferreira, M. A. Neves, P. Fonseca e Castro, (1990), "A two-phase roll cutting problem" European Journal of Operational Research, vol.44, pp. 185-196

8. A.I. Hinxman, (1980), "The trim loss and assortment problems: A survey" European Journal of Operational Research, vol. 5, pp. 8-18

9. E.G. Coffman, P.W. Shor, (1990), "Average-case analysis of cutting and packing in two dimension", European Journal of Operational Research, vol.44, pp. 134-144

10. C.H. Dagli, (1990), "Knowledge-based systems for cutting stock problems", European Journal of Operational Research, vol.44, pp. 160-166

11. H. Dickoff, (1990), "A typology of cuttin and packing problems", European Journal of Operational Research, vol. 44, pp. 145-159

12. H. Dychoff, G. Scheitauer, J. Terno, (1997), "Cutting and packing", in F. Maffioli, M. Dell'Amico and S. Martello, editors, Annotated bibliographies in Combinatorial Optimization, pp. 393-413, John Wiley & Sons

13. A.Colorni, (1988), "Elementi di Ricerca Operativa", Zanichelli, cap. 10, pp. 211-214

14. R. Morabito, M.N. Arenales, (1996), "Staged and unconstrained twodimensional guillotine cutting problems: An AND/OR-graph approach", European Journal of Operational Research, vol. 94, pp. 548-560

15. C. Pisinger, (1998), "A three-search heuristic for the container loading problem", Ricerca Operativa, vol. 81, pp. 31-47

RINGRAZIAMENTI

Alla fine di questo percorso, desidero ringraziare tutte le persone che mi hanno accompagnato lungo di esso, e quelle che ne hanno permesso il compimento.

Ringrazio quindi innanzitutto il professor Giovanni Boschetti per la sua disponibilità e gentilezza.

Ringrazio i miei compagni di corso, tra tutti, in particolar modo Niccolò ed Alessandro, con i quali ho condiviso innumerevoli ore di studio, ma anche esperienze e momenti da ricordare.

Ringrazio l' Ing D' Amico per la sua gentile, competente e puntuale assistenza.

Ringrazio gli amici e i parenti che in questi anni hanno creduto in me: zio Piero e zia Cristina, Alessandro, Demis, Denis, Eliana, Emanuel, Michela, Vanessa e Vittorio (rigorosamente in ordine alfabetico).

Ringrazio mio papà, che non mi ha mai fatto mancare nulla.

Ringrazio Luca e Francesca, per avermi sempre incitato e sostenuto.

Infine, ma non per caso, un pensiero alla persona che più mi ha amato e che teneva a questo traguardo forse più di quanto ci tenga io: mia mamma. A lei dedico questa tesi.