ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA GESTIONALE

Tesi di Laurea Magistrale in RESOURCES OPTIMIZATION

STRIP PACKING BIDIMENSIONALE CON ITEMS DEFORMABILI CON APPLICAZIONE ALLO SCHEDULING DI TASK IN AMBITO HPC

CANDIDATO

Jacopo Ferdinando Ferreri

RELATORE

Prof. Michele Monaci

CORRELATORE

Prof. Enrico Malaguti

Anno Accademico 2020-2021
Sessione III

Ringraziamenti

Un sincero ringraziamento al mio relatore Prof. Michele Monaci ed al mio correlatore correlatore Prof. Enrico Malaguti. Grazie per l'opportunità che mi è stata data con questa tesi, per l'assiduo supporto e per le sfidanti richieste: hanno contribuito fortemente alla mia crescita, che oltre che professionale sento molto personale.

Grazie alla mia famiglia per il supporto continuo e fondamentale durante tutti i miei studi, mi avete trasmesso l'importanza della cultura, inutile dire che dedico a voi questo traguardo. Grazie in particolare a mia madre, mi hai sempre chiesto il massimo, e anche se a volte non lo capivo, questo ha fatto la differenza. Io so quanto questo traguardo sia anche merito tuo.

Grazie a Laura, sei stata allo stesso tempo testimone e attrice di questo percorso dall'inizio alla fine, con te ho condiviso tutto, dai momenti più belli a quelli di maggior difficoltà, mi hai sempre sostenuto e mi hai aiutato a rimanere lucido ai bivi delle scelte più importanti.

Grazie ai miei amici per essere stati la fonte di leggerezza di cui avevo bisogno, il vostro contributo è stato fondamentale nei momenti in cui mi prendevo troppo sul serio.

Infine, durante questo percorso ho condiviso parte del cammino con molte persone, alcune delle quali ho imparato a chiamare amici. Sarebbe difficile citare tutti, quindi dico solo grazie a chiunque abbia camminato con me anche solo per una piccola parte del sentiero che questa tesi porta a conclusione.

Indice

Sommario

È tipico di diversi settori industriali, il dover affrontare un problema di minimizzazione dello spazio utilizzato, riempiendo un contenitore con oggetti di dimensioni finite, o il dover minimizzare gli scarti nel ricavare più componenti dal taglio di un singolo elemento. La risoluzione di questi problemi può essere affrontata tramite le tecniche tipiche della programmazione lineare intera, e in particolare un problema molto sfruttato in quest'ambito è lo *strip packing* bidimensionale. In questa variante del *bin packing*, si ha un insieme di elementi che devono essere posizionati all'interno di un contenitore di larghezza determinata e altezza infinita, solitamente minimizzando l'area o l'altezza totale utilizzata. Alcuni esempi di applicazioni in cui questo problema può essere sfruttato possono essere: il taglio di oggetti rettangolari da bobine di materiale tessile o metallico, lo stoccaggio della merce in un magazzino, e altre applicazioni in cui tramite la combinazione anche con altri problemi si possono ottimizzare processi manufatturieri, logistici o delle telecomunicazioni. Lo scopo di questa tesi è quello di studiare una variante del problema di *strip packing* bidimensionale, che permetta di minimizzare l'altezza totale usata attraverso la deformazione degli oggetti da collocare nel contenitore, ovvero lo *strip packing* bidimensionale con oggetti deformabili. In particolare, saranno presentati modelli esatti sviluppati come adattamento di modelli per lo *strip packing* classico, e algoritmi euristici sviluppati ad hoc. Sebbene la ricerca su questo tipo di problema non sia particolarmente ampia, l'utilità pratica proveniente della risoluzione del problema in esame può manifestarsi, ad esempio, in tutte le situazioni in cui è possibile modellare il contenitore come un insieme di risorse disponibili a suddividersi un certo carico di lavoro, come nel caso del *High Performance Computing*, in cui un gran numero di processori si suddividono la

risoluzione di più *task*. Modellando un simile problema come un problema di ottimizzazione, attraverso le dovute semplificazioni, è possibile pensare alla larghezza della *strip* come il numero di processori a disposizione, mentre gli oggetti rappresentano i *task* da risolvere; l'obiettivo sarà quindi determinare posizionamento e forma degli elementi all'interno del contenitore, con minimizzando il tempo totale utilizzato, ovvero l'altezza della *strip*.

Nella tesi sono introdotti tutti i concetti necessari a presentare il problema studiato, i metodi risolutivi dedicati, ed i risultati dei test computazionali condotti seguendo la struttura spiegata di seguito. Nel Capitolo 1 viene esposta una breve revisione bibliografica dei problemi di *cutting & packing*, con lo scopo di evidenziarne l'evoluzione storica e le applicazioni tipiche, andando anche a definire in maniera chiara i problemi di *bin packing* e *strip packing* bidimensionale. Nel Capitolo 2 sono presentati tre modelli esatti della letteratura, mirati a risolvere il problema dello *strip packing* bidimensionale classico e sui quali ci si è basati per lo sviluppo dei modelli esatti necessari a risolvere la variante con oggetti deformabili. La possibilità di modificare la forma degli oggetti è introdotta nel Capitolo 3, dove ne viene spiegata l'utilità pratica e la relativa definizione matematica. Procedendo, nel Capitolo 4 si introducono gli adattamenti operati ai tre modelli esatti già introdotti, e sei algoritmi euristici, cinque dei quali basati sull'uso di un *solver* commerciale. Infine, nel Capitolo 5 sono riportate le caratteristiche delle istanze usate per i test, che derivano dalle istanze delle classi NGCUT e C già presenti in letteratura, e sono presentati i risultati dei test computazionali svolti. I risultati, analizzati secondo più prospettive, portano a determinare quali modelli ed algoritmi si siano rivelati più adatti, ed in quali situazioni.

Capitolo 1

I problemi di *cutting & packing* bidimensionale

Nel tempo i problemi di *packing* bidimensionale sono stati studiati per applicazioni nei più svariati settori attraverso più varianti. A livello pratico, posizionare un elemento in un contenitore, con una rappresentazione bidimensionale, equivale al tagliare tale elemento da un'unità di materiale grezzo, è evidente quindi come i problemi di *cutting* e *packing* siano fortemente correlati, e come in entrambe i casi sia spesso possibile usare le stesse metodologie risolutive.

Un primo approccio al *packing* bidimensionale lo si trova in una pubblicazione del 1965 [Gilmore and Gomory, 1965], in cui gli autori, analizzando alcune varianti del problema di *cutting stock*, hanno presentato il problema del "taglio a ghigliottina in due stadi". Questo particolare problema di taglio non è altro che una prima introduzione del "*packing* bidimensionale a livelli", una variante del problema generale che sarà analizzata nel seguito. Nonostante siano passati anni dalla sua prima presentazione, il taglio a ghigliottina in due stadi resta largamente utilizzato in vari settori industriali, esso infatti permette di suddividere una superficie rettangolare in un certo numero di rettangoli, ricavando ogni elemento con massimo due tagli in linea retta, operati da una estemità all'altra della superficie di partenza: il primo divide la superficie di partenza in strisce lungo la sua lunghezza, il secondo taglia le strisce così ricavate lungo la loro larghezza. Il problema viene introdotto come problema di taglio (*cutting*), ma può essere applicato anche

a situazioni analoghe a quelle di *packing* semplicemente immaginando la superficie tagliata come il contenitore in cui devono essere inseriti gli oggetti, e quest'ultimi che corrispondono esattamente agli elementi tagliati su di essa.

È evidente, quindi, come questo tipo di taglio si presti alle più svariate situazioni applicative, in particolare per le applicazioni industriali in cui da un componente solido, approssimabile a due dimensioni, debbano essere ricavati componenti più piccoli attraverso tagli in linea retta, da lato a lato, minimizzando lo spreco di materiale. Sono esempi tipici di questo caso la produzione del vetro, il taglio delle lamine metalliche, plastiche o delle bobine di tessuto. In tutti questi casi è spesso possibile sfruttare i modelli che risolvono la variante a livelli del problema di taglio, proprio perché le macchine utilizzate possono dover tagliare velocemente in linea retta dei componenti dalla forma semplice, operando tagli da parte a parte e delegando ulteriori rifiniture ai successivi stadi della lavorazione.

Gilmore e Gomory operano una ulteriore distinzione tra taglio a ghigliottina "esatto" e "non esatto". Nel primo caso gli elementi tagliati in due stadi dallo stesso livello hanno tutti la stessa altezza, mentre nel secondo è permesso un terzo taglio (*trimming*) con il quale ogni elemento viene portato all'altezza desiderata.

Passando ad esaminare il problema generale del *packing* bidimensionale, una possibile definizione è la seguente: dati un contenitore ed un insieme di elementi di forma rettangolare, un *packing* ammissibile consiste nel posizionamento degli elementi all'interno del contenitore in maniera tale che essi vi siano completamente contenuti all'interno senza sovrapposizioni reciproche [Iori et al., 2021]. Inoltre, nel seguito si farà riferimento esclusivamente al caso del posizionamento degli elementi in maniera ortogonale, per cui è necessario che i lati degli elementi posizionati siano paralleli o alla base o ai lati del contenitore. Svincolandosi quindi dalla richiesta esplicita di livelli, e passando al caso generale, le applicazioni presenti in letteratura sono varie e tutte sfruttano i modelli tipici dei problemi di *cutting & packing* classici operando alcune modifche per adattarsi al caso specfico. Tra gli usi industriali più interessanti troviamo lo stoccaggio della merce nei magazzini o la collocazione all'interno dei container per il trasporto, caso che si può combinare con il problema di *routing* e per cui si può parlare di problema di *routing* con vincoli

di carico [Iori and Martello, 2010]. Ancora in combinazione con un altro problema, si riporta [Melega et al., 2018], una revisione della letteratura riguardante la combinazione del *lot-sizing* e del *cutting stock*, utile in tutte le applicazioni in cui lotti di grandi dimensioni devono essere suddivisi in oggetti di dimensioni inferiori, per esempio per far fronte alle richieste di un cliente. Considerare i problemi in maniera integrata permette di ottenere risultati migliori rispetto alla risoluzione separata, per questo l'approccio integrato ha una grossa rilevanza.

Altre applicazioni richiedono invece la sola applicazione delle tecniche tipiche dei problemi di *cutting* & *packing* con particolari adattamenti a seconda della situazione in cui si sta operando. In [Lodi et al., 2011] si affronta il problema della trasmissione entro un tempo limite dei pacchetti dati tramite una banda di frequenza frazionabile, sfruttando il *bin packing*. Rimanendo nell'ambito delle telecomunicazioni, in [Martello, 2014] si sfrutta sempre il *bin packing* per ottimizzare l'invio di pacchetti dati all'interno di "*frames*" di dimensioni variabili, ognuna rappresentante diverse combinazioni di frequenza e tempo di trasmissione. In [Silva et al., 2016] è mostrato l'uso che nel tempo la ricerca ha fatto di modelli ed algortimi tipici dei problemi di *cutting* & *packing* per risolvere il problema del *manufacturer's pallet loading problem* (MPLP), il quale è un problema in tre dimensioni nel quale si deve caricare più oggetti possibile di dimensioni identiche su un pallet, e che può essere portato ad un problema di *packing* in due dimensioni sfruttando le altezza identiche tra gli oggetti. In generale poi, si può dover affrontare il problema del taglio/posizionamento di oggetti durante l'impaginazione dei contenuti in riviste sia cartacee che online, in cui i contenuti pubblicitari e articoli devono essere inseriti in pagine prescritte, oppure minimizzando il numero di pagine usate. In questo campo [Strecker and Hennig, 2009] hanno sfruttato il *bin packing* per generare layout di pagina automaticamente in pagine di giornale, unendo anche l'uso di una funzione regolante il senso estetico dell'output, necessario in applicazioni di questo tipo. Infine, si riporta un esempio di applicazione delle tecniche di *packing* per organizzare il lavoro in un sito di assemblaggio di un cantiere navale [Kwon and Lee, 2015], in cui si punta a minimizzare il tempo di assemblaggio organizzando le zone di lavoro nelle varie fasi della lavorazione.

1.1 Definizione dei problemi di *packing*

Una volta definiti in maniera generale i problemi di *cutting* & *packing*, e dimostrata la loro utilità pratica, se ne può introdurre la notazione matematica necessaria in seguito a modellarli come problemi di ottimizzazione. Si noti che nelle versioni classiche dei problemi di *packing*, presentate nel seguito, si considera fisso l'orientamento degli elementi da posizionare, che non possono quindi essere ruotati di 90°. Questa possibilità, pur mantenendo l'ortogonalità dei tagli/posizionamenti, richiede modifiche ulteriori ai modelli matematici utilizzati per modellare il problema base.

Una prima distinzione in ambito di problemi di *packing* in due dimensioni può essere fatta dividendo tra "*bin packing* bidimensionale" (2BP) e "*strip packing*" bidomensionale (2SP) [Iori et al., 2021].

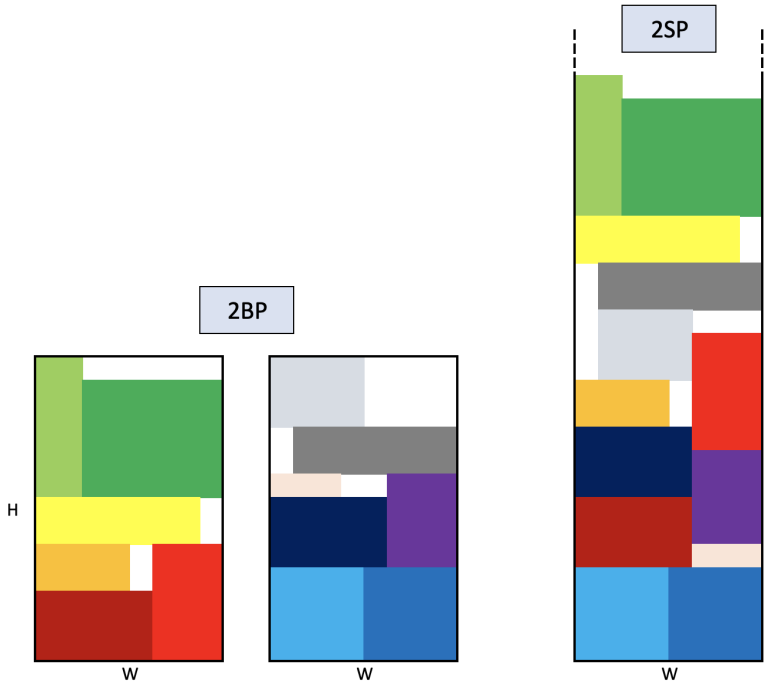

Figura 1.1. A sinistra un esempio di *bin packing* bidimensionale (2BP), a destra un esempio di *strip packing* bidimensionale (2SP) con gli stessi oggetti.

2BP. Nel 2BP è dato un insieme di n elementi rettangolari $j \in J = (1, \ldots, n)$, ciascuno di larghezza w_j e altezza h_j, ed un numero illimitato di contenitori (*bins*) di larghezza W e altezza H. L'obiettivo è collocare tutti gli elementi minimizzando il numero di contenitori utilizzati. Ad ulteriore testimonianza dei punti in comune tra problemi di *packing* e *cutting*, si noti come in caso siano presenti più oggetti di forma uguale (e eventualmente più tipi di oggetti), il problema del *bin packing* diventa un problema di *cutting stock*. In quest'ultimo, se si considera ogni elemento come oggetto "a sé", indipendentemente da quanti altri elementi simili siano presenti, il problema diventa ovviamente un 2BP. Ovviamente, affinché gli elementi siano collocabili all'interno del contenitore, è necessario che gli oggetti abbiano dimensioni positive e compatibili con quelle del contenitore:

$$0 < w_j \leq W, \;\; 0 < h_j \leq H \qquad\qquad \forall j \in J \tag{1.1}$$

2SP. Il 2SP è una importante variante del problema di 2BP, nel quale non si hanno infiniti contenitori, ma un solo contenitore di altezza infinita (*strip*), in cui è richiesto di collocare tutti gli elementi in modo da minimizzare l'altezza totale utilizzata. Anche in questo caso, affinché il *packing* sia ammissibile è necessario che gli oggetti abbiano dimensioni positive e compatibili con quelle della *strip*:

$$0 < w_j \leq W, \;\; h_j > 0 \qquad\qquad \forall j \in J \tag{1.2}$$

In termini di complessità computazionale, si può dimostrare che i due problemi presentati sono fortemente *NP*-hard. Il caso bidimensionale del *bin packing*, infatti, non è altro che una generalizzazione a due dimensioni del *bin packing* monodimensionale (1BP), la cui intrattabilità è ampiamente riconosciuta [Garey and Johnson, 1981]. Partendo da questo presupposto, la dimostrazione della complessità del 2BP e del 2SP può essere fatta immaginando di prendere come riferimento un'istanza del 1BP di n elementi $i \in I$ e per ogni elemento definirne uno in due dimensioni di larghezza w_i e altezza $h_i = 1$, ne consegue che una soluzione del 2SP con contenitore di larghezza W, risolve anche l'istanza del 1BP, la quale è risolta dalla soluzione del 2BP con contenitori di larghezza W e altezza $H = 1$. Dato quindi che risolvere le istanze bidimensionali derivate in

tempo polinomiale attraverso un algoritmo polinomiale significherebbe risolvere in tempo polinomiale il *bin packing* monodimensionale, si può dire che il *bin packing* e lo *strip packing* in due dimensioni sono *NP-hard* [Iori et al., 2021].

Quando si possono sfruttare tagli a ghigliotina, come visto nel paragrafo precedente, è possibile applicare una versione versione leggermente modificata del 2SP: lo "*strip packing* bidimensionale a livelli" (2LSP). Il 2LSP consiste nel posizionare gli elementi all'interno del contenitore partendo dalla base e da sinistra verso destra, o viceversa. Quando un elemento j non può essere posizionato sul livello L (*shelf*) perché lo spazio a disposizione è insufficiente ($w_j > W - \sum w_i, i < j, i \in L$), viene posizionato su un nuovo livello L+1, che ha la propria base sulla linea orizzontale coincidente con l'altezza dell'elemento di altezza maggiore del livello appena chiuso [Lodi et al., 2004]. Il vincolo di posizionamento a livelli degli oggetti può essere sfruttato anche nel problema di *bin packing*, dove ogni contenitore viene diviso in livelli via via che sono posizionati gli oggetti, e eventualmente si apre un livello in un nuovo contenitore quando è superata l'altezza H. In questo caso si parla di *bin packing* a livelli in due dimensioni (2LBP).

Come nel caso del 2SP e 2BP, sia il 2LSP che il 2LBP sono generalizzazioni del 1BP, quindi, seguendo il procedimento della dimostrazione per i problemi base, si può dire che anche la variante a livelli del problema di *packing* sia fortemente *NP*-hard.

Capitolo 2

Modelli PLI per lo *strip packing*

I modelli esatti ed alcuni degli euristici utlizzati in questa tesi per affrontare la variante del 2SP con elementi deformabili, si basano su modelli nati per risolvere la versione classica del problema, e che quindi non si può prescindere dal descrivere. Nel seguito saranno quindi presentati i modelli di programmazione lineare intera (PLI) utilizzabili per afforntare i problemi visti in precedenza. Sebbene la maggior parte di essi derivi la formulazione per lo *strip packing* bidimensionale da quella del *bin packing*, ai fini di questa tesi si presentano solo i modelli per il caso del singolo contenitore di altezza infinita. Ai tre modelli illustrati in questo capitolo sarà fatto riferimento attraverso i nomi "Modello A", "Modello B" e "Modello C", dove il primo affronta il problema sfruttando il vincolo del posizionamento a livelli, e gli altri due affrontano direttamente il problema del 2SP.

2.1 Modello A

Per quanto riguarda il 2LSP il lavoro si è basato sui modelli presentati in [Lodi et al., 2004] e [Lodi and Monaci, 2003], entrambi con un numero polinomiale di variabili e di vincoli. Il primo si basa specificamente sul problema di 2LSP, e per essere presentato attraverso una formulazione compatta richede di osservare che, per ogni soluzione ottimale a livelli, ne esiste una equivalente in cui:

1. l'elemento più a sinistra di ogni livello è il più alto di tale livello;
2. il primo livello in basso nella *strip* è il più alto tra tutti i livelli;

3. gli elementi sono ordinati per altezze non-crescenti $h_1 \geq h_2 \geq \ldots \geq h_n$.

Il modello si basa su due insiemi di variabili che gestiscono gli n livelli potenziali, ognuno inizializzabile da un elemento i e quindi avente altezza h_i. Le variabili decisionali sono:

$$y_i = \begin{cases} 1 & \text{se elem. } i \text{ inizializza il livello } i \\ 0 & \text{altrimenti} \end{cases} \qquad (i = 1, \ldots, n) \tag{2.1}$$

$$x_{ij} = \begin{cases} 1 & \text{se elem. } j \text{ è nel livello } i \\ 0 & \text{altrimenti} \end{cases} \qquad (i = 1, \ldots, n-1; j > i) \tag{2.2}$$

Il modello ILP è il seguente:

$$\min \sum_{i=1}^{n} h_i y_i \tag{2.3}$$

$$\sum_{i=1}^{j-1} x_{ij} + y_i = 1 \qquad (j = 1, \ldots, n) \tag{2.4}$$

$$\sum_{j=i+1}^{n} w_j x_{ij} \leq (W - w_i) y_i \qquad (i = 1, \ldots, n-1) \tag{2.5}$$

$$y_i \in \{0,1\} \qquad (i = 1, \ldots, n) \tag{2.6}$$

$$x_{ij} \in \{0,1\} \qquad (i = 1, \ldots, n-1; j > i) \tag{2.7}$$

Il secondo modello è simile, ma presentato come modello PLI per risolvere un problema di *knapsack bidimensionale in due stadi* (2TDK). Il *knapsack* differisce dal problema di *packing* in quanto ad ogni elemento j è associato un profitto p_j, e l'obiettivo finale è scegliere quali elementi collocare nel contenitore (di dimensioni finite) massimizzando il profitto totale. Il modello descritto da Lodi e Monaci, pur mirando a risolvere un problema diverso rispetto a quello di nostro interesse, utilizza una notazione che può essere implementata nel modello precedente per risolvere un problema di *strip packing*. Le variabili decisionali sono infatti solo di

un tipo:

$$x_{jk} = \begin{cases} 1 & \text{se l'elem. } j \text{ è nel livello } k \\ 0 & \text{altrimenti} \end{cases} \qquad (k = 1, \ldots, n; j = k, \ldots, n) \tag{2.8}$$

In questa notazione quindi, un elemento inizializza il livello di appartenenza k se $x_{kk} = 1$. Si noti che anche in questo caso gli elementi sono da considerare come posizionati secondo altezze non-crescenti. Il modello PLI è il seguente:

$$\max \sum_{i=1}^{n} p_j \sum_{k=1}^{j} x_{jk} \tag{2.9}$$

$$\sum_{k=1}^{j} x_{jk} \leq 1 \qquad (j = 1, \ldots, n) \tag{2.10}$$

$$\sum_{j=k+1}^{n} w_j x_{jk} \leq (W - w_k) x_{kk} \qquad (k = 1, \ldots, n-1) \tag{2.11}$$

$$\sum_{k=1}^{n} h_k x_{kk} \leq H \tag{2.12}$$

$$x_{jk} \in 0{,}1 \qquad (k = 1, \ldots, n; j = k, \ldots, n) \tag{2.13}$$

Adottando la notazione del secondo modello per risolvere un problema di 2LSP come visto nel primo, il modello che si ottiene è il seguente:

$$\min \sum_{k=1}^{n} h_k x_{kk} \tag{2.14}$$

$$\sum_{k=1}^{j} x_{jk} = 1 \qquad (j = 1, \ldots, n) \tag{2.15}$$

$$\sum_{j=k+1}^{n} w_j x_{jk} \leq (W - w_k) x_{kk} \qquad (k = 1, \ldots, n-1) \tag{2.16}$$

$$x_{jk} \in 0{,}1 \qquad (k = 1, \ldots, n; j = k, \ldots, n) \tag{2.17}$$

La funzione obiettivo (2.14) minimizza la somma delle altezze dei livelli inizializzati, il vincolo (2.15) impone che ogni elemento venga posizionato in un solo livello, il vincolo (2.16) richiede il rispetto della larghezza della strip. Nel seguito

si farà riferimento al modello appena descritto come "Modello a *shelf*" o "Modello A". Sebbene il vincolo di posizionamento a livelli restringa la probabilità di trovare una soluzione ottima al problema di 2SP classico, la sua utlità si dimostra nel fatto che i modelli ad esso associati, come quelli appena descritti, hanno un numero polinomiale di variabili e di vincoli, il cui numero è direttamente legato al numero di elementi presenti, e nella pratica si dimostrano relativamente efficienti nell'arrivare ad una soluzione. Questo sarà evidente quando nel Capitolo 5 dove saranno confrontati i risultati dei diversi modelli esatti.

2.2 Modello B

Si passa ora ad analizzare il primo dei modelli esatti che affrontano il problema dello *strip packing* bidimensionale in maniera diretta, per il quale si fa riferimento a [Costa et al., 2017]. Un modello PLI come quello qui presentato è definito "pseudopolinomiale", in questo tipo di modelli il numero di variabili e di vincoli non dipende solo dalla quantità di elementi nel problema, ma anche dalla capacità del contenitore in cui devono essere posizionati [Delorme et al., 2016]. Si noti che rispetto alla versione citata del modello viene qui presentata una versione in cui il problema ha uno sviluppo verticale. Il contenitore è suddiviso in quadrati di dimensione 1×1 indicati attraverso le coordinate $p = 0, \ldots, H-1$ per le righe e $q = 0, \ldots, W-1$ per le colonne. Le variabili decisionali sono le seguenti:

$$x_{pq}^{j} = \begin{cases} 1 & \text{se l'angolo inferiore sinistro dell'elem. } j \text{ è posizionato in } p, q \\ 0 & \text{altrimenti} \end{cases} \tag{2.18}$$

per $j = 1, \ldots, n$, $p \in H_j$, $q \in W_j$ Dove:

- $H_j = 0, \ldots, H - h_j$
- $W_j = 0, \ldots, W - w_j$

che indicano l'insieme di punti in cui l'angolo inferiore sinistro del generico elemento j può essere posizionato. Il modello ILP è il seguente:

$$\min z \tag{2.19}$$

$$\sum_{p\in H_j}\sum_{q\in W_j} x^j_{pq} = 1 \qquad (j = 1,\ldots,n) \tag{2.20}$$

$$\sum_{j=1}^{n}\sum_{\substack{p=s-h_j+1\\ p\in H_j}}^{s}\sum_{\substack{q=r-w_j+1\\ q\in W_j}}^{r} x^j_{pq} \leq 1 \quad (s = 1,\ldots,H-1; r = 1,\ldots,W-1) \tag{2.21}$$

$$\sum_{p\in H_j}\sum_{q\in W_j} (p+h_j)x^j_{pq} \leq z \qquad (j = 1,\ldots,n) \tag{2.22}$$

$$x^j_{pq} \in 0{,}1 \qquad (j = 1,\ldots,n; p \in H_j; q \in W_j) \tag{2.23}$$

La funzione obiettivo (2.19) richiede che venga minimizzata l'altezza totale usata misurata attraverso il vincolo (2.22), i vincoli (2.20) e (2.21) impongono rispettivamente che ogni elemento venga posizionato una e una sola volta, e che ogni unità di area venga occupata al massimo da un solo elemento, evitando così le sovrapposizioni. In seguito sarà fatto riferimento a questo modello come "Modello B".

2.3 Modello C

L'ultimo modello esatto presentato si basa sull'enumerazione dei possibili piazzamenti relativi tra coppie di oggetti [Chen et al., 1995]. In questo caso si tratta di un modello con un numero polinomiale di variabili e vincoli. Le variabili sono le seguenti:

$$x_j \in [0, W - w_j] \tag{2.24}$$

$$y_j \in [0, H - h_j] \tag{2.25}$$

che rappresentano le possibili coordinate dell'angolo inferiore sinistro dell'elemento j. Inoltre si ha:

$$\alpha_{jk} = \begin{cases} 1 \text{ se l'angolo inferiore sinistro dell'elem. } j \text{ è a sinistra} \\ \text{dell'angolo inferiore sinistro dell'elem. } k \\ 0 \text{ altrimenti} \end{cases} \tag{2.26}$$

$$\beta_{jk} = \begin{cases} 1 \text{ se l'angolo inferiore sinistro dell'elem. } j \text{ è a destra} \\ \text{dell'angolo inferiore sinistro dell'elem. } k \\ 0 \text{ altrimenti} \end{cases} \tag{2.27}$$

$$\gamma_{jk} = \begin{cases} 1 \text{ se l'angolo inferiore sinistro dell'elem. } j \text{ è sopra} \\ \text{all'angolo inferiore sinistro dell'elem. } k \\ 0 \text{ altrimenti} \end{cases} \tag{2.28}$$

$$\delta_{jk} = \begin{cases} 1 \text{ se l'angolo inferiore sinistro dell'elem. } j \text{ è sotto} \\ \text{all'angolo inferiore sinistro dell'elem. } k \\ 0 \text{ altrimenti} \end{cases} \tag{2.29}$$

$$\theta \ \textit{intera} \tag{2.30}$$

con $j, k = 1, \ldots, n; j \neq k$. Il modello ILP è il seguente:

$$\min \theta \tag{2.31}$$

$$(x_j + w_j) - x_k \leq W(1 - \alpha_{jk}) \qquad \forall (j,k), j \neq k \tag{2.32}$$

$$(x_k + w_k) - x_j \leq W(1 - \beta_{jk}) \qquad \forall (j,k), j \neq k \tag{2.33}$$

$$(y_k + h_k) - y_j \leq W(1 - \gamma_{jk}) \qquad \forall (j,k), j \neq k \tag{2.34}$$

$$(y_j + h_j) - y_k \leq W(1 - \delta_{jk}) \qquad \forall (j,k), j \neq k \tag{2.35}$$

$$\alpha_{jk} + \beta_{jk} + \gamma_{jk} + \delta_{jk} \geq 1 \qquad \forall (j,k), j \neq k \tag{2.36}$$

$$\theta \geq y_j + h_j \qquad \forall j \tag{2.37}$$

$$x_j, y_j \ \textit{intere} \qquad \forall j \tag{2.38}$$

$$\theta \ \textit{intera} \tag{2.39}$$

$$\alpha_{jk}, \beta_{jk}, \gamma_{jk}, \delta_{jk}, z_j \in 0{,}1 \qquad \forall (j,k), j \neq k \tag{2.40}$$

La funzione obiettivo (2.31) richiede di minimizzare l'altezza totale data dal vincolo (2.37). I vincoli da (2.32) a (2.35) controllano rispettivamente il posizionamento a sinistra, destra, sopra e sotto di j rispetto a k, mentre il vincolo (2.36) impone che gli elementi j e k non si sovrappongano. Nel seguito si farà riferimento a questo modello come "Modello C".

I modelli esatti appena presentati sono stati poi modificati per adattarli al caso di interesse degli elementi deformabili come illustrato nel Capitolo 4.

Capitolo 3

Lo *strip packing* con oggetti deformabili

Partendo dai concetti illustrati nei capitoli precedenti si passa ora ad illustrare il problema per cui questa tesi propone alcuni modelli esatti e algoritmi euristici, ovvero il 2SP nel caso in cui gli oggetti siano deformabili.

In [Baker et al., 1980] è riportato che tra gli utilizzi delle tecniche atte a risolvere i problemi di *packing* in due dimensioni c'è quello della divisione e programmazione dei compiti (*task*) in sistemi a risorse condivise. Proprio questa possibilità apre a quella che è un'applicazione del problema studiato in questa tesi, si può infatti immaginare un sistema di *High Performance Computing* (HPC) in cui un grande numero di processori deve svolgere più *task* che possono essere più o meno ripetitivi e prevedibili nel tempo. In particolare, possono esistere situazioni in cui l'insieme di compiti che devono essere svolti dal sistema è ripetuto nel tempo, come ad esempio nei centri per le previsioni meterorologiche, o situazioni in cui invece la risorsa condivisa svolge richieste da parte di più utenti in modo meno prevedibile e non ripetitivo, come nei centri di ricerca. Nei casi in cui l'insieme dei *task* è noto a priori e si può programmare in anticipo, si può pensare di utilizzare tecniche risolutive che minimizzino il tempo totale necessario allo svolgimento, come ad esempio i modelli matematici e gli algoritmi che mirano a risolvere il problema dello *strip packing*. In questa situazione pratica i vari *task* che devono essere eseguiti corrispondono agli elementi da posizionare, con la base della *strip* che,

suddivisa in unità, corrisponde all'insieme di processori a disposizione nel sistema. Gli elementi dovrebbero essere quindi posizionati con l'obiettivo di minimizzare l'altezza totale usata, che in questo caso corrisponde al tempo di esecuzione totale (*makespan*. Si noti che in questa rappresentazione del problema si considerano identici tutti i processori, e si esclude la possibilità di *preemption*, ovvero non si può interrompere l'esecuzione di un task una volta iniziata, per riprenderla poi successivamente, e inoltre non è possibile modificare il numero di processori dedicati ad un particolare *task* durante la sua esecuzione. Con queste assunzioni è possibile immaginare quindi i vari compiti come rettangoli da posizionare in un contenitore la cui altezza, che corrisponde al tempo di esecuzione, deve essere minimizzata.

Una delle prime descrizioni delle applicazioni multi-processore come problema di *packing* risale alla metà degli anni '70 [Garey and Graham, 1975], analizzando però questa formulazione del problema, si nota come il problema studiato dagli autori non sia totalmente simile a quello di interesse per questa ricerca. Nel lavoro presentato da Garey e Graham infatti, si considera la quantità di risorsa disponibile come la totalità di processori disponibili in un esatto istante, determinando la possibilità di affidargli l'esecuzione di un certo *task* anche se non contigui nell'eventuale rappresentazione da noi utilizzata, lasciando aperta la possibilità di "frammentare" i compiti da eseguire. Questo non è permesso nella nostra formulazione del problema, dove ad eseguire un processo in parallelo sono processori contigui nella rappresentazione logica.

Nel tempo le applicazioni multi-processore sono state studiate sempre più attraverso l'uso di problemi di *scheduling* piuttosto che di *packing*. Il prolema di *multi-processor scheduling* è simile a quello di *packing* sotto certi aspetti, ovvero l'assegnazione di un insieme di *task* a più processori che lavorano in parallelo minimizzando generalmente il tempo di esecuzione [Kling et al., 2017], ma sfrutta generalmente modelli ed euristiche diverse. Inoltre, la ricerca si è concentrata su una variante del problema che coinvolge anche una ulteriore "risorsa condivisa" che nella pratica può rappresentare la larghezza della banda o l'energia necessaria richiesta per il funzionamento del sistema, mettendo il focus sulla programmazione simultanea dei processi e della determinata risorsa condivisa. Per una recente raccolta bibliografica sul tema si rimanda a [Janiak et al., 2007].

L'obiettivo di questa tesi è invece presentare degli euristici per lo *strip packing*

che si concentrino solo sulla ricerca di una soluzione buona o ottima in termini di suddivisione dei compiti tra i vari processori, con l'ulteriore richiesta di considerare anche le possibili deformazioni degli elementi. Nel paragrafo seguente viene quindi presentato il problema studiato, illustrando come possa essere utilizzato per modellare la parallelizzazione dei vari *task* su più processori contigui.

3.1 Definizione del problema di *strip packing* con oggetti deformabili

Come fatto per la versione classica del problema di *strip packing*, anche nel caso degli elementi deormabili prima di presentare i modelli PLI è necessario presentarne la notazione utilizzata per modellare il problema.

Nello *strip packing* con elementi deformabili si ha un insieme M di *task* $j \in M = (1, \ldots, m)$, ciascuno con una dimensione (*effort*) e_j, ed una *strip* di larghezza W rappresentante l'insieme di risorse. Nell'applicazione oggetto di studio, ogni *task* j può essere eseguito su w macchine consecutive (con $1 \leq w \leq W$) per un tempo $t_{j,w}$ inversamente proporzionale al grado di parallelizzazione w dello stesso, ed è quindi rappresentabile graficamente attraverso un rettangolo di base $w_j = (1, \ldots, r)$ rappresenta il numero di risorse usate, e altezza h_j che è pari al tempo di esecuzione $t_{j,w}$, con $w_j h_j \geq e_j$. Considerando quindi diverse forme possibili per ogni processo, il risultato è la generazione di un insieme N di *items* $i \in N = (1, \ldots, n)$. L'obiettivo è quello di posizionare tutti i *task* decidendone collocazione e forma, in modo da minimizzare l'altezza totale della *strip*. Il tutto senza *preemption* e con ogni risorsa che può eseguire solo un *task* alla volta, evitando quindi sovrapposizioni.

Nelle applicazioni multi-processore l'altezza di ogni elemento corrisponde al tempo di esecuzione $t_{j,w}$, e la base al numero di processori su cui il *task* j è parallelizzato, con l'altezza totale del *packing* che corrisponde al tempo di esecuzione totale. Nel corso di questa ricerca si assume che l'effetto della parallelizzazione sul tempo di esecuzione di un compito, sia una diminuzione del tempo necessario alla sua esecuzione, con una relazione inversamente proporzionale tra il numero di

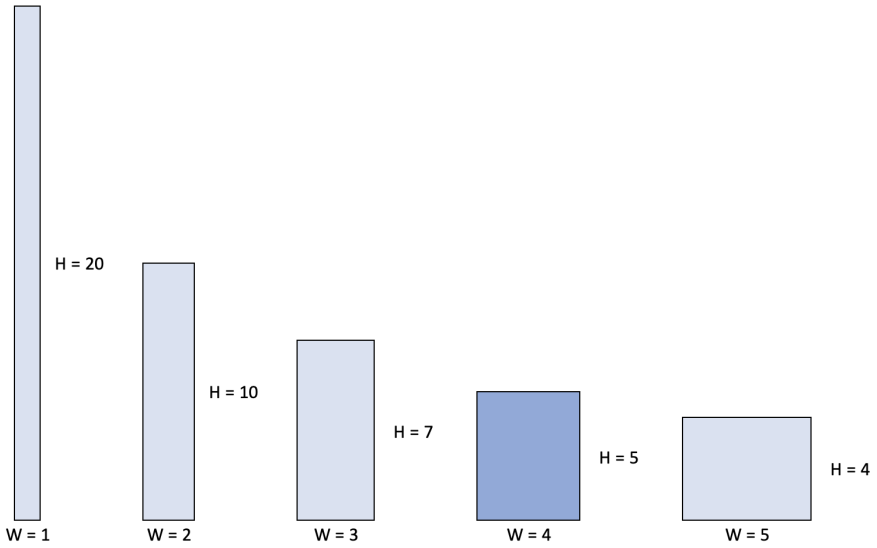

Figura 3.1. Esempio delle possibili forme del *task* 6 dell'istanza NGCUT03_05 a partire da un elemento con $w_6 = 4$, $h_6 = 5$ ed una larghezza massima pari a 5.

risorse usate ed il tempo di esecuzione, è da tenere però presente che la proporzionalità inversa pura è solo una assunzione esemplifictiva del problema, e che gli effetti della parallelizzazione sui tempi di esecuzione di un processo sono attualmente oggetto di studio nel campo del HPC. Uno dei primi studi sugli effetti della parallelizzazione tra più processori è quello presentato da [Amdahl, 1967], in cui è sottolineata la necessità di continuare a puntare sull'aumento della potenza dei singoli processori, piuttosto che aumentarne il numero, evidenziando un effetto collaterale negativo della parallelizzazione descrivibile attraveso la *Legge di Amdahl*, spesso sintetizzata nella seguente formula:

$$\frac{t_i(n)}{t_i(1)} = \frac{1}{s_i + (p_i(n)/n)} \tag{3.1}$$

Dove:

- $t_i(k)$ è il tempo di esecuzione di un *task* i su k processori;
- s_i è il tempo di esecuzione delle parti di *task* da eseguire essenzialmente in serie;
- $p_i(n)$ è il tempo di esecuzione delle parti di *task* parallelizzabili.

In pratica, Amdahl sotolineò che andando a parallelizzare l'esecuzione su più componenti si rendevano necessarie ulteriori risorse computazionali per coordinare e sincronizzare il lavoro dei processori, aumentando così il tempo di esecuzione rispetto a quello che si sarebbe potuto avere con una suddivisione ideale del *task* tra i processsori. Il lavoro di coordinamento tra processori si aggiunge a quello già necessario, aumento che Amdahl stimò essere circa il 10% del totale, mentre il lavoro che deve essere svolto obbligatoriamente in serie occupa circa il 25% delle risorse computazionali, lasciando a disposizione, nel caso più probabile, solo il 65% delle risorse computazionali per la parte in parallelo. Nel lavoro di generazione delle istanze considerate per questa tesi, comunque, è stata fatta assunzione di proporzionalità inversa semplice tra il tempo d'esecuzione ed il numero di risorse usate, semplificando quello che è il problema realmente affrontato dai centri di HPC.

Capitolo 4

Modelli PLI e algoritmi euristici per lo *strip packing* con oggetti deformabili

Ora che il problema dello *strip packing* con elementi deformabili è stato definito, si presentano le modifiche necessarie ad adattare i modelli presentati nel Paragrafo 1.1. Ognuno dei tre modelli opera un piccolo cambio di notazione adeguandosi a quella utilizzata per presentare il problema nel capitolo precedente, considerando con n la quantità di elementi i totali, e con m la quantità di *task* j. Inoltre, per tenere conto del fatto che gli n elementi generati sono ripartiti tra i diversi *task* viene introdotta la variabile I_j, che indica la famiglia di elementi i appartenenti al processo j.

4.1 Adattamento dei Modelli PLI

I modelli di programmazione lineare intera utilizzati nel corso della ricerca si basano sull'adattamento al problema specifico di quelli tipici della versione classica del problema di *strip packing*. Si presentano quindi tali adattamenti, che saranno poi sfruttati anche in alcuni degli euristici presentati nel seguito. Seguendo la logica usata in precedenza, ai modelli sarà fatto riferimento rispettivamente attraverso i nomi "Modello A adattato", "Modello B adattato" e "Modello C adattato".

Modello A adattato. Questo modello sfrutta come base il Modello A, ma le variabili decisionali x_{ik} sono riferite non al posizionamento di un *task* j bensì a quello di un elemento i. Si è reso inoltre necessario modificare il vincolo (2.15) per tenere in considerazione la necessità di posizionare un solo elemento per ogni *task*. Il modello è quindi il seguente:

$$\min \sum_{k=1}^{n} h_k x_{kk} \tag{4.1}$$

$$\sum_{i \in I_j} \sum_{k=1}^{i} x_{ik} = 1 \qquad (j = 1, \ldots, m) \tag{4.2}$$

$$\sum_{i=k+1}^{n} w_i x_{ik} \leq (W - w_k) x_{kk} \qquad (k = 1, \ldots, n-1) \tag{4.3}$$

$$x_{ik} \in 0{,}1 \qquad (k = 1, \ldots, n; i = k, \ldots, n) \tag{4.4}$$

Nel seguito sarà fatto riferimento a questo modello anche come "Modello a livelli adattato" o "Modello a *shelf* adattato".

Modello B adattato. L'adattamento del Modello B, come nel caso del precedente, parte dalla modifica delle variabili decisionali x^i_{pq} in modo che indichino il posizionamento dell'angolo in basso a sinistra non più di un *task* j ma di un oggetto i. È stata introdotta anche una nuova variabile decisionale y_i che indica se un determinato elemento debba essere posizionato o meno. Le variabili decisionali sono quindi:

$$x^i_{pq} = \begin{cases} 1 & \text{se l'angolo inferiore sinistro dell'elem. } i \text{ è posizionato in } p, q \\ 0 & \text{altrimenti} \end{cases} \tag{4.5}$$

per $i = 1, \ldots, n$. In aggiunta:

$$y_i = \begin{cases} 1 & \text{se elem. } i \text{ è selezionato} \\ 0 & \text{altrimenti} \end{cases} \qquad (i = 1, \ldots, n) \tag{4.6}$$

Si è reso inoltre necessario modificare il vincolo (2.20) richiedendo il posizionamento di ogni elemento una sola volta e solo se selezionato, ed è stato aggiunto un

vincolo (4.8) che richiede la selezione di un solo elemento per ogni *task*. Il modello diventa il seguente:

$$\min z \tag{4.7}$$

$$\sum_{\substack{i=1 \\ i \in I_j}}^{n} y_i = 1 \qquad (j = 1, \ldots, m) \tag{4.8}$$

$$\sum_{p \in H_i} \sum_{q \in W_i} x_{pq}^i = y^i \qquad (i = 1, \ldots, n) \tag{4.9}$$

$$\sum_{i=1}^{n} \sum_{\substack{p=s-h_i+1 \\ p \in H_i}}^{s} \sum_{\substack{q=r-w_i+1 \\ q \in W_i}}^{r} x_{pq}^i \leq 1 \quad (s = 1, \ldots, H-1; r = 1, \ldots, W-1) \tag{4.10}$$

$$\sum_{p \in H_i} \sum_{q \in W_i} (p + h_i) x_{pq}^i \leq z \qquad (i = 1, \ldots, n) \tag{4.11}$$

$$x_{pq}^i \in 0{,}1 \qquad (i = 1, \ldots, n; p \in H_i; q \in W_i) \tag{4.12}$$

$$y^i \in 0{,}1 \qquad (i = 1, \ldots, n) \tag{4.13}$$

Si evidenzia che in questo modello è richiesto l'uso di un valore di altezza massima H da assegnare al contenitore, dato che lo si sta però applicando allo *strip packing* è stato necessario decidere, a livello operativo, come assegnare il valore di H in modo *problem-specific*. Tale scelta è stata fatta bilanciando due aspetti: da un lato la necessità di non escludere soluzioni considerabili buone (ad esempio minori dell'altezza dell'elemento più alto), dall'altro tenere sotto controllo il numero di variabili e vincoli necessari data la pseudo-polinomialità del modello. In sintesi, la formula utilizzata è stata la seguente:

$$H = \left\lceil \frac{m}{W} \right\rceil h_i^{max} \tag{4.14}$$

Dove:

- m rappresenta il numero di *task* del problema;
- W rappresenta la larghezza del contenitore;
- h_i^{max} rappresenta l'altezza dell'elemento più alto del problema.

La formula sfrutta il fatto che ogni *task* presenta sempre, tra gli altri, un elemento di larghezza unitaria. Una semplice soluzione potrebbe quindi essere quella di scegliere, per ogni processo, l'elemento di larghezza pari a 1, affiancando via via tutti gli elementi e creando così una soluzione pari all'altezza h_i^{max} nel caso di $n \leq W$, e sicuramente minore o uguale a $\lceil n/W \rceil h_i^{max}$ nel caso di $n > W$.

Modello C adattato. Nel Modello C, oltre al cambio di notazione per indicare gli elementi i, è stato aggiunto il vincolo (4.17), che si basa sulla seguente variabile decisionale:

$$z_i = \begin{cases} 1 \text{ se l'elem. } i \text{ è selezionato} \\ 0 \text{ altrimenti} \end{cases} \tag{4.15}$$

Questo specifica la richiesta di un solo elemento per ogni *task* j. Un'ulteriore modifica è stata effettuata al vincolo (2.36), per assicurare che venga applicato a coppie di elementi in cui entrambi sono selezionati, ed al vincolo (2.37). Il modello adattato è il seguente:

$$\min \theta \tag{4.16}$$

$$\sum_{i \in I_j} z_i = 1 \qquad (j = 1, \ldots, m) \tag{4.17}$$

$$(x_i + w_i) - x_k \leq W(1 - \alpha_{ik}) \qquad \forall (i,k), i \neq k \tag{4.18}$$

$$(x_k + w_k) - x_i \leq W(1 - \beta_{ik}) \qquad \forall (i,k), i \neq k \tag{4.19}$$

$$(y_k + h_k) - y_i \leq W(1 - \gamma_{ik}) \qquad \forall (i,k), i \neq k \tag{4.20}$$

$$(y_i + h_i) - y_k \leq W(1 - \delta_{ik}) \qquad \forall (i,k), i \neq k \tag{4.21}$$

$$\alpha_{ik} + \beta_{ik} + \gamma_{ik} + \delta_{ik} \geq z_i + z_k - 1 \qquad \forall (i,k), i \neq k \tag{4.22}$$

$$\theta \geq y_i + h_i z_i \qquad \forall j \tag{4.23}$$

$$x_i, y_i \ \textit{intere} \qquad \forall i \tag{4.24}$$

$$\theta \ \textit{intera} \tag{4.25}$$

$$\alpha_{ik}, \beta_{ik}, \gamma_{ik}, \delta_{ik}, z_i \in 0{,}1 \qquad \forall (i,k), i \neq k \tag{4.26}$$

4.2 Algoritmi euristici

Per risolvere in tempi ragionevoli e con soluzioni di buona qualità il problema oggetto di studio, sono stati sviluppati alcuni algoritmi euristici. Tutti gli algoritmi presentati sono del tipo *off-line*, in quanto nel problema esaminato tutti gli oggetti sono noti a priori, caratteristica che ha permesso di fornire in input una lista degli elementi pre-ordinata per altezze non-crescenti. Inoltre, ogni algoritmo ha funzionamento deterministico, in quanto ad ogni esecuzione si ottiene sempre la stessa soluzione attraverso lo stesso procedimento. Il primo algoritmo presentato non utilizza nessun modello esatto ed è stato sviluppato in maniera indipendente dagli altri, i successivi invece si basano tutti sull'applicazione di due o più modelli tra quelli presentati nel paragrafo precedente e adottano tutti il paradigma della ricerca locale con alcune differenze tra i diversi casi.

Algoritmo A. L'algoritmo lavora attraverso due fasi che sono ripetute in successione per un numero di volte limitato superiormente dal numero di elementi n presenti nel problema. Il paradigma seguito è quello della ricerca locale iterativa, in quanto ad ogni iterazione si perturba la soluzione ottenuta a quella precedente, verificandone il valore ed aggiornando l'altezza trovata e le informazioni su posizionamento e scelta degli oggetti ogni volta che si trova una combinazione migliorativa. Le due fasi sono:

1. Fase di selezione;
2. *Bottom-Left* packing.

Nella fase 1 vengono selezionati gli elementi di cui poi si deve decidere il posizionamento. Alla prima iterazione l'algoritmo sceglie, per ogni *task*, l'elemento di altezza maggiore. Alle iterazioni successive, invece, l'algoritmo individua l'*item* di altezza maggiore tra tutti quelli selezionati in precedenza, lo deseleziona e procede a selezionarne uno appartenente allo stesso processo, che sia di altezza minore. Se l'elemento più alto dovesse appartenere ad un *task* già completamente esplorato non viene deselezionato, e l'algoritmo passa semplicemente al successivo di altezza maggiore, *task* non è stato completamente esplorato. Ad ogni iterazione, dopo la fase di selezione, si passa alla fase di packing (fase 2), che viene svolta

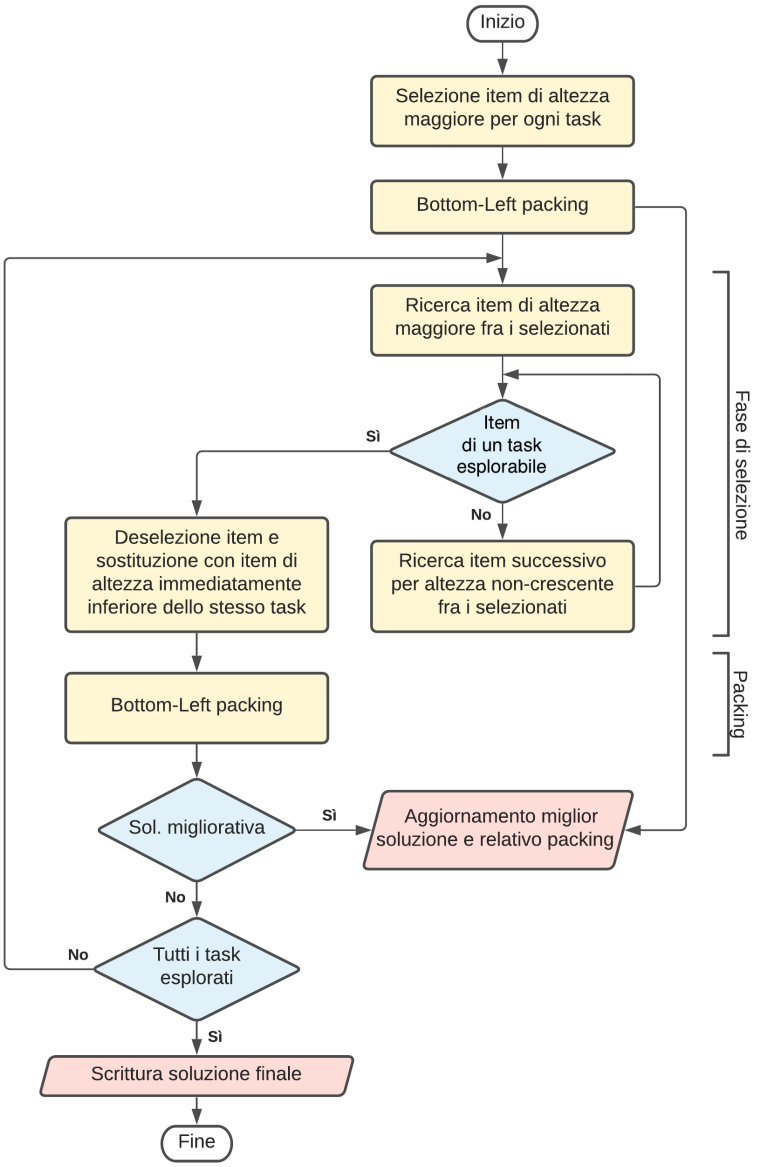

Figura 4.1. Diagramma di flusso rappresentante il funzionamento dell'Algoritmo A.

attraverso l'uso dell'algoritmo approssimativo *Bottom-Left* (BL) e che è stato applicato suddividendo il contenitore in $W \times H$ quadrati di area unitaria. Il BL è un'euristica costruttiva, che compone la soluzione da zero a partire da una lista di elementi che vengono posizionati uno alla volta nel punto più basso possibile e più a sinistra possibile tra quelli a disposizione, (ovvero tali per cui l'elemento può

essere posizionato interamente all'interno del contenitore senza sovrapposizioni). [Baker et al., 1980]. In questo caso la lista ordinata di elementi è quella fornita dal processo di selezione della fase 1.

Le iterazioni possono essere interrotte prematuramente in caso la selezione degli elementi resti la stessa per più di una volta o in caso di raggiungimento di un certo tempo limite $TDEF$ fissato dall'utente. Il tempo limite è stato fissato a 1h.

Algoritmo B. Questo algoritmo si basa sulla ricerca locale. La soluzione di partenza è determinata attraverso il Modello a livelli adattato, ed è quindi una soluzione a livelli che viene poi fornita come soluzione iniziale al Modello B adattato. La ricerca si basa perciò sull'intorno più ampio possibile, ovvero quello che

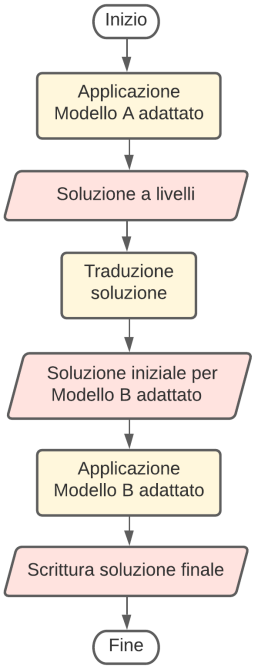

Figura 4.2. Diagramma di flusso rappresentante il funzionamento dell'Algoritmo B.

permette di considerare tutti gli elementi di tutti i processi. Il migliormento rispetto all'applicazione del solo Modello B adattato sta quindi nel fatto che si può far partire l'ottimizzazione da una soluzione di partenza ammissibile e di buona qualità, velocizzando la scrittura e l'esecuzione del modello pseudopolinomiale.

L'applicazione in sequenza dei due modelli è legittimata dal fatto che il Modello a *shelf* produce una soluzione ottima per il problema a livelli z^1 che sarà sempre maggiore o uguale alla soluzione ottima z^2 trovata senza il vincolo di impaccamento a livelli, essendo il secondo caso un rilassamento del primo. Per queso motivo, è lecito pensare di perturbare la soluzione iniziale cercando nell'intorno composto da tutti i posizionamenti possibili senza il vincolo a livelli rimettendo in gioco tutti gli elementi, non solo quelli già scelti dal Modello 1 adattato. Per lo stesso motivo, per semplificare il lavoro di scrittura e risoluzione del *solver*, il valore H fornito al Modello 2 adattato in questo algoritmo è proprio il valore della soluzione iniziale z^1. Il tempo limite di ottimizzazione a livello globale (ottmizzazione con il primo modello e ri-ottimizzazione) è stato di 1h.

Algoritmo C. In questo caso si sfrutta ancora il paradigma della ricerca locale e si parte sempre dalla soluzione fornita dal Modello a *shelf* adattato, ma l'intorno esplorato si basa solo sulle possibili rioganizzazioni di ogni livello in maniera separata attraverso l'adattamento del Modello B. Si ha quindi una ri-ottimizzazione per ogni *shelf*, e ad ognuna di esse si ha l'esecuzione del modello pseudopolinomiale al quale è data come soluzione iniziale il *packing* del livello stesso, e come valore H l'altezza di tale livello. Mantenendo i *task* già presenti si cerca una nuova combinazione in termini di elementi selezionati e posizionamento. L'eventuale abbassamento dell'altezza di uno o più livelli produce un miglioramento nella soluzione finale trovata, altrimenti la soluzione iniziale è lasciata invariata. Il tempo limite di ottimizzazione globale è stato di 1h.

Algoritmo D. Questo algoritmo sfrutta la ricerca locale partendo sempre da una soluzione iniziale determinata attraverso il Modello A adattato, ma l'intorno analizzato è quello delle possibili unioni e ri-ottimizzazioni tra coppie di livelli, che vengono riorganizzate attraverso il Modello B adattato. Come nell'Algoritmo C, ad ogni coppia analizzata, si sfrutta come soluzione iniziale per la fase di post-ottimizzazione il *packing* iniziale dei livelli. Dopo aver analizzato l'eventuale miglioramento ottenibile dall'unione di tutte le coppie possibili di *shelves* viene fornita come soluzione finale quella che implementa l'unione che permette di diminuire in misura maggiore l'altezza della *strip*. In caso nessuna unione migliorativa

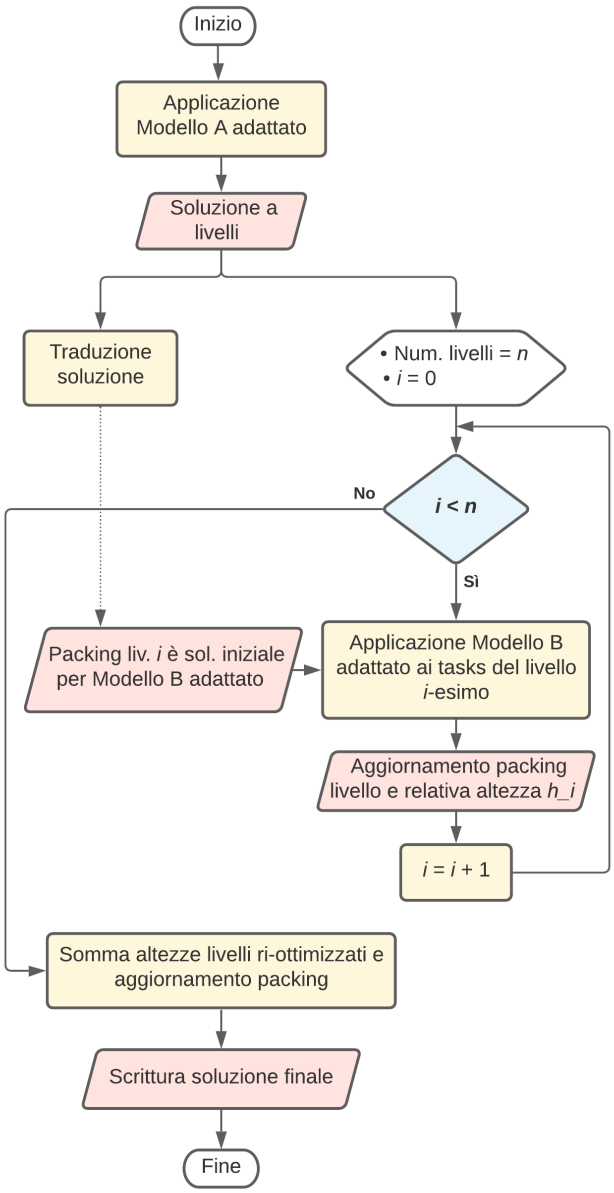

Figura 4.3. Diagramma di flusso rappresentante il funzionamento dell'Algoritmo C.

venga trovata, rimane invariata la soluzione iniziale. Il tempo limite di ottimizzazione globale è stato di 1h. Si noti che l'applicazione di questo algoritmo al caso in cui la soluzione iniziale è composta da un singolo o due livelli, corrisponderebbe all'applicazione dell'Algoritmo B, per questo motivo in questi casi è stata mantenuta

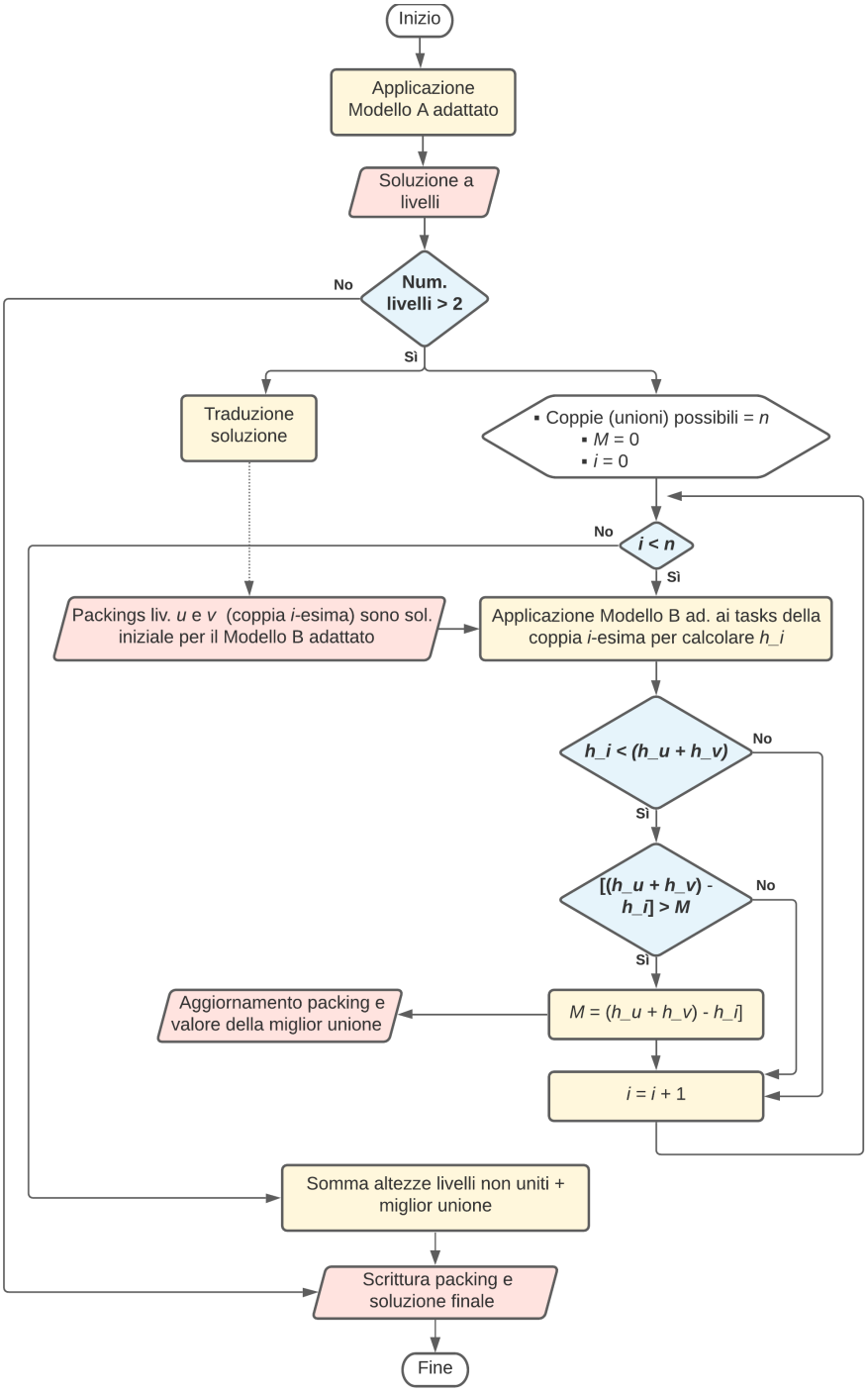

Figura 4.4. Diagramma di flusso rappresentante il funzionamento dell'Algoritmo D.

la soluzione iniziale, ovvero quella determinata attraverso il Modello A adattato, senza effettuare la ricerca di una soluzione migliore.

Algoritmo E. Questo algoritmo adotta lo schema della ricerca locale, ma seguendo un procedimento che vede due possibili alternative a seconda del numero di livelli prodotti dalla soluzione iniziale. Se il numero di livelli della soluzione iniziale è pari o inferiore a due, lo schema seguito è lo stesso dell'Algoritmo B, quindi vengono considerati tutti i possibili *task*. Altrimenti, se il numero di livelli della soluzione iniziale è maggiore di due, la ricerca locale si basa nuovamente sulla riorganizzazione di coppie di livelli seguendo lo stesso schema dell'Algoritmo D, ma i valori di altezza risultanti da ogni riorganizzazione vengono utilizzati per risolvere un problema di assegnamento, che determina quale sia la combinazione di coppie riorganizzate e *shelf* lasciati invariati, che porta al miglior risultato possibile. Questo lascia aperta la possibilità di unire più coppie nello stesso passaggio di ri-ottimizzazione. L'algoritmo si ferma nel caso in cui la risoluzione del problema di assegnamento non abbia prodotto nessuna soluzione migliorativa.

Un problema di assegnamento consiste nel determinare la combinazione migliore per assegnare un determinato numero di oggetti ad un certo numero di risorse massimizzando un profitto o minimizzando un determinata grandezza. Nel caso in esame, i livelli della soluzione di una certa istanza hanno il ruolo sia di oggetti che risorse, in quanto ognuno può essere assegnato a un altro o a se stesso (caso che significa che il determinato livello non è stato unito). Detto θ il numero di livelli generati nella soluzione iniziale, sia $h_u v$ il valore della miglior soluzione trovata nell'unione dei livelli $u = 0, \ldots, \theta - 1$ e $v = 0, \ldots, u$ numerando i livelli da 0 a $\theta - 1$, allora data la seguente variabile decisionale:

$$\alpha_{uv} = \begin{cases} 1 & \text{se livello } u \text{ unito al livello } v \\ 0 & \text{altrimenti} \end{cases} \tag{4.27}$$

si può definire il modello ILP seguente che minimizza l'altezza totale dell'assegnamento:

$$\min \sum_{u=0}^{\theta-1} \sum_{v=0}^{u} h_{uv} \alpha_{uv} \tag{4.28}$$

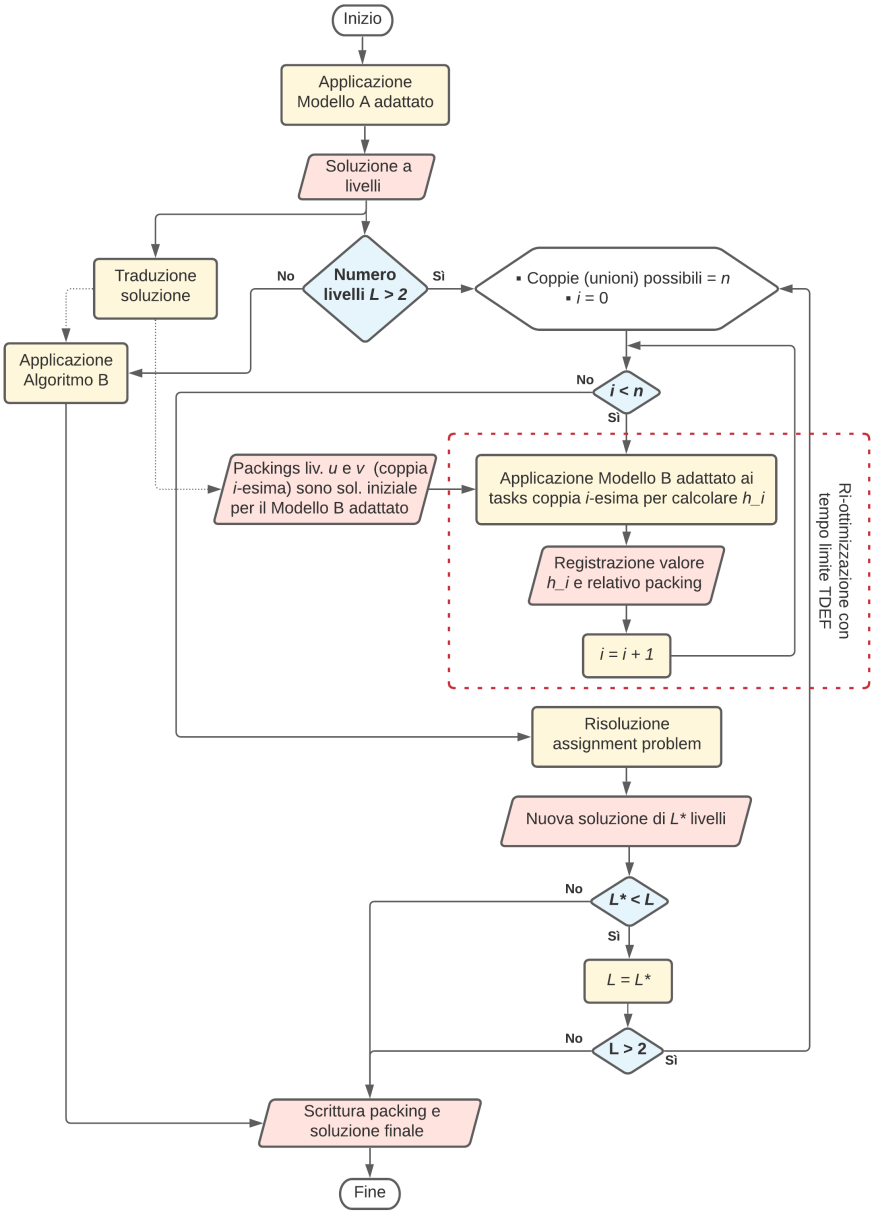

Figura 4.5. Diagramma di flusso rappresentante il funzionamento dell'Algoritmo E.

$$\sum_{v=1}^{s} \alpha_{sv} + \sum_{u=s+1}^{\theta-1} \alpha_{us} = 1 \qquad (s = 0, \ldots, \theta - 1) \tag{4.29}$$

$$\alpha_{uv} \in 0{,}1 \qquad (u = 0, \ldots, \theta - 1; v = 0, \ldots, u) \tag{4.30}$$

Si noti che nel caso in cui $v = u$, h_{uu} è semplicemente l'altezza del livello u, indicando così che tale *shelf* non è stato assegnato.

La fase di valutazione delle unioni e conseguente risoluzione del problema di assegnamento può ripetersi più volte in sequenza, perché se il problema di assegnamento precedente ha migliorato la soluzione, si è all'interno del tempo limite, ed il numero di livelli non è diventato pari o minore di due, allora il procedimento viene ripetuto per valutare le possibili combinazioni dei nuovi livelli generati dall'unione di una o più coppie presenti in precedenza.

Durante le ri-ottimizzazioni, sia nel caso in cui il numero di livelli fosse maggiore di due, che nel caso non lo fosse, l'esecuzione del Modello B adattato per riorganizzare i *task* nei livelli è stata vincolata al tempo limite $TDEF$ definito dall'utente, allo scadere del quale è stata presa la miglior soluzione prodotta dal *solver* fino a quel momento. L'algoritmo è stato testato utilizzando valori di $TDEF = 30, 45, 90s$. Per l'esecuzione del Modello a livelli adattato, necessario ad ottenere la soluzione a *shelf* iniziale, è stato invece impostato un tempo limite di 10 minuti.

Algoritmo F. Il funzionamento si basa esattamente su quello dell'Algoritmo E, ma con una modifica che permette di valutare in maniera più agevole e proficua istanze di grandi dimensioni per cui anche solo due livelli uniti avrebbero prodotto una quantità troppo alta di variabili e vincoli, impedendo una ri-ottimizzazione efficace entro il tempo limite $TDEF$. Quindi, nei casi in cui le dimensioni del problema di ri-ottimizzazione eccedano un certo valore MAX, per riorganizzare le coppie di livelli e produrre i valori da utilizzare nel problema di assegnamento l'algoritmo non sfrutta il Modello B adattato, bensì l'Algortmo A. Anche in questo caso, l'algoritmo agisce ri-organizzando solo i *task* appartenenti ai livelli la cui unione è in corso di valutazione. La regola utilizzata per determinare la dimensione limite MAX si è basata sull'analisi del tempo di risoluzione (o ottenimento di una buona soluione), da parte del Modello B adattato, di istanze di diverse dimensioni, individuando capire quali fossero quelle per cui erano prodotte soluzioni di qualità entro un certo tempo. Per valutare le dimensioni dell'istanza nella valutazione

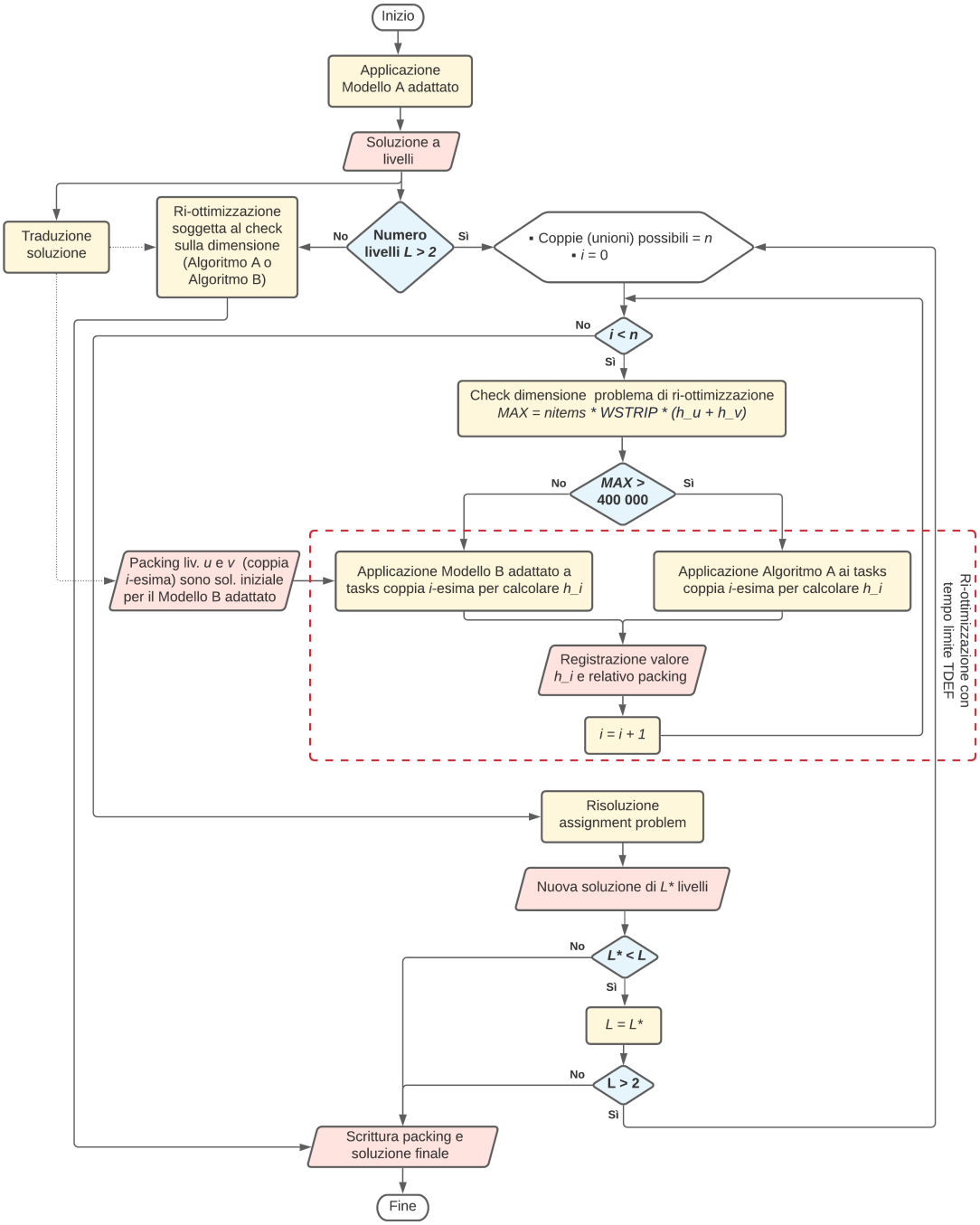

Figura 4.6. Diagramma di flusso rappresentante il funzionamento dell'Algoritmo F.

dell'unione delle coppie è stata usata la seguente formula:

$$MAX = nitems \times W \times H \tag{4.31}$$

Dove *nitems* corrisponde al numero degli elementi presenti in tutto il problema, senza quindi considerare solo quelli relativi ai livelli analizzati, W è la larghezza della *strip* e H la somma dell'altezza dei due livelli. In sintesi, il valore limite che permette un uso proficuo del modello esatto entro 120s si è rivelato essere pari a 400.000. Il tempo limite di 120s è stato usato considerando un $TDEF$ massimo di 90s, in modo da permettere ri-ottimizzazioni proficue entro tale tempistica.

Nel capitolo seguente si passa a presentare i risultati ottenuti applicando i modelli esatti e gli algoritmi euristici appena illustrati. Gli esperimenti computazionali sono stati compiuti su istanze anch'esse descritte nel capitolo successivo.

Capitolo 5

Risultati computazionali

I modelli esatti e gli algoritmi euristici presentati nel Capitolo 4 sono stati applicati per simulare alcune situazioni possibili durante la risoluzione di problemi di ottimizzazione modellabili come problemi di *packing* con oggetti deformabili. A questo scopo, sono state definite delle istanze di *strip packing* con oggetti deformabili, a partire da un insieme di problemi test introdotti nella letteratura scientifica per altri problemi di packing bidimensionale. Il prossimo paragrafo fornisce alcuni dettagli sulle istanze della letteratura da cui siamo partiti e sulla procedura utilizzata per definire istanze di *strip packing* con oggetti deformabili. I test computazionali sono stati svolti su un computer con processore Intel Core i5 dual-core 2,3GHz e RAM di 8GB, utilizzando per la risoluzione dei modelli esatti il solver commerciale Gurobi Optimizer 9.1.2. Tutti i modelli esatti e gli algoritmi sono stati programmati usando il linguaggio C, impostando sempre un limite di tempo globale di ottimizzazione pari a 1h. I risultati dei test sono presentati nel secondo paragrafo di questo capitolo.

5.1 Istanze testate

Le istanze con elementi deformabili sono state generate a partire da istanze base di riferimento già presenti in letteratura per testare i problemi di *packing* classici; queste istanze specificano il numero e le dimensioni dei *task* insieme alla larghezza della *strip*. Le istanze usate appartengono a due classi:

- NGCUT. Proposte originariamente da [Beasley, 1985] per il *knapsack* in due dimensioni. Le 12 istanze base comprese in questa categoria hanno un numero di *task* compreso tra 7 e 22, con una larghezza della *strip* compresa tra 10 e 30;

- C. Proposte originariamente da [Hopper and Turton, 2001] per lo *strip packing* bidimensionale. Di questa categoria sono state considerate 15 istanze base (da c1-p1 a c5-p3), aventi un numero di *task* compreso tra 16 e 73, con una larghezza della *strip* compresa tra 20 e 60.

La Tabella 5.1 riporta le caratteristiche fondamentali di ciascuna istanza base.

Tabella 5.1: Tabella riassuntiva delle caratteristiche delle istanze base.

Nome	**# *task***	**W**	**Nome**	**# *task***	**W**
NGCUT01	10	10	c1-p1	16	20
NGCUT02	17	10	c1-p2	17	20
NGCUT03	21	10	c1-p3	16	20
NGCUT04	7	15	c2-p1	25	40
NGCUT05	14	15	c2-p2	25	40
NGCUT06	15	10	c2-p3	25	40
NGCUT07	8	20	c3-p1	28	60
NGCUT08	13	20	c3-p2	29	60
NGCUT09	18	20	c3-p3	28	60
NGCUT10	13	30	c4-p1	49	60
NGCUT11	15	30	c4-p2	49	60
NGCUT12	22	30	c4-p3	49	60
			c5-p1	73	60
			c5-p2	73	60
			c5-p3	73	60

Generazione di istanze con *items* deformabili. Al fine di definire istanze con *items* deformabili, abbiamo implementato la seguente procedura. Si ha in input un insieme di n *task*, ciascuno caratterizzato da una certa area, un parametro intero $WMAX$, ed associa a ciascun *task* j una famiglia di elementi $i \in I_j$ (*items* rettangolari). Ciascun elemento è caratterizzato da una larghezza w_i ed altezza h_i, e il rettangolo risultante modella l'esecuzione del processo j su w_i macchine

impiegando un tempo di esecuzione pari ad t_i. In particolare, per ciascun *task* j, avente area e_j, la procedura genera al massimo $WMAX$ *items*, l'i-esimo dei quali ha larghezza $w_i = i$ ed altezza $t_i = \lceil \frac{e_j}{w_i} \rceil$ (per $i = 1, \ldots, WMAX$).
Alla fine della generazione, vengono eliminati tutti i rettangoli dominati per un certo *task*; detti i e k due generici *items* associati allo stesso *task*, l'*item* i domina l'*item* k se:

- $w_i \leq w_k$ e $t_i = t_k$, oppure
- $t_i \leq t_k$ e $w_i = w_k$.

Rispettando le caratteristiche appena descritte, sono state generate 5 istanze per ogni istanza base, ognuna con un numero di elementi che varia in base alla larghezza massima scelta ($WMAX$). I valori di larghezza scelti sono i seguenti:

- Per le istanze NGCUT si ha $WMAX$ = 3, 5, 8, 10 ed un valore casuale tra 3 e 7;
- Per le istanze C si ha $WMAX$ = 6, 10, 16, 20 ed un valore casuale tra 7 e 15;

Nel caso della scelta causale di $WMAX$ entro i limiti imposti, si noti che il valore può variare da processo a processo, quindi due *task* della stessa istanza base possono avere, entro i limiti, due valori di $WMAX$ diversi e quindi un diverso numero di elementi e forme possibili. In pratica, in ogni istanza generata, ad ogni *task* è assegnato, quando è permesso dalle dimensioni dello stesso, un numero di elementi pari al valore $WMAX$, ognuno con $w_i = (1, \ldots, WMAX)$.

Ai fini di presentare i risultati dei test computazionali in maniera ordinata, ogni istanza generata è stata rinominata partendo dal nome dell'istanza base ed usando come suffisso il valore di $WMAX$ con cui è stata generata, assegnando "_00" quando si è usato il valore casuale. Ad esempio, quelle generate dall'istanza base NGCUT01 con $WMAX$ pari a 3 e con il valore casuale sono state rinominate rispettivamente NGCUT01_03 e NGCUT01_00. Nella Tabella 5.2 sono riportate le caratteristiche delle istanze generate, e per ognuna si ha anche l'altezza minima teorica LB.

5.1.1 Il *lower bound*

Il calcolo di LB è stato effettuato prendendo in considerazione per ogni *task* j due distinti valori, il *lower bound* geometrico (LBg) ed il *lower bound* associato alle altezze degli elementi (LBh):

1. LBg: l'intero superiore del rapporto tra la somma totale delle aree minime di ogni *task* e la larghezza della *strip*:

$$LBg = \left\lceil \frac{\sum_j \min_{i \in I_j} w_i h_i}{W} \right\rceil \tag{5.1}$$

2. LBh: il massimo tra le altezze minime possibili associate ad ogni oggetto, in formula:

$$LBh = \max_j \min_{i \in I_j} h_i \tag{5.2}$$

Una volta determinati i due valori la formula usata per determinare il LB è la seguente:

$$LB = \max\{LBg, LBh\} \tag{5.3}$$

Si evidenzia che comunque solo rare volte l'altezza minima teorica corrisponderà al valore della soluzione ottimale per la determinata istanza.

Tabella 5.2: Tabella riassuntiva delle caratteristiche delle istanze usate durante i test.

NGCUT					C				
Nome	***#item***	***#task***	**W**	**LB**	**Nome**	***#item***	***#task***	**W**	**LB**
NGCUT01_00	44	10	10	19	c1-p1_00	104	16	20	20
NGCUT01_03	30	10	10	19	c1-p1_06	87	16	20	20
NGCUT01_05	48	10	10	19	c1-p1_10	108	16	20	20
NGCUT01_08	62	10	10	19	c1-p1_16	123	16	20	20
NGCUT01_10	68	10	10	19	c1-p1_20	128	16	20	20
NGCUT02_00	72	17	10	28	c1-p2_00	107	17	20	20
NGCUT02_03	50	17	10	28	c1-p2_06	85	17	20	20
NGCUT02_05	77	17	10	28	c1-p2_10	113	17	20	20
NGCUT02_08	98	17	10	28	c1-p2_16	127	17	20	20
NGCUT02_10	104	17	10	28	c1-p2_20	133	17	20	20
NGCUT03_00	89	21	10	28	c1-p3_00	101	16	20	20

Proseguimento Tabella 5.2

NGCUT					C				
Nome	***#item***	***#task***	**W**	**LB**	**Nome**	***#item***	***#task***	**W**	**LB**
NGCUT03_03	61	21	10	28	c1-p3_06	86	16	20	20
NGCUT03_05	90	21	10	28	c1-p3_10	109	16	20	20
NGCUT03_08	111	21	10	28	c1-p3_16	122	16	20	20
NGCUT03_10	116	21	10	28	c1-p3_20	126	16	20	20
NGCUT04_00	31	7	15	11	c2-p1_00	162	25	40	15
NGCUT04_03	21	7	15	11	c2-p1_06	132	25	40	15
NGCUT04_05	34	7	15	11	c2-p1_10	168	25	40	15
NGCUT04_08	46	7	15	11	c2-p1_16	190	25	40	15
NGCUT04_10	49	7	15	11	c2-p1_20	197	25	40	15
NGCUT05_00	63	14	15	24	c2-p2_00	157	25	40	15
NGCUT05_03	42	14	15	24	c2-p2_06	124	25	40	18
NGCUT05_05	67	14	15	24	c2-p2_10	156	25	40	15
NGCUT05_08	91	14	15	24	c2-p2_16	180	25	40	15
NGCUT05_10	99	14	15	24	c2-p2_20	184	25	40	15
NGCUT06_00	67	15	15	20	c2-p3_00	151	25	40	15
NGCUT06_03	44	15	15	20	c2-p3_06	125	25	40	15
NGCUT06_05	70	15	15	20	c2-p3_10	159	25	40	15
NGCUT06_08	92	15	15	20	c2-p3_16	181	25	40	15
NGCUT06_10	98	15	15	20	c2-p3_20	185	25	40	15
NGCUT07_00	29	8	20	14	c3-p1_00	221	28	60	35
NGCUT07_03	24	8	20	18	c3-p1_06	161	28	60	46
NGCUT07_05	33	8	20	11	c3-p1_10	232	28	60	30
NGCUT07_08	42	8	20	9	c3-p1_16	296	28	60	30
NGCUT07_10	48	8	20	9	c3-p1_20	317	28	60	30
NGCUT08_00	63	13	20	32	c3-p2_00	224	29	60	30
NGCUT08_03	39	13	20	32	c3-p2_06	158	29	60	32
NGCUT08_05	65	13	20	32	c3-p2_10	234	29	60	30
NGCUT08_08	96	13	20	32	c3-p2_16	296	29	60	30
NGCUT08_10	109	13	20	32	c3-p2_20	317	29	60	30
NGCUT09_00	83	18	20	49	c3-p3_00	394	28	60	30
NGCUT09_03	54	18	20	49	c3-p3_06	282	28	60	36
NGCUT09_05	86	18	20	49	c3-p3_10	420	28	60	30
NGCUT09_08	127	18	20	49	c3-p3_16	542	28	60	30
NGCUT09_10	150	18	20	49	c3-p3_20	585	28	60	30
NGCUT10_00	63	13	30	61	c4-p1_00	394	49	60	60
NGCUT10_03	39	13	30	81	c4-p1_06	282	49	60	60

Proseguimento Tabella 5.2

NGCUT					C				
Nome	***#item***	***#task***	**W**	**LB**	**Nome**	***#item***	***#task***	**W**	**LB**
NGCUT10_05	65	13	30	58	c4-p1_10	420	49	60	60
NGCUT10_08	101	13	30	58	c4-p1_16	542	49	60	60
NGCUT10_10	124	13	30	58	c4-p1_20	585	49	60	60
NGCUT11_00	72	15	30	50	c4-p2_00	379	49	60	60
NGCUT11_03	45	15	30	50	c4-p2_06	270	49	60	60
NGCUT11_05	75	15	30	50	c4-p2_10	392	49	60	60
NGCUT11_08	117	15	30	50	c4-p2_16	511	49	60	60
NGCUT11_10	142	15	30	50	c4-p2_20	560	49	60	60
NGCUT12_00	102	22	30	77	c4-p3_00	364	49	60	60
NGCUT12_03	66	22	30	77	c4-p3_06	270	49	60	76
NGCUT12_05	107	22	30	77	c4-p3_10	392	49	60	60
NGCUT12_08	161	22	30	77	c4-p3_16	492	49	60	60
NGCUT12_10	193	22	30	77	c4-p3_20	537	49	60	60
					c5-p1_00	555	73	60	90
					c5-p1_06	411	73	60	90
					c5-p1_10	590	73	60	90
					c5-p1_16	752	73	60	90
					c5-p1_20	817	73	60	90
					c5-p2_00	515	73	60	90
					c5-p2_06	375	73	60	90
					c5-p2_10	545	73	60	90
					c5-p2_16	704	73	60	90
					c5-p2_20	773	73	60	90
					c5-p3_00	567	73	60	90
					c5-p3_06	406	73	60	90
					c5-p3_10	600	73	60	90
					c5-p3_16	764	73	60	90
					c5-p3_20	835	73	60	90

5.2 Risultati

I test computazionali sono stati eseguiti iniziando dalle istanze della categoria NGCUT che pur mantenendo dimensioni contenute hanno permesso di arrivare ad un

giudizio chiaro sulle performance dei vari modelli ed algoritmi testati. Successivamente, sulla base di quanto ottenuto su quest'ultime, si è proceduto testando i modelli e gli algoritmi euristici più promettenti anche sulle istanze derivanti dalla classe C.

Risultati istanze NGCUT

Modelli esatti. I risultati numerici dell'applicazione dei modelli esatti adattati sono riassunti nella Tabella 5.3. Si può sicuramente dire di essere arrivati ad una soluzione ottima per l'istanza analizzata ogni volta che il modello B adattato o il modello C adattato terminano la risoluzione entro il tempo limite fissato di 1h. In tali casi, dato che a differenza del modello A adattato, questi non presentano il vincolo di *packing* a livelli, si è sicuramente arrivati alla miglior soluzione possibile. Un'altra condizione sufficiente a dichiarare l'ottimalità di una soluzione, indipendentemente dal fatto che sia stata trovata tramite un modello esatto o un euristico, si verifica quando il valore della soluzione è pari a quello del valore di LB calcolato per l'istanza in esame. Questo vale anche per i casi in cui la miglior soluzione sia stata trovata da un modello esatto ma senza terminare la risoluzione entro il tempo limite, anche in questo caso se la soluzione equivale il valore di LB può comunque essere dichiarata ottimale.

Dato che non in tutte le istanze si è giunti alla soluzione ottima, nell'analisi dei risultati occorrerà distinguere tra "soluzione ottima" e "miglior soluzione" (*best-known*), dove la seconda rappresenta la miglior soluzione trovata per una determinata istanza durante tutti i test, senza che questa sia necessariamente ottimale. Per chiarezza, in Tabella 5.6 sono mostrate la miglior soluzione di ogni istanza derivante dalle NGCUT, e quali modelli o algoritmi sono giunti a tale risultato.

In virtù di quanto dichiarato, analizzando i risultati ottenuti attraverso i test computazionali sulle istanze basate sulla classe NGCUT, si evidenzia che:

- Il Modello A adattato ha raggiunto la miglior soluzione conosciuta in 24 istanze sulle 60 testate (40%), con un tempo di calcolo medio di $0.44s$. Di

queste, 18 soluzioni sono decretabili come ottime per raggiungimento del valore di *lower bound* dell'istanza;

- il Modello B adattato giunge alla miglior soluzione in 52 istanze su 60 (86.7%), con un tempo medio di calcolo di 685.97s. Di queste, 51 soluzioni sono decretabili come ottime, mentre in un caso (NGCUT12_05) la soluzione è pari alla migliore disponibile, ma la risoluzione ha raggiunto il tempo limite;

- il Modello C adattato giunge alla miglior soluzione disponibile in 24 istanze su 60 (40%), con un tempo medio di calcolo di 2909.55s. Di queste, 16 soluzioni sono decretabili come ottime per raggiungimento del valore di *lower bound* dell'istanza.

I risultati relativi alle occasioni in cui i modelli hanno portato al raggiungimento della soluzione migliore o ottima sono visibili esplicitamente e nel dettaglio nella Tabella 5.6.

Per analizzare ulteriormente la prestazione dei modelli esatti applicati, oltre al numero di soluzioni può essere utile esaminare la distanza media della soluzione trovata, dalla miglior soluzione disponibile per ognuna delle istanze testate. Le prestazioni in termini percentuali di distanza media dalla miglior soluzione (*gap*) dei tre modelli esatti sono le seguenti:

- Per il modello A adattato il *gap* è stato del 4.44%;

- per il modello B adattato il *gap* è stato del 2.60%;

- per il modello C adattato non è stato possibile calcolare la distanza media percentuale, in quanto il modello non è arrivato ad alcuna soluzione per le cinque istanze del tipo NGCUT02. Tenendo conto solo delle soluzioni disponibili il valore del *gap* è 3.61%.

I risultati mostrano come il Modello A adattato sia il più efficiente, mentre il Modello B adattato il più efficace nel raggiungere soluzioni di buona qualità o ottime. Il modello C adattato si è rivelato considerevolmente più lento e quasi mai è giunto ad una soluzione entro il tempo limite, con ben cinque istanze in cui

non ha prodotto alcuna soluzione. Sulla base di questi risultati quindi, è stato deciso di escludere quest'ultimo modello da ulteriori test con istanze di dimensioni maggiori.

Tabella 5.3: Risultati dei tre modelli adattati A, B, C con le istanze NGCUT. Per ogni modello è indicata la soluzione trovata entro un'ora (3600 s), ed il rispettivo tempo di risoluzione. Per il modello A adattato è presente anche l'indicazione riguardo al numero di livelli di cui la soluzione è composta. (*) Soluzione euristica fornita dal solver, non è stato possibile risalire al packing preciso.

	Modello A ad.			**Modello B ad.**		**Modello C ad.**	
Istanza	**Sol.**	**Liv.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT01_00	21	3	0.10	20	8.76	20	3600.04
NGCUT01_03	21	2	0.16	20	0.35	20	3600.02
NGCUT01_05	20	4	0.03	20	12.91	20	3600.03
NGCUT01_08	20	5	0.03	19	5.18	20	3600.03
NGCUT01_10	20	6	0.03	19	10.22	20	3600.03
NGCUT02_00	29	5	0.10	28	234.79	no	3600.05
NGCUT02_03	29	4	0.05	28	56.12	no	3600.03
NGCUT02_05	29	5	0.08	28	278.31	no	3600.04
NGCUT02_08	29	5	0.10	28	236.30	no	3600.26
NGCUT02_10	29	7	0.12	28	210.33	no	3600.03
NGCUT03_00	28	7	0.06	28	25.93	29	3600.04
NGCUT03_03	29	5	0.28	28	34.32	29	3600.03
NGCUT03_05	28	7	0.05	28	87.97	29	3600.55
NGCUT03_08	28	8	0.17	28	56.21	30	3600.12
NGCUT03_10	28	8	0.09	28	29.54	30	3600.12
NGCUT04_00	12	1	0.04	12	1.52	12	199.49
NGCUT04_03	12	1	0.01	12	0.60	12	1.07
NGCUT04_05	12	2	0.01	11	1.28	11	64.07
NGCUT04_08	12	3	0.02	11	3.85	11	1151.56
NGCUT04_10	12	3	0.02	11	6.08	11	3600.02
NGCUT05_00	25	3	0.27	24	64.01	25	3600.03
NGCUT05_03	27	3	1.20	24	7.30	25	3600.02
NGCUT05_05	25	3	0.29	24	49.76	24	3600.08
NGCUT05_08	24	4	0.12	24	35.93	24	3600.17
NGCUT05_10	24	4	0.11	24	59.88	25	3600.13
NGCUT06_00	20	3	0.50	20	34.91	21	3600.04
NGCUT06_03	22	2	0.96	20	5.07	20	3600.02

Proseguimento Tabella 5.3

	Modello A ad.			Modello B ad.		Modello C ad.	
Istanza	**Sol.**	**Liv.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT06_05	20	4	0.09	20	18.34	21	3600.03
NGCUT06_08	20	5	0.30	20	51.69	21	3600.14
NGCUT06_10	20	4	0.12	20	46.10	21	3600.07
NGCUT07_00	14	1	0.01	14	2.36	14	0.76
NGCUT07_03	18	1	0.01	18	1.11	18	0.21
NGCUT07_05	11	1	0.01	11	3.42	11	0.59
NGCUT07_08	10	1	0.02	9	6.63	9	151.86
NGCUT07_10	10	2	0.02	9	8.54	9	168.66
NGCUT08_00	35	2	0.19	32	137.39	34	3600.20
NGCUT08_03	42	2	0.25	34	22.00	34	3600.02
NGCUT08_05	34	2	1.43	32	120.05	34	3600.06
NGCUT08_08	32	2	0.23	32	309.31	33	3600.05
NGCUT08_10	32	3	0.34	32	422.38	33	3600.10
NGCUT09_00	54	3	0.77	49	980.91	52	3600.05
NGCUT09_03	60	2	0.63	50	150.65	50	3600.02
NGCUT09_05	52	3	0.65	49	1550.65	52	3600.14
NGCUT09_08	50	5	0.40	50	3600.08	54	3600.09
NGCUT09_10	50	7	0.70	50	3600.10	55	3600.11
NGCUT10_00	61	1	0.02	61	285.00	61	9.66
NGCUT10_03	81	1	0.02	81	9.54	81	1.09
NGCUT10_05	61	1	0.02	61	212.32	61	3600.03
NGCUT10_08	61	3	0.65	58	1939.16	60	3600.24
NGCUT10_10	59	3	0.17	64	3600.50	62	3600.18
NGCUT11_00	59	2	0.88	52	555.81	53	3600.05
NGCUT11_03	63	1	0.03	63	17.93	63	4.17
NGCUT11_05	52	2	0.13	52	190.62	52	3600.03
NGCUT11_08	52	3	0.89	53	3600.17	54	3600.21
NGCUT11_10	51	3	0.56	78	3600.16	56	3601.64
NGCUT12_00	85	3	1.24	80	3601.29	81	3600.08
NGCUT12_03	101	2	1.27	84	155.06	84	3600.04
NGCUT12_05	84	3	3.35	78	3600.15	80	3600.33
NGCUT12_08	79	4	3.60	107*	3600.55	85	3600.07
NGCUT12_10	78	5	2.12	109*	3600.75	101	3613.99

Algoritmi euristici. I risultati dell'applicazione degli algoritmi euristici sono riassunti in Tabella 5.4 e Tabella 5.5. Per tutti gli euristici qui presentati, ad eccezione dell'Algoritmo B, si può dichiarare l'ottimalità della soluzione solo nell'eventualità in cui il valore coincida con quello del *lower bound* dell'istanza. Nel caso dell'algoritmo B invece, essendo la combinazione di due modelli esatti in successione, si può dichiarare l'ottimalità della soluzione anche nel caso in cui questa sia stata trovata entro il tempo limite di un'ora. Si noti che nell'Algoritmo E, essendo la ri-ottimizzazione dei casi in cui la soluzione iniziale presenta solo uno o due livelli, uguale all'applicazione dell'Algoritmo B, anche in questi casi si può dichiarare la soluzione ottima se la ri-ottimizzazione termina entro il tempo limite, che però in questo caso sarà dato dal TDEF impostato. Lo stesso ragionamento può essere applicato nelle stesse situazioni anche all'algoritmo F, tenendo però conto solo dei casi in cui la ri-ottimizzazione sia stata effettuata con il modello esatto e non con l'euristico. Analizzando quindi i risultati dei test computazionali sulle istanze derivanti dalla classe NGCUT, si evidenzia che:

- L'Algoritmo A è arrivato alla miglior soluzione disponibile in 18 istanze su 60 (30%), di cui in 12 casi con soluzione dichiarabile ottima per eguaglianza con il valore di LB dell'istanza. Il tempo medio di calcolo è stato di $0.65s$. Il *gap* è stato del 3.57%;

- l'Algoritmo B è arrivato alla miglior soluzione in 58 istanze su 60 (96.7%), di cui in 55 casi con soluzione ottimale. Il tempo medio di calcolo è stato di $535.83s$. Il *gap* è stato del 0.05%;

- l'Algoritmo C è giunto alla miglior soluzione disponibile in 24 istanze su 60 (40%), di cui in 18 casi con soluzione dichiarabile ottima per eguaglianza con il valore di LB dell'istanza. Il tempo medio di calcolo è stato di $1.92s$. Il *gap* è stato del 4.09%;

- l'Algoritmo D è giunto alla miglior soluzione disponibile in 35 istanze su 60 (58.3%), di cui in 27 casi con soluzione dichiarabile ottima per eguaglianza con il valore di LB dell'istanza. Il tempo medio di calcolo è stato di $176.07s$. Il *gap* è stato del 4.41%;

- l'Algoritmo E ($TDEF = 30s$) è arrivato alla miglior soluzione disponibile in 38 istanze su 60 (63.3%), di cui in 35 casi con soluzione dichiarabile ottima. Il tempo medio di calcolo è stato di $18.71s$. Il *gap* è stato del 1.61%;
- l'Algoritmo E ($TDEF = 45s$) è arrivato alla miglior soluzione disponibile in 40 istanze su 60 (66.7%), di cui in 36 casi con soluzione dichiarabile ottima. Il tempo medio di calcolo è stato di $25.74s$. Il *gap* è stato del 1.36%;
- l'Algoritmo E ($TDEF = 90s$) è arrivato alla miglior soluzione disponibile in 43 istanze su 60 (71.7%), di cui in 39 casi con soluzione dichiarabile ottima. Il tempo medio di calcolo è stato di $37.24s$. Il *gap* è stato del 1.14%;
- l'Algoritmo F ($TDEF = 90s$) è arrivato alla miglior soluzione disponibile in 43 istanze su 60 (71.7%), di cui in 39 casi con soluzione dichiarabile ottima. Il tempo medio di calcolo è stato di $37.30s$. Il *gap* è stato del 1.14%.

I risultati relativi alle occasioni in cui gli algoritmi euristici hanno portato al raggiungimento della soluzione migliore o ottima sono visibili esplicitamente e nel dettaglio nella Tabella 5.6.

Le prestazioni assolute degli euristici possono essere ulteriormente approfondite ponendole in relazione all'eventuale miglioramento apportato alla soluzione iniziale fornita dal modello a livelli (ad eccezione dell'Algoritmo A). I risultati sono stati i seguenti:

- L'Algoritmo B è stato migliorativo in 35 istanze su 60 (58.3%);
- l'Algoritmo C è stato migliorativo in 2 istanze su 60 (3.33%);
- l'Algoritmo D è stato migliorativo in 18 istanze su 60 (30%);
- l'Algoritmo E ($TDEF = 30s$) è stato migliorativo in 24 istanze su 60 (40%);
- l'Algoritmo E ($TDEF = 45s$) è stato migliorativo in 26 istanze su 60 (43.3%);
- l'Algoritmo E ($TDEF = 90s$) è stato migliorativo in 27 istanze su 60 (45%);
- l'Algoritmo F ($TDEF = 90s$) è stato migliorativo in 27 istanze su 60 (45%).

Dai risultati ottenuti si apprende che l'Algoritmo più veloce è stato l'Algoritmo A, seguito poi dal C, che nonostante abbia una velocità comparabile a quella del primo, è risultato migliorativo solo nel 3.33% delle istanze sulle quali è stato applicato. L'algoritmo B si è rivelato particolarmente lento (824 volte più lento dell'Alg. A e circa 14 volte più lento dell'Alg. E ($TDEF = 90s$), nonostante ciò si è dimostrato molto più efficace degli altri nel migliorare la soluzione iniziale. Analizzando nello specifico i risultati ottenuti attraverso gli algoritmi E ed F si nota come, prevedibilmente, aumentare il tempo a disposizione di ogni ri-ottimizzazione permetta di trovare più soluzioni migliorative, mentre nel caso dell'algoritmo F, il comportamento deve essere ulteriormente studiato su istanze più grandi rispetto a quelle derivanti dalle istanze NGCUT, in quanto in queste istanze non si è mai attivata la procedura che affida la ri-ottimizzazione all'algoritmo euristico piuttosto che al modello esatto. Per tale motivo le prestazioni saranno commentate in relazione ai test sulle istanze derivanti dalla classe C. L'Algoritmo D ha contribuito a migliorare la soluzione iniziale solo nel 30% dei casi, con un tempo comunque considerevolmente maggiore rispetto ad algoritmi come E e F che si sono dimostrati anche più efficaci.

Sulla base dei risultati ottenuti, è stato quindi deciso di testare nella fase successiva, con le istanze derivate dalla classe C, gli euristici B, E, e F. I due euristici C e D infatti, sono risultati meno migliorativi rispetto agli altri algoritmi che sfruttano le ri-ottimizzazioni delle soluzioni a livelli.

Tabella 5.4: Risultati degli euristici A, B, C, D con le istanze NGCUT. Per ogni algoritmo è indicata la soluzione trovata entro un'ora (3600 s), ed il rispettivo tempo di risoluzione.

	Alg. A		**Alg. B**		**Alg. C**		**Alg. D**	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT01_00	20	0.10	20	10.04	21	0.14	20	0.88
NGCUT01_03	20	0.09	20	0.44	21	0.18	21	0.16
NGCUT01_05	20	0.10	20	10.91	20	0.05	20	0.28
NGCUT01_08	20	0.13	19	2.30	20	0.05	20	0.51
NGCUT01_10	20	0.11	19	3.32	20	0.06	20	0.49
NGCUT02_00	29	0.19	28	65.4	29	0.14	28	2.03
NGCUT02_03	29	0.16	28	15.33	29	0.08	29	0.62
NGCUT02_05	29	0.17	28	130.22	29	0.15	28	2.97
NGCUT02_08	29	0.23	28	118.62	29	0.26	28	2.44

Proseguimento Tabella 5.4

	Alg. A		Alg. B		Alg. C		Alg. D	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT02_10	29	0.22	28	41.19	29	0.24	28	2.58
NGCUT03_00	29	0.23	28	8.72	28	0.11	28	1.76
NGCUT03_03	29	0.21	28	12.03	29	0.30	29	0.96
NGCUT03_05	29	0.22	28	7.26	28	0.10	28	1.61
NGCUT03_08	29	0.27	28	10.19	28	0.26	28	2.38
NGCUT03_10	29	0.27	28	12.25	28	0.20	28	2.72
NGCUT04_00	12	0.10	12	0.27	12	0.23	12	0.04
NGCUT04_03	12	0.10	12	0.06	12	0.01	12	0.01
NGCUT04_05	12	0.10	11	0.20	12	0.03	12	0.01
NGCUT04_08	12	0.11	11	0.92	12	0.03	11	0.47
NGCUT04_10	12	0.11	11	1.69	12	0.13	11	0.68
NGCUT05_00	25	0.23	24	41.66	25	0.34	25	2.68
NGCUT05_03	26	0.23	24	4.08	27	1.07	26	1.85
NGCUT05_05	25	0.21	24	14.85	25	0.47	24	4.06
NGCUT05_08	25	0.27	24	10.07	24	0.32	24	2.25
NGCUT05_10	25	0.28	24	11.7	24	0.42	24	2.66
NGCUT06_00	20	0.24	20	4.17	20	0.68	20	1.72
NGCUT06_03	20	0.21	20	5.19	22	1.10	22	0.96
NGCUT06_05	20	0.22	20	4.78	20	0.13	20	1.15
NGCUT06_08	20	0.27	20	11.95	20	0.40	20	2.12
NGCUT06_10	20	0.27	20	11.97	20	0.35	20	2.49
NGCUT07_00	14	0.15	14	0.16	14	0.11	14	0.01
NGCUT07_03	18	0.15	18	0.10	18	0.01	18	0.01
NGCUT07_05	11	0.15	11	0.08	11	0.01	11	0.01
NGCUT07_08	10	0.15	9	0.16	10	0.02	10	0.02
NGCUT07_10	9	0.16	9	1.02	10	0.21	10	0.02
NGCUT08_00	35	0.30	32	40.69	35	0.69	35	0.19
NGCUT08_03	36	0.25	34	14.58	42	0.59	42	0.25
NGCUT08_05	34	0.30	32	98.55	33	2.12	34	1.43
NGCUT08_08	34	0.45	32	23.89	32	0.70	32	6.91
NGCUT08_10	33	0.42	32	34.5	32	1.07	32	12.53
NGCUT09_00	52	0.51	49	1194.2	54	2.2	51	300.00
NGCUT09_03	54	0.41	50	76.51	60	1.61	60	0.63
NGCUT09_05	51	0.48	49	1531.66	52	1.15	50	53.83
NGCUT09_08	51	1.06	49	1568.07	50	1.17	50	65.58
NGCUT09_10	51	0.82	49	1455.76	50	1.17	50	16.76

Proseguimento Tabella 5.4

	Alg. A		Alg. B		Alg. C		Alg. D	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT10_00	61	0.64	61	9.33	61	9.21	61	0.02
NGCUT10_03	81	0.34	81	4.05	81	0.02	81	0.02
NGCUT10_05	61	0.84	61	8.18	61	0.02	61	0.02
NGCUT10_08	61	1.96	59	3600.14	61	2.32	59	779.74
NGCUT10_10	61	1.67	58	2585.00	59	3.2	58	300.06
NGCUT11_00	56	1.00	52	412.23	59	8.55	59	0.88
NGCUT11_03	63	0.42	63	2.45	63	0.03	63	0.03
NGCUT11_05	54	0.85	52	15.50	52	1.33	52	0.13
NGCUT11_08	52	1.9	50	2695.18	52	2.76	51	132.83
NGCUT11_10	52	1.68	50	1745.21	51	3.45	50	135.23
NGCUT12_00	81	2.14	78	3600.14	85	4.76	80	2749.32
NGCUT12_03	88	1.35	84	60.32	86	33.81	101	1.27
NGCUT12_05	81	2.01	78	3600.18	84	7.31	81	3600.07
NGCUT12_08	81	6.16	79	3600.18	79	10.74	78	2213.09
NGCUT12_10	79	4.52	78	3600.27	78	6.54	78	147.74

Tabella 5.5: Risultati degli euristici E ed F con le istanze NGCUT. Per ogni algoritmo è indicata la soluzione trovata entro un'ora (3600 s), il rispettivo tempo di risoluzione e il TDEF utilizzato.

	Alg. E TDEF= 30s		Alg. E TDEF= 45s		Alg. E TDEF= 90s		Alg. F TDEF= 90s	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT01_00	20	0.74	20	0.74	20	0.74	20	0.74
NGCUT01_03	20	0.45	20	0.45	20	0.45	20	0.45
NGCUT01_05	20	0.29	20	0.29	20	0.29	20	0.29
NGCUT01_08	20	0.55	20	0.55	20	0.55	20	0.55
NGCUT01_10	20	0.52	20	0.52	20	0.52	20	0.52
NGCUT02_00	28	3.78	28	3.78	28	3.78	28	3.78
NGCUT02_03	29	0.62	29	0.62	29	0.62	29	0.62
NGCUT02_05	28	6.14	28	6.14	28	6.14	28	6.14
NGCUT02_08	28	5.41	28	5.41	28	5.41	28	5.41
NGCUT02_10	28	5.07	28	5.07	28	5.07	28	5.07
NGCUT03_00	28	1.76	28	1.76	28	1.76	28	1.76
NGCUT03_03	29	0.97	29	0.97	29	0.97	29	0.97
NGCUT03_05	28	1.64	28	1.64	28	1.64	28	1.64

Proseguimento Tabella 5.5

	Alg. E TDEF= 30s		Alg. E TDEF= 45s		Alg. E TDEF= 90s		Alg. F TDEF= 90s	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT03_08	28	2.45	28	2.45	28	2.45	28	2.45
NGCUT03_10	28	2.79	28	2.79	28	2.79	28	2.79
NGCUT04_00	12	0.24	12	0.24	12	0.24	12	0.24
NGCUT04_03	12	0.06	12	0.06	12	0.06	12	0.06
NGCUT04_05	11	0.20	11	0.20	11	0.20	11	0.20
NGCUT04_08	11	0.47	11	0.47	11	0.47	11	0.47
NGCUT04_10	11	0.78	11	0.78	11	0.78	11	0.78
NGCUT05_00	25	2.69	25	2.69	25	2.69	25	2.69
NGCUT05_03	26	1.81	26	1.81	26	1.81	26	1.81
NGCUT05_05	24	4.08	24	4.08	24	4.08	24	4.08
NGCUT05_08	24	2.28	24	2.28	24	2.28	24	2.28
NGCUT05_10	24	2.65	24	2.65	24	2.65	24	2.65
NGCUT06_00	20	1.73	20	1.73	20	1.73	20	1.73
NGCUT06_03	20	5.02	20	5.02	20	5.02	20	5.02
NGCUT06_05	20	1.35	20	1.35	20	1.35	20	1.35
NGCUT06_08	20	2.18	20	2.18	20	2.18	20	2.18
NGCUT06_10	20	2.59	20	2.59	20	2.59	20	2.59
NGCUT07_00	14	0.10	14	0.10	14	0.10	14	0.10
NGCUT07_03	18	0.10	18	0.10	18	0.10	18	0.10
NGCUT07_05	11	0.08	11	0.08	11	0.08	11	0.08
NGCUT07_08	9	0.73	9	0.73	9	0.73	9	0.73
NGCUT07_10	9	0.93	9	0.93	9	0.93	9	0.93
NGCUT08_00	33	30.22	32	40.43	32	40.43	32	40.43
NGCUT08_03	34	13.95	34	13.95	34	13.95	34	13.95
NGCUT08_05	33	31.43	33	46.42	33	91.44	33	91.44
NGCUT08_08	32	7.16	32	7.16	32	7.16	32	7.16
NGCUT08_10	32	12.20	32	12.20	32	12.20	32	12.20
NGCUT09_00	52	44.24	52	44.24	52	44.24	52	44.24
NGCUT09_03	51	30.76	51	45.64	50	76.40	50	76.40
NGCUT09_05	50	50.99	50	50.01	50	50.91	50	50.91
NGCUT09_08	50	44.73	50	59.65	50	62.53	50	62.53
NGCUT09_10	50	16.44	50	16.44	50	16.44	50	16.44
NGCUT10_00	61	9.63	61	9.63	61	9.63	61	9.63
NGCUT10_03	81	4.11	81	4.11	81	4.11	81	4.11
NGCUT10_05	61	8.47	61	8.47	61	8.47	61	8.47

Proseguimento Tabella 5.5

	Alg. E TDEF= 30s		**Alg. E TDEF= 45s**		**Alg. E TDEF= 90s**		**Alg. F TDEF= 90s**	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
NGCUT10_08	59	47.15	59	62.12	59	107.12	59	107.12
NGCUT10_10	59	50.93	59	65.08	59	110.28	59	110.28
NGCUT11_00	59	30.90	59	45.97	59	90.93	59	90.93
NGCUT11_03	63	2.48	63	2.48	63	2.48	63	2.48
NGCUT11_05	52	15.59	52	15.59	52	15.59	52	15.59
NGCUT11_08	52	66.21	51	88.16	51	133.17	51	133.17
NGCUT11_10	51	77.68	51	106.68	50	134.38	50	134.38
NGCUT12_00	80	68.84	80	99.04	80	189.03	80	189.03
NGCUT12_03	99	31.52	92	46.5	84	60.24	84	60.24
NGCUT12_05	79	81.17	79	94.53	79	139.18	79	139.18
NGCUT12_08	79	154.85	78	348.6	78	605.98	78	605.98
NGCUT12_10	78	127.77	78	147.99	78	145.05	78	148.24

Tabella 5.6: Riassunto delle prestazioni di modelli esatti adattati ed algoritmi euristici in termini di miglior soluzione trovata ed eventuale ottimalità con le istanze derivate dalla classe NGCUT. Si denota con "x*" se la soluzione è dichiarabile ottima mentre "x" indica che la soluzione può solo essere considerata come miglior soluzione per la data istanza.

			Mod. esatti			**Algoritmi euristici**							
Istanza	**Best**	**Opt**	**A**	**B**	**C**	**A**	**B**	**C**	**D**	**E 30s**	**E 45s**	**E 90s**	**F 90s**
NGCUT01_00	20	sì		x*	x	x	x*		x	x	x	x	x
NGCUT01_03	20	sì		x*	x	x	x*			x*	x*	x*	x*
NGCUT01_05	20	sì	x	x*	x	x	x*	x	x	x	x	x	x
NGCUT01_08	19	sì		x*			x*						
NGCUT01_10	19	sì		x*			x*						
NGCUT02_00	28	sì		x*			x*		x*	x*	x*	x*	x*
NGCUT02_03	28	sì		x*			x*						
NGCUT02_05	28	sì		x*			x*		x*	x*	x*	x*	x*
NGCUT02_08	28	sì		x*			x*		x*	x*	x*	x*	x*
NGCUT02_10	28	sì		x*			x*		x*	x*	x*	x*	x*
NGCUT03_00	28	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT03_03	28	sì		x*			x*						
NGCUT03_05	28	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT03_08	28	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT03_10	28	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT04_00	12	sì	x	x*	x*	x	x*	x	x	x*	x*	x*	x*
NGCUT04_03	12	sì	x	x*	x*	x	x*	x	x	x*	x*	x*	x*
NGCUT04_05	11	sì		x*	x*		x*			x*	x*	x*	x*
NGCUT04_08	11	sì		x*	x*		x*		x*	x*	x*	x*	x*
NGCUT04_10	11	sì		x*	x*		x*		x*	x*	x*	x*	x*
NGCUT05_00	24	sì		x*			x*						
NGCUT05_03	24	sì		x*			x*						
NGCUT05_05	24	sì		x*	x*		x*		x*	x*	x*	x*	x*
NGCUT05_08	24	sì	x*	x*	x*		x*	x*	x*	x*	x*	x*	x*
NGCUT05_10	24	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT06_00	20	sì	x*	x*		x*	x*	x*	x*	x*	x*	x*	x*
NGCUT06_03	20	sì		x*	x*	x*	x*			x*	x*	x*	x*
NGCUT06_05	20	sì	x*	x*		x*	x*	x*	x*	x*	x*	x*	x*
NGCUT06_08	20	sì	x*	x*		x*	x*	x*	x*	x*	x*	x*	x*
NGCUT06_10	20	sì	x*	x*		x*	x*	x*	x*	x*	x*	x*	x*
NGCUT07_00	14	sì	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Proseguimento Tabella 5.6

			Mod. esatti			Algoritmi euristici							
Istanza	**Best**	**Opt**	**A**	**B**	**C**	**A**	**B**	**C**	**D**	**E 30s**	**E 45s**	**E 90s**	**F 90s**
NGCUT07_03	18	sì	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
NGCUT07_05	11	sì	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
NGCUT07_08	9	sì		x*	x*		x*			x*	x*	x*	x*
NGCUT07_10	9	sì		x*	x*	x*	x*			x*	x*	x*	x*
NGCUT08_00	32	sì		x*			x*				x*	x*	x*
NGCUT08_03	34	sì		x*	x		x*			x*	x*	x*	x*
NGCUT08_05	32	sì		x*			x*						
NGCUT08_08	32	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT08_10	32	sì	x*	x*			x*	x*	x*	x*	x*	x*	x*
NGCUT09_00	49	sì		x*			x*						
NGCUT09_03	50	sì		x*	x		x*					x*	x*
NGCUT09_05	49	sì		x*			x*						
NGCUT09_08	49	sì					x*						
NGCUT09_10	49	sì					x*						
NGCUT10_00	61	sì	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
NGCUT10_03	81	sì	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
NGCUT10_05	61	sì	x	x*	x	x	x*	x	x	x*	x*	x*	x*
NGCUT10_08	58	sì		x*									
NGCUT10_10	58	sì					x*		x*				
NGCUT11_00	52	sì		x*			x*						
NGCUT11_03	63	sì	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*
NGCUT11_05	52	sì	x	x*	x		x*	x	x	x*	x*	x*	x*
NGCUT11_08	50	sì					x*						
NGCUT11_10	50	sì					x*		x*			x*	x*
NGCUT12_00	78	no					x						
NGCUT12_03	84	sì		x*	x		x*					x	x
NGCUT12_05	78	no		x			x						
NGCUT12_08	78	no							x		x	x	x
NGCUT12_10	78	no	x				x	x	x	x	x	x	x

Risultati istanze C

Le istanze derivanti dalla classe qui analizzata sono in media considerevolmente più grandi di quelle della classe NGCUT. Per questo motivo non è stato possibile testare 30 delle 75 istanze (in particolare, da c4-p1 a c5-p3) per alcuni degli algoritmi e per il Modello B. Il tentativo di risoluzione di quest'ultime entro tempi compatibli con questa ricerca sarebbero stati comunque vani, come mostrato dai risultati ottenuti attraverso l'applicazione dei modelli esatti e degli algoritmi euristici sulle istanze immediatamente precedenti; questi esperimenti, infatti, evidenziano un notevole peggiormanto della qualità delle soluzioni trovate, o comunque la scarsa capacità di ottenere ri-ottimizzazioni proficue. Per queste ragioni, le suddette istanze sono state analizzate solo attraverso il Modello A adattato, l'Algoritmo A e l'Algoritmo F. Pertanto, nel presentare i risultati di quest'ultimi modelli esatti ed algoritmi, saranno riportate le statistiche sia nel caso in cui si considerino tutte le 75 istanze derivate dalla classe C, che nel caso si considerino solo le prime 45. Questo accorgimento permette di confrontare i risultati in maniera più coerente.

Modelli esatti. I risultati dell'applicazione modelli esatti adattati sono riassunti nella Tabella 5.7 e sono ulteriormente esplicitate le occasioni di raggiungimento delle soluzioni migliori o ottime nella Tabella 5.10. Come anticipato, sono stati testati solo il Modello A adattato ed il Modello B adattato perché giudicati più promettenti sulla base dei risultati ottenuti nella prima fase di test. Le prestazioni in termini assoluti possono essere così riassunte:

- Il modello A adattato ha raggiunto la miglior soluzione conosciuta in 24 istanze sulle 75 testate (32%), con un tempo di calcolo medio di 45.83s ed un *gap* medio del 4.35%. Di queste, solo 1 soluzione è decretabile come ottima per raggiungimento del valore di *lower bound* dell'istanza.

  Prendendo in considerazione solo le prime 45 istanze invece, si è arrivati alla miglior soluzione conosciuta in 7 casi (15.6%), con un tempo di calcolo medio di 2.53s ed un *gap* medio del 5.36%;

- Il modello B adattato giunge alla miglior soluzione in 32 istanze delle 45 sulle quali è stato possibile testarlo (71.1%), con un tempo medio di calcolo

di 1650.66s ed un *gap* medio del 163.99%. Di queste, tutte le 32 soluzioni sono decretabili come ottime.

A testimonianza di quanto anticipato, il *gap* medio del Modello B adattato è molto alto a causa dei risultati ottenuti nelle 15 istanze da c3-p1 a c3-p3, mostrando un repentino peggioramento delle prestazioni del modello all'aumentare delle dimensioni delle istanze analizzate sia in termini di tempo che di qualità delle soluzioni trovate.

Tabella 5.7: Risultati dei modelli adattati A e B con le istanze C. Per ogni modello è indicata la soluzione trovata entro un'ora (3600 s), ed il rispettivo tempo di risoluzione. Per il modello A adattato è presente anche l'indicazione riguardo al numero di livelli di cui la soluzione è composta. (*) Soluzione euristica fornita dal solver, non è stato possibile risalire al packing preciso.

	Modello A ad.			**Modello B ad.**	
Istanza	**Sol.**	**Liv.**	**T (s)**	**Sol.**	**T (s)**
c1-p1_00	21	4	0.18	20	139.23
c1-p1_06	22	3	0.52	20	101.62
c1-p1_10	21	3	0.11	20	326.52
c1-p1_16	21	6	0.12	20	448.75
c1-p1_20	21	6	0.12	20	505.03
c1-p2_00	21	4	0.12	20	248.87
c1-p2_06	21	3	0.34	20	144.08
c1-p2_10	21	5	0.18	20	236.62
c1-p2_16	21	6	0.09	20	527.06
c1-p2_20	21	7	0.08	20	293.28
c1-p3_00	21	4	0.08	20	268.30
c1-p3_06	22	3	0.87	21	59.58
c1-p3_10	21	5	0.06	20	152.89
c1-p3_16	21	5	0.15	20	281.69
c1-p3_20	21	5	0.09	20	358.26
c2-p1_00	16	3	1.28	15	603.17
c2-p1_06	17	2	2.02	15	491.54
c2-p1_10	16	4	1.45	15	1526.61
c2-p1_16	16	5	0.52	15	2851.71
c2-p1_20	16	5	0.46	15	3242.33
c2-p2_00	16	4	0.46	15	414.01
c2-p2_06	19	2	0.53	18	95.18

Proseguimento Tabella 5.7

	Modello A ad.			Modello B ad.	
Istanza	**Sol.**	**Liv.**	**T (s)**	**Sol.**	**T (s)**
c2-p2_10	16	3	1.51	15	597.07
c2-p2_16	16	4	0.27	15	1517.20
c2-p2_20	16	5	0.91	15	1393.45
c2-p3_00	16	3	2.08	15	335.73
c2-p3_06	17	2	0.53	16	295.90
c2-p3_10	16	3	0.89	15	941.07
c2-p3_16	16	5	0.41	15	1489.30
c2-p3_20	16	6	0.21	15	1505.56
c3-p1_00	37	2	4.11	271*	3624.06
c3-p1_06	46	1	0.91	54	3607.69
c3-p1_10	33	2	4.07	266*	3649.87
c3-p1_16	31	4	5.03	271*	3648.72
c3-p1_20	31	4	2.16	269*	4024.16
c3-p2_00	33	3	3.34	191*	4395.00
c3-p2_06	38	2	5.17	32	2143.25
c3-p2_10	32	3	3.76	191*	3608.31
c3-p2_16	31	4	4.80	184*	3709.55
c3-p2_20	31	5	5.26	191*	3915.19
c3-p3_00	33	2	1.29	214*	3629.20
c3-p3_06	37	2	0.75	36	2059.73
c3-p3_10	32	2	45.19	216*	3639.18
c3-p3_16	31	3	9.96	205*	3615.36
c3-p3_20	31	4	1.31	209*	3618.92
c4-p1_00	63	5	75.58	-	-
c4-p1_06	69	2	28.96	-	-
c4-p1_10	63	4	59.56	-	-
c4-p1_16	61	5	54.45	-	-
c4-p1_20	61	8	32.35	-	-
c4-p2_00	62	5	91.21	-	-
c4-p2_06	68	3	600.05	-	-
c4-p2_10	62	4	61.06	-	-
c4-p2_16	61	5	41.31	-	-
c4-p2_20	61	9	5.71	-	-
c4-p3_00	69	3	28.05	-	-
c4-p3_06	81	2	8.96	-	-
c4-p3_10	65	5	30.48	-	-

Proseguimento Tabella 5.7

	Modello A ad.			Modello B ad.	
Istanza	**Sol.**	**Liv.**	**T (s)**	**Sol.**	**T (s)**
c4-p3_16	62	6	19.70	-	-
c4-p3_20	61	7	8.90	-	-
c5-p1_00	93	6	69.65	-	-
c5-p1_06	105	4	138.28	-	-
c5-p1_10	94	6	230.45	-	-
c5-p1_16	91	9	205.11	-	-
c5-p1_20	91	9	37.02	-	-
c5-p2_00	93	6	151.55	-	-
c5-p2_06	96	5	71.52	-	-
c5-p2_10	92	7	72.38	-	-
c5-p2_16	91	9	71.89	-	-
c5-p2_20	91	10	28.34	-	-
c5-p3_00	97	6	176.59	-	-
c5-p3_06	105	3	132.62	-	-
c5-p3_10	95	5	600.15	-	-
c5-p3_16	91	9	141.37	-	-
c5-p3_20	91	10	50.59	-	-

Algoritmi euristici. I risultati dell'applicazione degli algoritmi euristici A, B, E e F sono riassunti in Tabella 5.8 e Tabella 5.9 e le occasioni in cui hanno portato alla miglior soluzione o alla soluzione ottima sono visibili dettagliatamente in Tabella 5.10. I risultati dei test computazionali degli algoritmi euristici con le istanze derivanti dalla classe C sono presentati di seguito:

- L'Algoritmo A è arrivato alla miglior soluzione disponibile in 23 istanze su 75 (30.7%), di cui in 6 casi con soluzione dichiarabile ottima per eguaglianza con il valore di LB dell'istanza. Il tempo medio di calcolo è stato di $43.05s$. Il *gap* è stato del 3.38%.

  Prendendo in considerazione solo le prime 45 istanze invece, l'algoritmo è arrivato alla miglior soluzione in 7 casi (15.5%), con un tempo medio di calcolo di $2.40s$ ed un *gap* medio del 5.08%;

- l'algoritmo B è arrivato alla miglior soluzione in 45 istanze delle 45 sulle quali è stato testato (100%), di cui in 38 casi con soluzione ottimale. Il tempo medio di calcolo è stato di 1046.67s. Il *gap* è stato dello 0%;

- l'Algoritmo E ($TDEF = 30s$) è arrivato alla miglior soluzione in 11 istanze delle 45 sulle quali è stato testato (24.4%), di cui in 4 casi con soluzione ottimale. Il tempo medio di calcolo è stato di 33.11s. Il *gap* è stato dello 4.60%;

- l'Algoritmo E ($TDEF = 45s$) è arrivato alla miglior soluzione in 11 istanze delle 45 sulle quali è stato testato (24.4%), di cui in 4 casi con soluzione ottimale. Il tempo medio di calcolo è stato di 41.92s. Il *gap* è stato dello 4.60%;

- l'Algoritmo E ($TDEF = 90s$) è arrivato alla miglior soluzione in 12 istanze delle 45 sulle quali è stato testato (26.7%), di cui in 5 casi con soluzione ottimale. Il tempo medio di calcolo è stato di 62.67s. Il *gap* è stato dello 4.20%;

- l'Algoritmo F ($TDEF = 90s$) è arrivato alla miglior soluzione in 37 istanze su 75 sulle quali è stato testato (49.3%), di cui in 6 casi con soluzione ottimale. Il tempo medio di calcolo è stato di 176.63s. Il *gap* è stato dello 2.59%.

  Prendendo in considerazione solo le prime 45 istanze invece, l'algoritmo è arrivato alla miglior soluzione in 12 casi (26.6%), con un tempo medio di calcolo di 46.99s ed un *gap* medio del 4.13%.

Anche in questo caso, per completare l'analisi dei risultati degli algoritmi euristici che sfruttano le ri-ottimizzazioni è utile andare a controllare quante volte sono risultati migliorativi sulle soluzioni ottenute tramite il modello A a livelli:

- L'Algoritmo B è stato migliorativo in 38 istanze su 45 (84,4%);

- l'algorimo E ($TDEF = 30s$) in 7 istanze su 45 (15.6%);

- l'algorimo E ($TDEF = 45s$) in 7 istanze su 45 (15.6%);

- l'algorimo E ($TDEF = 90s$) in 9 istanze su 45 (20%);

- l'algorimo F ($TDEF = 90s$) in 22 istanze su 75 (29.3%), considerando solo le prime 45 istanze la soluzione iniziale è stat migliorata 10 volte (22.2%).

L'Algoritmo B si conferma quello che, dove applicabile, porta alla maggior probabilità di migliorare la soluzione iniziale entro il tempo limite, per contro però il tempo di risoluzione medio è molto più alto. Considerando infatti le prime 45 istanze, l'Algoritmo B è stato 436 volte più lento dell'Algoritmo A, circa 17 volte più lento dell'Algortimo E ($TDEF = 90s$) e circa 22 volte più lento dell'Algoritmo F ($TDEF = 90s$). Inoltre, si nota come all'aumentare delle dimensioni delle istanze, le difficoltà da parte dell'Algoritmo B a migliorare la soluzione a livelli fornita in partenza aumentino, ed è ragionevole supporre che pochi o nessun miglioramento sarebbe stato apportato, entro il tempo limite, alle istanze di dimensioni ben superiori (da c4-p1 a c5-p3) che non è stato possibile testare. Confrontando invece i risultati ottenuti attraverso l'Algoritmo E e l'Algoritmo F, si nota come nonstante il secondo si affidi alla procedura euristica per le ri-ottimizzazioni di dimensioni troppo grandi, la percentuale di soluzioni iniziali migliorate risulta ben superiore a quella dell'Algoritmo E con lo stesso $TDEF$, questo sia nel caso in cui si considerino solo le prime 45 istanze, che nel caso in cui si considerino tutte. Se confrontati sullo stesso numero di istanze, la prestazione in termini di soluzioni migliorate è superiore nel caso dell'Algoritmo F, con 10 istanze in cui si è avuto un miglioramento, contro le 9 dell'Algoritmo E ($TDEF = 90s$). Inoltre, l'algortimo che sfrutta anche la procedura euristica si rivela, a parità di $TDEF$, 1.3 volte più veloce rispetto all'Algoritmo E.

Infine, un confronto tra i risultati dell'Algoritmo A e dell'Algoritmo F. Entrambe gli algoritmi sono stati sfruttati per calcolare una soluzione di tutte le istanze scelte per la classe C, è quindi possibile confrontarli sfruttando le statistiche che comprendono tutti i risultati generati. L'algoritmo F si conferma anche in questo caso superiore in termini di qualità delle soluzioni trovate, con un *gap* medio percentuale che è meno della metà di quello dell'Algoritmo A. In termini di velocità l'Algoritmo A risulta circa 4.1 volte più veloce, ma l'incremento del tempo medio necessario alla risoluzione di un'istanza al crescere delle dimensioni delle stesse suggerisce di indagare più a fondo la relazione tra la velocità di risoluzione e le dimensioni delle istanze, in quanto istanze di dimensioni ancora superiori potrebbero

rendere l'Algoritmo F quello preferibile anche in termini di velocità.

Tabella 5.8: Risultati degli euristici A e B con le istanze della classe C. Per ogni algoritmo è indicata la soluzione trovata entro un'ora (3600 s), ed il rispettivo tempo di risoluzione.

	Alg. A		**Alg. B**	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**
c1-p1_00	22	0.37	20	39.35
c1-p1_06	21	0.33	20	51.57
c1-p1_10	22	0.36	20	123.19
c1-p1_16	21	0.38	20	374.45
c1-p1_20	21	0.41	20	131.92
c1-p2_00	21	0.43	20	132.85
c1-p2_06	21	0.35	20	82.68
c1-p2_10	21	0.36	20	176.76
c1-p2_16	21	0.42	20	216.13
c1-p2_20	21	0.41	20	240.8
c1-p3_00	21	0.37	20	94.68
c1-p3_06	22	0.35	21	21.39
c1-p3_10	22	0.38	20	146.07
c1-p3_16	21	0.39	20	249.45
c1-p3_20	21	0.40	20	96.10
c2-p1_00	16	1.04	15	74.87
c2-p1_06	16	0.97	15	443.83
c2-p1_10	16	1.09	15	331.86
c2-p1_16	16	1.16	15	1807.15
c2-p1_20	16	1.19	15	2432.41
c2-p2_00	16	1.07	15	469.66
c2-p2_06	18	0.91	18	3.49
c2-p2_10	18	0.92	15	183.14
c2-p2_16	16	1.17	15	572.14
c2-p2_20	16	1.16	15	1788.67
c2-p3_00	16	1.03	15	84.30
c2-p3_06	16	0.94	16	19.30
c2-p3_10	16	1.08	15	170.81
c2-p3_16	16	1.19	15	873.91
c2-p3_20	16	1.18	15	790.61
c3-p1_00	35	4.13	35	75.87
c3-p1_06	46	1.69	46	50.09

Proseguimento Tabella 5.8

	Alg. A		Alg. B	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**
c3-p1_10	32	5.35	31	1221.19
c3-p1_16	32	8.99	31	3600.33
c3-p1_20	32	9.06	31	3600.24
c3-p2_00	32	5.46	31	3405.55
c3-p2_06	32	2.73	32	388.18
c3-p2_10	32	5.04	31	3600.14
c3-p2_16	32	8.62	31	3600.23
c3-p2_20	31	8.39	31	3600.34
c3-p3_00	33	4.88	31	1843.25
c3-p3_06	36	1.82	36	22.79
c3-p3_10	32	4.73	30	2667.86
c3-p3_16	32	8.19	31	3600.22
c3-p3_20	32	7.03	31	3600.25
c4-p1_00	62	37.65	-	-
c4-p1_06	62	12.52	-	-
c4-p1_10	62	28.43	-	-
c4-p1_16	63	74.03	-	-
c4-p1_20	62	72.93	-	-
c4-p2_00	63	51.02	-	-
c4-p2_06	62	14.31	-	-
c4-p2_10	62	27.65	-	-
c4-p2_16	62	75.09	-	-
c4-p2_20	62	60.2	-	-
c4-p3_00	63	26.4	-	-
c4-p3_06	76	5.06	-	-
c4-p3_10	63	25.57	-	-
c4-p3_16	62	63.7	-	-
c4-p3_20	62	48.94	-	-
c5-p1_00	93	157.92	-	-
c5-p1_06	92	41.46	-	-
c5-p1_10	93	98.52	-	-
c5-p1_16	93	309.14	-	-
c5-p1_20	93	248.43	-	-
c5-p2_00	93	219.19	-	-
c5-p2_06	92	40.15	-	-
c5-p2_10	93	81.14	-	-

Proseguimento Tabella 5.8

	Alg. A		Alg. B	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**
c5-p2_16	93	278.06	-	-
c5-p2_20	92	209.91	-	-
c5-p3_00	93	165.89	-	-
c5-p3_06	93	37.44	-	-
c5-p3_10	92	91.57	-	-
c5-p3_16	93	308.39	-	-
c5-p3_20	92	210.42	-	-

Tabella 5.9: Risultati degli euristici E ed F con le istanze NGCUT. Per ogni algoritmo è indicata la soluzione trovata entro un'ora (3600 s), il rispettivo tempo di risoluzione e il TDEF utilizzato.

	Alg. E TDEF= 30s		Alg. E TDEF= 45s		Alg. E TDEF= 90s		Alg. F TDEF= 90s	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
c1-p1_00	21	4.78	21	4.78	21	4.78	21	4.78
c1-p1_06	21	8.68	21	8.68	21	8.68	21	8.68
c1-p1_10	21	5.85	21	5.85	21	5.85	21	5.85
c1-p1_16	21	4.15	21	4.15	21	4.15	21	4.15
c1-p1_20	21	4.54	21	4.54	21	4.54	21	4.54
c1-p2_00	21	3.24	21	3.24	21	3.24	21	3.24
c1-p2_06	21	2.71	21	2.71	21	2.71	21	2.71
c1-p2_10	21	3.53	21	3.53	21	3.53	21	3.53
c1-p2_16	21	5.30	21	5.30	21	5.30	21	5.30
c1-p2_20	21	4.42	21	4.42	21	4.42	21	4.42
c1-p3_00	21	2.91	21	2.91	21	2.91	21	2.91
c1-p3_06	21	8.17	21	8.17	21	8.17	21	8.17
c1-p3_10	21	3.44	21	3.44	21	3.44	21	3.44
c1-p3_16	21	4.60	21	4.60	21	4.60	21	4.60
c1-p3_20	21	4.88	21	4.88	21	4.88	21	4.88
c2-p1_00	16	19.74	16	19.74	16	19.74	16	19.74
c2-p1_06	16	32.22	16	47.07	16	92.08	16	92.08
c2-p1_10	16	9.62	16	9.62	16	9.62	16	9.62
c2-p1_16	16	11.66	16	11.66	16	11.66	16	11.66
c2-p1_20	16	11.44	16	11.44	16	11.44	16	11.44
c2-p2_00	16	8.27	16	8.27	16	8.27	16	8.27

Proseguimento Tabella 5.9

	Alg. E TDEF= 30s		**Alg. E TDEF= 45s**		**Alg. E TDEF= 90s**		**Alg. F TDEF= 90s**	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
c2-p2_06	18	3.55	18	3.55	18	3.55	18	3.55
c2-p2_10	16	20.09	16	20.09	16	20.09	16	20.09
c2-p2_16	16	38.84	16	53.27	16	98.12	16	98.12
c2-p2_20	16	22.91	16	22.91	16	22.91	16	22.91
c2-p3_00	16	37.58	16	52.49	16	97.50	16	97.50
c2-p3_06	16	19.12	16	19.12	16	19.12	16	19.12
c2-p3_10	16	34.38	16	49.66	16	94.54	16	94.54
c2-p3_16	16	12.61	16	12.61	16	12.61	16	12.61
c2-p3_20	16	9.02	16	9.02	16	9.02	16	9.02
c3-p1_00	37	34.27	37	49.34	35	76.71	35	19.18
c3-p1_06	46	32.18	46	46.03	46	51.53	46	12.31
c3-p1_10	33	33.91	33	49.27	33	94.02	32	20.90
c3-p1_16	31	106.75	31	148.12	31	238.68	31	159.78
c3-p1_20	31	125.83	31	157.93	31	211.16	31	210.63
c3-p2_00	32	75.28	32	105.35	32	196.26	32	122.78
c3-p2_06	38	35.39	38	50.29	34	95.44	34	95.44
c3-p2_10	32	78.87	32	109.16	32	198.98	32	198.97
c3-p2_16	31	124.15	31	132.02	31	176.86	31	96.41
c3-p2_20	31	129.93	31	147.87	31	154.52	31	155.13
c3-p3_00	33	31.36	33	46.53	33	91.48	33	17.15
c3-p3_06	36	23.29	36	23.29	36	23.29	36	23.29
c3-p3_10	32	74.37	32	109.50	32	134.57	32	60.25
c3-p3_16	31	99.45	31	134.44	31	240.13	31	163.06
c3-p3_20	31	122.7	31	155.55	31	235.18	31	158.01
c4-p1_00	-	-	-	-	-	-	63	220.55
c4-p1_06	-	-	-	-	-	-	62	80.86
c4-p1_10	-	-	-	-	-	-	62	304.27
c4-p1_16	-	-	-	-	-	-	61	174.05
c4-p1_20	-	-	-	-	-	-	61	184.77
c4-p2_00	-	-	-	-	-	-	62	240.56
c4-p2_06	-	-	-	-	-	-	63	767.22
c4-p2_10	-	-	-	-	-	-	62	158.76
c4-p2_16	-	-	-	-	-	-	61	168.47
c4-p2_20	-	-	-	-	-	-	61	174.93
c4-p3_00	-	-	-	-	-	-	65	135.85

Proseguimento Tabella 5.9

	Alg. E TDEF= 30s		**Alg. E TDEF= 45s**		**Alg. E TDEF= 90s**		**Alg. F TDEF= 90s**	
Istanza	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**	**Sol.**	**T (s)**
c4-p3_06	-	-	-	-	-	-	76	45.35
c4-p3_10	-	-	-	-	-	-	63	350.91
c4-p3_16	-	-	-	-	-	-	62	181.79
c4-p3_20	-	-	-	-	-	-	61	171.23
c5-p1_00	-	-	-	-	-	-	93	335.72
c5-p1_06	-	-	-	-	-	-	92	567.89
c5-p1_10	-	-	-	-	-	-	92	878.74
c5-p1_16	-	-	-	-	-	-	91	490.65
c5-p1_20	-	-	-	-	-	-	91	370.55
c5-p2_00	-	-	-	-	-	-	93	447.63
c5-p2_06	-	-	-	-	-	-	93	521.03
c5-p2_10	-	-	-	-	-	-	92	388.81
c5-p2_16	-	-	-	-	-	-	91	365.53
c5-p2_20	-	-	-	-	-	-	91	300.70
c5-p3_00	-	-	-	-	-	-	93	916.36
c5-p3_06	-	-	-	-	-	-	93	267.97
c5-p3_10	-	-	-	-	-	-	93	1126.00
c5-p3_16	-	-	-	-	-	-	91	453.89
c5-p3_20	-	-	-	-	-	-	91	341.12

Tabella 5.10: Riassunto delle prestazioni di modelli esatti adattati ed algoritmi euristici in termini di miglior soluzione trovata ed eventuale ottimalità con le istanze derivate dalla classe C. Si denota con "x*" se la soluzione è dichiarabile ottima mentre "x" indica che la soluzione può solo essere considerata come miglior soluzione per la data istanza.

			M. esatti		**Algoritmi euristici**					
Istanza	**Best**	**Opt**	**A**	**B**	**A**	**B**	**E 30s**	**E 45s**	**E 90s**	**F 90s**
c1-p1_00	20	sì		x*		x*				
c1-p1_06	20	sì		x*		x*				
c1-p1_10	20	sì		x*		x*				
c1-p1_16	20	sì		x*		x*				
c1-p1_20	20	sì		x*		x*				
c1-p2_00	20	sì		x*		x*				
c1-p2_06	20	sì		x*		x*				
c1-p2_10	20	sì		x*		x*				
c1-p2_16	20	sì		x*		x*				
c1-p2_20	20	sì		x*		x*				
c1-p3_00	20	sì		x*		x*				
c1-p3_06	21	sì		x*		x*	x	x	x	x
c1-p3_10	20	sì		x*		x*				
c1-p3_16	20	sì		x*		x*				
c1-p3_20	20	sì		x*		x*				
c2-p1_00	15	sì		x*		x*				
c2-p1_06	15	sì		x*		x*				
c2-p1_10	15	sì		x*		x*				
c2-p1_16	15	sì		x*		x*				
c2-p1_20	15	sì		x*		x*				
c2-p2_00	15	sì		x*		x*				
c2-p2_06	18	sì		x*	x*	x*	x*	x*	x*	x*
c2-p2_10	15	sì		x*		x*				
c2-p2_16	15	sì		x*		x*				
c2-p2_20	15	sì		x*		x*				
c2-p3_00	15	sì		x*		x*				
c2-p3_06	16	sì		x*	x	x*	x*	x*	x*	x*
c2-p3_10	15	sì		x*		x*				
c2-p3_16	15	sì		x*		x*				
c2-p3_20	15	sì		x*		x*				
c3-p1_00	35	sì			x*	x*			x*	x*

Proseguimento Tabella 5.10

			M. esatti		Algoritmi euristici					
Istanza	**Best**	**Opt**	**A**	**B**	**A**	**B**	**E 30s**	**E 45s**	**E 90s**	**F 90s**
c3-p1_06	46	sì	x*		x*	x*	x*	x*	x*	x*
c3-p1_10	31	sì				x*				
c3-p1_16	31	no	x			x	x	x	x	x
c3-p1_20	31	no	x			x	x	x	x	x
c3-p2_00	31	sì				x*				
c3-p2_06	32	sì		x*	x*	x*				
c3-p2_10	31	no				x				
c3-p2_16	31	no	x			x	x	x	x	x
c3-p2_20	31	no	x		x	x	x	x	x	x
c3-p3_00	31	sì				x*				
c3-p3_06	36	sì		x*	x*	x*	x*	x*	x*	x*
c3-p3_10	30	sì				x*				
c3-p3_16	31	no	x			x	x	x	x	x
c3-p3_20	31	no	x			x	x	x	x	x
c4-p1_00	62	no			x					
c4-p1_06	62	no			x					x
c4-p1_10	62	no			x					x
c4-p1_16	61	no	x							x
c4-p1_20	61	no	x							x
c4-p2_00	62	no	x							x
c4-p2_06	62	no			x					
c4-p2_10	62	no	x		x					x
c4-p2_16	61	no	x							x
c4-p2_20	61	no	x							x
c4-p3_00	63	no			x					
c4-p3_06	76	sì			x*					x*
c4-p3_10	63	no			x					x
c4-p3_16	62	no	x		x					x
c4-p3_20	61	no	x							x
c5-p1_00	93	no	x		x					x
c5-p1_06	92	no			x					x
c5-p1_10	92	no								x
c5-p1_16	91	no	x							x
c5-p1_20	91	no	x							x
c5-p2_00	93	no	x		x					x

Proseguimento Tabella 5.10

			M. esatti		Algoritmi euristici					
Istanza	**Best**	**Opt**	**A**	**B**	**A**	**B**	**E 30s**	**E 45s**	**E 90s**	**F 90s**
c5-p2_06	92	no			x					
c5-p2_10	92	no	x							x
c5-p2_16	91	no	x							x
c5-p2_20	91	no	x							x
c5-p3_00	93	no			x					x
c5-p3_06	93	no			x					x
c5-p3_10	92	no			x					
c5-p3_16	91	no	x							x
c5-p3_20	91	no	x							x

A conclusione del capitolo è utile confrontare in maniera integrata i risultati ottenuti attraverso i modelli esatti e gli algoritmi euristici; i parametri su cui si basa il confronto sono riassunti in Tabella 5.11 e Tabella 5.12. Essendo le istanze derivanti dalla classe C di dimensioni rilevanti, e più probabilmente simili ad applicazioni reali, saranno utilizzati i dati basati su quest'ultime. I risultati con le istanze derivanti dalla classe NGCUT sono comunque stati utili a determinare in anticipo quali modelli ed algoritmi valesse la pena testare su problemi più grandi e quali invece dovessero essere scartati.

Per confrontare l'efficacia è utile guardare al *gap* medio percentuale, per cui il metodo migliore, considerando solo le prime 45 istanze si è rivelato essere l'Algoritmo B con un *gap* nullo, mentre il peggiore si è rivelato essere la sua versione senza soluzione iniziale pre-fornita, ovvero il Modello B adattato. In termini di velocità di risoluzione, il metodo con il tempo medio di risoluzione minore è stato l'Algoritmo A, di poco più veloce del Modello A adattato, mentre il più lento è il Modello B adattato. Infine, in termini di numero di soluzioni *best-known* trovate, il metodo migliore è risultato essere l'Algoritmo B, mentre i peggiori il Modello A ad. e l'Algortimo A.

Considerando invece i metodi con cui sono state risolte tutte e 75 le istanze, il metodo più efficace si è rivelato l'Algoritmo F, mentre il meno efficace il Modello A adattato. Il più veloce è stato l'Algoritmo A di poco pù veloce del Modello

A adattato, mentre l'Algoritmo F è stato il più lento. Infine, l'Algoritmo F ha trovato più soluzioni *best-known*, mentre sotto questo aspetto l'Algoritmo A è stato il peggiore.

Tabella 5.11: Riassunto dei parametri prestazionali principali di tutti i metodi applicati alle prime 45 istanze derivate dalla classe C.

	Gap	**T. medio (s)**	**#best-known**
Modello A ad.	5.36%	2.53	7
Modello B ad.	163.99%	1650.56	32
Algoritmo A	5.08%	2.40	7
Algoritmo B	0%	1046.67	45
Algoritmo E (TDEF=30s)	4.60%	33.11	11
Algoritmo E (TDEF=45s)	4.60%	41.92	11
Algoritmo E (TDEF=90s)	4.20%	62.67	12
Algoritmo F (TDEF=90s)	2.13%	46.99	12

Tabella 5.12: Riassunto dei parametri prestazionali principali dei metodi applicati a tutte le istanze derivate dalla classe C.

	Gap	**T. medio (s)**	**#best-known**
Modello A ad.	4.35%	45.83	24
Algoritmo A	3.38%	43.05	23
Algoritmo F (TDEF=90s)	2.59%	176.63	37

Di seguito, in Tabella 5.13 e Tabella 5.14, sono aggiunti alcuni indicatori alle statistiche presentate in precedenza in riferimento alle prestazioni dei vari metodi con entrambe le classi di istanze. Ad esempio, viene tenuto conto in maniera esplicita del numero di volte che non è stato possibile arrivare ad una soluzione. Inoltre, è riportato un conteggio delle soluzioni ottime trovate dai modelli/algoritmi, che tenga conto solo dell'eventuale coincidenza del valore della soluzione trovata con quello della soluzione ottima dell'istanza, considerando tale anche una soluzione la cui ottimalità non possa essere effettivamente dimostrata dal relativo metodo. Questo ci permette di comparare, secondo una prospettiva più ampia, le prestazioni dei metodi riportati senza tenere conto degli aspetti teorici sottostanti l'ottimalità, che possono penalizzare alcuni algoritmi eursitici o il modello a livelli.

Tabella 5.13: Riassunto risultati dei test computazionali di tutti i metodi applicati alle istanze NGCUT. Dei due conteggi relativi alle soluzioni ottimali il primo indica le ottimalità dimostrate dal metodo, il secondo indica il numero di istanze per cui il metodo è arrivato al valore della soluzione ottima indipendentemente dalla dimostrabilità.

Analisi su 60 istanze NG-CUT	**Gap**	**T.(s)**	**#best-known**	**#opt.**	**#opt. (value)**	**#Fail.**
Modello A ad.	4.44%	0.44	24	18	23	0
Modello B ad.	2.60%	685.97	52	51	51	0
Modello C ad.	3.61%	2909.55	24	16	24	5
Algoritmo A	3.57%	0.65	18	12	18	0
Algoritmo B	0.05%	535.83	58	55	55	0
Algoritmo C	4.09%	1.92	24	18	23	0
Algoritmo D	3.16%	176.07	35	27	33	0
Algoritmo E (TDEF=30s)	1.61%	18.71	38	35	37	0
Algoritmo E (TDEF=45s)	1.36%	25.74	40	36	38	0
Algoritmo E (TDEF=90s)	1.14%	37.24	43	39	41	0
Algoritmo F (TDEF=90s)	1.14%	37.30	43	39	41	0

Tabella 5.14: Riassunto risultati dei test computazionali di tutti i metodi applicati alle istanze C. Dei due conteggi relativi alle soluzioni ottimali il primo indica le ottimalità dimostrate dal metodo, il secondo indica il numero di istanze per cui il metodo è arrivato al valore della soluzione ottima indipendentemente dalla dimostrabilità.

Analisi su 75 istanze C	**Gap**	**T.(s)**	**#best-known**	**#opt.**	**#opt. (value)**	**#Fail.**
Modello A ad.	4.35%	45.33	24	1	1	0
Modello B ad.	163.99%	1650.56	32	32	32	30
Algoritmo A	3.38%	43.05	23	6	7	0
Algoritmo B	0.00%	1046.67	45	38	38	30
Algoritmo E (TDEF=30s)	4.60%	33.11	11	4	5	30
Algoritmo E (TDEF=45s)	4.60%	41.92	11	4	5	30
Algoritmo E (TDEF=90s)	4.20%	62.67	12	5	6	30
Algoritmo F (TDEF=90s)	2.59%	176.63	37	6	7	0

Conclusioni

In questa tesi è stato studiato un problema di ottimizzazione combinatoria derivante da una applicazione di *high-performance computing*, nella quale è richiesto di schedulare un insieme di *task* su un insieme di processori paralleli, identici e consecutivi in modo da minimizzare il tempo di completamento di tutti i *task* (*makespan*). Associando a ciascun processo le possibili realizzazioni consentite (in termini di numero di processori e tempo di *scheduling*), il problema può essere modellato come una variante del problema dello *strip packing* bidimensionale nella quale gli oggetti sono deformabili.

Per questo problema, sono state introdotte tre diverse formulazioni di Programmazione Lineare Intera (PLI) derivate da formulazioni proposte per il problema classico con oggetti non deformabili. Data la complessità computazionale del problema, sono stati inoltre sviluppati alcuni algoritmi euristici e metaeuristici che consentono di gestire anche istanze di grandi dimensioni in tempi di calcolo ragionevoli.

Tutti i modelli e gli algoritmi sono stati testati computazionalmente su una serie di problemi test derivanti da istanze proposte nella letteratura scientifica per problemi di *packing* bidimensionale. Al fine di valutare sperimentalmente le prestazioni degli algortimi proposti, è stata inoltre sviluppata una procedura per il calcolo di una stima ottimistica (*lower bound*) del valore della soluzione ottima. I risultati computazionali hanno mostrato che:

- Il modello di PLI che posiziona gli oggetti a livelli (Modello A adattato) è tipicamente in grado di risolvere anche istanze di grandi dimensioni in tempi relativamente contenuti. In generale, però, il vincolo di posizionamento a

shelf non consente di determinare una soluzione ottima, ma solo una soluzione approssimata. Si osserva però che la soluzione prodotta è comunque molto vicina alla soluzione ottima, soprattutto per istanze nelle quali il numero di possibili realizzazioni associate a ciascun *task* è grande. Tale risultato si può rivelare molto utile quando un'eventuale applicazione di HPC permette svariate possibilità di deformazione, ovvero quando il grado di istruzioni da eseguire in serie in un determinato processo è abbastanza basso.

- Il modello di PLI caratterizzato da un numero pseudo-polinomiale di variabili e vincoli, indicato come Modello B adattato in questa tesi, è risultato essere il più efficace in termini di soluzione esatta del problema. Le prestazioni di questo modello sono molto condizionate dalla disponibilità di una buona soluzione di partenza. Per questo motivo, risulta conveniente applicare questo modello partendo dalla soluzione iniziale ottenuta dall'applicazione del Modello A adattato; nonostante questo, il modello rimane inadeguato per risolvere in modo esatto istanze di grandi dimensioni.

- Il terzo modello di PLI si è rivelato inefficace nel risolvere anche istanze di dimensioni contenute, nonostante sia caratterizzato da un numero di variabili e vincoli che è polinomiale nella dimensione dell'istanza.

- Tra gli algoritmi euristici, quelli che forniscono le prestazioni migliori sono l'Algoritmo A e l'Algoritmo F. Il primo sfrutta in maniera iterativa due fasi di selezione e posizionamento degli elementi, con l'obiettivo, ad ogni iterazione, di diminuire l'altezza totale del *packing* riselezionando gli elementi con alcune variazioni. L'algoritmo F fa invece parte del gruppo di algoritmi che possono sfruttare al loro interno dei modelli PLI per risolvere alcuni sotto-problemi e per questo richiede l'uso di un *solver* commerciale. Tra i due, l'algoritmo più veloce nell'arrivare ad una soluzione per una data istanza è stato l'Algoritmo A; l'Algoritmo F ha performato meglio in termini di qualità delle soluzioni trovate per via del maggior numero di soluzioni *best-known*, ed il minor *gap* medio. Inolte, entrambe i metodi si sono rivelati adatti anche a istanze di grandi dimensioni, a differenza degli algortmi B ed E.

Ovviamente, le prestazioni dei vari algoritmi sono influenzate dal tempo di calcolo a disposizione; le considerazioni sopra enunciate valgono, con qualche approssimazione, con tutti i diversi tempi di calcolo che sono stati testati nella nostra analisi.

Questo lavoro di tesi apre alcune interessanti direzioni di ricerca per il futuro:

- Dal punto di vista teorico, è interessante capire il livello di approssimazione nel caso peggiore del modello a *shelf* rispetto alla soluzione ottima del problema.

- L'analisi computazionale richiede di valutare il comportamento dei modelli di PLI e degli algoritmi eursitici per istanze caratterizzate da un elevato numero di processori. Questa analisi richiede una potenza computazionale non trascurabile e potrebbe beneficiare di algoritmi di calcolo paralleli.

- Altro possibile sviluppo futuro può essere la richiesta della ulteriore "risorsa condivisa" che si trova nelle ricerche sul *multi-processor scheduling* che quasi mai presentano un approccio basato sulle tecniche risolutive tipiche dei problemi di *packing*. Tra gli esempi possibili, si può pensare di aggiungere l'ulteriore richiesta di tenere conto della disponibilità di memoria necessaria a gestire correttamente i processi in via di risoluzione, dell'energia necessaria al funzionamento del sitema, o della larghezza della banda in caso i dati debbano essere trasmessi tra centri diversi.

- Dal punto di vista applicativo, è interessante adattare gli algoritmi in modo da gestire alcuni possibili vincoli aggiuntivi. Ad esempio, in alcune applicazioni, tutti i *task* dello stesso cliente devono essere schedulati in modo analogo (cioè utilizzando lo stesso numero di processori), mentre in altre applicazioni alcuni *task* possono essere schedulati solo su alcuni processori. Questi vincoli, difficilmente modellabili in una formulazione di PLI, possono essere gestiti attraverso algoritmi euristici, per consentire di ottenere soluzioni che rispecchino il più possibile le esigenze dell'applicazione in questione.

Bibliografia

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities.

[Baker et al., 1980] Baker, B. S., Coffman, Jr., E. G., and Rivest, R. L. (1980). Orthogonal Packings in Two Dimensions. *SIAM Journal on Computing*, 9(4):846–855.

[Beasley, 1985] Beasley, J. E. (1985). An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure. *Operations Research*, 33(1):49–64.

[Chen et al., 1995] Chen, C., Lee, S., and Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76.

[Costa et al., 2017] Costa, G., Delorme, M., Iori, M., Malaguti, E., and Martello, S. (2017). Training software for orthogonal packing problems. *Computers & Industrial Engineering*, 111:139–147.

[Delorme et al., 2016] Delorme, M., Iori, M., and Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20.

[Garey and Graham, 1975] Garey, M. R. and Graham, R. L. (1975). Bounds for Multiprocessor Scheduling with Resource Constraints. *SIAM Journal on Computing*, 4(2):187–200.

[Garey and Johnson, 1981] Garey, M. R. and Johnson, D. S. (1981). Approximation Algorithms for Bin Packing Problems: A Survey. In Ausiello, G. and

Lucertini, M., editors, *Analysis and Design of Algorithms in Combinatorial Optimization*, pages 147–172. Springer Vienna, Vienna.

[Gilmore and Gomory, 1965] Gilmore, P. C. and Gomory, R. E. (1965). Multistage Cutting Stock Problems of Two and More Dimensions. *Operations Research*, 13(1):94–120.

[Hopper and Turton, 2001] Hopper, E. and Turton, B. (2001). An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1):34–57.

[Iori et al., 2021] Iori, M., de Lima, V. L., Martello, S., Miyazawa, F. K., and Monaci, M. (2021). Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, 289(2):399–415.

[Iori and Martello, 2010] Iori, M. and Martello, S. (2010). Routing problems with loading constraints. *TOP*, 18(1):4–27.

[Janiak et al., 2007] Janiak, A., Janiak, W., and Lichtenstein, M. (2007). Resource Management in Machine Scheduling Problems: A Survey. *Decision Making in Manufacturing and Services*, 1(2):59–89.

[Kling et al., 2017] Kling, P., Mäcker, A., Riechers, S., and Skopalik, A. (2017). Sharing is Caring: Multiprocessor Scheduling with a Sharable Resource. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 123–132, Washington DC USA. ACM.

[Kwon and Lee, 2015] Kwon, B. and Lee, G. M. (2015). Spatial scheduling for large assembly blocks in shipbuilding. *Computers & Industrial Engineering*, 89:203–212.

[Lodi et al., 2011] Lodi, A., Martello, S., Monaci, M., Cicconetti, C., Lenzini, L., Mingozzi, E., Eklund, C., and Moilanen, J. (2011). Efficient Two-Dimensional Packing Algorithms for Mobile WiMAX. *Management Science*, 57(12):2130–2144.

[Lodi et al., 2004] Lodi, A., Martello, S., and Vigo, D. (2004). Models and Bounds for Two-Dimensional Level Packing Problems. *Journal of Combinatorial Optimization*, 8(3):363–379.

[Lodi and Monaci, 2003] Lodi, A. and Monaci, M. (2003). Integer linear programming models for 2-staged two-dimensional Knapsack problems. *Mathematical Programming*, 94(2-3):257–278.

[Martello, 2014] Martello, S. (2014). Two-dimensional packing problems in telecommunications. *Pesquisa Operacional*, 34(1):31–38.

[Melega et al., 2018] Melega, G. M., de Araujo, S. A., and Jans, R. (2018). Classification and literature review of integrated lot-sizing and cutting stock problems. *European Journal of Operational Research*, 271(1):1–19.

[Silva et al., 2016] Silva, E., Oliveira, J. F., and Wäscher, G. (2016). The pallet loading problem: a review of solution methods and computational experiments: The pallet loading problem: a review of solution methods and computational experiments. *International Transactions in Operational Research*, 23(1-2):147–172.

[Strecker and Hennig, 2009] Strecker, T. and Hennig, L. (2009). Automatic Layouting of Personalized Newspaper Pages. In Fleischmann, B., Borgwardt, K.-H., Klein, R., and Tuma, A., editors, *Operations Research Proceedings 2008*, pages 469–474. Springer Berlin Heidelberg, Berlin, Heidelberg.