

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Generazione di istanze a scarto nullo per il problema del 2D Multiple Bin Packing con tagli a ghigliottina

Candidato:
Edoardo Ballerini

Relatore:
Prof. Fabrizio Marinelli

Anno Accademico 2025-2026



UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Generazione di istanze a scarto nullo per il problema del 2D Multiple Bin Packing con tagli a ghigliottina

Candidato:
Edoardo Ballerini

Relatore:
Prof. Fabrizio Marinelli

Anno Accademico 2025-2026

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

Ringraziamenti

Ringrazio innanzitutto il Prof. Fabrizio Marinelli per la sua disponibilità e per l'entusiasmo contagioso e l'interesse che ha dimostrato per l'argomento.

Ringrazio la mia famiglia per il sostegno che non mi ha mai fatto mancare. Un ringraziamento particolare va a mia madre, per essere il motore irrefrenabile che mi spinge sempre un passo più in là.

Ancona, Ottobre 2026

Edoardo Ballerini

Sommario

Il lavoro riguarda il problema del 2D *Multiple Bin Packing* con tagli a ghigliottina, nel quale pezzi rettangolari con domanda assegnata devono essere ricavati da pannelli di uno o più formati minimizzando il materiale impiegato. Il problema, ricorrente nel taglio industriale di pannelli, è NP-difficile in senso forte e si affronta di norma con metodi euristici, la cui distanza dall'ottimo resta in generale incognita.

Per misurarla è stato sviluppato un generatore di istanze a scarto nullo. Costruendo l'istanza a partire da una suddivisione completa dei pannelli in pezzi, il generatore garantisce l'esistenza di una soluzione priva di scarto, cosicché lo scarto di qualunque soluzione coincide con la sua distanza dall'ottimo. Le istanze generate sono state impiegate per valutare un'euristica industriale per il taglio di pannelli in legno, con un disegno sperimentale fattoriale completo di 3072 combinazioni di caratteristiche dell'istanza e parametri dell'algoritmo, analizzato con strumenti statistici non parametrici.

I risultati mostrano che la qualità delle soluzioni dipende soprattutto dalla domanda e che i parametri dell'algoritmo con l'effetto più ampio incidono principalmente sulle istanze a domanda bassa, le più difficili. Lo sforzo di ricerca maggiore recupera circa un quinto della distanza dall'ottimo a fronte di un tempo di calcolo circa venti volte superiore, mentre lo schema di taglio a due stadi comporta in mediana una perdita inferiore a un punto percentuale rispetto a quello a tre stadi. La sovraproduzione dei pezzi, infine, riduce lo scarto riportato dall'algoritmo ma aumenta il materiale impiegato: misurata con una metrica che tiene conto dei pezzi in eccesso, essa peggiora la qualità delle soluzioni.

Indice

Introduzione **1**
Contesto e obiettivo 1
Contributo 1
Risultati principali 2
Struttura del documento 2

1. Il problema del 2D Multiple Bin Packing con tagli a ghigliottina **5**
1.1. I problemi di taglio e di *packing* 5
1.2. Definizione del problema 7
1.3. Vincoli di taglio 9
1.4. Modello matematico 10
1.5. Complessità computazionale 12
1.6. Rassegna della letteratura 13
1.7. Istanze di test e istanze con ottimo noto 14

2. Il software di ottimizzazione **17**
2.1. Il problema affrontato dal *solver* 17
2.2. L'euristica a correzione sequenziale dei valori 19
2.3. Generazione degli schemi di taglio 21
2.4. Soluzioni non dominate e scelta della soluzione 23
2.5. Le fasi di ottimizzazione 24
2.6. Formato di input e output 25
2.7. Ruolo nel contesto della tesi 28

3. Generazione di istanze a scarto nullo **29**
3.1. Definizione di istanza a scarto nullo 29
3.2. Parametri del generatore 30
3.3. Algoritmo di generazione 31
3.3.1. Ciclo principale 31
3.3.2. Generazione del primo schema di taglio 32
3.3.3. Generazione degli schemi di taglio successivi 33
3.3.4. Taglio monodimensionale 33
3.4. Formato di output 34

4. Risultati computazionali **37**
4.1. Disegno sperimentale 37

4.2. Metriche di valutazione 39
4.3. Strumenti e metodi statistici 40
4.3.1. Statistiche descrittive e *boxplot* 40
4.3.2. Correlazione di Spearman 41
4.3.3. Test di Wilcoxon dei ranghi con segno 41
4.3.4. Test di Mann-Whitney 42
4.3.5. Test di Kruskal-Wallis 42
4.3.6. Frontiera di Pareto 43
4.3.7. Impianto metodologico 43
4.4. Risultati 43
4.4.1. Effetti principali e ordinamento dei fattori 45
4.4.2. Le leve più forti del *tuning* incidono soprattutto sulle istanze difficili 45
4.4.3. La variabilità tra istanze 47
4.4.4. La sovraproduzione riduce lo scarto riportato, non quello effettivo 48
4.4.5. Lo sforzo di ricerca recupera una frazione stabile della distanza dall'ottimo 49
4.4.6. Il prezzo della semplicità del taglio a due stadi 52
4.4.7. Frontiera di Pareto e configurazione raccomandata 52
4.4.8. L'effetto della configurazione di pannelli 55
4.5. Sintesi e raccomandazioni 55

Conclusioni e sviluppi futuri **59**
Riepilogo del lavoro 59
Considerazioni sui risultati 60
Limiti 60
Sviluppi futuri 62

A. Codice sorgente **63**
A.1. `main.py` 63
A.2. `file_reader.py` 65
A.3. `panel_generator.py` 68
A.4. `cutter.py` 71
A.5. `image_generator.py` 72

B. Tabelle di dettaglio dei risultati **75**
B.1. Statistiche descrittive per fattore 75
B.2. Effetti appaiati dei fattori algoritmici 75
B.3. Configurazioni non dominate 78

Elenco delle figure

1.1. (a) Schema di taglio a ghigliottina; (b) schema che non può essere ottenuto con tagli a ghigliottina, poiché ogni taglio da un bordo all'altro del pannello attraversa almeno un pezzo. 9

1.2. Schemi di taglio a stadi con tagli del primo stadio orizzontali, indicati dalle linee spesse: (a) 2 stadi esatto; (b) 2 stadi con rifilatura, in cui le aree grigie sono lo scarto separato dalla rifilatura; (c) 3 stadi, in cui il terzo stadio divide le colonne in più pezzi. 10

2.1. Struttura di uno schema di taglio a 3 stadi con tagli del primo stadio orizzontali. Le linee spesse sono i tagli del primo stadio, che delimitano le strisce di altezza h_j; all'interno di ogni striscia i tagli del secondo stadio separano le colonne, e nella striscia centrale i tagli del terzo stadio separano le copie sovrapposte di uno stesso *part-type*. Le aree grigie sono scarto; il sotto-schema in colore più scuro riempie l'altezza residua R. Negli schemi a 2 stadi ogni colonna contiene un solo pezzo, come nella striscia inferiore. 22

2.2. Costruzione di uno schema con pretaglio. (a) Schema p: la linea tratteggiata indica la posizione x del taglio del secondo stadio più vicina a metà larghezza. (b) Il pannello è diviso in corrispondenza di x: la parte sinistra conserva i pezzi di p che non attraversano il taglio, la parte destra è liberata. (c) La parte destra è riempita con un nuovo sotto-schema, in arancione, e l'unione delle due parti fornisce lo schema p'. 23

3.1. Schema esemplificativo di un pannello generato da `generate_first` (taglio a 3 stadi). Il primo taglio suddivide l'altezza del pannello in strisce orizzontali; il secondo taglio suddivide la larghezza di ciascuna striscia in colonne; il terzo taglio suddivide l'altezza di ciascuna colonna nei singoli pezzi, etichettati con le dimensioni $w \times h$. 32

3.2. Schema esemplificativo di un pannello generato da `generate_next` (taglio a 2 stadi). La porzione superiore è prodotta da `generate_first` sull'altezza residua. Ogni striscia di riutilizzo contiene, nella porzione destra, copie identiche di un *part-type* preesistente (evidenziate con lo stesso colore) e, nella porzione sinistra, pezzi nuovi. 34

3.3. Esempio di taglio monodimensionale a scarto nullo. In `cut_min` ogni pezzo rispetta la dimensione minima $n_{\min}$ e il residuo inferiore a $n_{\min}$ viene assorbito nell'ultimo pezzo, che si ingrandisce (evidenziato in arancione). In `cut_max` ogni pezzo rispetta la dimensione massima $n_{\max}$ e il residuo finale è emesso come pezzo autonomo (evidenziato in viola). In entrambi i casi la lunghezza totale del segmento è ripartita senza scarto. 35

4.1. Correlazione di Spearman ρ tra i fattori e le metriche di output. Il colore rosso indica associazione positiva, il blu negativa. 44
4.2. Entità dell'effetto dei fattori algoritmici sullo scarto, misurata dalla mediana del valore assoluto delle differenze appaiate e calcolata separatamente per i due livelli del fattore di domanda. Il valore assoluto non ne rappresenta il segno, ossia la direzione dell'effetto. 46
4.3. Variabilità tra le istanze generate da una stessa configurazione di istanza, misurata dalla deviazione standard: (a) dello scarto per fattore di domanda, (b) dello scarto per gestione a pacco, (c) del tempo per fattore di domanda. Il tempo è in scala logaritmica. . . . 47
4.4. Effetto dello sforzo *SVC* alto rispetto a quello basso. (a) Frazione della distanza dall'ottimo recuperata, in mediana, per fascia di scarto, sullo scarto e sullo scarto effettivo. (b) Frazione recuperata nella configurazione di riferimento e nella variante di ciascun fattore, per strato; n.s. indica una differenza non significativa al test di Mann-Whitney. (c) Rapporto mediano tra i tempi con sforzo alto e basso, totale e alla migliore soluzione. 51
4.5. Prezzo della semplicità dello schema a due stadi: differenza di scarto rispetto allo schema a tre stadi, a parità di ogni altro fattore. 53
4.6. Frontiere di Pareto delle 64 configurazioni algoritmiche: (a) nel piano scarto–tempo, (b) nel piano scarto effettivo–tempo. Il tempo è in scala logaritmica. Sulla metrica robusta alla sovraproduzione le configurazioni efficienti cambiano. 54
4.7. Distribuzione dello scarto e del tempo per le quattro configurazioni di pannelli. Il tempo è in scala logaritmica. 56

Elenco delle tabelle

1.1. Tipi intermedi dei problemi di minimizzazione dell'input 7

2.1. Esempio illustrativo, con valori fittizi, di tre soluzioni non dominate per un'istanza con area domandata pari a 114 m^2. In grassetto il valore migliore di ciascun indicatore. Nessuna soluzione è migliore delle altre su tutti gli indicatori: con il peso interamente sull'area utilizzata il *solver* restituisce s_1, con il peso sullo scarto s_2, con il peso sul numero di schemi s_3. Lo scarto minimo di s_2 è ottenuto con la sovraproduzione. 24

2.2. Campi del file dell'istanza e valori assunti nelle istanze di questa tesi. 26

2.3. Parametri del *solver* rilevanti per questa tesi. Nella parte superiore i parametri variati nel disegno sperimentale, nella parte inferiore quelli mantenuti fissi. 27

4.1. Fattori controllati nell'esperimento e relativi livelli. 38

4.2. Le quattro configurazioni di pannelli impiegate. La direzione del primo taglio è verticale (V) oppure scelta casualmente (L). Il limite (w, h) è pari a un decimo della media delle dimensioni dei pannelli della configurazione. 38

4.3. Effetto appaiato medio di ciascun fattore algoritmico (livello alto meno livello basso), in punti percentuali, nello strato con sovraproduzione attiva. La direzione dell'effetto è concorde tra le due metriche per ogni fattore. 49

B.1. Statistiche descrittive dello scarto, in punti percentuali, per livello di ciascun fattore. 76

B.2. Statistiche descrittive del tempo di calcolo totale, in secondi, per livello di ciascun fattore. 77

B.3. Effetti appaiati dei fattori algoritmici sulle metriche di qualità, in punti percentuali (livello alto, indicato tra parentesi, meno livello basso), con la percentuale di coppie in cui la metrica diminuisce e il p-value nominale del test di Wilcoxon. Ogni riga è calcolata su 1536 coppie. 77

B.4. Effetti appaiati dei fattori algoritmici sul tempo di calcolo totale e sul tempo alla migliore soluzione: differenza mediana in secondi (livello alto meno livello basso), rapporto mediano tra i tempi (livello alto su livello basso), percentuale di coppie in cui il livello alto è più rapido e p-value nominale del test di Wilcoxon. Ogni riga è calcolata su 1536 coppie. 78
B.5. Configurazioni algoritmiche non dominate nel piano scarto–tempo. . 78
B.6. Configurazioni algoritmiche non dominate nel piano scarto effettivo–tempo. 79

Introduzione

Contesto e obiettivo

Il taglio di pannelli rettangolari in pezzi più piccoli è un'operazione ricorrente in numerosi settori industriali, tra cui la lavorazione del legno, del vetro, della carta e dei metalli [1, 2]. Pianificare il taglio significa decidere quali pannelli impiegare e come ripartirli nei pezzi richiesti, con l'obiettivo di ridurre il materiale impiegato e quindi lo *scarto*, cioè la parte dei pannelli che non costituisce pezzi. Questa tesi considera il problema del 2D *Multiple Bin Packing* con tagli a ghigliottina, nel quale i pezzi, raggruppati in *part-type* ciascuno con una propria domanda, devono essere ricavati da pannelli di uno o più formati mediante tagli che attraversano il pannello da un bordo a quello opposto, organizzati in un numero limitato di stadi.

Il problema è NP-difficile in senso forte (Sezione 1.5), e per le istanze di dimensioni industriali si impiegano metodi euristici, che ottengono in tempi brevi soluzioni di buona qualità senza garantirne l'ottimalità [3]. Valutare tali soluzioni è a sua volta difficile. La loro qualità è di norma misurata rispetto a un limite inferiore, e quando la soluzione non lo raggiunge non è possibile stabilire quanta parte della differenza sia dovuta all'euristica e quanta alla distanza tra il limite e l'ottimo; numerose istanze di *benchmark* della letteratura restano infatti prive di una soluzione ottima nota [4].

La tesi affronta questa difficoltà per un caso specifico: il *solver* sviluppato da Arbib et al. per il taglio di pannelli in legno su macchine industriali [5], un'euristica che costruisce gli schemi di taglio correggendo iterativamente i valori attribuiti ai *part-type*. Sulle istanze di *benchmark* dell'articolo di riferimento la soluzione del *solver* raggiunge il limite inferiore, e ne è quindi certificata l'ottimalità rispetto al materiale impiegato, in circa quattro casi su cinque [5]; nei restanti la distanza dall'ottimo non è nota. L'obiettivo del lavoro è valutare il *solver* su istanze il cui ottimo è noto per costruzione, così da misurare direttamente la qualità delle soluzioni e studiare come essa e il tempo di calcolo dipendano dalle caratteristiche dell'istanza e dalla configurazione dell'algoritmo.

Contributo

Il primo contributo è un generatore di *istanze a scarto nullo*, cioè di istanze che ammettono un piano di taglio privo di scarto. Il generatore costruisce l'istanza a partire dalla soluzione: suddivide con tagli successivi pannelli interi in pezzi compatibili con i vincoli di taglio e raccoglie i pezzi ottenuti nella domanda dell'istanza.

Il piano di taglio così costruito è ottimo, e il valore ottimo dello scarto è noto e pari a zero. Il generatore è configurabile nei formati di pannello, nei limiti dimensionali dei pezzi, nel numero di *part-type* distinti, in un fattore che moltiplica la domanda e nel tipo di taglio, a 2 o a 3 stadi. Le istanze prodotte combinano più formati di pannello, domanda per *part-type* e rotazione dei pezzi, una combinazione che le istanze con ottimo noto disponibili in letteratura non coprono [6].

Il secondo contributo è una campagna sperimentale che impiega il generatore per valutare il *solver*. Il disegno fattoriale completo e bilanciato combina quattro fattori di istanza e sei fattori algoritmici, per un totale di 3072 combinazioni, e per ciascuna misura lo scarto, che sulle istanze a scarto nullo coincide con la distanza dall'ottimo, e il tempo di calcolo. I dati sono analizzati con strumenti statistici non parametrici, che individuano dapprima i fattori più influenti e ne quantificano poi gli effetti e le interazioni.

Risultati principali

La qualità delle soluzioni dipende soprattutto dall'istanza: il fattore di domanda è quello più associato allo scarto, che diminuisce al crescere della dimensione dell'istanza, mentre il tempo di calcolo cresce soprattutto con lo sforzo di ricerca e con la dimensione dell'istanza. L'effetto dei parametri più incisivi del *solver*, la sovraproduzione e la gestione a pacco, si concentra sulle istanze a domanda unitaria, le più difficili, e diventa contenuto su quelle a domanda elevata. Tra i fattori algoritmici, la gestione a pacco e la risoluzione *greedy* del sottoproblema *knapsack* peggiorano la qualità; lo sforzo di ricerca alto recupera circa un quinto della distanza dall'ottimo, a fronte di un tempo di calcolo circa venti volte maggiore; lo schema di taglio a due stadi comporta in mediana una perdita di qualità inferiore a un punto percentuale rispetto a quello a tre stadi ed è quasi sempre più rapido. La sovraproduzione, infine, riduce lo scarto riportato dal *solver* ma aumenta il materiale impiegato: valutata con lo *scarto effettivo*, una metrica introdotta per tenere conto dei pezzi prodotti in eccesso, essa peggiora la qualità e non compare tra le configurazioni efficienti.

Struttura del documento

Il Capitolo 1 inquadra il problema nella famiglia dei problemi di taglio e di *packing*, ne fornisce la definizione formale, il modello matematico e la complessità computazionale, e passa in rassegna i metodi risolutivi e le istanze di test della letteratura. Il Capitolo 2 descrive il *solver*: il problema che affronta, l'algoritmo e i parametri impiegati negli esperimenti. Il Capitolo 3 presenta il generatore di istanze a scarto nullo, con i suoi parametri e gli algoritmi di generazione. Il Capitolo 4 illustra il disegno sperimentale, le metriche e gli strumenti statistici, e riporta i risultati con la loro interpretazione. Le conclusioni riepilogano il lavoro e ne discutono i

limiti e gli sviluppi futuri; l'Appendice A riporta il codice sorgente del generatore e l'Appendice B le tabelle di dettaglio dei risultati.

Capitolo 1.

Il problema del 2D Multiple Bin Packing con tagli a ghigliottina

Il presente capitolo introduce il problema di ottimizzazione nel cui contesto si colloca la tesi. Il problema viene dapprima inquadrato nella famiglia dei problemi di taglio e di *packing*, attraverso le tipologie proposte in letteratura; seguono la sua definizione formale, i vincoli di taglio a ghigliottina e a stadi, e la presentazione di un modello matematico e del limite inferiore che ne deriva. Dopo la discussione della complessità computazionale, il capitolo si chiude con una rassegna dei metodi risolutivi e delle istanze di test impiegate in letteratura, tra le quali quelle con ottimo noto.

1.1. I problemi di taglio e di *packing*

I problemi di taglio e di *packing* (*cutting and packing problems*) riguardano l'assegnazione di un insieme di *pezzi* a uno o più *oggetti* di dimensioni maggiori [3]. Nei problemi di taglio i pezzi sono ricavati dagli oggetti, mentre nei problemi di *packing* sono disposti al loro interno: si tratta di due letture della stessa struttura, che la letteratura raccoglie in un'unica famiglia di problemi. Il risultato di un problema di taglio è un *piano di taglio*, cioè un insieme di schemi di taglio, ciascuno associato al numero di oggetti su cui viene ripetuto; le parti degli oggetti che non costituiscono pezzi sono considerate *scarto* (*trim loss*) [3]. Problemi di questo tipo ricorrono in numerosi settori industriali, tra cui la lavorazione del legno, del vetro, della carta e dei metalli [1, 2]. Nella terminologia di questa tesi gli oggetti sono i pannelli, le classi di pezzi uguali sono i *part-type* e uno schema di taglio descrive come è ripartito un pannello.

I problemi si distinguono anzitutto per il numero di dimensioni rilevanti [3]. Nei problemi monodimensionali conta una sola dimensione, come nel taglio di un rotolo di carta in tratti di lunghezza assegnata; nei problemi bidimensionali ne contano due, come quando da una lastra di legno si ricavano pezzi dello stesso spessore della lastra; nei problemi tridimensionali, come la disposizione di colli in un contenitore, contano tutte e tre. Si parla inoltre di problemi a dimensione aperta quando una delle dimensioni dell'oggetto non è fissata ed è essa stessa da determinare. I pezzi possono

avere forma regolare, come rettangoli e cerchi, o irregolare, e possono essere disposti parallelamente ai lati dell'oggetto, cioè in modo ortogonale, oppure con orientamento qualsiasi [3]. Questa tesi tratta il caso bidimensionale con pezzi rettangolari disposti in modo ortogonale.

La varietà dei problemi ha portato a definire tipologie che li raggruppano secondo caratteristiche comuni. La prima, proposta da Dyckhoff [7], classifica un problema secondo quattro criteri: la dimensionalità; il tipo di assegnazione, che distingue i problemi nei quali gli oggetti disponibili non bastano per tutti i pezzi, e si sceglie quali pezzi ricavarne, da quelli nei quali tutti i pezzi devono essere ottenuti, e si sceglie quali oggetti impiegare; l'assortimento degli oggetti; l'assortimento dei pezzi. Ne deriva una notazione a quattro campi, in cui per esempio 2/V/I/M indica un problema bidimensionale in cui tutti i pezzi, numerosi e di molte forme diverse, devono essere ricavati da oggetti identici. Per superare alcuni limiti di questa classificazione, che non sempre consentiva di distinguere problemi simili ma con caratteristiche diverse, Wäscher, Haußner e Schumann [8] hanno proposto una tipologia migliorata, alla quale fanno riferimento le rassegne più recenti [4, 3]. Essa si basa su cinque criteri: la dimensionalità, il tipo di assegnazione, l'assortimento dei pezzi, l'assortimento degli oggetti e la forma dei pezzi. Il tipo di assegnazione corrisponde ai due casi già descritti, che la tipologia denomina *massimizzazione dell'output* e *minimizzazione dell'input*: nel primo si massimizza il valore dei pezzi ottenuti, nel secondo si minimizza il valore, il numero o la dimensione degli oggetti impiegati. La combinazione del tipo di assegnazione e dell'assortimento dei pezzi definisce i tipi di problema di base; l'aggiunta dell'assortimento degli oggetti porta ai tipi intermedi, e quella della dimensionalità e della forma dei pezzi ai tipi raffinati [8].

Tra i problemi di minimizzazione dell'input, ai quali appartiene il problema di questa tesi, la tipologia distingue i problemi di *bin packing*, nei quali i pezzi formano un assortimento fortemente eterogeneo, con pochi pezzi uguali tra loro, dai problemi di *cutting stock*, nei quali l'assortimento è debolmente eterogeneo, cioè composto da pochi *part-type* ripetuti in molte copie [8]. Ciascuno dei due si articola poi secondo l'assortimento degli oggetti, che possono essere identici, appartenere a pochi tipi diversi oppure essere tutti diversi tra loro, come accade quando si impiegano i residui di lavorazioni precedenti. Ne risultano i sei tipi intermedi della Tabella 1.1. La distinzione tra *bin packing* e *cutting stock* riguarda dunque l'assortimento dei pezzi e non la struttura del problema: con lievi modifiche della formulazione l'uno può essere riformulato come l'altro [2].

Il problema affrontato in questa tesi prevede pochi tipi di pannello, disponibili in quantità sufficiente, e una domanda da soddisfare esattamente per ciascun *part-type*; l'obiettivo è minimizzare il materiale impiegato. Secondo la tipologia di Wäscher et al., si tratta di un problema bidimensionale rettangolare di tipo *MBSBPP* quando i *part-type* sono numerosi e richiesti in poche copie, e di tipo *MSSCSP* quando sono pochi e ripetuti in molte copie; con un solo tipo di pannello si ricade nei corrispondenti casi a oggetti identici, *SBSBPP* e *SSSCSP*. Le istanze impiegate negli esperimenti,

Tabella 1.1.: Tipi intermedi dei problemi di minimizzazione dell'input con oggetti di dimensioni fissate, secondo la tipologia di Wäscher et al. [8].

Oggetti	Pezzi debolmente eterogenei	Pezzi fortemente eterogenei
identici	*Single Stock Size Cutting Stock Problem* (*SSSCSP*)	*Single Bin Size Bin Packing Problem* (*SBSBPP*)
debolmente eterogenei	*Multiple Stock Size Cutting Stock Problem* (*MSSCSP*)	*Multiple Bin Size Bin Packing Problem* (*MBSBPP*)
fortemente eterogenei	*Residual Cutting Stock Problem* (*RCSP*)	*Residual Bin Packing Problem* (*RBPP*)

nelle quali variano sia il numero di *part-type* sia la domanda (Sezione 4.1), coprono entrambe le situazioni; per questo nel seguito il problema è indicato con il nome complessivo di 2D *Multiple Bin Packing* con tagli a ghigliottina. Nella notazione a tre campi introdotta da Lodi, Martello e Vigo, come riportata in [4], il caso con oggetti identici e tagli a ghigliottina è indicato come 2BP|O|G se l'orientamento dei pezzi è fisso e come 2BP|R|G se ne è ammessa la rotazione. La sezione successiva definisce formalmente il problema.

1.2. Definizione del problema

Il problema considerato in questa tesi è definito sui seguenti dati. È dato un insieme $\mathcal{S}$ di tipi di pannello: un pannello di tipo $i \in \mathcal{S}$ ha larghezza W_i e altezza H_i, e ogni tipo è disponibile in quantità illimitata. È dato inoltre un insieme $\mathcal{R}$ di *part-type*: il *part-type* $j \in \mathcal{R}$ ha larghezza w_j, altezza h_j e domanda d_j, cioè devono esserne ottenuti esattamente d_j pezzi. La domanda è espressa per *part-type* anziché elencando i singoli pezzi: un *part-type* di domanda d_j equivale a d_j pezzi identici [4]. Tutte le dimensioni sono intere e positive, e ogni *part-type* può essere contenuto in almeno un tipo di pannello.

Uno *schema di taglio* per un pannello di tipo i stabilisce quali pezzi ne sono ricavati e in quale posizione. Il pannello è rappresentato nel piano cartesiano con i lati paralleli agli assi e il vertice in basso a sinistra nell'origine; a ciascun pezzo k assegnato al pannello corrisponde la coppia (x_k, y_k) delle coordinate del suo vertice in basso a sinistra [4]. Indicando con K l'insieme dei pezzi assegnati al pannello e, con lieve abuso di notazione, con w_k e h_k le dimensioni del *part-type* del pezzo k, lo schema è ammissibile se ogni pezzo è interamente contenuto nel pannello,

$$0 \le x_k \le W_i - w_k, \qquad 0 \le y_k \le H_i - h_k \qquad (k \in K), \tag{1.1}$$

e se due pezzi qualsiasi non si sovrappongono, cioè se le loro proiezioni sono disgiunte

su almeno uno dei due assi:

$$\begin{aligned} & [x_k, x_k + w_k) \cap [x_l, x_l + w_l) = \emptyset \\ \text{oppure} \quad & [y_k, y_k + h_k) \cap [y_l, y_l + h_l) = \emptyset \end{aligned} \qquad (k, l \in K,\ k \neq l). \tag{1.2}$$

Poiché le dimensioni sono intere, considerare soltanto coordinate intere non comporta perdita di generalità [4]. I pezzi hanno orientamento fisso: ciascuno è collocato con il lato di lunghezza w_k parallelo alla larghezza del pannello. Lo schema deve infine poter essere realizzato con tagli a ghigliottina in un numero limitato di stadi, secondo le condizioni descritte nella Sezione 1.3.

Una soluzione del problema è un piano di taglio: un insieme di schemi, ciascuno ripetuto su un certo numero di pannelli del tipo corrispondente. Il piano è ammissibile se produce esattamente d_j pezzi di ciascun *part-type* $j \in \mathcal{R}$. Indicando con n_i il numero di pannelli di tipo i impiegati dal piano, l'obiettivo è minimizzare l'area utilizzata

$$U = \sum_{i \in \mathcal{S}} W_i H_i \, n_i. \tag{1.3}$$

Poiché la domanda deve essere soddisfatta esattamente, l'area complessiva dei pezzi prodotti non dipende dal piano di taglio ed è pari a

$$A^* = \sum_{j \in \mathcal{R}} d_j \, w_j h_j. \tag{1.4}$$

Ne segue che minimizzare l'area utilizzata equivale a minimizzare lo *scarto*, definito come la frazione dell'area utilizzata che non è ripartita in pezzi:

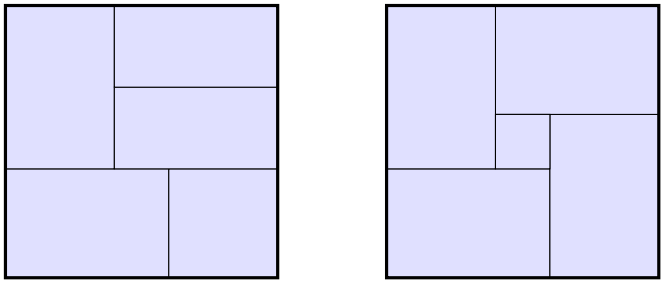
$$s = \frac{U - A^*}{U} = 1 - \frac{A^*}{U}. \tag{1.5}$$

Con un solo tipo di pannello, minimizzare l'area equivale anche a minimizzare il numero di pannelli impiegati, che è l'obiettivo classico del *bin packing* [4]; con più tipi di pannello le due grandezze non sono più equivalenti, poiché uno stesso numero di pannelli può corrispondere ad aree diverse. La definizione (1.5) coincide con quella dello scarto effettivo introdotta nella Sezione 4.4.4: le due misure differiscono soltanto quando la produzione può superare la domanda, possibilità esclusa dal problema qui definito.

Poiché i pezzi prodotti sono contenuti nei pannelli impiegati e non si sovrappongono, vale $U \geq A^*$, e lo scarto di qualunque soluzione ammissibile è non negativo. Se un'istanza ammette una soluzione con $U = A^*$, cioè una soluzione che ripartisce interamente in pezzi i pannelli impiegati, tale soluzione è ottima e il valore ottimo dello scarto è noto e pari a zero. Le istanze con questa proprietà, dette istanze a scarto nullo, sono quelle costruite dal generatore descritto nel Capitolo 3, nella variante con rotazione dei pezzi descritta nella Sezione 1.3.

1.3. Vincoli di taglio

Le macchine da taglio automatiche producono di norma i pezzi mediante una sequenza di tagli a ghigliottina [4]. Un taglio a ghigliottina procede da un bordo all'altro del rettangolo in lavorazione, parallelamente a uno dei suoi lati, e lo divide in due rettangoli, ciascuno dei quali può essere un pezzo, scarto oppure un nuovo rettangolo da tagliare [6]. Uno schema di taglio è detto a ghigliottina se tutti i suoi pezzi possono essere ottenuti con tagli di questo tipo. Non tutti gli schemi lo sono: nello schema della Figura 1.1b nessun taglio da un bordo all'altro del pannello evita di attraversare un pezzo, e i pezzi non possono quindi essere separati. Il vincolo restringe l'insieme degli schemi ammissibili e può condurre a soluzioni peggiori di quelle ottenibili senza di esso [4].



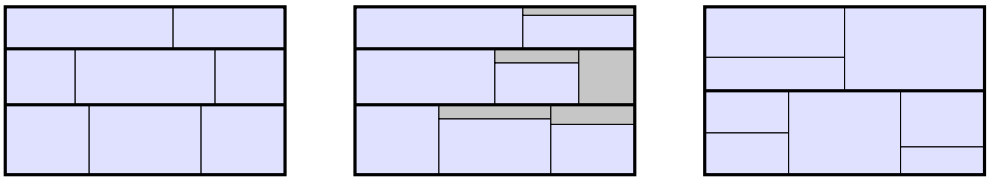
(a) (b)

Figura 1.1.: (a) Schema di taglio a ghigliottina; (b) schema che non può essere ottenuto con tagli a ghigliottina, poiché ogni taglio da un bordo all'altro del pannello attraversa almeno un pezzo.

Spesso le macchine eseguono i tagli per *stadi*, alternando la direzione orizzontale e quella verticale [4]. Nel primo stadio il pannello è diviso da tagli paralleli in una stessa direzione; nel secondo stadio ciascuna delle parti ottenute è divisa da tagli ortogonali ai precedenti, e così via. Nel seguito, con tagli del primo stadio orizzontali, le parti prodotte dal primo stadio sono dette *strisce* e quelle prodotte dal secondo *colonne*, mentre il terzo stadio divide le colonne in pezzi. Se il numero di stadi con cui ogni pezzo deve essere ottenuto è limitato da un valore k, il problema è detto a k stadi; in caso contrario è detto senza limite di stadi [3]. Nella maggior parte delle applicazioni k vale 2 o 3 [4]. La direzione dei tagli del primo stadio può essere imposta per ciascun tipo di pannello oppure lasciata libera; nelle istanze di questa tesi è un dato associato al tipo di pannello (Capitolo 3).

Negli schemi a stadi può essere ammesso un taglio aggiuntivo, detto di *rifilatura* (*trimming*), che riduce un pezzo alle sue dimensioni esatte separandolo dallo scarto; il problema è detto non esatto se la rifilatura è ammessa ed esatto in caso contrario [3]. La Figura 1.2 confronta uno schema a 2 stadi esatto, nel quale ogni colonna coincide con un pezzo, con uno a 2 stadi con rifilatura e con uno a 3 stadi. Il caso a 2 stadi con rifilatura corrisponde al *packing* per livelli, nel quale i pezzi sono affiancati su

ripiani di larghezza pari a quella del pannello e altezza pari a quella del pezzo più alto [1, 4].



(a) (b) (c)

Figura 1.2.: Schemi di taglio a stadi con tagli del primo stadio orizzontali, indicati dalle linee spesse: (a) 2 stadi esatto; (b) 2 stadi con rifilatura, in cui le aree grigie sono lo scarto separato dalla rifilatura; (c) 3 stadi, in cui il terzo stadio divide le colonne in più pezzi.

Uno schema a 3 stadi in cui il terzo stadio non è utilizzato, cioè in cui ogni colonna contiene un solo pezzo, è uno schema a 2 stadi. L'insieme degli schemi a 2 stadi è quindi contenuto in quello degli schemi a 3 stadi e, più in generale, l'insieme degli schemi a k stadi è contenuto in quello degli schemi a $k+1$ stadi; ne segue che il valore ottimo dello scarto non può aumentare al crescere del numero di stadi ammessi.

La rotazione dei pezzi di 90°, esclusa nella definizione della Sezione 1.2 come nella maggior parte della letteratura [4], è una variante diffusa, possibile quando il materiale è uniforme in tutte le direzioni [3]: consente schemi più compatti, ma aumenta il numero di decisioni da prendere [4]. Le istanze di questa tesi sono costruite e risolte in questa variante, con schemi a 2 o a 3 stadi: il generatore impiega i *part-type* anche ruotati (Capitolo 3), e la soluzione a scarto nullo che ne risulta può non essere ammissibile senza rotazione. Poiché tale soluzione è costruita con schemi esatti, il suo valore è ottimo sia nella versione esatta sia in quella non esatta del problema; la variante gestita dal *solver* è descritta nelle Sezioni 2.1 e 2.3.

1.4. Modello matematico

Il problema può essere formulato come problema di programmazione lineare intera seguendo l'approccio proposto da Gilmore e Gomory per il *cutting stock*, come riportato in [3, 1], nel quale le variabili non descrivono la posizione dei singoli pezzi ma il numero di volte in cui ciascuno schema di taglio viene impiegato. Sia $\mathcal{P}_i$ l'insieme degli schemi ammissibili per un pannello di tipo $i \in \mathcal{S}$, cioè degli schemi che soddisfano le condizioni delle Sezioni 1.2 e 1.3, e sia $\mathcal{P} = \bigcup_{i \in \mathcal{S}} \mathcal{P}_i$. Per ogni schema $p \in \mathcal{P}$ siano a_{jp} il numero di pezzi del *part-type* j che esso contiene, in uno qualsiasi dei due orientamenti nella variante con rotazione, e x_p la variabile intera che indica

il numero di pannelli tagliati secondo lo schema p. Il problema è allora

$$\min \quad \sum_{i \in \mathcal{S}} W_i H_i \sum_{p \in \mathcal{P}_i} x_p \tag{1.6}$$

$$\text{s.a.} \quad \sum_{p \in \mathcal{P}} a_{jp}\, x_p \geq d_j \qquad j \in \mathcal{R}, \tag{1.7}$$

$$x_p \in \mathbb{Z}_{\geq 0} \qquad p \in \mathcal{P}. \tag{1.8}$$

La funzione obiettivo (1.6) è l'area utilizzata (1.3), poiché il numero di pannelli di tipo i impiegati è $n_i = \sum_{p \in \mathcal{P}_i} x_p$. I vincoli (1.7) impongono che la domanda di ciascun *part-type* sia soddisfatta. Essi sono espressi come disuguaglianze, come nella formulazione originale, ma non ammettono soluzioni migliori di quelle con domanda esatta: sostituendo con scarto un pezzo in eccesso si ottiene ancora uno schema ammissibile, che impiega la stessa area. In presenza di una disponibilità limitata b_i dei pannelli di tipo i si aggiungerebbero i vincoli $\sum_{p \in \mathcal{P}_i} x_p \leq b_i$, come nel problema affrontato dal *solver* (Sezione 2.1).

Il numero di schemi ammissibili è in generale esponenziale [4], e già per istanze di piccole dimensioni non è pratico enumerarli tutti [3]. Gilmore e Gomory hanno perciò proposto un procedimento di *generazione di colonne*: si risolve il rilassamento lineare del problema, ottenuto sostituendo i vincoli (1.8) con $x_p \geq 0$, considerando inizialmente un sottoinsieme ridotto di schemi; a ogni iterazione un sottoproblema, detto di *pricing*, individua lo schema di costo ridotto più negativo rispetto ai prezzi duali dei vincoli di domanda, che viene aggiunto al problema. Il procedimento termina quando nessuno schema ha costo ridotto negativo, e la soluzione ottenuta è ottima per il rilassamento [3]. Nel caso monodimensionale il sottoproblema è un *knapsack* limitato; per gli schemi a 2 stadi, nella versione in cui il numero di copie dei pezzi non è limitato, può essere risolto con un *knapsack* per ciascun tipo di striscia, uno per *part-type*, e con un ulteriore *knapsack* che sceglie quali strisce combinare nel pannello, come mostrato da Gilmore e Gomory e riportato in [3]. Poiché la soluzione del rilassamento può essere frazionaria, per ottenere una soluzione intera occorre un passo ulteriore, come l'arrotondamento delle frequenze degli schemi o l'impiego di un altro algoritmo sul problema residuo [3]. La procedura di generazione degli schemi del *solver* descritto nel Capitolo 2 ha la stessa struttura a due livelli di *knapsack*, ma sostituisce i prezzi duali con valori stimati e aggiornati in modo euristico (Sezioni 2.2 e 2.3).

Il rilassamento più semplice si ottiene dividendo ogni pezzo in quadrati di lato unitario, che possono essere collocati liberamente [4]. Per il problema qui considerato esso fornisce il limite inferiore $U \geq A^*$ all'area utilizzata, già osservato nella Sezione 1.2; con un solo tipo di pannello, di dimensioni $W \times H$, ne discende il limite sul numero n di pannelli

$$n \geq \left\lceil \frac{A^*}{WH} \right\rceil . \tag{1.9}$$

Questo limite, detto continuo, si calcola in tempo lineare ed è impiegato in quasi tutti i lavori sui problemi bidimensionali [4], ma in generale può essere lontano dall'ottimo: per il *bin packing* bidimensionale con pannelli identici, Martello e Vigo hanno mostrato che il suo rapporto di prestazione nel caso peggiore è 1/4, come riportato in [3]. Un limite inferiore consente di certificare l'ottimalità di una soluzione che lo raggiunge, ed è con questo scopo che il *solver* lo impiega per arrestare la ricerca (Sezione 2.5). Sulle istanze a scarto nullo il limite continuo coincide con il valore ottimo, poiché la soluzione ottima utilizza un'area pari ad A^*.

Oltre alla formulazione a schemi, la letteratura propone modelli basati sulle coordinate in cui i pezzi possono essere collocati, come quello di Beasley riportato in [1], modelli con un numero polinomiale di variabili per il caso a 2 stadi, nel quale i pezzi sono disposti per livelli [1], e modelli a flusso su archi, nei quali uno schema è rappresentato da un cammino in un grafo [3]. Una rassegna sistematica, che classifica le formulazioni secondo la loro dimensione, è riportata in [4].

1.5. Complessità computazionale

Il problema è NP-difficile in senso forte. Il risultato discende dal *bin packing* monodimensionale, nel quale pezzi di lunghezza w_j devono essere ripartiti nel minimo numero di contenitori di capacità W, noto per essere NP-difficile in senso forte (Garey e Johnson, come riportato in [4]). Data un'istanza di tale problema, si considerino pannelli di dimensioni $W \times 1$ e pezzi di dimensioni $w_j \times 1$: la soluzione dell'istanza bidimensionale fornisce quella dell'istanza monodimensionale [1, 4]. Lo stesso argomento vale per il problema di questa tesi, che comprende queste istanze come caso particolare. Esse hanno un solo tipo di pannello, per il quale minimizzare l'area equivale a minimizzare il numero di pannelli (Sezione 1.2), e i vincoli della Sezione 1.3 non le modificano: pezzi alti quanto il pannello possono soltanto essere affiancati in un'unica fila, separabile con tagli a ghigliottina in un solo stadio, e un pezzo ruotato entra nel pannello solo se è quadrato, nel qual caso la rotazione non ne cambia l'ingombro. Il problema è quindi anch'esso NP-difficile in senso forte.

Un algoritmo è detto pseudo-polinomiale se il suo tempo di calcolo è polinomiale nella dimensione dell'istanza e nel valore dei numeri che vi compaiono, come le dimensioni dei pannelli e dei pezzi. Il *knapsack* monodimensionale, per esempio, si risolve in tempo pseudo-polinomiale con la programmazione dinamica [4]. Un problema NP-difficile in senso forte resta invece tale anche quando quei valori sono piccoli: la sua difficoltà dipende dalla struttura combinatoria e non dall'entità dei numeri, e non ammette algoritmi pseudo-polinomiali a meno che non sia P = NP [4].

Sul piano pratico, i metodi esatti hanno compiuto progressi considerevoli negli ultimi decenni, ma esistono ancora istanze di *benchmark* con un numero relativamente ridotto di pezzi che non sono state risolte all'ottimo [4]. Per le istanze di dimensioni industriali si ricorre quindi a metodi non esatti, che ottengono soluzioni di buona qualità con risorse di calcolo molto inferiori, senza tuttavia garantirne l'ottimalità [3].

La qualità della soluzione di un metodo non esatto può essere valutata confrontandone il valore con un limite inferiore: se lo raggiunge, la soluzione è ottima; in caso contrario la distanza dall'ottimo resta incognita, poiché non è noto quanta parte della differenza sia dovuta al metodo e quanta alla distanza tra il limite e l'ottimo, che per il limite continuo può essere considerevole (Sezione 1.4). Le istanze con ottimo noto, discusse nella Sezione 1.7, eliminano questa incertezza.

1.6. Rassegna della letteratura

La letteratura sui metodi risolutivi per i problemi bidimensionali di taglio e di *packing* è molto ampia; questa sezione ne richiama le linee principali per i problemi con pannelli e tagli a ghigliottina, rimandando alle rassegne di Lodi et al. [1], Iori et al. [4] e Oliveira et al. [3] per una trattazione completa. Tra i metodi esatti, Martello e Vigo hanno risolto il *bin packing* bidimensionale con un algoritmo di *branch-and-bound* basato su nuovi limiti inferiori, mentre algoritmi di *branch-and-price*, nei quali a ogni nodo del *branch-and-bound* il rilassamento lineare della formulazione a schemi è risolto per generazione di colonne (Sezione 1.4), sono stati proposti da Pisinger e Sigurd per il caso con tagli a ghigliottina e da Puchinger e Raidl per quello a 3 stadi; Silva et al. hanno formulato un modello di programmazione lineare intera per il *cutting stock* a 2 e a 3 stadi, esteso da Furini et al. anche al caso con più formati di pannello, come riportato in [3]. La presenza di pannelli di dimensioni diverse è gestita, nei metodi enumerativi e nel *pricing*, considerando i diversi formati nella generazione dei nodi o delle colonne, al prezzo di un maggiore sforzo di calcolo [4].

Tra i metodi non esatti, le euristiche costruttive più semplici dispongono i pezzi per livelli, producendo schemi a 2 stadi con rifilatura (Sezione 1.3) e scomponendo il problema in due problemi monodimensionali. L'euristica *hybrid first-fit* di Chung et al., per esempio, ordina i pezzi per altezza non crescente e li colloca uno alla volta nella prima striscia che ha spazio sufficiente, aprendone una nuova, alta quanto il pezzo, quando nessuna striscia può accoglierlo; le strisce così formate sono poi assegnate ai pannelli come in un *bin packing* monodimensionale, con la regola *first-fit decreasing*, come riportato in [3, 1]. Altre euristiche sfruttano la formulazione a schemi senza ricorrere al *pricing* esatto: quella di Monaci e Toth genera un ampio insieme di schemi con regole *greedy* diversificate e ne seleziona poi la combinazione migliore risolvendo un problema di copertura, così che la qualità della soluzione dipende dalla varietà degli schemi generati più che dall'ottimalità di ciascuno [3].

Sono state infine proposte numerose metaeuristiche, cioè strategie di ricerca di carattere generale che guidano euristiche più semplici nell'esplorazione dello spazio delle soluzioni, consentendo di allontanarsi dagli ottimi locali in cui una ricerca puramente migliorativa si arresterebbe. Per il *bin packing* bidimensionale con tagli a ghigliottina Lodi, Martello e Vigo hanno proposto algoritmi di ricerca tabù, mentre Alvelos et al. hanno sviluppato una ricerca a vicinato variabile per i casi a 2 e a 3

stadi; algoritmi genetici e altre metaeuristiche sono stati applicati alle diverse varianti del problema, come riportato in [3].

Una famiglia di euristiche è particolarmente vicina al *solver* impiegato in questa tesi. Anziché risolvere in modo esatto il sottoproblema di *pricing* con i prezzi duali, come nella generazione di colonne, queste euristiche costruiscono gli schemi uno alla volta impiegando valori dei pezzi stimati, che vengono corretti tra un'iterazione e la successiva [5]. L'euristica di Boschetti e Mingozzi aggiorna i valori dei pezzi al termine di ogni soluzione e ne genera una nuova con i valori aggiornati, mentre quella di Cui et al. corregge i valori dopo la generazione di ogni schema, favorendo i pezzi che si combinano male con lo schema corrente e quelli più grandi, più difficili da collocare, come riportato in [3]. Il *solver* descritto nel Capitolo 2 appartiene a questa famiglia, applicata a un problema *rich* con più formati di pannello e più criteri di ottimizzazione [5].

1.7. Istanze di test e istanze con ottimo noto

Il confronto tra metodi risolutivi richiede insiemi di istanze condivisi, sui quali valutare le soluzioni ottenute e i tempi di calcolo. Il gruppo di lavoro europeo sui problemi di taglio e di *packing* (*ESICUP*) mette a disposizione *dataset* e generatori di istanze proposti da diversi autori, e Oliveira et al. hanno raccolto oltre 80 *dataset* e sei generatori relativi ai problemi bidimensionali, convertendo le istanze in un formato comune, come riportato in [3]. Con un obiettivo analogo, la libreria 2DPackLib [6] raccoglie in un formato unificato 25 *benchmark* della letteratura, per un totale di oltre 3000 istanze, per rimediare ad alcune difficoltà ricorrenti: la corrispondenza non sempre chiara tra varianti del problema e *benchmark*, la varietà dei formati e l'irreperibilità di istanze pubblicate su pagine personali non più disponibili.

Tra i *benchmark* per il *bin packing* e il *cutting stock* bidimensionali, l'insieme più impiegato comprende 500 istanze suddivise in dieci classi, con pannelli identici all'interno di ciascuna classe; un altro insieme, composto da 43 istanze tratte dall'industria del mobile, richiede tagli a ghigliottina, con pannelli identici in ciascuna istanza, di uno fra tre formati [6]. Il formato unificato della libreria prevede tuttavia pannelli identici: i *benchmark* con pannelli di dimensioni diverse sono pochi e non vi sono inclusi [6].

Molte di queste istanze non sono ancora risolte all'ottimo. Dopo oltre vent'anni, numerose istanze delle dieci classi con almeno 60 pezzi restano aperte, con una differenza di un pannello tra il miglior limite superiore e il miglior limite inferiore noti; delle 500 istanze con pannelli di dimensioni e costi diversi ottenute da Pisinger e Sigurd modificando le dieci classi, ne sono state risolte soltanto 164 [4]. Su istanze di questo tipo la distanza dall'ottimo delle soluzioni ottenute con un metodo non esatto non è nota, per le ragioni discusse nella Sezione 1.5.

Esistono anche istanze generate in modo che l'ottimo sia noto in anticipo. Tra i *benchmark* della 2DPackLib esse riguardano lo *strip packing*, nel quale i pezzi devono

essere disposti in un unico oggetto di larghezza fissata minimizzandone l'altezza utilizzata: l'insieme proposto da Hopper e Turton è generato in modo che la soluzione ottima sia nota, e quello proposto da Burke et al. in modo che l'ottimo consista in una disposizione priva di scarto, ottenibile con tagli a ghigliottina; per queste istanze i file riportano l'altezza ottima [6]. La libreria segnala inoltre generatori di istanze per i problemi bidimensionali, come quello sviluppato da Silva et al. [6].

Il generatore descritto nel Capitolo 3 applica l'idea delle istanze con ottimo noto al problema definito nelle Sezioni 1.2 e 1.3: più formati di pannello, domanda per *part-type*, schemi a ghigliottina a 2 o a 3 stadi e rotazione dei pezzi. Si tratta di una combinazione che i *benchmark* con ottimo noto descritti sopra non coprono, e che riguarda la classe di problemi nella quale le istanze ancora aperte mostrano quanto sia difficile valutare la qualità delle soluzioni.

Capitolo 2.

Il software di ottimizzazione

Il presente capitolo descrive il software di ottimizzazione impiegato negli esperimenti, un'euristica per il taglio di pannelli sviluppata per un'applicazione industriale [5]. Viene dapprima presentato il problema affrontato dal *solver*, con le caratteristiche che lo distinguono da quello del Capitolo 1; segue la descrizione dell'algoritmo, articolata nell'euristica a correzione sequenziale dei valori, nella procedura di generazione degli schemi di taglio, nel confronto tra le soluzioni basato sulla dominanza di Pareto e nelle fasi di ottimizzazione. Sono poi illustrati i file di input e di output, insieme ai parametri impiegati negli esperimenti. Il capitolo si chiude con il ruolo del *solver* nel contesto della tesi e con il motivo per cui la sua valutazione richiede istanze con ottimo noto.

2.1. Il problema affrontato dal *solver*

Il *solver* è stato sviluppato per il taglio di pannelli in legno su una famiglia di macchine industriali, e il problema che risolve include tutti i vincoli tecnici di tali macchine, così che le soluzioni prodotte possano essere eseguite direttamente sull'impianto. Problemi di questo tipo sono detti *rich* [5]. Un modello semplificato, infatti, trascura dettagli rilevanti nella pratica: uno schema di taglio che non tiene conto del rifilo dei bordi del pannello può non essere eseguibile dalla macchina, mentre considerare uno spessore della lama di frazioni di millimetro richiede all'algoritmo una precisione maggiore, che può rallentarlo [5].

Nella formulazione di riferimento il problema è una variante bidimensionale del *cutting stock* con più formati di pannello, e ha lo stesso nucleo del problema del Capitolo 1: ricavare da pannelli rettangolari di più tipi, disponibili in numero limitato o illimitato, i pezzi richiesti per ciascun *part-type*. L'obiettivo principale è la minimizzazione dell'area di pannello utilizzata; altri criteri, come il numero di pannelli impiegati, il numero di schemi di taglio distinti e il numero di pretagli, possono essere considerati in modo selettivo o congiunto [5], secondo il meccanismo descritto nella Sezione 2.4.

Le caratteristiche geometriche del problema sono le seguenti [5]:

- i tagli sono a ghigliottina, cioè procedono da un bordo all'altro del pannello, o della parte di pannello in lavorazione, parallelamente a uno dei suoi lati;

- i pezzi possono essere ruotati di 90°; la rotazione è esclusa per i *part-type* che presentano una venatura, il cui orientamento sul pannello è vincolato;

- gli schemi di taglio sono a 2 oppure a 3 stadi; nell'articolo di riferimento lo schema a 2 stadi ammette la rifilatura (*trimming*), mentre la versione del *solver* impiegata offre lo schema a 2 stadi sia con sia senza rifilatura, oltre a schemi con un numero di stadi arbitrario.

Ogni schema di taglio è costruito a partire da una *configurazione*, la coppia formata da un tipo di pannello e dall'orientamento, orizzontale o verticale, dei tagli del primo stadio. Nel primo stadio il pannello è diviso da tagli paralleli secondo tale orientamento; ciascuna delle parti ottenute è poi ruotata di 90° e lo stadio si ripete su di essa. La definizione generale degli stadi è data nella Sezione 1.3, mentre la struttura degli schemi a 2 e a 3 stadi descritta in [5] è illustrata nella Sezione 2.3. Negli esperimenti di questa tesi sono impiegate la modalità a 2 stadi senza rifilatura e la modalità a 3 stadi. Per ogni tipo di pannello l'orientamento del primo taglio può essere imposto oppure lasciato libero; nel secondo caso entrambe le configurazioni del tipo di pannello sono disponibili per la costruzione degli schemi.

Il problema ammette inoltre, come passo preliminare facoltativo, il *pretaglio*: un taglio che divide il pannello in due parti, ciascuna ripartita poi con un proprio sotto-schema [5]. Il numero di pretagli è tra i criteri che il *solver* cerca di ridurre, in una fase dedicata dell'algoritmo (Sezione 2.5). Negli esperimenti di questa tesi il pretaglio è ammesso, mentre la fase di riduzione dei pretagli è disattivata.

La domanda va di norma soddisfatta esattamente. Per ogni *part-type* si può tuttavia ammettere una tolleranza, così che il numero di pezzi prodotti possa discostarsi dalla domanda entro quantità ammesse di sottoproduzione e di sovraproduzione. Un parametro del *solver* consente inoltre di regolare la sovraproduzione in modo automatico: il vincolo di domanda si allenta, la produzione deve essere almeno pari alla domanda anziché uguale a essa e la quantità in eccesso è stabilita dall'algoritmo. Rilassare il vincolo di domanda può condurre a soluzioni con scarto ridotto, al prezzo di un numero elevato di pezzi in eccesso che accresce i pannelli necessari e con essi i costi di produzione [5]. Nelle istanze di questa tesi le tolleranze sono nulle: la sovraproduzione deriva soltanto dal parametro automatico, il cui effetto sulle istanze a scarto nullo è analizzato nella Sezione 4.4.4.

Il vincolo operativo principale riguarda gli *open stack*. I pezzi tagliati sono depositati presso le stazioni di scarico della macchina in pile, una per ciascun *part-type* in lavorazione, e una pila resta aperta finché il relativo *part-type* non è completato. Il numero di pile aperte contemporaneamente non è un obiettivo da minimizzare, ma è limitato dal numero di stazioni di scarico disponibili: la costruzione degli schemi rispetta tale limite, e una procedura di sequenziamento degli schemi cerca poi di ridurre le pile aperte dalla soluzione [5]. Nelle istanze di questa tesi i *part-type* non sono assegnati alle stazioni di scarico e il vincolo non interviene.

Rispetto al problema del Capitolo 1, quello affrontato dal *solver* aggiunge quindi la rotazione dei pezzi, il pretaglio, la disponibilità limitata dei pannelli, la tolleranza sulla domanda e i vincoli operativi della macchina. Nelle istanze di questa tesi la disponibilità dei pannelli è illimitata, le tolleranze sono nulle e gli *open stack* non sono vincolati; delle caratteristiche aggiuntive restano attive la rotazione, consentita per tutti i *part-type*, il pretaglio e la sovraproduzione automatica, quando il relativo parametro è attivo. La sezione successiva descrive l'euristica con cui il *solver* costruisce le soluzioni.

2.2. L'euristica a correzione sequenziale dei valori

Il *solver* costruisce le soluzioni mediante un'euristica a correzione sequenziale dei valori (*Sequential Value Correction*, *SVC*) [5]. A ciascun *part-type* è assegnato un valore, che ne esprime la convenienza a essere inserito in uno schema di taglio, e gli schemi sono costruiti uno dopo l'altro, scegliendo ogni volta tra i pezzi non ancora assegnati quelli che massimizzano il valore complessivo dello schema. Dopo ogni schema i valori vengono corretti, e la costruzione di una soluzione completa è ripetuta più volte per ottenere soluzioni diverse. A differenza dei metodi che calcolano i valori in modo esatto, come i prezzi duali di un modello di programmazione lineare, l'euristica li stima e li aggiorna con regole semplici: rinuncia al rigore del calcolo esatto, ma costruisce ogni schema in tempi ridotti [5].

I valori sono definiti sull'insieme $\mathcal{R}'$ formato dai *part-type* e dalle loro versioni ruotate, ove la rotazione è ammessa, e ciascun valore v_j è inizializzato all'area del pezzo corrispondente [5]. All'inizio, quindi, il valore di uno schema coincide con l'area che esso ripartisce in pezzi, e uno schema di valore massimo è uno schema di scarto minimo sul singolo pannello. Le correzioni successive modificano questo criterio puramente geometrico, tenendo conto anche della domanda ancora da soddisfare.

La procedura è riportata nell'Algoritmo 1, in una forma semplificata che omette i dettagli relativi al pretaglio. Ciascuna delle N iterazioni costruisce una soluzione completa s. Finché resta domanda da soddisfare, per ogni configurazione c viene generato uno schema di taglio di valore massimo rispetto ai valori correnti e alla domanda residua (riga 9), secondo la procedura descritta nella Sezione 2.3. Tra gli schemi generati è scelto quello di valore maggiore, cui è associato un *livello di attivazione* t, cioè il numero di volte in cui lo schema viene ripetuto, espresso in pacchi: con pacchi di ℓ' pannelli lo schema è tagliato su $t\ell'$ pannelli. Il livello di attivazione è stabilito in base alla domanda residua e alla disponibilità dei pannelli (riga 10). Gli spazi rimasti vuoti nello schema scelto sono quindi riempiti, ove possibile, con pezzi ancora da assegnare, e la domanda residua viene aggiornata (riga 11).

Dopo ogni schema i valori vengono corretti, promuovendo i *part-type* con domanda residua elevata e con poche copie nello schema appena scelto (riga 13) [5]. Lo scopo della correzione è rendere più convenienti, negli schemi successivi, i *part-type* ancora molto richiesti, che altrimenti rischierebbero di restare per ultimi e di dover essere

Algoritmo 1 Euristica *SVC* in forma semplificata, adattata da [5].

Dati: domanda $\mathbf{d}$, dimensione del pacco ℓ, numero di iterazioni N

1: $v_j \leftarrow$ area del pezzo j, per ogni $j \in \mathcal{R}'$
2: **per** $n = 1, \ldots, N$ **esegui**
3: $\quad g \leftarrow$ valore casuale in $[1, \rho]$
4: $\quad \mathbf{r} \leftarrow \mathbf{d}, \quad s \leftarrow \emptyset, \quad \ell' \leftarrow \ell$
5: $\quad$ **ripeti**
6: $\quad\quad \mathbf{d}' \leftarrow \lfloor \mathbf{r}/\ell' \rfloor, \quad \mathbf{r}' \leftarrow \mathbf{r} \bmod \ell'$
7: $\quad\quad$ **finché** $\mathbf{d}' \neq \mathbf{0}$ **esegui**
8: $\quad\quad\quad$ **per** ogni configurazione $c \in \mathcal{C}$ **esegui**
9: $\quad\quad\quad\quad p_c \leftarrow$ GeneraSchema$(c, \mathbf{v}, \mathbf{d}')$
10: $\quad\quad\quad$ $(a, t) \leftarrow$ schema di valore massimo tra i p_c e suo livello di attivazione
11: $\quad\quad\quad$ Riempi(a, t); aggiungi a a s con livello t e aggiorna $\mathbf{d}'$
12: $\quad\quad\quad$ $h \leftarrow$ valore casuale in $[1/g, g]$
13: $\quad\quad\quad$ $\mathbf{v} \leftarrow$ CorreggiValori$(a, \mathbf{v}, h)$
14: $\quad\quad$ $\mathbf{r} \leftarrow \mathbf{r}', \quad$ riduci ℓ'
15: $\quad$ **fino a quando** $\mathbf{r} = \mathbf{0}$
16: $\quad$ Sequenzia(s)
17: $\quad$ **se** s non è dominata **allora**
18: $\quad\quad$ aggiungi s al *pool*

collocati in schemi poco efficienti. La correzione dipende da un fattore casuale h, estratto nell'intervallo $[1/g, g]$, la cui ampiezza g è a sua volta estratta a ogni iterazione in $[1, \rho]$, con $\rho \geq 1$ parametro dell'euristica (riga 3): la perturbazione fa sì che iterazioni diverse esplorino costruzioni diverse. Le estrazioni casuali sono governate da un seme, parametro del *solver*: a parità di seme, di istanza e di parametri la sequenza delle estrazioni è la stessa. Negli esperimenti di questa tesi il seme è mantenuto costante. I valori, inoltre, non vengono reinizializzati tra un'iterazione e la successiva (riga 1), e ogni soluzione eredita le correzioni accumulate durante la costruzione delle precedenti.

Completata una soluzione, una procedura di sequenziamento riordina gli schemi, scambiando le coppie che riducono maggiormente il numero di *open stack* (riga 16), e la soluzione è inserita nel *pool* se non è dominata da quelle già presenti (Sezione 2.4) [5]. Il numero N di iterazioni misura lo sforzo di ricerca dell'euristica: nel *solver* è pari al prodotto di due parametri, il numero di cicli e il numero di ripetizioni all'interno di ciascun ciclo, e costituisce uno dei fattori analizzati nel Capitolo 4.

La procedura riceve infine la dimensione del pacco ℓ, cioè il numero di pannelli sovrapposti, detti *fogli*, che vengono tagliati contemporaneamente secondo lo stesso schema [5]. La domanda residua è espressa in pacchi, $\mathbf{d}' = \lfloor \mathbf{r}/\ell \rfloor$, e gli schemi sono attivati in multipli di ℓ (riga 6). La parte di domanda che non costituisce un pacco intero, $\mathbf{r} \bmod \ell$, è soddisfatta in un secondo momento con pacchi di dimensione progressivamente minore, fino al pannello singolo (riga 14). Con $\ell = 1$, che nel file dei parametri si indica con `lotsize` pari a -1, la gestione a pacco è assente

e ogni schema può essere attivato su un numero qualsiasi di pannelli. Lo stesso raggruppamento della domanda, con pacchi di dimensione crescente, è impiegato nella fase di minimizzazione degli schemi (Sezione 2.5).

2.3. Generazione degli schemi di taglio

Questa sezione descrive la procedura GeneraSchema dell'Algoritmo 1, che per una configurazione c costruisce uno schema di taglio di valore massimo rispetto ai valori correnti $\mathbf{v}$ e alla domanda residua $\mathbf{d}'$ [5]. Si considera il caso in cui i tagli del primo stadio sono paralleli alla larghezza W_i del pannello e lo dividono quindi in *strisce* orizzontali; il caso opposto si ottiene scambiando i ruoli di larghezza e altezza. I tagli del secondo stadio dividono ogni striscia in *colonne*, e quelli dell'eventuale terzo stadio dividono le colonne in pezzi. La struttura che ne risulta è illustrata nella Figura 2.1.

Per ogni *part-type* $j \in \mathcal{R}'$ si definisce una striscia orizzontale di altezza h_j e la si riempie risolvendo un problema *knapsack* intero limitato, la cui capacità è la larghezza W_i del pannello [5]. Gli elementi del *knapsack* sono colonne, formate con i *part-type* k di altezza $h_k \leq h_j$, domanda residua positiva e valore positivo, ciascuna di larghezza pari a quella del pezzo che contiene. Negli schemi a 2 stadi, nella forma descritta in [5], ogni colonna contiene un solo pezzo, e la differenza di altezza $h_j - h_k$ resta scarto, separato con una rifilatura (Sezione 1.3). Nella modalità senza rifilatura, impiegata negli esperimenti, lo schema deve essere a 2 stadi esatto: una colonna può contenere soltanto un pezzo alto quanto la striscia. Negli schemi a 3 stadi, nella stessa forma, una colonna può contenere fino a $\lfloor h_j/h_k \rfloor$ copie sovrapposte dello stesso *part-type*, separate da un terzo stadio di tagli, e il suo valore è calcolato di conseguenza. Il valore di una striscia è la somma dei valori dei pezzi che contiene.

Le strisce generate formano un insieme $\mathcal{Q}_c$, e a ciascuna è associato un livello di attivazione rispetto alla domanda residua. Lo schema si ottiene risolvendo un secondo *knapsack* intero limitato, che seleziona le strisce da sovrapporre nel pannello: la capacità è l'altezza H_i, e ogni striscia ha come peso la propria altezza e come valore il proprio [5]. Per conservare la struttura del *knapsack*, il modello considera soltanto i livelli di attivazione delle strisce, e trascura sia le molteplicità dei *part-type* al loro interno sia la presenza contemporanea di un *part-type* e della sua versione ruotata: la soluzione può quindi prevedere più pezzi di quanti la domanda ne richieda. Per questo le strisce selezionate sono ordinate per valore decrescente del rapporto tra valore e area, e aggiunte allo schema una alla volta secondo tale ordine, aggiornando la domanda residua; una striscia è scartata se produce pezzi oltre la domanda o se supera il limite sugli *open stack*.

Le strisce scartate possono lasciare nel pannello un'altezza residua R non occupata. In tal caso l'intera procedura, generazione e selezione delle strisce, è ripetuta su un pannello fittizio di dimensioni $W_i \times R$, e il sotto-schema ottenuto è sovrapposto allo schema e unito a esso; l'operazione prosegue finché il residuo non può più essere

riempito [5]. Lo schema completo entra infine tra i candidati tra cui l'Algoritmo 1 sceglie quello di valore massimo (riga 10).

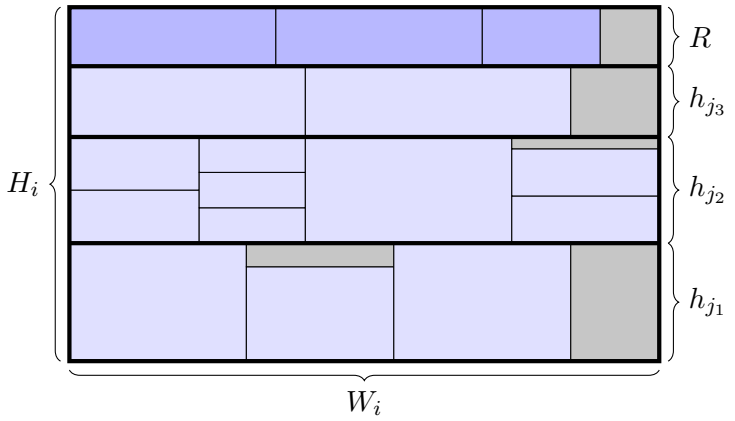


Figura 2.1.: Struttura di uno schema di taglio a 3 stadi con tagli del primo stadio orizzontali. Le linee spesse sono i tagli del primo stadio, che delimitano le strisce di altezza h_j; all'interno di ogni striscia i tagli del secondo stadio separano le colonne, e nella striscia centrale i tagli del terzo stadio separano le copie sovrapposte di uno stesso *part-type*. Le aree grigie sono scarto; il sotto-schema in colore più scuro riempie l'altezza residua R. Negli schemi a 2 stadi ogni colonna contiene un solo pezzo, come nella striscia inferiore.

Poiché i valori dei *part-type* sono numeri reali, ciascun *knapsack* intero è trasformato in un *knapsack* 0-1, codificando le variabili intere con variabili binarie, e risolto in modo esatto [5]. In alternativa il *solver* consente di risolvere i *knapsack* con un'euristica *greedy*, che costruisce la soluzione inserendo gli elementi uno alla volta secondo un ordine prefissato, senza rimettere in discussione le scelte già compiute: la risoluzione è più rapida, ma gli schemi ottenuti possono avere valore inferiore a quello massimo. La modalità di risoluzione è uno dei fattori analizzati nel Capitolo 4.

Per ogni schema p, infine, la procedura genera anche una variante p' con pretaglio e valuta se sostituirla a p [5]. Tra i tagli del secondo stadio, che separano le colonne, è scelto quello la cui posizione x è più vicina alla metà della larghezza W_i. Il pannello è diviso in corrispondenza di x in due parti, di dimensioni $x \times H_i$ e $(W_i - x) \times H_i$: nella parte sinistra sono conservati i pezzi di p che non attraversano il taglio, mentre la parte destra è riempita con un nuovo sotto-schema, costruito con la stessa procedura. L'unione delle due parti costituisce lo schema p' (Figura 2.2). Lo schema con pretaglio sostituisce p se la differenza tra i costi ridotti dei due schemi, che ne misura la diversa convenienza, supera una soglia di opportunità δ. La soglia è nulla nelle prime due fasi dell'algoritmo e cresce nella fase di riduzione dei pretagli, così da rendere il pretaglio progressivamente meno conveniente (Sezione 2.5). Negli esperimenti di questa tesi tale fase è disattivata e la soglia resta nulla.

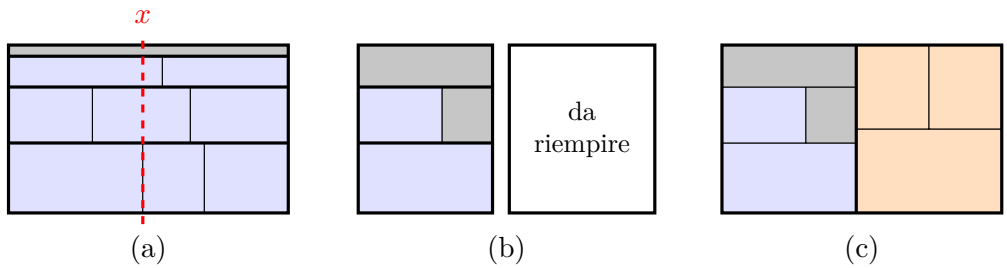


Figura 2.2.: Costruzione di uno schema con pretaglio. (a) Schema p: la linea tratteggiata indica la posizione x del taglio del secondo stadio più vicina a metà larghezza. (b) Il pannello è diviso in corrispondenza di x: la parte sinistra conserva i pezzi di p che non attraversano il taglio, la parte destra è liberata. (c) La parte destra è riempita con un nuovo sotto-schema, in arancione, e l'unione delle due parti fornisce lo schema p'.

2.4. Soluzioni non dominate e scelta della soluzione

Le soluzioni costruite dall'euristica sono confrontate su più indicatori di prestazione. Nella formulazione di riferimento i criteri sono l'area di pannello utilizzata, lo scarto, il numero di pannelli impiegati, il numero di schemi di taglio distinti e il numero di pretagli [5]; il *solver* vi aggiunge il costo dei pannelli e l'area dei pezzi prodotti, oltre a due indicatori relativi alla disposizione dei pezzi tagliati nelle pile di scarico, funzione non attiva negli esperimenti di questa tesi. L'area prodotta è sempre almeno pari all'area domandata, e la supera in presenza di sovraproduzione. Lo scarto è la percentuale dell'area utilizzata non ripartita in pezzi, calcolata al netto dello spessore della lama e del rifilo dei bordi del pannello; negli esperimenti di questa tesi entrambi sono nulli, e lo scarto coincide con la definizione adottata nel Capitolo 4 (Sezione 4.2).

Gli indicatori misurano aspetti distinti della soluzione, che in generale non sono ottimizzati tutti dalla stessa soluzione. Con più tipi di pannello uno stesso numero di pannelli può corrispondere ad aree diverse; in presenza di sovraproduzione, un'area maggiore può accompagnarsi a uno scarto minore, poiché il materiale aggiuntivo è occupato da pezzi non richiesti [5]. Se invece la produzione eguaglia la domanda, l'area dei pezzi prodotti è fissata e minimizzare l'area utilizzata equivale a minimizzare lo scarto.

Il confronto tra le soluzioni si basa sulla dominanza di Pareto. Una soluzione s *domina* una soluzione s' se non è peggiore di s' su alcun indicatore ed è strettamente migliore su almeno uno; una soluzione è *non dominata* se nessun'altra la domina. L'insieme delle soluzioni non dominate costituisce la *frontiera di Pareto*: ciascuna delle sue soluzioni rappresenta un compromesso tra gli indicatori, poiché nessuna delle altre la migliora su un indicatore senza peggiorarla su un altro. Ogni soluzione costruita dall'euristica è inserita in un *pool* se non è dominata da quelle già presenti, e al termine dell'esecuzione il *pool* è ridotto alla frontiera delle soluzioni non dominate [5].

La Tabella 2.1 ne riporta un esempio. La dominanza così definita, interna al *solver*, confronta le soluzioni di una stessa istanza sugli indicatori elencati; lo stesso concetto è impiegato nel Capitolo 4 per confrontare le configurazioni del *solver* rispetto allo scarto e al tempo di calcolo (Sezione 4.4.7).

Tabella 2.1.: Esempio illustrativo, con valori fittizi, di tre soluzioni non dominate per un'istanza con area domandata pari a 114 m^2. In grassetto il valore migliore di ciascun indicatore. Nessuna soluzione è migliore delle altre su tutti gli indicatori: con il peso interamente sull'area utilizzata il *solver* restituisce s_1, con il peso sullo scarto s_2, con il peso sul numero di schemi s_3. Lo scarto minimo di s_2 è ottenuto con la sovraproduzione.

Soluzione	Area utilizzata (m^2)	Area prodotta (m^2)	Scarto (%)	Schemi distinti
s_1	**120,0**	**114,0**	5,0	12
s_2	126,0	120,7	**4,2**	9
s_3	124,0	115,9	6,5	**6**

Quando la disponibilità dei pannelli non consente di soddisfare l'intera domanda, il *pool* può contenere soltanto soluzioni parziali; in tal caso sono conservate quelle che soddisfano la frazione di domanda più alta, detta livello di servizio [5], e a parità di livello di servizio valgono le stesse regole di dominanza. Le soluzioni parziali sono eliminate non appena si ottiene una soluzione completa. Nelle istanze di questa tesi la disponibilità dei pannelli è illimitata e il caso non si presenta.

Dal *pool* il *solver* estrae infine la soluzione da restituire: quella che minimizza una combinazione lineare degli indicatori a coefficienti non negativi, i cui pesi sono parametri del *solver*. A parità di valore della combinazione prevale la soluzione migliore secondo un ordine prefissato degli indicatori: costo, area utilizzata, area prodotta, scarto, numero di pannelli, numero di schemi e numero di pretagli. I pesi determinano quindi quale compromesso della frontiera viene restituito, come mostra la Tabella 2.1. Nei test dell'articolo di riferimento la soluzione restituita è quella non dominata di area utilizzata minima [5]; negli esperimenti di questa tesi il peso è attribuito interamente allo scarto, e il *solver* restituisce la soluzione non dominata di scarto minimo.

2.5. Le fasi di ottimizzazione

L'esecuzione del *solver* si articola in tre fasi successive, ciascuna rivolta a un indicatore di prestazione diverso e realizzata mediante l'euristica dell'Algoritmo 1 [5]. La prima fase è sempre eseguita, mentre la seconda e la terza sono facoltative. Le soluzioni prodotte da tutte le fasi confluiscono nello stesso *pool*, al quale si applicano le regole di dominanza e di scelta della Sezione 2.4. A ciascuna fase è assegnato un tempo massimo di esecuzione, pari negli esperimenti di questa tesi a 60 secondi.

La prima fase minimizza l'area utilizzata, e quindi lo scarto, con la domanda non raggruppata, salvo la gestione a pacco [5]. Essa esegue le N iterazioni dell'euristica descritte nella Sezione 2.2, e termina in anticipo se l'area utilizzata dalla soluzione corrente eguaglia il limite duale, cioè il limite inferiore all'area necessaria per soddisfare la domanda: in tal caso nessuna soluzione può utilizzare un'area minore, e proseguire la ricerca non migliorerebbe l'indicatore. Ulteriori criteri di arresto, basati su soglie fissate dall'utente per il numero di pannelli e per l'area utilizzata, non sono attivi negli esperimenti di questa tesi.

La seconda fase riduce il numero di schemi di taglio distinti raggruppando la domanda in pacchi di dimensione crescente [5]. Quando la domanda è raggruppata, ogni schema viene ripetuto su un numero di pannelli multiplo della dimensione del pacco, e la stessa domanda è soddisfatta con un numero minore di schemi distinti. La fase procede per passi: il primo impiega pacchi di dimensione 2, e a ogni passo successivo la dimensione è scelta come il più grande intero compreso tra la dimensione corrente aumentata di uno e la domanda massima dei *part-type* che minimizza l'area della domanda residua, cioè della parte che non costituisce un pacco intero. Ogni passo esegue N iterazioni dell'euristica, e la fase termina quando un passo non produce nuove soluzioni non dominate, quando il numero di schemi raggiunge il relativo limite duale oppure dopo un numero massimo di passi. Poiché le soluzioni della seconda fase entrano nello stesso *pool* della prima e la soluzione restituita è quella di scarto minimo, l'attivazione di questa fase non può peggiorare lo scarto della soluzione restituita, e lo migliora quando uno dei passi produce una soluzione con scarto inferiore.

La terza fase riduce il numero di pretagli: la soglia di opportunità δ introdotta nella Sezione 2.3 è aumentata progressivamente, rendendo il pretaglio sempre meno conveniente, finché le soluzioni generate non ne contengono più [5]. Negli esperimenti di questa tesi questa fase è disattivata.

Il *solver* registra due misure di tempo: il tempo totale di calcolo e il tempo impiegato a trovare la soluzione restituita. Le due misure differiscono perché, dopo aver trovato tale soluzione, l'algoritmo prosegue con le iterazioni e con le fasi previste, a meno che non intervenga l'arresto anticipato della prima fase o il tempo massimo di una fase. La scelta tra le due misure di tempo ai fini dell'analisi è discussa nella Sezione 4.2.

2.6. Formato di input e output

Il *solver* è realizzato come libreria di funzioni. Negli esperimenti di questa tesi è stato invocato da un programma a riga di comando, una volta per ciascuna coppia formata da un'istanza e da un file di parametri. Ogni esecuzione riceve tre file di testo: la descrizione dell'istanza, i parametri dell'algoritmo e la configurazione delle stazioni di *stacking*, cioè delle aree in cui i pezzi tagliati vengono depositati.

Quest'ultimo file è richiesto dall'interfaccia ma non ha effetto sulle soluzioni, poiché la funzione di *stacking* è disattivata.

Il file dell'istanza si apre con una riga che riporta il numero di tipi di pannello, il numero di *part-type* e il numero di pezzi destinati esclusivamente al riempimento degli spazi residui, nullo nelle istanze di questa tesi. Seguono il blocco `STOCKS`, con una riga per ciascun tipo di pannello, e il blocco `PARTS`, con una riga per ciascun *part-type*; all'interno di una riga i campi sono separati da virgole. Il significato dei campi e i valori assunti nelle istanze di questa tesi sono riportati nella Tabella 2.2. Ne è riportato di seguito un estratto, relativo a un'istanza con un solo tipo di pannello:

```
1, 52, 0
STOCKS
1                   , 4000, 4000,   10, -1, 2, 0, 0, 0
PARTS
1                   ,  417,  443, 0,  9, 0, 0, 0, 0, -1, 0, 0, -1
2                   ,  417,  470, 0,  7, 0, 0, 0, 0, -1, 0, 0, -1
...
```

Il file è prodotto dal generatore descritto nel Capitolo 3, che ne determina le dimensioni, le domande e la direzione del primo taglio.

Tabella 2.2.: Campi del file dell'istanza e valori assunti nelle istanze di questa tesi.

Blocco	Campo	Significato	Valore
`STOCKS`	codice	identificativo del tipo di pannello	progressivo
	larghezza, altezza	dimensioni del pannello in mm	dal generatore
	spessore	spessore del pannello in mm	10
	disponibilità	numero di pannelli disponibili (-1: illimitata)	-1
	primo taglio	0: indifferente, 1: solo orizzontale, 2: solo verticale	0 o 2
	materiale	codice del materiale	0
	residuo	0: pannello ordinario, 1: residuo di lavorazioni precedenti	0
	costo	costo unitario del pannello	0
`PARTS`	codice	identificativo del *part-type*	progressivo
	larghezza, altezza	dimensioni del pezzo in mm	dal generatore
	sottoproduzione	pezzi in meno ammessi rispetto alla domanda	0
	domanda	numero di pezzi richiesti	dal generatore
	sovraproduzione	pezzi in più ammessi rispetto alla domanda	0
	venatura	0: rotazione ammessa, 1: rotazione esclusa	0
	tipo di scarico	-1: pezzo non gestito dalle stazioni di scarico	-1
	altri campi	materiale, commessa, destinazione e raggruppamento dei pezzi	costanti

Il file dei parametri elenca un parametro per volta su due righe: la prima contiene il nome, preceduto dal carattere `p`, la seconda il valore. Le righe di commento sono ignorate, e i parametri non specificati assumono un valore predefinito. La Tabella 2.3 riporta i parametri rilevanti per questa tesi: i sei parametri variati nel disegno sperimentale, i cui livelli corrispondono ai fattori algoritmici della Tabella 4.1, e i principali parametri mantenuti fissi in tutte le esecuzioni.

Tabella 2.3.: Parametri del *solver* rilevanti per questa tesi. Nella parte superiore i parametri variati nel disegno sperimentale, nella parte inferiore quelli mantenuti fissi.

Parametro	Significato	Valore
`iterSVC`, `iterSVC_p`	cicli e ripetizioni dell'euristica; N è il loro prodotto	3 o 20
`patternType`	tipo di schema: 2 a 2 stadi, 16 a 3 stadi	2 o 16
`reduce_setups`	attivazione della fase di minimizzazione degli schemi	0 o 1
`lotsize`	dimensione del pacco (-1: gestione a pacco assente)	-1 o 5
`automatic_overproduction`	regolazione automatica della sovraproduzione	0 o 1
`greedyKnap`	risoluzione dei *knapsack*: 0 esatta, 1 *greedy*	0 o 1
`maxRunningTime`	tempo massimo di ciascuna fase, in secondi	60
`precut`	pretaglio (3: ammesso in entrambi gli orientamenti)	3
`reduce_precuts`	attivazione della fase di riduzione dei pretagli	0
`coeff_waste_perc`	peso dello scarto nella scelta della soluzione; gli altri pesi sono nulli	1
`seed`	seme delle estrazioni casuali	0
`bladeThickness`, `croppingWidth_X`, `croppingWidth_Y`	spessore della lama e rifilo dei bordi, in mm	0
`outbuffers_automatic`, `outbuffers_manual`	limite sugli *open stack* (-1: illimitato)	-1
`min_numCuts`, `min_obj_val`, `min_numpatterns`	soglie di arresto anticipato sul numero di pannelli, sull'area utilizzata e sul numero di schemi (0: non attive)	0
`filling`, `Stack_management`	riempimento con pezzi dedicati e *stacking*	0

Al termine dell'esecuzione il *solver* scrive un file con gli schemi di taglio della soluzione restituita e, se richiesto, un file per ciascuna soluzione non dominata del *pool*. Ogni esecuzione aggiunge inoltre un *record* a un file di riepilogo, con campi separati da punto e virgola, che riporta:

- gli indicatori della soluzione restituita, tra cui l'area utilizzata (`usedArea`), l'area prodotta (`producedArea`), lo scarto (`waste`), il numero di pannelli, di schemi e di pretagli e il numero di pezzi prodotti oltre la domanda (`pzOverProd`);
- l'area domandata (`PartsTotNetArea`), i limiti duali e il livello di servizio;

- il tempo totale di calcolo (`tot_cputime`) e il tempo impiegato a trovare la soluzione restituita (`cputime`);
- il valore migliore di ciascun indicatore tra le soluzioni del *pool*;
- una copia dei valori di tutti i parametri impiegati.

Da questi *record* derivano i dati analizzati nel Capitolo 4: lo scarto corrisponde al campo `waste`, il tempo di calcolo al campo `tot_cputime`, e lo scarto effettivo è calcolato dall'area utilizzata e dall'area domandata (Sezione 4.4.4). La copia dei parametri consente inoltre di verificare, per ogni esecuzione, la configurazione effettivamente impiegata.

2.7. Ruolo nel contesto della tesi

Nell'articolo di riferimento la qualità delle soluzioni è valutata, su un insieme di 52 istanze di *benchmark*, mediante il limite continuo dato dall'area complessiva dei pezzi domandati: quando l'area utilizzata eguaglia il limite, l'ottimalità della soluzione rispetto all'impiego di materiale è certificata. Ciò avviene nell'80,8% delle istanze con il pretaglio disattivato e nell'82,7% con il pretaglio ammesso [5]. Nei casi restanti la distanza dall'ottimo resta incognita, per le ragioni discusse nella Sezione 1.5. Le istanze a scarto nullo, descritte nel Capitolo 3 e collocate rispetto alle istanze di test della letteratura nella Sezione 1.7, eliminano questa incertezza: il loro ottimo ha scarto nullo per costruzione, e lo scarto della soluzione restituita dal *solver* ne misura la distanza dall'ottimo.

La descrizione di questo capitolo fornisce inoltre il riferimento per i fattori algoritmici analizzati nel Capitolo 4. Lo sforzo SVC corrisponde al numero N di iterazioni dell'euristica e la gestione a pacco al raggruppamento della domanda (Sezione 2.2); il tipo di schema di taglio e la modalità di risoluzione dei *knapsack* intervengono nella generazione degli schemi (Sezione 2.3); la sovraproduzione modifica il vincolo di domanda (Sezione 2.1) e il confronto tra le soluzioni (Sezione 2.4); la minimizzazione degli schemi corrisponde alla seconda fase dell'algoritmo (Sezione 2.5). Il capitolo successivo descrive il generatore con cui sono costruite le istanze a scarto nullo impiegate negli esperimenti.

Capitolo 3.

Generazione di istanze a scarto nullo

Il presente capitolo descrive il generatore di istanze a scarto nullo sviluppato nell'ambito di questo lavoro di tesi. Viene dapprima formalizzato il concetto di istanza a scarto nullo e ne viene motivata l'utilità ai fini della validazione sperimentale; seguono la descrizione dei parametri configurabili del generatore e la presentazione degli algoritmi di generazione, articolata nel ciclo principale di controllo, nelle due strategie di suddivisione dei pannelli e nella procedura di taglio monodimensionale. Il capitolo si chiude con la descrizione del formato dei file prodotti.

3.1. Definizione di istanza a scarto nullo

Nel contesto del problema del 2D *Multiple Bin Packing* con tagli a ghigliottina, definito nel Capitolo 1 e considerato nella variante con rotazione dei pezzi (Sezione 1.3), si definisce *istanza a scarto nullo* un'istanza per la quale esiste almeno un piano di taglio ammissibile che ripartisce interamente in pezzi i pannelli impiegati, senza lasciare materiale inutilizzato. La condizione richiede che l'area utilizzata (1.3) sia uguale all'area dei pezzi domandati (1.4):

$$\sum_{i \in \mathcal{S}} W_i H_i \, n_i = \sum_{j \in \mathcal{R}} d_j \, w_j h_j, \tag{3.1}$$

dove n_i è il numero di pannelli di tipo i impiegati dal piano e le altre grandezze sono quelle della Sezione 1.2.

Come osservato nella Sezione 1.2, una soluzione che soddisfa la (3.1) è ottima, e il valore ottimo dello scarto è noto e pari a zero: il piano di taglio utilizzato per generare l'istanza ne costituisce un esempio. Questa proprietà rende le istanze a scarto nullo utili nella validazione di algoritmi di ottimizzazione: la qualità della soluzione prodotta dal *solver* può essere confrontata direttamente con l'ottimo, senza ricorrere a tecniche aggiuntive di verifica.

Il generatore costruisce le istanze percorrendo il problema a ritroso: anziché ricercare uno schema di taglio a partire da un insieme di pezzi richiesti, si parte dal pannello intero e lo si suddivide, con tagli successivi, in strisce, colonne e pezzi compatibili con i vincoli imposti, accumulando i *part-type* prodotti in un contatore. Questo approccio garantisce per costruzione il soddisfacimento della condizione (3.1).

3.2. Parametri del generatore

Il generatore riceve in ingresso un file di testo strutturato in sezioni, ciascuna introdotta da un'etichetta che ne indica il tipo. Di seguito è riportato un esempio di file di parametri:

```
panels
4800 2400 0
2400 1200 1

limits
400 300 0

quantity
10 0.5

demand
5

cut
1
```

Le sezioni e i relativi parametri sono descritti di seguito.

`panels` Definisce i tipi di pannello disponibili. Ogni riga della sezione specifica un tipo di pannello tramite tre variabili: larghezza, altezza e *flag* di direzione del primo taglio. Il *flag* può assumere il valore 0 (direzione del primo taglio scelta casualmente, con probabilità uguale tra orizzontale e verticale), 1 (primo taglio orizzontale) oppure 2 (primo taglio verticale). Poiché l'algoritmo esegue il primo taglio sempre in direzione orizzontale, il primo taglio verticale viene ottenuto scambiando larghezza e altezza del pannello prima di applicare la suddivisione.

`limits` Definisce i vincoli sulle dimensioni dei pezzi. La sezione contiene un'unica riga con tre variabili: il limite orizzontale, il limite verticale e un *flag* che specifica se i valori rappresentano dimensioni minime (0) o massime (1). Nella modalità a dimensioni minime ogni pezzo deve avere larghezza almeno pari al limite orizzontale e altezza almeno pari al limite verticale; nella modalità a dimensioni massime le disuguaglianze si invertono.

`quantity` Specifica il numero desiderato di *part-type* distinti da produrre nell'istanza. La sezione contiene un intero q e un valore reale τ che rappresentano rispettivamente il numero obiettivo e la tolleranza relativa: il generatore si ritiene soddisfatto quando il numero di *part-type* distinti prodotti è compreso nell'intervallo $[q \cdot (1 - \tau),\ q \cdot (1 + \tau)]$.

`demand` Definisce un fattore moltiplicativo applicato uniformemente alle domande di tutti i *part-type*. Se un *part-type* compare n volte negli schemi di taglio prodotti, la sua domanda nel file di output è $n \cdot d$, dove d è il valore di questa sezione. Il parametro consente di scalare la dimensione dell'istanza senza modificare la struttura degli schemi di taglio. In assenza della sezione, il valore predefinito è 1.

`cut` Specifica il tipo di taglio a ghigliottina: 0 per il taglio a 2 stadi, 1 per il taglio a 3 stadi.

3.3. Algoritmo di generazione

L'algoritmo è suddiviso in quattro componenti: il ciclo principale di controllo, le due funzioni di generazione degli schemi di taglio e la procedura di taglio monodimensionale. Di seguito ciascuno viene descritto; il codice sorgente è riportato nell'Appendice A.

3.3.1. Ciclo principale

Il ciclo principale gestisce la generazione iterativa degli schemi di taglio e la politica di accettazione degli schemi di taglio prodotti. Vengono mantenuti due contatori: `total_pieces`, che raccoglie i *part-type* accettati fino all'iterazione corrente, e `test`, copia temporanea usata per verificare il numero di *part-type* che risulterebbe dall'accettazione dello schema corrente.

A ogni iterazione si seleziona casualmente un tipo di pannello tra quelli disponibili. In base al *flag* di direzione del primo taglio, si determina l'orientamento del pannello; nella modalità a dimensioni minime, se il pannello così orientato è più piccolo dei limiti, la scelta viene ripetuta. Lo schema di taglio viene quindi generato richiamando `generate_first` finché `total_pieces` è vuoto, cioè fino all'accettazione del primo schema, e `generate_next` per i pannelli successivi.

I *part-type* prodotti vengono accumulati in `test`. Se il numero di *part-type* distinti in `test` non supera il limite superiore della tolleranza, lo schema è accettato: `total_pieces` viene aggiornato e lo schema viene incluso nell'output grafico. Altrimenti lo schema viene scartato, il contatore dei tentativi falliti viene incrementato e si aggiorna il registro ausiliario `best_attempt`, che conserva il contenuto di `test` con il minor numero di *part-type* distinti tra tutti i tentativi falliti. Il contatore non viene azzerato dalle accettazioni: quando un tentativo fallisce dopo che ne sono già falliti 30, il ciclo viene interrotto e il contenuto di `best_attempt` viene adottato come soluzione finale. Questo meccanismo di uscita evita cicli potenzialmente infiniti nei casi in cui il vincolo di tolleranza risulti incompatibile con la variabilità introdotta dai nuovi pannelli. Poiché `best_attempt` è formato dagli schemi accettati fino al tentativo corrispondente e da uno schema completo, l'istanza adottata resta a scarto

nullo; in questo caso, tuttavia, l'output grafico può non coincidere con gli schemi che la compongono.

Al termine del ciclo viene scritto il file di istanza e il file PDF viene chiuso.

3.3.2. Generazione del primo schema di taglio

La funzione `generate_first` genera uno schema di taglio per un singolo pannello senza riutilizzare *part-type* preesistenti, seguendo la struttura classica del taglio a ghigliottina a 2 o 3 stadi.

L'altezza totale del pannello viene suddivisa in strisce orizzontali tramite un primo taglio. Per ciascuna striscia, la larghezza viene suddivisa indipendentemente in colonne tramite un secondo taglio. Se il tipo di taglio è 3 stadi, ogni colonna viene ulteriormente suddivisa in altezza, ottenendo i singoli pezzi; se il tipo di taglio è 2 stadi, ogni colonna viene considerata come se fosse stata suddivisa in un unico pezzo, allo scopo di mantenere uniforme il formato di ritorno. La struttura dei tre stadi di taglio è illustrata nella Figura 3.1.

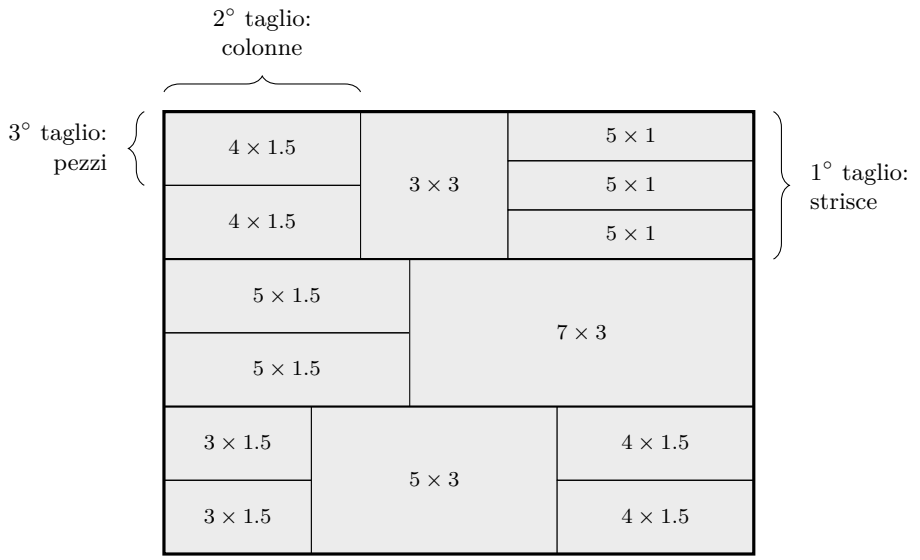


Figura 3.1.: Schema esemplificativo di un pannello generato da `generate_first` (taglio a 3 stadi). Il primo taglio suddivide l'altezza del pannello in strisce orizzontali; il secondo taglio suddivide la larghezza di ciascuna striscia in colonne; il terzo taglio suddivide l'altezza di ciascuna colonna nei singoli pezzi, etichettati con le dimensioni $w \times h$.

La funzione restituisce due strutture, `piece_width` e `piece_height`: `piece_width[i][j]` è la larghezza della colonna j nella striscia i; `piece_height[i][j][k]` è l'altezza del pezzo k all'interno di quella colonna. Queste strutture annidate sono consumate dal ciclo principale per l'accumulo dei *part-type* e dal modulo di output grafico per la rappresentazione degli schemi.

3.3.3. Generazione degli schemi di taglio successivi

La funzione `generate_next` estende la logica di `generate_first` con il riutilizzo di *part-type* già presenti in `total_pieces`. Il riutilizzo aumenta la condivisione di *part-type* tra schemi di taglio distinti: in assenza di riutilizzo la maggior parte dei *part-type* comparirebbe in un unico schema, mentre con il riutilizzo gli schemi condividono *part-type* tra loro con maggiore frequenza.

Per ciascun pannello viene estratto casualmente, nell'intervallo $[0, 5]$, il numero di tentativi di riutilizzo; a ogni tentativo si sceglie a caso un *part-type* tra quelli già presenti, eventualmente lo stesso più volte. Per il *part-type* scelto si valutano i due orientamenti possibili (originale e ruotato di 90°), scartando quelli che non rispettano i vincoli di dimensione o che eccedono la larghezza del pannello o lo spazio verticale residuo; tra gli orientamenti ammissibili se ne sceglie uno a caso. Se nessun orientamento risulta ammissibile, il *part-type* viene ignorato. Se l'altezza residua del pannello scende al di sotto del 25% dell'altezza totale, il ciclo di riutilizzo viene interrotto anticipatamente: questo vincolo limita l'altezza occupata dai *part-type* riutilizzati, così da evitare situazioni in cui il ciclo principale non riesca a superare la soglia minima di *part-type* distinti. Nella modalità a dimensioni minime è presente un ulteriore controllo: dopo aver assegnato la striscia di riutilizzo deve restare almeno uno spazio verticale pari alla dimensione minima, affinché la successiva chiamata a `generate_first` possa produrre almeno una striscia ammissibile.

Per ciascun *part-type* accettato, si determina quante copie affiancarne orizzontalmente (nell'intervallo $[1, 5]$, troncato se la larghezza disponibile non è sufficiente) e si costruisce la striscia corrispondente: la porzione sinistra è riempita con pezzi nuovi secondo la logica di `generate_first`, mentre la porzione destra contiene le copie del *part-type* riutilizzato. Al termine del ciclo di riutilizzo, l'altezza ancora libera viene riempita richiamando `generate_first` come se fosse un pannello autonomo, e le strisce di riutilizzo vengono accodate al risultato; la struttura complessiva è illustrata nella Figura 3.2.

3.3.4. Taglio monodimensionale

Il modulo `cutter.py` implementa le funzioni `cut_min` e `cut_max`, che suddividono un segmento di lunghezza intera in pezzi rispettando rispettivamente un vincolo di dimensione minima e uno di dimensione massima. Entrambe le funzioni adottano un meccanismo di *bias* per orientare la distribuzione delle dimensioni verso valori vicini al limite del vincolo.

In `cut_min`, con dimensione minima $n_{\min}$, la dimensione di ciascun pezzo viene determinata come

$$n = n_{\min} + \lfloor r \cdot (l - n_{\min}) \rfloor, \tag{3.2}$$

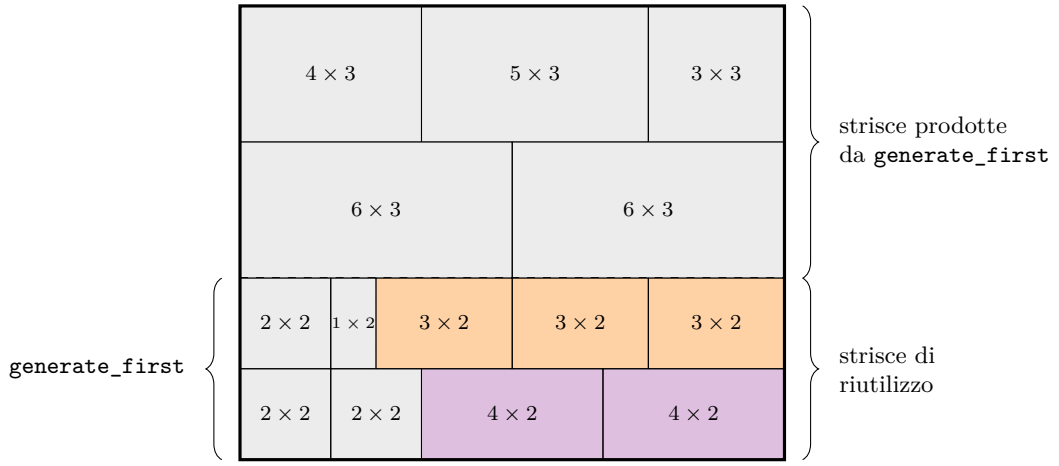


Figura 3.2.: Schema esemplificativo di un pannello generato da `generate_next` (taglio a 2 stadi). La porzione superiore è prodotta da `generate_first` sull'altezza residua. Ogni striscia di riutilizzo contiene, nella porzione destra, copie identiche di un *part-type* preesistente (evidenziate con lo stesso colore) e, nella porzione sinistra, pezzi nuovi.

dove l è la lunghezza residua del segmento e $r = U^b$, con $U \sim \mathcal{U}[0, 1)$ e $b = 3$. L'elevamento alla terza concentra r vicino a 0, orientando n verso $n_{\min}$: senza questo *bias*, una scelta uniforme su $[n_{\min}, l]$ produrrebbe spesso al primo taglio un pezzo di dimensione prossima alla metà della lunghezza disponibile, portando a schemi con pochi *part-type* e scarsa varietà. Quando la lunghezza residua scende al di sotto di $n_{\min}$, il residuo viene assorbito nell'ultimo pezzo emesso, incrementandone la dimensione: poiché non esiste un limite superiore, il pezzo risultante rimane valido, e questa è la proprietà che garantisce lo scarto nullo nella modalità a dimensioni minime.

In `cut_max`, con dimensione massima $n_{\max}$, la dimensione viene determinata come

$$n = n_{\max} - \lfloor r \cdot (n_{\max} - 1) \rfloor, \tag{3.3}$$

con il medesimo meccanismo di *bias* a segno invertito: n tende ora a $n_{\max}$, evitando la produzione frequente di pezzi di dimensione unitaria che si verificherebbe con una scelta uniforme su $[1, n_{\max}]$. In questo caso non esiste un limite inferiore, quindi il residuo finale viene accettato come pezzo autonomo. Le due modalità sono illustrate nella Figura 3.3.

In entrambe le funzioni i tratti sono emessi in gruppi consecutivi di uguale lunghezza, con la dimensione del gruppo scelta casualmente nell'intervallo $[1, 5]$.

3.4. Formato di output

Al termine della generazione il software produce due tipi di output: un file di testo con la descrizione dell'istanza (che verrà poi usato come input per il *solver*) e un

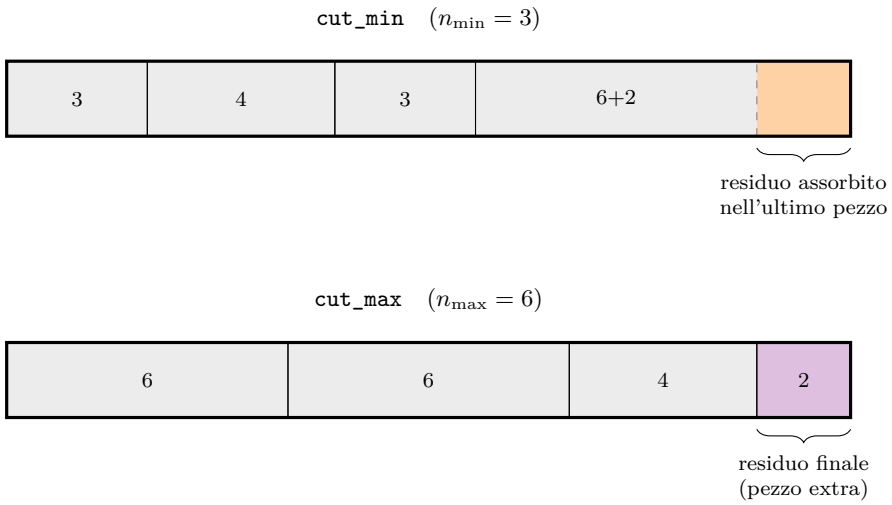


Figura 3.3.: Esempio di taglio monodimensionale a scarto nullo. In `cut_min` ogni pezzo rispetta la dimensione minima $n_{\min}$ e il residuo inferiore a $n_{\min}$ viene assorbito nell'ultimo pezzo, che si ingrandisce (evidenziato in arancione). In `cut_max` ogni pezzo rispetta la dimensione massima $n_{\max}$ e il residuo finale è emesso come pezzo autonomo (evidenziato in viola). In entrambi i casi la lunghezza totale del segmento è ripartita senza scarto.

file PDF con la rappresentazione grafica degli schemi di taglio (per permettere la consultazione degli schemi generati all'utente).

Il file di testo segue il formato di input del *solver* descritto nella Sezione 2.6, dove la Tabella 2.2 riporta il significato dei campi e i valori assunti nelle istanze generate. Il generatore riporta le dimensioni e la direzione del primo taglio di ciascun tipo di pannello, ricevute in ingresso, e determina le dimensioni e la domanda di ciascun *part-type*, calcolata come prodotto tra il numero di occorrenze del *part-type* negli schemi generati e il parametro `demand`; gli altri campi assumono valori costanti. I *part-type* sono ordinati per domanda decrescente e, a parità di domanda, per dimensioni crescenti. Un *part-type* e la sua versione ruotata sono considerati lo stesso *part-type*, le cui dimensioni sono riportate con la larghezza non superiore all'altezza.

Il file PDF è prodotto dal modulo `image_generator.py` tramite la libreria *matplotlib*. Ogni pagina del documento corrisponde a un pannello e ne mostra lo schema di taglio generato: i pezzi sono rappresentati come rettangoli con etichetta che riporta l'indice progressivo e le dimensioni. L'asse delle ordinate è invertito in modo che l'origine si trovi in alto a sinistra, coerentemente con l'ordine di accumulo delle strisce nell'algoritmo.

Capitolo 4.

Risultati computazionali

Il presente capitolo analizza il comportamento del *solver* descritto nel Capitolo 2 quando applicato alle istanze a scarto nullo generate con il metodo del Capitolo 3. Come stabilito nella Sezione 1.2, l'ottimo di queste istanze ha scarto nullo, poiché nessuna soluzione può avere scarto negativo; per questo motivo lo scarto prodotto dal *solver* corrisponde alla sua distanza dall'ottimo. Viene prima illustrato il disegno sperimentale, con la distinzione tra fattori di istanza e fattori algoritmici; seguono la definizione delle metriche di valutazione e la presentazione degli strumenti statistici impiegati. La parte centrale del capitolo riporta i risultati, organizzati per tema, assieme alla loro spiegazione. Chiudono il capitolo una sintesi e alcune conclusioni sulle configurazioni del *solver* che l'analisi suggerisce essere più efficaci o efficienti.

4.1. Disegno sperimentale

L'esperimento sottopone il *solver* a un insieme di istanze a scarto nullo, facendo variare in modo sistematico sia le caratteristiche del problema sia la configurazione dell'algoritmo. Le variabili controllate si dividono in due categorie:

Fattori di istanza sono i parametri forniti in ingresso all'algoritmo di generazione descritto nel Capitolo 3 e determinano quindi l'istanza prodotta, caratterizzando il problema da risolvere. Corrispondono alle sezioni del file di parametri del generatore: la configurazione di pannelli (`panels`), il numero di *part-type* distinti (`quantity`), il fattore di domanda (`demand`) e il tipo di taglio a ghigliottina con cui l'istanza è generata (`cut`). Il limite dimensionale dei pezzi (`limits`) non è scelto in modo indipendente, ma calcolato dalle dimensioni dei pannelli, come descritto più avanti.

Fattori algoritmici sono i parametri che controllano il comportamento del *solver*: lo sforzo dell'euristica *SVC*, il tipo di schema di taglio da utilizzare, l'attivazione della fase di minimizzazione degli schemi, la gestione a pacco, la sovraproduzione e la modalità di risoluzione del sottoproblema *knapsack*.

I livelli assunti da ciascun fattore sono riportati nella Tabella 4.1.

Le quattro configurazioni di pannelli sono riportate nella Tabella 4.2. Le prime due impiegano un solo tipo di pannello, le altre due un insieme di tre tipi; queste ultime

Tabella 4.1.: Fattori controllati nell'esperimento e relativi livelli.

Blocco	Fattore	Livelli
Istanza	configurazione di pannelli	configurazioni A, B, C, D
	numero di *part-type*	10, 50, 100
	fattore di domanda	1, 20
	tipo di taglio dell'istanza	2 stadi, 3 stadi
Algoritmico	sforzo SVC	3, 20
	tipo di schema di taglio	2 stadi, 3 stadi
	minimizzazione degli schemi	disattivata, attivata
	gestione a pacco (fogli)	assente, 5
	sovraproduzione	disattivata, attivata
	risoluzione *knapsack*	esatta, *greedy*

differiscono unicamente per la direzione del primo taglio. Il limite dimensionale dei pezzi è fissato, nel file di parametri del generatore, a un decimo della media delle dimensioni dei pannelli della configurazione, ed è quindi determinato da quest'ultima. In tutte le istanze i limiti sono intesi come dimensioni minime dei pezzi, e la tolleranza sul numero di *part-type* distinti è pari a 0,5 (Capitolo 3).

Tabella 4.2.: Le quattro configurazioni di pannelli impiegate. La direzione del primo taglio è verticale (V) oppure scelta casualmente (L). Il limite (w, h) è pari a un decimo della media delle dimensioni dei pannelli della configurazione.

Config.	Pannelli ($W \times H$, mm)	Primo taglio	Limite (w, h)
A	4000×4000	V	$(400, 400)$
B	2000×4000	V	$(200, 400)$
C	2500×3000, 2000×4000, 1500×5000	V	$(200, 400)$
D	2500×3000, 2000×4000, 1500×5000	L	$(200, 400)$

I fattori algoritmici sono stati combinati in tutti i modi possibili, dando luogo a un fattoriale completo di $2^6 = 64$ configurazioni. Analogamente, sono state impiegate tutte le combinazioni dei fattori di istanza variati liberamente: le quattro configurazioni di pannelli, i tre valori del numero di *part-type*, i due del fattore di domanda e i due del tipo di taglio, per un totale di $4 \times 3 \times 2 \times 2 = 48$ configurazioni di istanza. Ciascuna configurazione di istanza è stata infine risolta con ciascuna configurazione algoritmica: sono state pertanto eseguite tutte le $48 \times 64 = 3072$ combinazioni di configurazione, ottenendo un disegno fattoriale completo e bilanciato. A ogni combinazione corrispondono dieci istanze distinte, generate dalla stessa configurazione di istanza mediante l'algoritmo di generazione stocastico del Capitolo 3 e risolte separatamente, per un totale di 30 720 esecuzioni; di ciascuna metrica si considera il valore medio sulle soluzioni restituite dal *solver* (Sezione 2.4).

La natura completa e bilanciata del disegno è la proprietà su cui si basano gli strumenti statistici della Sezione 4.3: la presenza, per ciascun fattore, di configurazioni che differiscono nel solo fattore considerato consente di valutarne l'effetto isolato.

Due fattori richiedono una precisazione sul modo in cui sono trattati nell'analisi. I due parametri che determinano lo sforzo *SVC* (Sezione 2.2) assumono nella campagna sperimentale sempre lo stesso valore: il loro effetto non è separabile e sono analizzati come un unico fattore, i cui livelli 3 e 20 corrispondono rispettivamente a 9 e a 400 iterazioni. La configurazione di pannelli, a sua volta, non deriva da un incrocio libero delle dimensioni ma da una lista fissa, e il limite dimensionale di taglio è calcolato dalle dimensioni dei pannelli: dimensioni e limite non sono quindi fattori indipendenti, e la configurazione è analizzata come un unico fattore di istanza categoriale (Sezione 4.3). Queste scelte costituiscono limiti dello studio, discussi con gli altri nella conclusione (Limiti).

I parametri del *solver* non variati nel disegno sono mantenuti fissi in tutte le esecuzioni (Tabella 2.3). Tra questi, il pretaglio è ammesso e la fase di riduzione dei pretagli è disattivata: lo schema con pretaglio è quindi adottato ogni volta che risulta più conveniente (Sezione 2.3). Il 58% delle soluzioni restituite contiene almeno un pretaglio, il 64% con lo schema a 2 stadi e il 52% con quello a 3 stadi. Il confronto tra i due tipi di schema di taglio riguarda pertanto due modalità del *solver* con pretaglio ammesso, e non schemi a 2 e a 3 stadi in senso stretto.

4.2. Metriche di valutazione

La valutazione si concentra su due metriche principali, che rispondono alle due domande dell'analisi: quanto è buona la soluzione prodotta e quanto costa ottenerla.

La prima metrica è lo *scarto*, definito come la percentuale di area dei pannelli non ripartita in pezzi: uno scarto del 10% indica che un decimo del materiale impiegato resta inutilizzato. Trattandosi di istanze a scarto ottimo nullo, tale valore misura direttamente la distanza percentuale della soluzione dall'ottimo; quando è attiva la sovraproduzione questa lettura richiede una correzione, discussa nella Sezione 4.4.4. La seconda metrica è il *tempo di calcolo* totale impiegato dal *solver* per completare l'esecuzione, espresso in secondi. Le due misure di tempo registrate dal *solver* (Sezione 2.5) sono correlate ($\rho = 0{,}85$), ma il tempo totale è in mediana circa tre volte maggiore di quello alla migliore soluzione. Come metrica si adotta il tempo totale, che rappresenta il costo sostenuto da chi utilizza il *solver*; il tempo alla migliore soluzione è richiamato soltanto dove conduce a conclusioni diverse (Sezioni 4.4.4 e 4.4.5). Il tempo totale è inoltre limitato dal tempo massimo assegnato a ciascuna fase, pari a 60 secondi (Sezione 2.5). Il limite è raggiunto con certezza in 684 esecuzioni, il 2,2% del totale: in 346, prive della seconda fase, il tempo totale si arresta a circa 60 secondi; in 338 entrambe le fasi si arrestano al limite, con un tempo totale di circa 120 secondi. In queste esecuzioni il tempo registrato sottostima quello che il *solver* impiegherebbe in assenza del limite, e lo scarto restituito può essere maggiore di

quello che si otterrebbe proseguendo la ricerca. La distribuzione del tempo presenta una marcata asimmetria positiva: la maggior parte delle configurazioni si concentra su valori bassi, con una coda di configurazioni molto più lente; per questo motivo, ove il tempo è rappresentato graficamente, si adotta una scala logaritmica e se ne descrive la tendenza tramite mediana e percentili anziché tramite la sola media.

Accanto allo scarto si considera una seconda metrica di qualità, lo *scarto effettivo*, che imputa come spreco anche i pezzi prodotti in eccesso rispetto alla domanda e coincide con lo scarto in assenza di sovraproduzione. Come mostra la Tabella 4.3, la direzione dell'effetto di ciascun fattore algoritmico coincide sullo scarto e sullo scarto effettivo, con la sola eccezione della sovraproduzione, che agisce direttamente sull'insieme dei pezzi prodotti. Lo scarto, metrica nativa del *solver*, è perciò adottato come riferimento nell'analisi; lo scarto effettivo si affianca ad esso soltanto nella sua analisi diretta (Sezione 4.4.4), dove ne è data la definizione formale, e nella frontiera di Pareto (Sezione 4.4.7).

Entrambe le metriche sono calcolate come media sui risultati delle esecuzioni indipendenti di ciascuna configurazione. Accanto ad esse si considera, dove utile a corroborare l'interpretazione, un insieme di metriche secondarie prodotte dal *solver*, tra cui l'area di pannello utilizzata, il numero di pannelli e di schemi di taglio impiegati e la quantità di pezzi prodotti in eccesso rispetto alla domanda, e la *variabilità* dei risultati tra le diverse istanze generate da una stessa configurazione di istanza, misurata dalla deviazione standard dello scarto e del tempo. Poiché l'algoritmo di generazione è stocastico, una stessa configurazione di istanza dà luogo a istanze distinte: una deviazione standard elevata segnala che l'esito dipende sensibilmente dalla particolare istanza generata, oltre che dai parametri che la definiscono.

4.3. Strumenti e metodi statistici

I fattori controllati assumono un numero ridotto di valori discreti e le metriche di risposta presentano distribuzioni fortemente asimmetriche. Per questo motivo l'analisi ricorre a strumenti non parametrici, che non richiedono ipotesi sulla forma della distribuzione e operano sui *ranghi* dei dati anziché sui loro valori: il rango di un'osservazione è la sua posizione nell'ordinamento crescente dei dati, cosicché conta soltanto l'ordine e non l'entità dei valori. Questa sezione descrive ciascuno strumento indicandone lo scopo, la ragione della scelta e la formulazione essenziale.

4.3.1. Statistiche descrittive e *boxplot*

Le distribuzioni delle metriche sono sintetizzate mediante media, mediana, deviazione standard e quartili. La mediana e i quartili sono preferiti alla sola media quando la distribuzione è asimmetrica, perché meno sensibili ai valori estremi. La rappresentazione grafica adottata è il *boxplot*, in cui il rettangolo copre il 50% centrale dei dati (dal primo quartile Q_1 al terzo quartile Q_3), la linea interna indica

la mediana, i *baffi* si estendono fino a una distanza pari a 1,5 volte l'ampiezza del rettangolo ($Q_3 - Q_1$) e i punti oltre i baffi sono i valori anomali. Un rettangolo ampio segnala una forte dispersione dei risultati, uno stretto una concentrazione attorno alla mediana.

4.3.2. Correlazione di Spearman

La correlazione di Spearman ρ misura l'intensità e il verso dell'associazione *monotona* tra due variabili, cioè la tendenza dell'una a crescere o decrescere al variare dell'altra, senza presupporne una relazione lineare. Si definisce come il coefficiente di correlazione di Pearson calcolato sui ranghi delle osservazioni anziché sui valori originali; assume valori in $[-1, 1]$, con il segno che indica il verso dell'associazione e il valore assoluto la sua intensità. La forma semplificata

$$\rho = 1 - \frac{6 \sum_i d_i^2}{n\,(n^2 - 1)}, \tag{4.1}$$

dove d_i è la differenza tra i ranghi della i-esima osservazione e n è il numero totale di osservazioni, vale solo in assenza di ranghi ripetuti; poiché ogni fattore assume pochi valori distinti su molte osservazioni, i ranghi ripetuti sono frequenti e si adotta la definizione generale basata sui ranghi medi. La correlazione di Pearson non è invece appropriata, poiché presuppone una relazione lineare tra variabili continue. A ogni valore di ρ si associa un p-value che quantifica la probabilità di osservare, qualora le due variabili fossero indipendenti, un'associazione in valore assoluto pari o superiore a quella riscontrata. La scala convenzionale, di seguito adottata, attribuisce a valori di $|\rho| < 0{,}1$ un'associazione trascurabile, 0,1–0,3 debole, 0,3–0,5 moderata e oltre 0,5 forte. La correlazione di Spearman è impiegata, per semplicità, soltanto per lo *screening* iniziale dei fattori; per una trattazione più rigorosa dal punto di vista quantitativo bisognerebbe tenere conto del fatto che la presenza di molti ranghi ripetuti comprime il valore assoluto di ρ: un fattore a pochi livelli non raggiunge $|\rho| = 1$ nemmeno sotto un'associazione monotona perfetta, e il massimo ottenibile, benché cresca con il numero di livelli, resta comunque al di sotto di tale valore. I valori di ρ non sono quindi pienamente comparabili tra fattori di cardinalità diversa e vanno intesi come indicazione di massima. La quantificazione degli effetti è invece affidata ai test descritti nel seguito.

4.3.3. Test di Wilcoxon dei ranghi con segno

Il disegno fattoriale completo e bilanciato consente di confrontare i due livelli di ciascun fattore algoritmico su *coppie appaiate*: per ogni combinazione degli altri fattori esistono una configurazione a livello basso e una a livello alto, la cui differenza riflette l'effetto del solo fattore in esame. Il test di Wilcoxon dei ranghi con segno stabilisce se il fattore sposta sistematicamente la metrica di risposta; la sua ipotesi nulla è che le differenze appaiate siano distribuite simmetricamente attorno allo zero,

condizione che corrisponde all'assenza di effetto. Per verificarla, il test ordina le differenze per valore assoluto e assegna a ciascuna un rango secondo tale ordine; somma quindi separatamente i ranghi delle differenze positive e di quelle negative e confronta le due somme. Se il fattore non incide, le due somme si equivalgono, mentre uno squilibrio marcato segnala uno spostamento sistematico nella direzione prevalente. Tenendo conto sia del segno sia dell'entità relativa di ogni differenza, il test è più informativo del semplice conteggio dei miglioramenti; operando sui ranghi anziché sui valori, resta inoltre robusto rispetto all'asimmetria delle distribuzioni. L'entità dell'effetto è riportata come mediana e media delle differenze appaiate, accompagnate dalla percentuale di coppie in cui la metrica migliora.

4.3.4. Test di Mann-Whitney

Il test di Mann-Whitney confronta due campioni *indipendenti*, cioè due insiemi di osservazioni distinte senza corrispondenza uno a uno, e verifica se i valori di uno tendono a essere sistematicamente maggiori di quelli dell'altro; la sua ipotesi nulla è che i due campioni provengano dalla stessa distribuzione. A tale scopo ordina congiuntamente tutte le osservazioni e confronta la somma dei ranghi dei due gruppi: se uno di essi tende ad assumere valori più alti, i suoi ranghi risultano sistematicamente maggiori. Nell'analisi il test serve a stabilire se l'entità dell'effetto di un fattore algoritmico dipende da una caratteristica dell'istanza: gli effetti appaiati del fattore, presi in valore assoluto, vengono suddivisi in due gruppi secondo il livello di tale caratteristica (per esempio il fattore di domanda) e se ne confrontano le distribuzioni. Poiché i due gruppi sono calcolati su istanze diverse, sono indipendenti, il che rende appropriato questo test anziché quello appaiato. Un esito significativo indica che l'entità dell'effetto cambia con la caratteristica, ossia la presenza di un'interazione tra i due fattori. L'entità di tale interazione è riportata come mediana del valore assoluto degli effetti nei due gruppi: il divario tra i due valori quantifica quanto l'effetto dipende dalla caratteristica.

4.3.5. Test di Kruskal-Wallis

Quando un fattore assume più di due livelli privi di un ordinamento naturale, come la configurazione di pannelli, la correlazione di Spearman non è applicabile, poiché misura un'associazione monotona e richiede quindi livelli ordinati. Si ricorre allora al test di Kruskal-Wallis, che generalizza il test di Mann-Whitney a più di due gruppi ed è l'analogo non parametrico dell'analisi della varianza. Il test ordina congiuntamente tutte le osservazioni e confronta il rango medio di ciascun gruppo: la sua ipotesi nulla è che i gruppi provengano dalla stessa distribuzione, condizione in cui i ranghi medi si equivalgono; un rango medio sistematicamente diverso in almeno un gruppo indica invece che il fattore influenza la metrica. All'esito del test, riassunto dalla statistica H e dal relativo p-value, si affianca una misura dell'*effect*

size tramite il coefficiente

$$\varepsilon^2 = \frac{H - k + 1}{n - k}, \tag{4.2}$$

dove k è il numero di gruppi e n il numero totale di osservazioni; ε^2 indica quanta parte delle differenze osservate nella metrica è dovuta al fattore: vale 0 quando il fattore non incide e tende a 1 quando ne spiega la maggior parte.

4.3.6. Frontiera di Pareto

Il confronto tra le configurazioni rispetto allo scarto e al tempo di calcolo, due obiettivi in conflitto, è condotto con la dominanza di Pareto (Sezione 2.4): le configurazioni non dominate costituiscono la frontiera di Pareto dell'analisi.

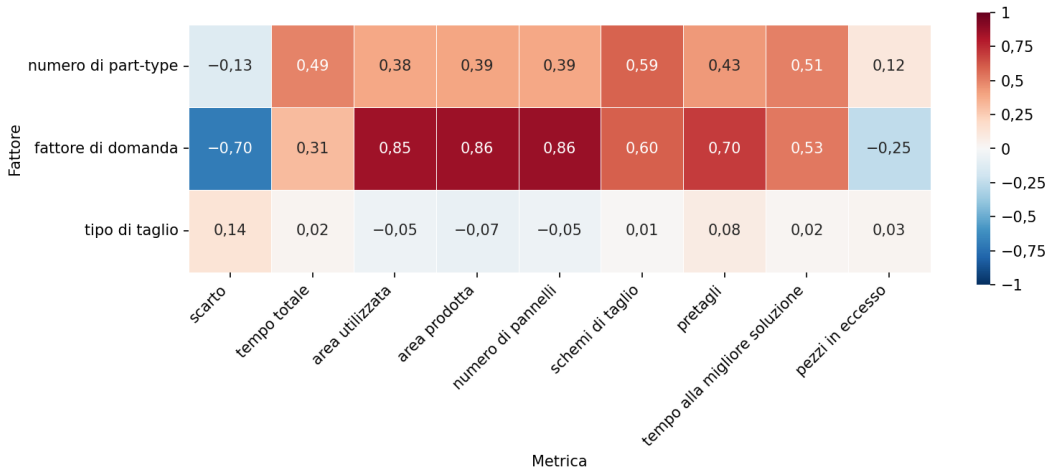
4.3.7. Impianto metodologico

L'analisi procede in due fasi. Nella prima, di *screening*, la correlazione di Spearman ordina i fattori in base all'intensità della loro associazione con ciascuna metrica, individuando quelli meritevoli di approfondimento. Nella seconda, di caratterizzazione, l'effetto dei fattori rilevanti è quantificato con i test appaiati e con le altre tecniche sopra descritte, sfruttando la struttura del disegno sperimentale.

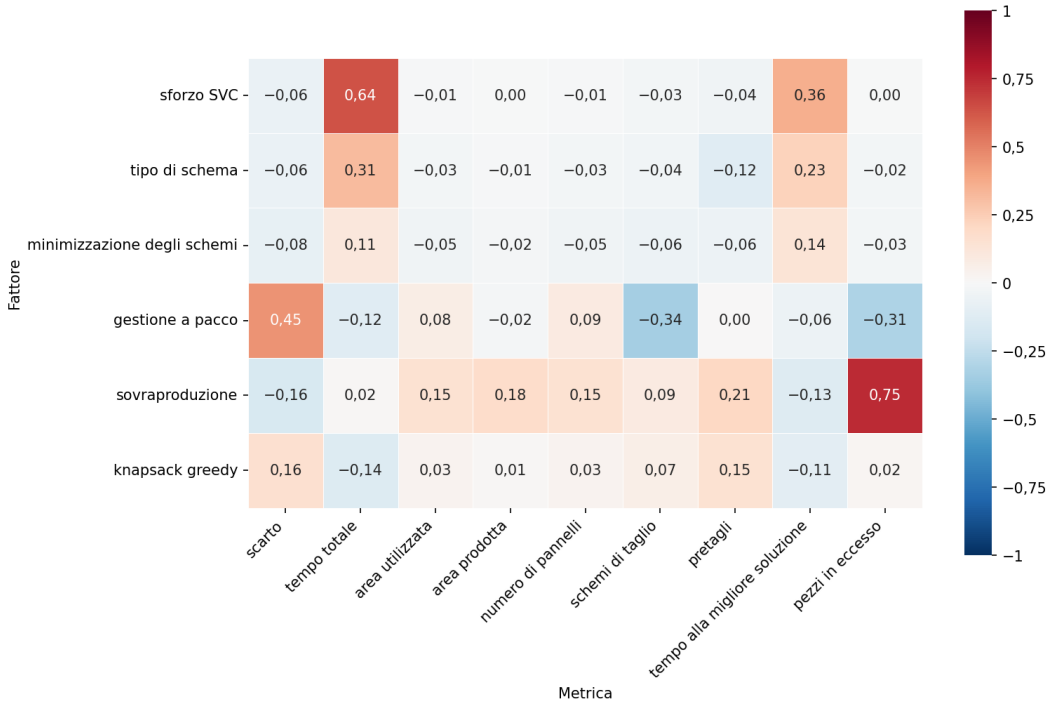
Va infine osservato che i p-value riportati vanno interpretati con cautela. Le osservazioni su cui operano i test, siano esse le singole configurazioni o le differenze appaiate, non sono indipendenti: quelle relative a una stessa configurazione di istanza condividono le stesse istanze generate e differiscono soltanto per i fattori algoritmici. Il numero di osservazioni indipendenti è quindi molto inferiore a quello considerato dai test, e i p-value nominali risultano più piccoli di quanto il disegno giustifichi. Inoltre, con un numero elevato di osservazioni anche effetti di entità trascurabile risultano statisticamente significativi, e la significatività non implica la rilevanza pratica dell'effetto. Per questi motivi l'interpretazione privilegia l'*entità* degli effetti (misurata dalle differenze appaiate, dalle medie per gruppo e dagli *effect size*) rispetto alla sola significatività statistica. Anche questi aspetti sono ripresi tra i limiti dello studio nella conclusione (Limiti).

4.4. Risultati

I risultati sono presentati per tema. Per ciascuno si riporta l'evidenza empirica, corredata degli effetti misurati e dei relativi test, e si formula un'ipotesi che ne tenta la giustificazione in termini di comportamento del *solver*. La fase di *screening* fornisce il quadro d'insieme: la Figura 4.1 riporta la correlazione di Spearman tra ciascun fattore e le metriche di output, separatamente per i due blocchi. Le statistiche descrittive per fattore, gli effetti appaiati completi e le configurazioni non dominate sono riportati nell'Appendice B.



(a) Fattori di istanza



(b) Fattori algoritmici

Figura 4.1.: Correlazione di Spearman ρ tra i fattori e le metriche di output. Il colore rosso indica associazione positiva, il blu negativa.

4.4.1. Effetti principali e ordinamento dei fattori

Sullo scarto, l'associazione più forte è quella del fattore di domanda ($\rho = -0{,}70$): al crescere della domanda lo scarto diminuisce. Segue la gestione a pacco ($\rho = +0{,}45$), unico fattore algoritmico con associazione moderata. Il numero di *part-type* presenta invece un'associazione debole con lo scarto ($\rho = -0{,}13$). Nell'intervallo di livelli considerato, sul tempo di calcolo prevale lo sforzo SVC ($\rho = +0{,}64$), seguito dal numero di *part-type* ($\rho = +0{,}49$) e, a pari intensità, dal fattore di domanda e dal tipo di schema di taglio ($\rho = +0{,}31$).

Questi effetti principali corrispondono a quanto atteso e ammettono una spiegazione comune. Ogni pannello è ripartito in numerosi pezzi: la taglia minima fissata dal limite dimensionale, pari a un decimo della media delle dimensioni dei pannelli della configurazione, ne consente al più un centinaio per pannello. Lo scarto dell'euristica si concentra allora in pochi pannelli riempiti solo in parte: i pezzi che non completano un pannello ne aprono uno nuovo, lasciato in gran parte vuoto. Questo costo, in valore assoluto, è grosso modo fisso: sulle istanze considerate l'equivalente di un paio di pannelli sprecati, poco sensibile alla taglia dell'istanza; lo scarto percentuale ne è quindi il rapporto con il materiale complessivo, e diminuisce man mano che l'istanza cresce.

Tanto il fattore di domanda quanto il numero di *part-type* ingrandiscono l'istanza: la domanda moltiplicando le copie di ciascun pezzo, il numero di *part-type* aggiungendo tipi distinti, che il generatore produce come ulteriori pannelli, e per questa via diluiscono il costo fisso e riducono lo scarto. La domanda vi incide molto di più perché, negli intervalli considerati, aggiunge assai più materiale: da qui la sua associazione forte con lo scarto, contro quella debole del numero di *part-type*. Entrambi, ampliando il problema combinatorio, ne aumentano invece il tempo di risoluzione. Poiché tali effetti sono prevedibili, l'interesse dell'analisi si concentra sui risultati meno immediati, esposti nei paragrafi seguenti: le interazioni tra i fattori, i compromessi tra qualità e tempo e il comportamento dei fattori a combinazione fissa.

4.4.2. Le leve più forti del *tuning* incidono soprattutto sulle istanze difficili

L'entità dell'effetto sullo scarto dei due fattori algoritmici più incisivi, la sovraproduzione e la gestione a pacco, non è costante, ma dipende dalla difficoltà dell'istanza, qui rappresentata dal fattore di domanda. La scelta di questo fattore come misura di difficoltà è motivata dal fatto che, tra le grandezze che dimensionano il problema, la domanda è l'unica a incidere marcatamente sullo scarto, mentre il numero di *part-type*, pur allungando i tempi di calcolo, vi influisce solo debolmente. Considerando la sovraproduzione (`automatic_overproduction`), la mediana del valore assoluto del suo effetto sullo scarto è di 14,3 punti percentuali sulle istanze a domanda unitaria, contro appena 0,2 punti su quelle a domanda elevata; per la gestione a pacco (`lotsize`) il divario è tra 24,1 e 4,0 punti. In entrambi i casi il test di Mann-Whitney conferma

che le due distribuzioni di effetti differiscono in modo netto (p nominale $< 10^{-50}$, da interpretare secondo quanto osservato nella Sezione 4.3). La Figura 4.2 riporta lo stesso confronto anche per la minimizzazione degli schemi e per la risoluzione *greedy*, i cui effetti sono di entità minore e non seguono questo andamento: a domanda elevata risultano anzi lievemente maggiori.

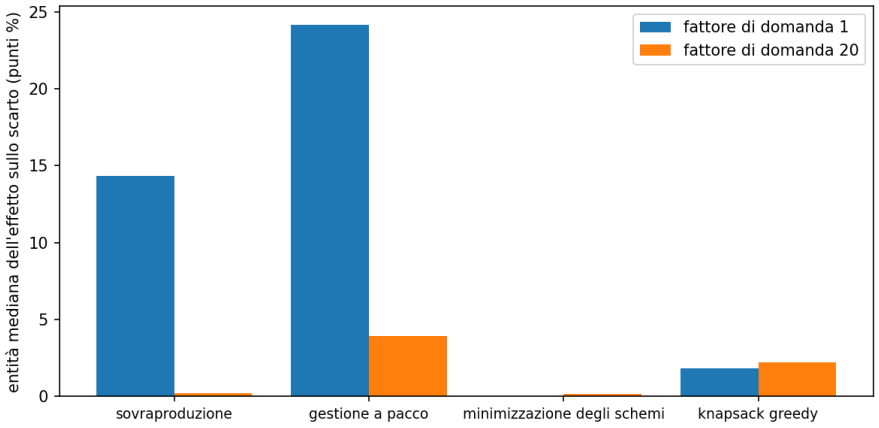


Figura 4.2.: Entità dell'effetto dei fattori algoritmici sullo scarto, misurata dalla mediana del valore assoluto delle differenze appaiate e calcolata separatamente per i due livelli del fattore di domanda. Il valore assoluto non ne rappresenta il segno, ossia la direzione dell'effetto.

Una possibile giustificazione è la seguente. Quando la domanda è elevata i pannelli sono riempiti da un gran numero di pezzi e la ripartizione risulta prossima alla completezza quasi indipendentemente dalla configurazione: le scelte algoritmiche trovano poco margine per incidere. Quando invece la domanda è unitaria i pezzi disponibili sono pochi e ogni decisione di collocazione pesa sul risultato finale, rendendo determinante il comportamento del *solver*. Per questi due fattori la scelta dei parametri ha quindi valore soprattutto sulle istanze difficili, mentre su quelle facili è in larga misura ininfluente.

A questa spiegazione generale se ne può affiancare, come ipotesi, una specifica per la gestione a pacco, legata alla divisibilità della domanda; il disegno non consente però di separarne l'effetto da quello del livello di domanda (Limiti). Poiché tale gestione attiva ogni schema di taglio in multipli della dimensione del pacco (Sezione 2.2), qui pari a cinque fogli, quando la domanda è un multiplo di tale dimensione (come nel caso della domanda pari a 20) la regola si integrerebbe senza difficoltà e l'effetto sullo scarto sarebbe modesto; quando invece la domanda non è un multiplo (come nel caso della domanda unitaria) la regola non potrebbe essere rispettata in modo efficiente e l'algoritmo ripiegherebbe su schemi meno compatti, con un forte aumento dello scarto. Coerentemente con questa ipotesi, la gestione a pacco non aumenta i pezzi prodotti in eccesso, con i quali è anzi associata negativamente (Figura 4.1): l'aumento dello scarto non deriva quindi dalla sovraproduzione.

4.4.3. La variabilità tra istanze

Poiché l'algoritmo di generazione è stocastico, una stessa configurazione di istanza dà luogo a più istanze distinte, risolte separatamente. La variabilità dei risultati tra queste istanze, misurata dalla deviazione standard dello scarto e del tempo (Sezione 4.2), quantifica quanto l'esito dipende dalla particolare istanza generata piuttosto che dai soli parametri che la definiscono. La Figura 4.3 ne riassume gli andamenti principali.

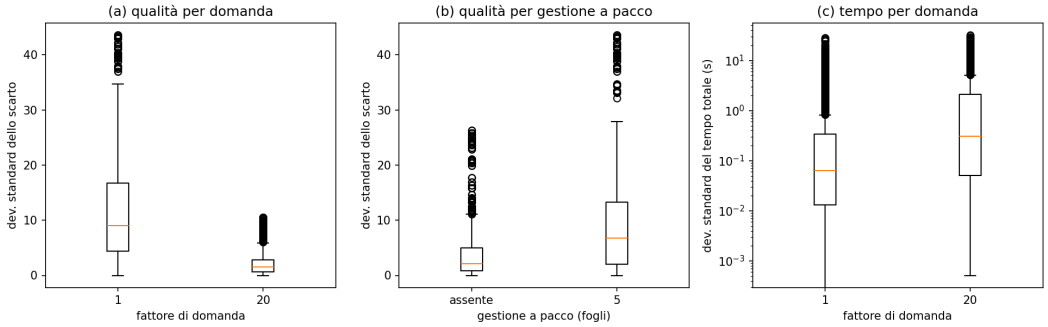


Figura 4.3.: Variabilità tra le istanze generate da una stessa configurazione di istanza, misurata dalla deviazione standard: (a) dello scarto per fattore di domanda, (b) dello scarto per gestione a pacco, (c) del tempo per fattore di domanda. Il tempo è in scala logaritmica.

La variabilità dello scarto segue la stessa difficoltà che governa lo scarto medio. Sulle istanze a domanda unitaria la deviazione standard dello scarto ha mediana di circa 9 punti percentuali, contro circa 1,5 punti su quelle a domanda elevata: la variabilità della qualità si concentra quindi sulle istanze difficili, le stesse in cui sovraproduzione e gestione a pacco incidono di più. Nella stessa direzione agisce la gestione a pacco, che innalza la deviazione standard dello scarto (mediana da circa 2 a circa 7 punti), coerentemente con il ripiego su schemi meno compatti descritto nella Sezione 4.4.2. Le leve algoritmiche che agiscono sulla ricerca a insieme di pezzi fissato, ossia lo sforzo *SVC*, il tipo di schema di taglio e la risoluzione del sottoproblema *knapsack*, lasciano invece la variabilità dello scarto pressoché invariata.

La variabilità del tempo mostra il quadro opposto: cresce con le grandezze che dimensionano il problema, il fattore di domanda e il numero di *part-type*, e con lo sforzo di ricerca, cioè con i fattori che ne allungano il valore medio. Ne risulta un contrasto netto: la qualità è più variabile sulle istanze difficili, quelle a bassa domanda, mentre il tempo è più variabile su quelle grandi, quelle ad alta domanda.

Il fenomeno condivide la radice con la dipendenza dell'effetto del *tuning* dalla difficoltà. Quando la domanda è elevata i pannelli sono riempiti da un gran numero di pezzi e la ripartizione è prossima alla completezza per qualunque istanza generata dalla configurazione, cosicché il risultato è tanto buono quanto uniforme; quando la domanda è unitaria i pezzi disponibili sono pochi e la loro particolare combinazione determina quanto compiutamente i pannelli possono essere riempiti, così che l'esito

varia sensibilmente da un'istanza all'altra. Sulle istanze difficili, dunque, non solo le scelte algoritmiche ma anche la specifica istanza generata pesano marcatamente sul risultato.

4.4.4. La sovraproduzione riduce lo scarto riportato, non quello effettivo

L'attivazione della sovraproduzione riduce lo scarto nel 71% delle coppie appaiate (test di Wilcoxon, p nominale $< 10^{-50}$). Sul tempo di calcolo non emerge invece alcun effetto: l'esecuzione è più rapida nel 46% delle coppie, con un rapporto mediano tra i tempi pari a 1,01 (p nominale = 0,19). La sovraproduzione anticipa soltanto il ritrovamento della migliore soluzione, che avviene prima nel 72% delle coppie (p nominale $< 10^{-50}$): il *solver* giunge prima alla soluzione che restituirà, ma prosegue l'esecuzione per un tempo analogo a quello richiesto in assenza di sovraproduzione. L'esame delle metriche secondarie mostra inoltre un costo: il parametro aumenta i pezzi prodotti oltre la domanda (in mediana 67 pezzi), l'area prodotta, il numero di pannelli utilizzati (in mediana 2,5 pannelli in più) e con essi l'area di pannello utilizzata.

Lo scarto misura quanto bene i pezzi prodotti ripartiscono i pannelli. Con la sovraproduzione attiva il *solver* risolve un problema diverso, in cui la produzione deve soddisfare almeno la domanda anziché eguagliarla (Sezione 2.1): su tale problema uno scarto minore è realmente ottenibile riempiendo pannelli aggiuntivi con pezzi non richiesti. Lo scarto riportato resta corretto, ma misura la ripartizione dell'insieme prodotto, non il soddisfacimento dell'insieme domandato. Sulle istanze a scarto nullo i due insiemi coincidono solo in assenza di sovraproduzione; il minore scarto non corrisponde quindi a una ripartizione più efficiente della domanda, ma si ottiene allontanandosi dall'ottimo e consumando più materiale.

Per valutare la qualità rispetto alla domanda si introduce lo *scarto effettivo*, definito come

$$s_{\text{eff}} = \frac{U - A^*}{U}, \tag{4.3}$$

dove U è l'area di pannello utilizzata e A^* l'area netta dei pezzi domandati. Esso conta come spreco tutto il materiale impiegato oltre l'ottimo, sia l'area non ripartita sia i pezzi prodotti in eccesso, e coincide con lo scarto quando la produzione eguaglia la domanda. Misurato su questa metrica, l'effetto della sovraproduzione si inverte: l'effetto appaiato medio passa da $-7{,}79$ punti sullo scarto a $+18{,}03$ punti sullo scarto effettivo, e la sovraproduzione peggiora la qualità nel 94% delle coppie.

L'inversione riguarda la sola sovraproduzione. Per gli altri fattori la direzione dell'effetto si conserva anche sotto lo scarto effettivo, come mostra la Tabella 4.3. I valori sono calcolati nello strato più sfavorevole a tale invarianza, quello in cui la sovraproduzione è attiva su entrambi i livelli di ogni coppia: per ciascun fattore, il disegno fattoriale completo e bilanciato garantisce che, fissati tutti gli altri fattori, esista una coppia formata dal livello basso e dal livello alto; se ne calcola la differenza sulla metrica e se ne riporta la media, separatamente per lo scarto e per lo

scarto effettivo. Nello strato in cui la sovraproduzione è disattivata le due metriche coincidono per costruzione. La direzione dell'effetto è concorde per tutti i fattori; la sua intensità può invece cambiare, come per la gestione a pacco, il cui effetto medio sullo scarto (+14,31 punti) si riduce quasi a zero sullo scarto effettivo (+0,08), segno che gran parte di tale effetto, in quel regime, riflette la metrica anziché una perdita di qualità reale.

Tabella 4.3.: Effetto appaiato medio di ciascun fattore algoritmico (livello alto meno livello basso), in punti percentuali, nello strato con sovraproduzione attiva. La direzione dell'effetto è concorde tra le due metriche per ogni fattore.

Fattore	scarto	scarto effettivo
sforzo *SVC*	−0,69	−1,16
tipo di schema di taglio	−2,35	−3,04
minimizzazione degli schemi	−2,03	−6,87
gestione a pacco (fogli)	+14,31	+0,08
risoluzione *knapsack*	+2,43	+4,74

4.4.5. Lo sforzo di ricerca recupera una frazione stabile della distanza dall'ottimo

L'aumento dello sforzo *SVC* dal livello basso a quello alto, cioè da 9 a 400 iterazioni, migliora lo scarto in misura modesta in termini assoluti (la differenza mediana appaiata è di 0,31 punti percentuali), mentre il tempo di calcolo cresce in tutte le coppie, di circa 21 volte in mediana. Il tempo alla migliore soluzione cresce invece di circa 7,6 volte (Figura 4.4c): la migliore soluzione tende quindi a essere trovata prima del termine delle iterazioni previste.

Un guadagno di pochi decimi di punto ha tuttavia un peso diverso a seconda dello scarto da cui si parte. Poiché lo scarto ottimo è nullo, lo scarto ottenuto con lo sforzo basso coincide con la distanza dall'ottimo, e la quantità

$$r = \frac{s_{\mathrm{b}} - s_{\mathrm{a}}}{s_{\mathrm{b}}}, \tag{4.4}$$

dove s_{b} e s_{a} sono gli scarti ottenuti con sforzo basso e alto, misura la frazione di tale distanza recuperata dallo sforzo alto. Per studiarne la dipendenza dal livello di scarto, le coppie sono raggruppate in fasce secondo la media dei due scarti anziché secondo s_{b}, in modo che la regressione verso la media non distorca il confronto.

Sull'insieme delle coppie la frazione recuperata decresce al crescere dello scarto (Figura 4.4a): in mediana è pari al 41% sotto l'1% di scarto, al 17% tra il 2 e il 5% e prossima a zero oltre il 20% (correlazione di Spearman tra livello di scarto e frazione recuperata $\rho = -0{,}47$). L'andamento suggerirebbe che lo sforzo sia più utile vicino all'ottimo, in contrasto con l'attesa di rendimenti decrescenti. Il confronto è però

trasversale, tra problemi e configurazioni diversi, e non segue la convergenza di una stessa istanza, che con due soli livelli di sforzo non è osservabile. L'andamento risulta infatti da un effetto di composizione.

Nella configurazione priva di fattori che peggiorano lo scarto (gestione a pacco assente, risoluzione *knapsack* esatta, schema a tre stadi e sovraproduzione disattivata) la frazione recuperata è simile nei due livelli di domanda: 22,5% a domanda unitaria, con uno scarto mediano del 20,7%, e 21,0% a domanda elevata, con uno scarto mediano dell'1,4%. La correlazione con il livello di scarto non è significativa ($\rho = -0{,}19$, p nominale $= 0{,}07$, su 92 coppie), e il confronto appaiato tra i due livelli di domanda indica semmai un recupero lievemente maggiore a domanda unitaria. In assenza di tali fattori, lo sforzo alto recupera circa un quinto della distanza dall'ottimo senza una dipendenza apprezzabile dalla sua ampiezza.

Il calo osservato sull'insieme delle coppie ha due origini. La prima sono le configurazioni con domanda unitaria e sovraproduzione attiva, nelle quali lo scarto con sforzo alto coincide con quello con sforzo basso nell'85% delle coppie. Le iterazioni aggiuntive sono eseguite, poiché il tempo di calcolo cresce di circa 18 volte, ma la migliore soluzione è trovata nello stesso tempo. Si può ipotizzare che ciò dipenda dal meccanismo dell'euristica, che corregge i valori dei *part-type* promuovendo quelli con domanda residua elevata (Sezione 2.2): con domanda unitaria e sovraproduzione, che in queste configurazioni produce in mediana oltre cento pezzi in eccesso, la domanda non vincola la costruzione degli schemi e la correzione perde il segnale che la guida. Queste configurazioni hanno uno scarto mediano del 15%, con metà delle coppie tra il 7 e il 31%, e costituiscono il 45% delle coppie nella fascia tra il 20 e il 40%: spiegano quindi soltanto in parte il recupero nullo delle fasce più alte.

La seconda origine sono i fattori che peggiorano lo scarto, di natura diversa tra loro (Figura 4.4b). La gestione a pacco è una regola dell'algoritmo, che attiva gli schemi in multipli di cinque fogli (Sezioni 2.2 e 4.4.2): lo scarto che ne deriva non è riducibile correggendo i valori dei pezzi, e la frazione recuperata scende dall'11,1% allo 0,8% a domanda unitaria e dal 22,5% al 13,6% a domanda elevata (test di Mann-Whitney, p nominale $< 10^{-4}$ in entrambi i casi). La risoluzione *greedy* del sottoproblema *knapsack* non modifica il problema ma la procedura di costruzione degli schemi, che a parità di valori dei pezzi produce schemi meno efficienti. A domanda elevata la frazione recuperata scende dal 25,4% al 16,4% (p nominale $< 10^{-3}$), sebbene il guadagno assoluto sia maggiore (0,75 contro 0,32 punti), mentre a domanda unitaria non si osserva differenza (p nominale $= 0{,}44$). Lo schema a due stadi, infine, ha un effetto trascurabile, visibile soltanto a domanda unitaria.

Lo sforzo alto recupera dunque circa un quinto della distanza dall'ottimo a qualunque livello di scarto, a prezzo di un tempo di calcolo circa venti volte maggiore. Recupera meno dove lo scarto dipende in parte dalla gestione a pacco o dalla risoluzione *greedy*, e nulla con domanda unitaria e sovraproduzione attiva. Stabilire se il guadagno giustifichi il tempo dipende dal costo del materiale rispetto a quello del calcolo, aspetto che esula dall'analisi.

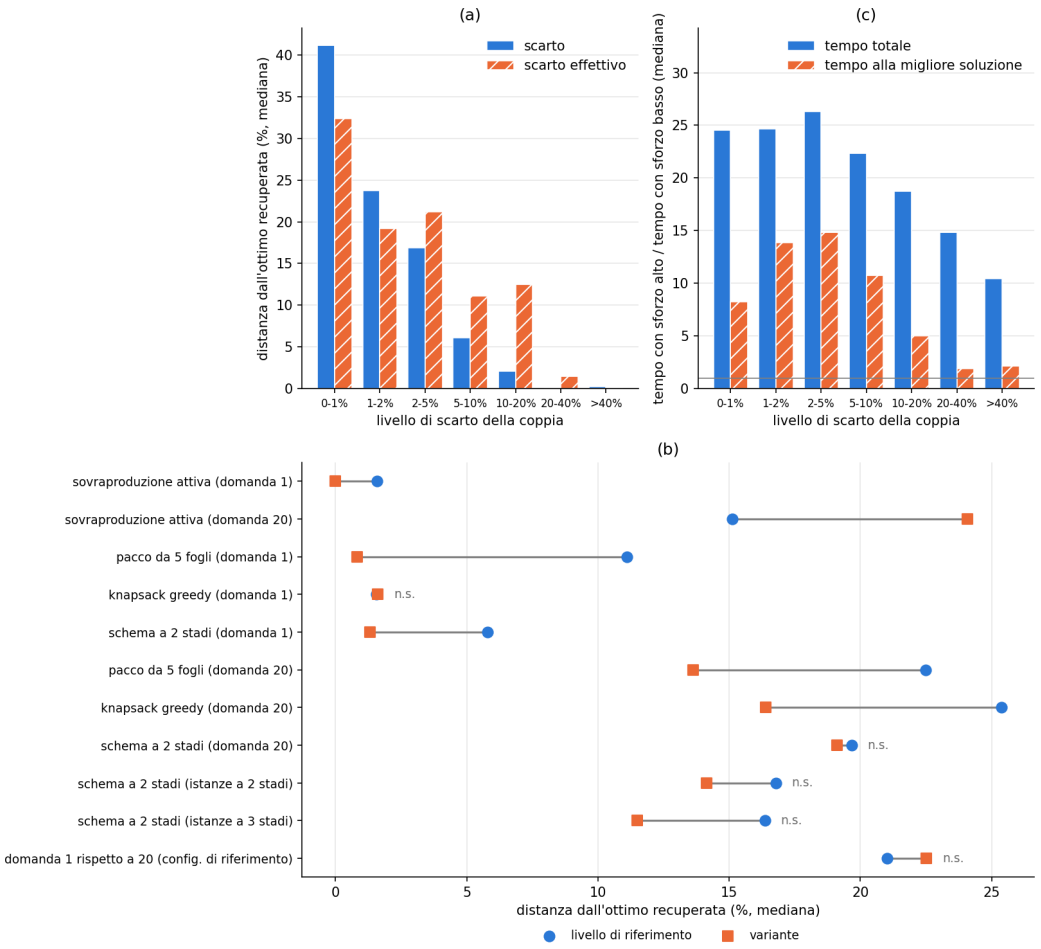


Figura 4.4.: Effetto dello sforzo *SVC* alto rispetto a quello basso. (a) Frazione della distanza dall'ottimo recuperata, in mediana, per fascia di scarto, sullo scarto e sullo scarto effettivo. (b) Frazione recuperata nella configurazione di riferimento e nella variante di ciascun fattore, per strato; n.s. indica una differenza non significativa al test di Mann-Whitney. (c) Rapporto mediano tra i tempi con sforzo alto e basso, totale e alla migliore soluzione.

4.4.6. Il prezzo della semplicità del taglio a due stadi

Il confronto tra i due tipi di schema di taglio richiede una cautela. Poiché ogni schema a due stadi è un caso particolare di schema a tre stadi (Sezione 1.3), il valore ottimo con lo schema a tre stadi non è per costruzione superiore a quello con lo schema a due stadi. Un minore scarto del tre stadi è quindi atteso per una ragione strutturale, e non indica di per sé una migliore prestazione dell'euristica, che d'altra parte non raggiunge sempre l'ottimo: con lo schema a tre stadi lo scarto risulta minore soltanto nel 67% delle coppie (Tabella B.3).

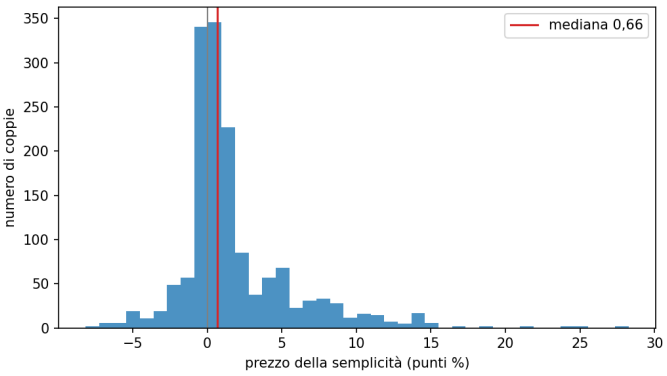
Si introduce allora il *prezzo della semplicità*, definito come la differenza tra lo scarto ottenuto imponendo lo schema a due stadi e quello ottenuto con lo schema a tre stadi, a parità di ogni altro fattore. Esso misura quanto scarto aggiuntivo si accetta rinunciando alla maggiore generalità del tre stadi in cambio di uno schema più semplice. Il confronto è condotto marginalizzando sul tipo di taglio dell'istanza, poiché tale caratteristica, benché determini se lo schema a tre stadi sia realmente necessario, non è osservabile a priori da chi utilizza il *solver*.

La distribuzione del prezzo della semplicità è riportata nella Figura 4.5. La mediana è pari a 0,66 punti percentuali: nel 47% dei casi lo schema a due stadi non comporta alcuna perdita apprezzabile, mentre nel 15% la perdita supera i 5 punti, con una coda che raggiunge i 28. Lo schema a due stadi è inoltre quasi sempre più rapido. Tra le caratteristiche osservabili dell'istanza, il fattore di domanda è il predittore più informativo del prezzo (che vale in media 2,96 punti a domanda unitaria contro 0,82 a domanda elevata, e in mediana 1,01 contro 0,55, come mostra la Figura 4.5b), ma il fattore realmente determinante, il tipo di taglio dell'istanza, resta latente.

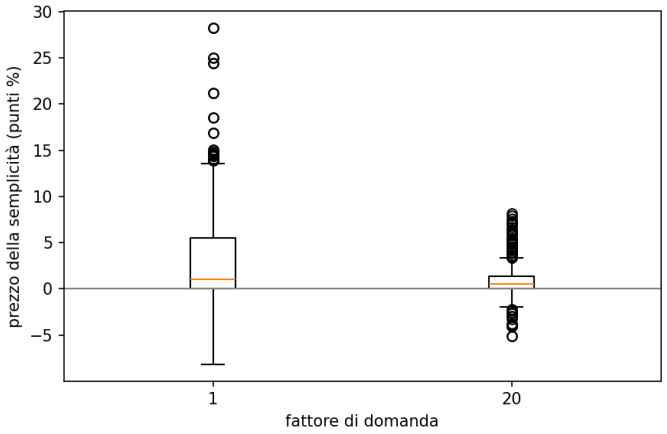
La giustificazione proposta è che la maggiore generalità dello schema a tre stadi si traduce in un vantaggio effettivo solo quando la soluzione ottima richiede realmente un terzo stadio di taglio, condizione che dipende dalla struttura dell'istanza e che le caratteristiche osservabili predicono solo debolmente. Il costo in tempo dell'esplorazione dello spazio di schemi più ampio è invece sistematico. Ne segue che lo schema a due stadi costituisce una scelta ragionevole in assenza di informazioni sull'istanza: raramente oneroso, quasi sempre più rapido, e con una perdita di qualità che, quando si presenta, è governata da una proprietà non nota a priori.

4.4.7. Frontiera di Pareto e configurazione raccomandata

La scelta della configurazione algoritmica è un problema a due obiettivi in conflitto, la minimizzazione dello scarto (o dello scarto effettivo) e quella del tempo. La Figura 4.6 riporta le due frontiere alla pari. Sul piano scarto–tempo 11 configurazioni delle 64 sono non dominate, e la sovraproduzione è attiva in 8 di esse; le tre in cui è disattivata si trovano nella regione a scarto superiore al 15%. Sul piano scarto effettivo–tempo il quadro si inverte: le configurazioni non dominate sono ancora 11, ma la sovraproduzione non compare in nessuna; l'intera regione di alta qualità è a sovraproduzione disattivata e con la gestione a pacco assente.

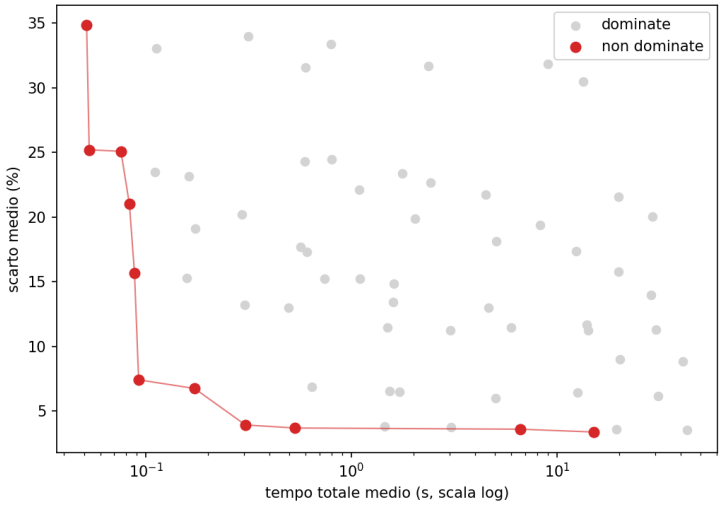


(a) Distribuzione del prezzo

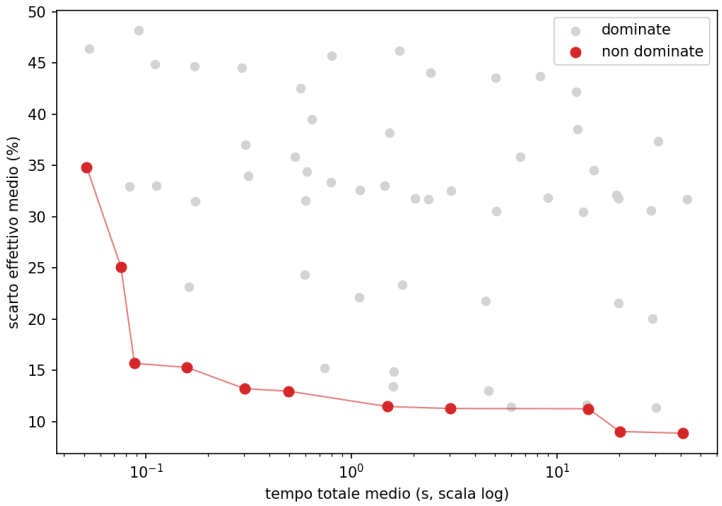


(b) Prezzo per fattore di domanda

Figura 4.5.: Prezzo della semplicità dello schema a due stadi: differenza di scarto rispetto allo schema a tre stadi, a parità di ogni altro fattore.



(a) Scarto



(b) Scarto effettivo

Figura 4.6.: Frontiere di Pareto delle 64 configurazioni algoritmiche: (a) nel piano scarto–tempo, (b) nel piano scarto effettivo–tempo. Il tempo è in scala logaritmica. Sulla metrica robusta alla sovraproduzione le configurazioni efficienti cambiano.

Il punto di ginocchio della frontiera (sforzo *SVC* basso, schema a due stadi, minimizzazione degli schemi attiva, gestione a pacco assente, sovraproduzione attiva e risoluzione *knapsack* esatta) raggiunge uno scarto medio di circa il 3,7% in 0,53 secondi, un valore prossimo a quello della configurazione più accurata, che richiede però un tempo circa ventotto volte maggiore. La composizione della frontiera è coerente con i risultati precedenti: le configurazioni efficienti attivano per lo più la sovraproduzione (8 su 11), che riduce lo scarto senza aumentare il tempo, ed evitano per lo più la gestione a pacco (7 su 11), che deteriora la qualità e compare soltanto nelle configurazioni più rapide e a scarto più alto, e adottano lo sforzo di ricerca alto soltanto all'estremo di massima qualità, dove il recupero di pochi decimi di punto richiede un tempo molto maggiore.

La frontiera sul solo scarto premia quindi la sovraproduzione, che su quella metrica migliora la qualità senza aumentare il tempo; la frontiera sullo scarto effettivo, che imputa il materiale consumato in eccesso, la esclude dalle configurazioni efficienti. Le due letture non si contraddicono: rispondono a obiettivi diversi, la minimizzazione dello scarto riportato e quella del materiale impiegato per soddisfare la domanda.

4.4.8. L'effetto della configurazione di pannelli

La configurazione di pannelli, analizzata come fattore categoriale a quattro livelli, incide in modo significativo sullo scarto: il test di Kruskal-Wallis fornisce $H \approx 120$ con p-value nominale inferiore a 10^{-20}, a fronte di un *effect size* $\varepsilon^2 \approx 0{,}038$. La Figura 4.7 mostra che le due configurazioni con un solo tipo di pannello producono uno scarto più alto, in media attorno al 18% e in mediana attorno al 10%, mentre le due con tre tipi di pannello si attestano in media intorno al 14% e in mediana tra il 5 e il 6%.

Questo risultato non è imputabile alla particolare geometria di un pannello, bensì alla varietà dei pannelli disponibili: le configurazioni che offrono tre tipi di pannello anziché uno solo consentono al *solver* un maggior numero di combinazioni con cui ripartire i pezzi, e quindi una ripartizione più completa dei pannelli. Sul tempo di calcolo, la configurazione con tre tipi di pannello e direzione del primo taglio scelta casualmente risulta la più lenta, verosimilmente perché, con il primo taglio libero, il *solver* costruisce uno schema per ciascuno dei due orientamenti di ogni tipo di pannello (Sezione 2.1), raddoppiando il numero di configurazioni da esplorare.

4.5. Sintesi e raccomandazioni

L'analisi restituisce un quadro coerente del comportamento del *solver* sulle istanze a scarto nullo. La qualità della soluzione è governata principalmente dal fattore di domanda, mentre il tempo di calcolo dipende, nell'intervallo di livelli considerato, soprattutto dallo sforzo di ricerca e dalla dimensione dell'istanza. Tra i fattori algoritmici, la sovraproduzione riduce lo scarto riportato senza incidere sul tempo,

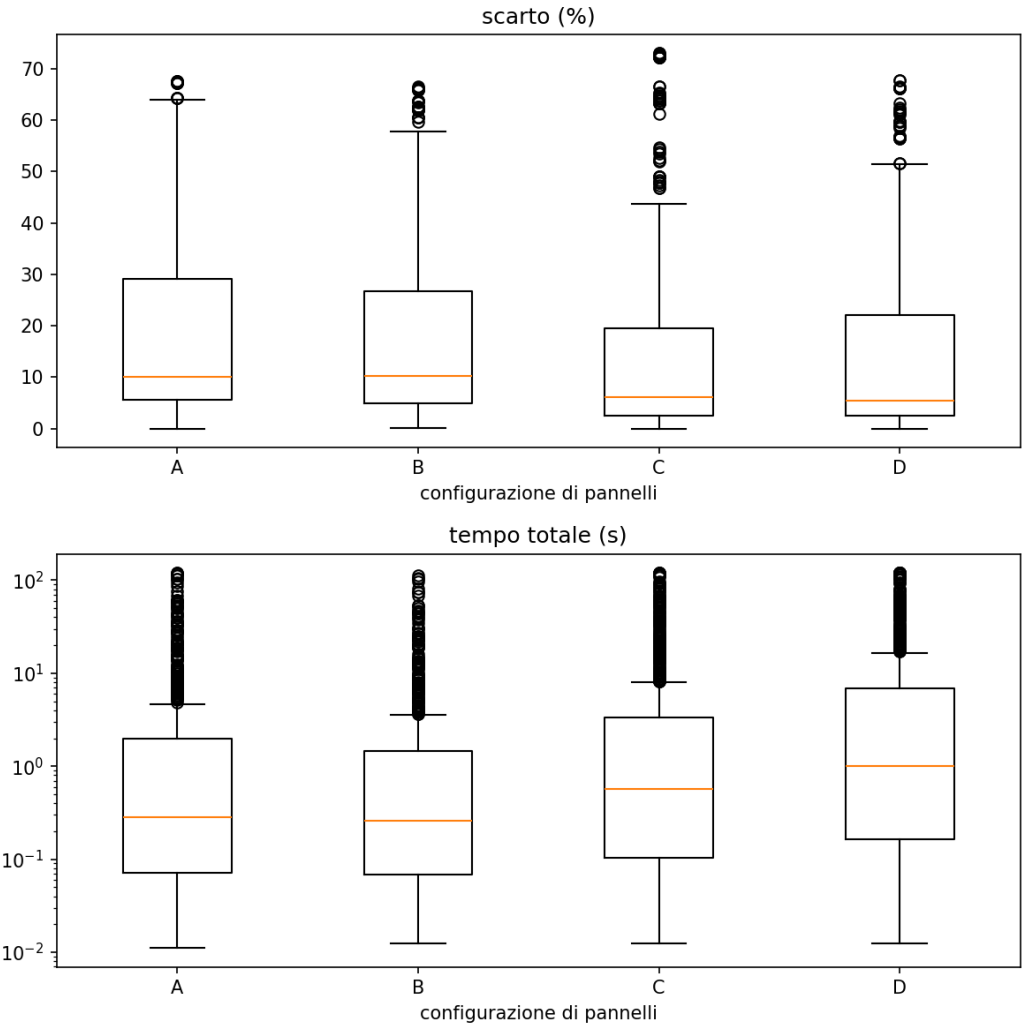


Figura 4.7.: Distribuzione dello scarto e del tempo per le quattro configurazioni di pannelli. Il tempo è in scala logaritmica.

ma aumenta il materiale consumato: valutata con lo scarto effettivo, che imputa tale materiale, essa peggiora la qualità ed esce dalle configurazioni efficienti. Il suo impiego è quindi un compromesso guidato dall'obiettivo. La gestione a pacco e la risoluzione *greedy* del sottoproblema *knapsack* deteriorano invece la qualità della soluzione; lo schema di taglio a tre stadi comporta un costo in tempo a fronte di un guadagno in qualità modesto, mentre lo sforzo di ricerca alto recupera circa un quinto della distanza dall'ottimo, meno in presenza della gestione a pacco o della risoluzione *greedy*, al prezzo di un tempo circa venti volte maggiore. L'incidenza della sovraproduzione e della gestione a pacco, le scelte algoritmiche con l'effetto più ampio, si manifesta soprattutto sulle istanze difficili, mentre su quelle facili è in larga parte ininfluente.

Da questi risultati discende una configurazione consigliata: gestione a pacco assente e risoluzione *knapsack* esatta, con la minimizzazione degli schemi attiva quando si desidera un miglioramento dello scarto, che si verifica in circa metà delle coppie e non lo peggiora mai (Sezione 2.5), a costo moderato. Lo sforzo SVC va mantenuto basso quando il tempo di calcolo è vincolante e innalzato quando il costo del materiale giustifica un tempo circa venti volte maggiore per recuperare circa un quinto dello scarto residuo. La sovraproduzione va attivata solo se l'obiettivo è la riduzione dello scarto riportato; se conta il materiale impiegato per soddisfare la domanda, ossia la vicinanza all'ottimo reale misurata dallo scarto effettivo, va invece disattivata, come indica la frontiera di Pareto costruita su tale metrica. In assenza di informazioni sulla struttura dell'istanza, lo schema a due stadi rappresenta una scelta di riferimento ragionevole, in virtù del prezzo della semplicità contenuto discusso nella Sezione 4.4.6.

I limiti che condizionano queste raccomandazioni e i possibili sviluppi del lavoro sono discussi nella conclusione, rispettivamente in Limiti e in Sviluppi futuri.

Conclusioni e sviluppi futuri

Il presente capitolo conclude la tesi. Viene dapprima riepilogato il lavoro svolto, dal problema affrontato al metodo con cui è stato studiato; seguono alcune considerazioni d'insieme sui risultati del Capitolo 4, il cui dettaglio è già sintetizzato nella Sezione 4.5. Il capitolo si chiude con i limiti dello studio e con i possibili sviluppi futuri.

Riepilogo del lavoro

La tesi si colloca nell'ambito del problema del 2D *Multiple Bin Packing* con tagli a ghigliottina, nel quale un insieme di *part-type*, ciascuno con la propria domanda, deve essere ricavato da pannelli di uno o più formati minimizzando l'area utilizzata, con schemi di taglio realizzabili mediante tagli a ghigliottina a 2 o a 3 stadi (Capitolo 1). Il problema è NP-difficile in senso forte (Sezione 1.5), e per le istanze di dimensioni industriali si ricorre a metodi non esatti, la cui distanza dall'ottimo resta in generale incognita: un limite inferiore certifica l'ottimalità di una soluzione soltanto quando questa lo raggiunge. Il *solver* oggetto dello studio, un'euristica a correzione sequenziale dei valori sviluppata per il taglio di pannelli in legno (Capitolo 2), non fa eccezione: sulle istanze di *benchmark* dell'articolo di riferimento l'ottimalità rispetto al materiale impiegato è certificata in circa quattro casi su cinque, mentre nei restanti la qualità della soluzione non è nota.

Per rimuovere questa incertezza è stato sviluppato un generatore di istanze a scarto nullo (Capitolo 3). Il generatore percorre il problema a ritroso: anziché cercare una ripartizione di pezzi assegnati, suddivide con tagli successivi pannelli interi in pezzi compatibili con i vincoli di taglio, cosicché l'istanza ammette per costruzione un piano di taglio privo di scarto e il suo valore ottimo è noto e pari a zero. Le istanze prodotte combinano più formati di pannello, domanda per *part-type*, schemi a 2 o a 3 stadi e rotazione dei pezzi, una combinazione che i *benchmark* con ottimo noto della letteratura non coprono (Sezione 1.7). Il generatore è stato quindi impiegato in una campagna sperimentale con disegno fattoriale completo e bilanciato, che incrocia 48 configurazioni di istanza con 64 configurazioni del *solver*, per un totale di 3072 combinazioni (Capitolo 4). Le soluzioni sono state valutate rispetto allo scarto, che su queste istanze coincide con la distanza dall'ottimo, e al tempo di calcolo; i dati sono stati analizzati con strumenti statistici non parametrici, adatti a fattori con pochi livelli e a distribuzioni fortemente asimmetriche.

Considerazioni sui risultati

Oltre alle indicazioni sull'impiego del *solver*, i risultati suggeriscono tre considerazioni sul metodo di valutazione adottato.

La prima riguarda le misure rese possibili dall'ottimo noto. Alcuni risultati non sarebbero ottenibili sulle istanze di *benchmark*, per le ragioni richiamate nella sezione precedente: la frazione della distanza dall'ottimo recuperata dallo sforzo di ricerca (Sezione 4.4.5), che richiede di conoscere tale distanza e non soltanto di confrontare due soluzioni, e la lettura in termini assoluti della qualità raggiunta, come lo scarto mediano di circa il 20% che il *solver* lascia sulle istanze a domanda unitaria anche in assenza dei fattori che peggiorano lo scarto.

La seconda riguarda la composizione degli insiemi di istanze di test. Il fattore di domanda, uno dei parametri del generatore, determina quanto margine resta alle scelte algoritmiche: sulle istanze a domanda elevata la ripartizione è prossima alla completezza per quasi ogni configurazione, e l'effetto dei fattori algoritmici più incisivi, la sovraproduzione e la gestione a pacco, si riduce marcatamente (Sezione 4.4.2). Un insieme di test composto soprattutto da istanze di questo tipo farebbe apparire equivalenti le configurazioni che sulle istanze difficili differiscono di molti punti percentuali. Per confrontare le configurazioni di un *solver* è quindi opportuno che l'insieme comprenda istanze difficili, che il generatore consente di ottenere attraverso il fattore di domanda.

La terza riguarda la metrica di qualità. Lo scarto riportato dal *solver* coincide con quello del problema definito nella Sezione 1.2 soltanto in assenza di sovraproduzione: quando questa è ammessa il *solver* risolve un problema diverso, e la sua metrica nativa premia una configurazione che consuma più materiale (Sezione 4.4.4). Il confronto tra configurazioni va quindi condotto su una metrica che corrisponda all'obiettivo del problema che si intende risolvere, verificando tale corrispondenza per ogni parametro che modifica il problema anziché il modo di risolverlo.

Limiti

I risultati presentati vanno letti alla luce dei limiti dello studio, di seguito raggruppati per natura. Alcuni sono già stati richiamati nel Capitolo 4, nei punti in cui condizionano un risultato specifico; qui se ne fornisce il quadro complessivo.

Disegno e scelta dei livelli

- Lo sforzo *SVC* è governato da due parametri che nella campagna sperimentale assumono sempre lo stesso valore. Il loro effetto non è separabile e i due parametri sono trattati come un unico fattore, cosicché l'analisi non ne distingue i contributi.

- La configurazione di pannelli non deriva da un incrocio libero delle dimensioni ma da una lista fissa, e il limite dimensionale di taglio è calcolato dalle dimensioni dei pannelli. L'effetto del limite è quindi confuso con quello della configurazione, analizzata come un unico fattore categoriale.

- Quasi tutti i fattori assumono due soli livelli. Il disegno rileva perciò il segno e l'entità dell'effetto tra i due livelli considerati, ma non la forma della risposta tra di essi, né eventuali andamenti non monotoni o valori intermedi più convenienti. Per la stessa ragione, l'ordine di importanza dei fattori stabilito dalle correlazioni dipende dall'ampiezza dell'intervallo scelto per ciascuno: un livello alto dello sforzo *SVC* più vicino a quello basso, per esempio, ne ridurrebbe l'associazione con il tempo di calcolo.

- Il livello elevato del fattore di domanda, pari a 20, è un multiplo della dimensione del pacco (`lotsize`), pari a 5. L'interazione tra i due fattori osservata nella Sezione 4.4.2 riflette perciò in parte questa scelta dei livelli, e non soltanto una proprietà generale del *solver*.

- Il limite dimensionale dei pezzi è fissato in ogni istanza a un decimo della media delle dimensioni dei pannelli della configurazione. La granularità dei pezzi rispetto al pannello resta perciò costante in tutta la campagna, e le conclusioni sull'entità dello scarto, legate a tale granularità (Sezione 4.4.1), non sono verificate su valori diversi.

- Il pretaglio è ammesso in tutte le esecuzioni e non è variato nel disegno: il suo effetto sulla qualità e sul tempo non è quindi misurato (Sezione 4.1).

- Il tempo massimo di 60 secondi assegnato a ciascuna fase interrompe almeno il 2,2% delle esecuzioni. In queste il tempo registrato sottostima quello necessario a completare la ricerca, e lo scarto può essere maggiore di quello che si otterrebbe proseguendola (Sezione 4.2).

Metodo di analisi

- Le osservazioni su cui operano i test non sono indipendenti, poiché quelle relative a una stessa configurazione di istanza condividono le stesse istanze generate. I test le trattano invece come indipendenti, e i p-value nominali risultano perciò più piccoli di quanto il disegno giustifichi, come già osservato nella Sezione 4.3; per questo l'interpretazione si fonda sull'entità degli effetti più che sulla loro significatività.

- Le istanze generate da ciascuna configurazione sono sintetizzate nel loro valore medio prima dell'analisi. La variabilità tra istanze è descritta ma non sottoposta a test, e la media, calcolata su un numero limitato di istanze e su distribuzioni asimmetriche, resta sensibile ai valori estremi.

- I confronti condotti sono numerosi e non è applicata alcuna correzione per la loro molteplicità. La probabilità di riscontrare associazioni apparenti per effetto del caso è quindi maggiore di quella suggerita dal singolo p-value.
- L'analisi quantifica gli effetti principali dei fattori e alcune interazioni a due fattori. Le interazioni di ordine superiore, che il disegno fattoriale completo consentirebbe di stimare, non sono esaminate.

Validità esterna

- Le conclusioni si riferiscono alle istanze a scarto nullo considerate e al *solver* impiegato; la loro estensione ad altri tipi di istanza o ad altri algoritmi non è immediata.
- Il fattore che meglio spiega il prezzo della semplicità dello schema a due stadi, ossia il tipo di taglio dell'istanza, non è osservabile a priori da chi utilizza il *solver* (Sezione 4.4.6).

Sviluppi futuri

Alcuni sviluppi discendono direttamente dai limiti appena esposti. Sul piano del disegno sperimentale si segnalano l'ampliamento dell'intervallo di variazione dei fattori con l'introduzione di livelli intermedi, che consentirebbe di ricostruire la forma della risposta e non il solo segno dell'effetto; la separazione dei due parametri che governano lo sforzo *SVC*, al fine di stimarne gli effetti distinti; l'analisi del punto della ricerca in cui l'euristica trova la migliore soluzione, che, insieme a più livelli di sforzo, consentirebbe di osservarne la convergenza su una stessa istanza e di distinguere, per la risoluzione *greedy*, un avvicinamento più lento all'ottimo da un arresto a distanza maggiore; la scelta di livelli del fattore di domanda non multipli della dimensione del pacco, che isolerebbe l'effetto della gestione a pacco da quello della divisibilità; la variazione del limite dimensionale dei pezzi, per verificare la dipendenza dei risultati dalla granularità. Sul piano dell'analisi, l'adozione di correzioni per la molteplicità dei confronti e di stime dell'incertezza robuste alla non-indipendenza delle osservazioni, come il *bootstrap* a livello di istanza, renderebbe più solida la quantificazione degli effetti. Sul piano del problema, infine, si segnalano l'estensione dell'analisi a istanze con scarto ottimo non nullo e lo studio di caratteristiche osservabili dell'istanza in grado di predire con maggiore affidabilità la convenienza dello schema a tre stadi.

Appendice A.

Codice sorgente

Di seguito è riportato il codice sorgente del generatore di istanze a scarto nullo, suddiviso nei cinque moduli che compongono il progetto. Il modulo `main.py` costituisce il punto di ingresso del programma e contiene il ciclo principale di generazione; `file_reader.py` si occupa del *parsing* del file di parametri; `panel_generator.py` implementa le funzioni di generazione degli schemi di taglio; `cutter.py` implementa le funzioni di taglio monodimensionale; `image_generator.py` produce la rappresentazione grafica degli schemi.

A.1. `main.py`

```
from file_reader import read_file
from cutter import cut_min, cut_max
from panel_generator import generate_first, generate_next
from image_generator import draw
from image_generator import close_pdf
import random
import os
import sys
from collections import Counter

if __name__ == "__main__" :

    if len(sys.argv) < 3:
        print("Error: missing parameters")
        sys.exit(1)

    input_file = sys.argv[1]
    pdf_file = f"ris_{sys.argv[2]}.pdf"

    panel, size_range, min_max_flag, desired_quantity, demand,
       type_of_cut = read_file(input_file)
    pieces = Counter()
    test = Counter()
    total_pieces = Counter()
    panel_number = 0
```

```
25      attempt = 0
26      best_attempt = Counter()
27
28      while len(total_pieces) < (desired_quantity[0] * (1 -
            desired_quantity[1])):
29          # selects a random type of panel
30          selected_panel = random.randint(0, len(panel) - 1)
31          print(f"using the {panel[selected_panel][0]} x {panel[
                selected_panel][1]} panel")
32          # rotates the panel
33          if panel[selected_panel][2] == 0 :
34              rotation = random.randint(0, 1)
35          else :
36              rotation = panel[selected_panel][2] - 1
37          if rotation :
38              print("Panel rotated")
39          total_height = panel[selected_panel][1 - rotation]
40          total_width = panel[selected_panel][0 + rotation]
41
42          # checks the panel isn't too small
43          if not min_max_flag and (total_width < size_range[0] or
                total_height < size_range[1]) :
44              print("Panel too small")
45              continue
46
47          # after the first generation the program starts re-
                using pieces
48          if not total_pieces :
49              piece_width, piece_height = generate_first(
                    total_width, total_height, size_range,
                    min_max_flag, type_of_cut)
50          else :
51              piece_width, piece_height = generate_next(
                    total_width, total_height, size_range,
                    min_max_flag, type_of_cut, total_pieces)
52
53          # pieces are saved in a buffer to check that their
                quantity doesn't go over the desired amount
54          test = total_pieces.copy()
55          for i, sublist in enumerate(piece_width):
56              for j, w in enumerate(sublist):
57                  for k, h in enumerate(piece_height[i][j]) :
58                      # pieces are sorted so that a A x B piece
                            is the same as a B x A piece
59                      key = tuple(sorted((w, h)))
60                      test[key] += demand
61
```

```
62          # the program tries to find a combination that doesn't
               go over the maximum quantity 30 times, then settles
               for the lowest amount of pieces
63          if len(test) <= (desired_quantity[0] * (1 +
               desired_quantity[1])) :
64              panel_number += 1
65              total_pieces = test.copy()
66              draw(panel_number, total_width, total_height,
                   piece_width, piece_height, pdf_file = pdf_file)
67          elif attempt < 30 :
68              attempt += 1
69              if (not best_attempt) or (len(test) < len(
                   best_attempt)) :
70                  best_attempt = test.copy()
71          else :
72              panel_number += 1
73              total_pieces = best_attempt.copy()
74              draw(panel_number, total_width, total_height,
                   piece_width, piece_height, pdf_file = pdf_file)
75              break
76
77      # the list of pieces is saved in "c01.txt"
78      pieces_list = sorted(total_pieces.items(), key=lambda x: (-
           x[1], x[0]))
79      pieces_quantity = sum(total_pieces.values())
80      with open(f"{sys.argv[2]}.txt", "w") as f :
81          f.write(f"{len(panel)}, {len(total_pieces)}, 0\n")
82          f.write("STOCKS\n")
83          for i, p in enumerate(panel, start=1):
84              f.write(f"{str(i).ljust(20)}, {str(p[0]).rjust(4)},
                       {str(p[1]).rjust(4)},    10, -1, {p[2]}, 0, 0,
                   0\n")
85          f.write("PARTS\n")
86          for i, ((dimX,dimY), count) in enumerate(pieces_list,
               start=1):
87              f.write(f"{str(i).ljust(20)}, {str(dimX).rjust(4)},
                       {str(dimY).rjust(4)}, 0, {str(count).rjust(2)},
                       0, 0, 0, 0, -1, 0, 0, -1\n")
88      close_pdf()
89      print(f"Generation completed: {len(total_pieces)} different
           pieces, {pieces_quantity} total pieces")
```

Listing A.1: `main.py`: ciclo principale di generazione.

A.2. `file_reader.py`

```
# PARAMETERS FILE LAYOUT:
#     panels
#     panel1_x panel1_y first_cut_flag1(0 -> random direction, 1 -> horizontal cut, 2 -> vertical cut)
#     panel2_x panel2_y first_cut_flag2
#     ...
#
#     limits
#     x_limit y_limit min_max_flag(0 -> minimum dimensions, 1 -> maximum dimensions)
#
#     quantity
#     desired_quantity tolerance
#
#     cut
#     type_of_cut(0 -> 2-stage cut, 1 -> 3-stage cut)

def read_file (file_name : str):
    panel : list[int] = []
    size_range : list[int] = []
    min_max_flag : int = 0
    desired_quantity: list = []
    demand : int = 1
    type_of_cut : int = 0
    section : str = " "

    with open(file_name, "r") as f:

        for line in f:

            line = line.strip()

            if not line:
                continue

            # determines what parameter is being read
            if line.lower() == "panels":
                section = "panels"
                print("Loading panels sizes")
                continue
            elif line.lower() == "limits":
                section = "limits"
                print("Loading piece size limits")
                continue
            elif line.lower() == "quantity":
                section = "quantity"
                print("Loading quantity of pieces")
```

```
46                  continue
47              elif line.lower() == "demand":
48                  section = "demand"
49                  print("Loading pieces demand")
50                  continue
51              elif line.lower() == "cut":
52                  section = "cut"
53                  print("Loading type of cut")
54                  continue
55
56              # reads the parameters
57              if section == "panels":
58                  try:
59                      width, height, first_cut_flag = map(int,
                            line.split())
60                      panel.append((width, height, first_cut_flag
                            ))
61                  except ValueError:
62                      print(f"Error in panel size at line: {line}
                            ")
63                      exit(1)
64              elif section == "limits":
65                  try:
66                      range_x, range_y, min_max_flag = map(int,
                            line.split())
67                      size_range = (range_x, range_y)
68                  except ValueError:
69                      print(f"Error in piece size at line: {line}
                            ")
70                      exit(1)
71              elif section == "quantity" :
72                  try:
73                      quantity, tollerance = line.split()
74                      desired_quantity = (int(quantity), float(
                            tollerance))
75                  except ValueError:
76                      print(f"Error in desired quantity of pieces
                             at line: {line}")
77                      exit(1)
78              elif section == "demand" :
79                  try:
80                      demand = int(line.strip())
81                  except ValueError:
82                      print(f"Error in demand of pieces at line:
                            {line}")
83                      exit(1)
84              elif section == "cut" :
```

```
85                  try:
86                      type_of_cut = int(line.strip())
87                  except ValueError:
88                      print(f"Error in desired quantity of pieces
                            at line: {line}")
89                      exit(1)
90
91      print("Loading completed")
92      return panel, size_range, min_max_flag, desired_quantity,
           demand, type_of_cut
```

Listing A.2: `file_reader.py`: lettura del file di parametri.

A.3. `panel_generator.py`

```
1  from cutter import cut_min, cut_max
2  from collections import Counter
3  import random
4
5  def generate_first(total_width : int, total_height : int,
       size_range : list[int], min_max_flag : int, type_of_cut :
       int):
6      first_cut : list[int] = []
7      second_cut : list[list[int]] = []
8      third_cut : list[list[int]] = []
9      piece_height : list[list[list[int]]] = []
10
11     if min_max_flag == 0 :
12         cut = cut_min
13     else :
14         cut = cut_max
15
16     # first cuts horizontally, then vertically
17     first_cut = cut(total_height, size_range[1])
18     for i in range(len(first_cut)) :
19         second_cut.append(cut(total_width, size_range[0]))
20
21     # if the cut is 2-stage the program treats it as if it cuts
           each piece in 1
22     # so that the function always returns in the same format
23     for i, sublist in enumerate(second_cut) :
24         for j, width in enumerate(sublist) :
25             if type_of_cut == 1 :
26                 third_cut.append(cut(first_cut[i], size_range
                       [1]))
27             else :
```

```
28                  third_cut.append([first_cut[i]])
29
30          piece_height.append(third_cut)
31          third_cut = []
32
33      piece_width = second_cut
34
35      return piece_width, piece_height
36
37  def generate_next(total_width : int, total_height : int,
        size_range : list[int], min_max_flag : int, type_of_cut :
        int, old_pieces : Counter):
38      pre_generated_width : list[list[int]] = []
39      pre_generated_height : list[list[list[int]]] = []
40
41      if min_max_flag == 0 :
42          cut = cut_min
43      else :
44          cut = cut_max
45
46      remaining_height = total_height
47      # randomly decides a number of pieces to re-use
48      pieces_number = random.randint(0, 5)
49      for i in range(pieces_number) :
50          # at least a quarter of the panel is made of new pieces
51          if remaining_height < int(total_height * 0.25) :
52              break
53
54          selected_piece = random.choice(list(old_pieces.keys()))
55          possible_selection = []
56          # randomly rotates the piece making sure it fits the
                size limitations
57          if (selected_piece[0] <= total_width and selected_piece
                [1] <= remaining_height) and ((selected_piece[0] >=
                size_range[0] and selected_piece[1] >= size_range[1]
                 and not min_max_flag) or (selected_piece[0] <=
                size_range[0] and selected_piece[1] <= size_range[1]
                 and min_max_flag)) :
58              possible_selection.append(selected_piece)
59          if (selected_piece[1] <= total_width and selected_piece
                [0] <= remaining_height) and ((selected_piece[0] >=
                size_range[1] and selected_piece[1] >= size_range[0]
                 and not min_max_flag) or (selected_piece[0] <=
                size_range[1] and selected_piece[1] <= size_range[0]
                 and min_max_flag)) :
60              possible_selection.append((selected_piece[1],
                    selected_piece[0]))
```

```
61          if not possible_selection:
62              continue
63          selected_piece = random.choice(possible_selection)
64
65          if remaining_height < selected_piece[1] + (size_range
                [1] * (1 - min_max_flag)) :
66              continue
67          print(f"Re-using the {selected_piece[0]} x {
                selected_piece[1]} piece")
68
69          # a part of the panel is cut using the selected piece
                as the root
70          piece_quantity = random.randint(1, 5)
71          for i in range(piece_quantity) :
72              if total_width < selected_piece[0] * (i+1) + (
                    size_range[0] * (1 - min_max_flag)) :
73                  piece_quantity = i
74                  break
75          if piece_quantity == 0 :
76              continue
77
78          remaining_height -= selected_piece[1]
79
80          buffer_width = cut(total_width - (selected_piece[0] *
                piece_quantity), size_range[0])
81          buffer_height : list[list[int]] = []
82          for j in range(len(buffer_width)) :
83              if type_of_cut == 1 :
84                  buffer_height.append(cut(selected_piece[1],
                        size_range[1]))
85              else:
86                  buffer_height.append([selected_piece[1]])
87          for i in range(piece_quantity) :
88              buffer_width.append(selected_piece[0])
89              buffer_height.append([selected_piece[1]])
90
91          pre_generated_width.append(buffer_width)
92          pre_generated_height.append(buffer_height)
93
94      # the rest of the panel is cut as if it was a smaller whole
            panel
95      piece_width, piece_height = generate_first(total_width,
            remaining_height, size_range, min_max_flag, type_of_cut)
96      piece_height.extend(pre_generated_height)
97      piece_width.extend(pre_generated_width)
98
99      return piece_width, piece_height
```

Listing A.3: panel_generator.py: generazione degli schemi di taglio.

A.4. cutter.py

```
import random

def cut_min (total_length : int, min_size : int) -> list[int] :
    pieces : list[int] = []

    if total_length < min_size :
        print(f"There isn't enough length to cut: {total_length} < {min_size}")
        exit(1)

    remaining = total_length
    while remaining > 0 :
        # the length of each piece is selected randomly, but small sized pieces
        # have increased probability; bias can be tweaked to preference
        bias = 3
        r = random.random() ** bias
        new_piece = min_size + int(r * (remaining - min_size))

        quantity = random.randint(1, 5)
        for i in range(quantity) :
            if remaining >= new_piece :
                pieces.append(new_piece)
                remaining -= new_piece
            else : break

        # when there isn't enough length left to make a new piece
        # the leftover is added to the last piece
        if remaining < min_size:
            pieces[-1] += remaining
            remaining -= remaining

    return pieces

def cut_max (total_length : int, max_size : int) -> list[int] :
    pieces : list[int] = []

    remaining = total_length
    while remaining > 0 :
```

```
38          # the length of each piece is selected randomly, but
                large sized pieces
39          # have increased probability; bias can be tweaked to
                preference
40          bias = 3
41          r = random.random() ** bias
42          new_piece = max_size - int(r * (max_size - 1))
43
44          quantity = random.randint(1, 5)
45          for i in range(quantity) :
46              if remaining >= new_piece :
47                  pieces.append(new_piece)
48                  remaining -= new_piece
49              # this condition prevents the program from creating
                    very small pieces too often
50              elif remaining > 0 :
51                  pieces.append(remaining)
52                  remaining -= remaining
53                  break
54
55      return pieces
```

Listing A.4: `cutter.py`: taglio monodimensionale.

A.5. `image_generator.py`

```
1  import os
2  import matplotlib.pyplot as plt
3  import matplotlib.patches as patches
4  from matplotlib.backends.backend_pdf import PdfPages
5
6  # Variabile globale per il PDF aperto
7  _pdf_pages = None
8  _pdf_file_path = None
9
10 def draw(panel_number: int,
11          total_width: int,
12          total_height: int,
13          piece_width: list[list[int]],
14          piece_height: list[list[list[int]]],
15          folder: str = None,
16          pdf_file: str = None):
17
18     global _pdf_pages, _pdf_file_path
19
20     # Se vogliamo salvare su PDF
```

```
21      if pdf_file:
22          if _pdf_pages is None or _pdf_file_path != pdf_file:
23              # Inizializza il PDF solo la prima volta o se
                    cambia il file
24              if _pdf_pages is not None:
25                  _pdf_pages.close()
26              _pdf_pages = PdfPages(pdf_file)
27              _pdf_file_path = pdf_file
28
29      # Crea figura e asse
30      fig, ax = plt.subplots()
31      current_y = 0
32      piece_number = 1
33      for i, sublist in enumerate(piece_width):
34          current_x = 0
35          for j, w in enumerate(sublist):
36              new_y = current_y
37              for k, h in enumerate(piece_height[i][j]):
38                  color = "#cccccc"
39                  rect = patches.Rectangle((current_x, new_y), w,
                        h, edgecolor='black', facecolor=color)
40                  ax.add_patch(rect)
41                  ax.text(current_x + w/2, new_y + h/2, f"{
                        piece_number}\n{int(w)}x{int(h)}",
42                          color='black', ha='center', va='center'
                            , fontsize=4)
43                  new_y += h
44                  piece_number += 1
45              current_x += w
46          current_y = new_y
47
48      ax.set_xlim(0, total_width)
49      ax.set_ylim(0, total_height)
50      ax.set_aspect('equal')
51      ax.axis('off')
52      ax.invert_yaxis()
53      ax.text(total_width / 2, -5, total_width , ha='center', va=
            'bottom', fontsize=6)
54      ax.text(-5, total_height / 2, total_height, ha='right', va=
            'center', fontsize=6, rotation='vertical')
55
56      # Salvataggio
57      if pdf_file:
58          _pdf_pages.savefig(fig, dpi=300, pad_inches=0.1)
59      else:
60          if folder is None:
61              folder = "."
```

```
62          file_path = os.path.join(folder, f"pannello_{
                panel_number}.png")
63          plt.savefig(file_path, dpi=300, pad_inches=0.1)
64
65      plt.close(fig)
66
67  def close_pdf():
68      global _pdf_pages
69      if _pdf_pages is not None:
70          _pdf_pages.close()
71          _pdf_pages = None
```

Listing A.5: `image_generator.py`: output grafico PDF.

Appendice B.

Tabelle di dettaglio dei risultati

La presente appendice raccoglie in forma tabellare i risultati della campagna sperimentale del Capitolo 4, di cui il testo riporta soltanto i valori necessari all'argomentazione: le statistiche descrittive dello scarto e del tempo di calcolo per livello di ciascun fattore, gli effetti appaiati dei fattori algoritmici e le configurazioni non dominate delle due frontiere di Pareto. Tutti i valori sono calcolati sulle 3072 combinazioni del disegno sperimentale, ciascuna rappresentata dal valore medio delle metriche sulle istanze che le corrispondono (Sezione 4.1); il tempo di calcolo è quello totale (Sezione 4.2). I p-value sono nominali e vanno interpretati secondo quanto osservato nella Sezione 4.3.

B.1. Statistiche descrittive per fattore

Le Tabelle B.1 e B.2 riportano, per ciascun livello di ogni fattore, il numero n di combinazioni, la media, la mediana, la deviazione standard e i valori minimo e massimo rispettivamente dello scarto e del tempo di calcolo. Il confronto tra i livelli di un fattore è marginale rispetto a tutti gli altri fattori, e non coincide quindi con gli effetti appaiati della Sezione B.2, che li tengono fissi. Le configurazioni di pannelli sono quelle della Tabella 4.2.

B.2. Effetti appaiati dei fattori algoritmici

Le Tabelle B.3 e B.4 riportano, per ciascun fattore algoritmico, l'esito del confronto appaiato tra il livello alto e il livello basso descritto nella Sezione 4.3, condotto sulle coppie di combinazioni che differiscono nel solo fattore considerato. Le differenze sono espresse come livello alto meno livello basso, cosicché un valore negativo indica una riduzione della metrica. La Tabella B.3 riguarda lo scarto e lo scarto effettivo; la Tabella B.4 il tempo di calcolo totale e il tempo alla migliore soluzione, per i quali, data l'asimmetria della distribuzione, si riporta anche il rapporto mediano tra i tempi.

Tabella B.1.: Statistiche descrittive dello scarto, in punti percentuali, per livello di ciascun fattore.

Fattore	Livello	n	Media	Mediana	Dev. std.	Min	Max
configurazione di pannelli	A	768	18,75	10,09	17,69	0,00	67,58
	B	768	17,71	10,19	16,52	0,11	66,63
	C	768	13,95	6,05	16,97	0,00	73,03
	D	768	14,29	5,44	16,68	0,00	67,71
numero di *part-type*	10	1024	18,27	10,55	18,07	0,00	73,03
	50	1024	17,46	9,22	17,99	0,17	67,71
	100	1024	12,78	6,09	14,47	0,16	64,31
fattore di domanda	1	1536	27,11	25,00	17,98	0,00	73,03
	20	1536	5,23	4,01	4,66	0,00	20,00
tipo di taglio dell'istanza	2 stadi	1536	14,00	7,00	15,59	0,00	67,71
	3 stadi	1536	18,34	9,49	18,22	0,20	73,03
sforzo *SVC*	3	1536	16,88	9,22	17,21	0,00	73,03
	20	1536	15,47	8,05	16,95	0,00	72,62
tipo di schema di taglio	2 stadi	1536	17,12	8,82	17,60	0,00	73,03
	3 stadi	1536	15,23	8,32	16,52	0,00	72,56
minimizzazione degli schemi	disattivata	1536	17,96	9,27	18,83	0,00	73,03
	attivata	1536	14,39	7,98	14,96	0,00	73,03
gestione a pacco (fogli)	assente	1536	8,81	4,87	9,94	0,00	50,00
	5	1536	23,54	18,00	19,42	0,00	73,03
sovraproduzione	disattivata	1536	20,07	13,20	19,91	0,00	73,03
	attivata	1536	12,28	7,23	12,56	0,00	53,46
risoluzione *knapsack*	esatta	1536	15,01	6,55	17,40	0,00	72,79
	greedy	1536	17,33	9,67	16,71	0,00	73,03

Tabella B.2.: Statistiche descrittive del tempo di calcolo totale, in secondi, per livello di ciascun fattore.

Fattore	Livello	n	Media	Mediana	Dev. std.	Min	Max
configurazione di pannelli	A	768	4,709	0,286	14,721	0,011	120,114
	B	768	3,516	0,262	11,301	0,012	113,576
	C	768	7,964	0,574	20,102	0,012	120,131
	D	768	11,835	1,003	25,304	0,013	120,128
numero di *part-type*	10	1024	0,573	0,101	1,568	0,011	19,436
	50	1024	5,381	0,587	13,354	0,023	116,830
	100	1024	15,065	1,969	27,977	0,033	120,131
fattore di domanda	1	1536	3,007	0,208	9,788	0,011	102,055
	20	1536	11,006	1,062	24,228	0,017	120,131
tipo di taglio dell'istanza	2 stadi	1536	6,625	0,441	18,223	0,011	120,131
	3 stadi	1536	7,387	0,470	19,556	0,012	120,128
sforzo *SVC*	3	1536	0,678	0,102	2,369	0,011	40,638
	20	1536	13,334	2,318	25,081	0,069	120,131
tipo di schema di taglio	2 stadi	1536	2,425	0,209	8,779	0,011	115,880
	3 stadi	1536	11,588	1,059	24,408	0,014	120,131
minimizzazione degli schemi	disattivata	1536	4,338	0,319	11,139	0,012	60,117
	attivata	1536	9,674	0,627	24,010	0,011	120,131
gestione a pacco (fogli)	assente	1536	8,807	0,624	21,720	0,012	120,131
	5	1536	5,206	0,334	15,380	0,011	120,081
sovraproduzione	disattivata	1536	6,990	0,433	18,941	0,011	120,131
	attivata	1536	7,022	0,473	18,870	0,013	120,128
risoluzione *knapsack*	esatta	1536	8,688	0,669	21,218	0,011	120,131
	greedy	1536	5,324	0,306	16,092	0,012	120,128

Tabella B.3.: Effetti appaiati dei fattori algoritmici sulle metriche di qualità, in punti percentuali (livello alto, indicato tra parentesi, meno livello basso), con la percentuale di coppie in cui la metrica diminuisce e il p-value nominale del test di Wilcoxon. Ogni riga è calcolata su 1536 coppie.

	Scarto				Scarto effettivo			
Fattore	Med.	Media	%	p	Med.	Media	%	p
sforzo *SVC* (20)	−0,31	−1,41	59,6	$1{,}1 \cdot 10^{-149}$	−0,35	−1,64	58,3	$7{,}4 \cdot 10^{-144}$
tipo di schema di taglio (3 stadi)	−0,66	−1,89	67,3	$1{,}1 \cdot 10^{-88}$	−1,15	−2,24	71,7	$8{,}2 \cdot 10^{-113}$
minimizzazione degli schemi (attivata)	0,00	−3,57	49,0	$6{,}5 \cdot 10^{-125}$	0,00	−5,99	49,0	$6{,}4 \cdot 10^{-125}$
gestione a pacco (5 fogli)	10,52	14,73	1,2	$9{,}6 \cdot 10^{-252}$	4,81	7,61	24,3	$2{,}9 \cdot 10^{-88}$
sovraproduzione (attivata)	−1,29	−7,79	70,6	$1{,}3 \cdot 10^{-141}$	8,84	18,03	5,8	$1{,}1 \cdot 10^{-208}$
risoluzione *knapsack* (*greedy*)	2,00	2,32	7,0	$1{,}9 \cdot 10^{-170}$	2,03	3,48	8,3	$6{,}4 \cdot 10^{-174}$

Tabella B.4.: Effetti appaiati dei fattori algoritmici sul tempo di calcolo totale e sul tempo alla migliore soluzione: differenza mediana in secondi (livello alto meno livello basso), rapporto mediano tra i tempi (livello alto su livello basso), percentuale di coppie in cui il livello alto è più rapido e p-value nominale del test di Wilcoxon. Ogni riga è calcolata su 1536 coppie.

	Tempo totale				Tempo alla migliore soluzione		
Fattore	Diff.	Rapp.	%	p	Rapp.	%	p
sforzo *SVC* (20)	2,216	21,07	0,0	$1{,}4 \cdot 10^{-252}$	7,57	13,0	$2{,}5 \cdot 10^{-216}$
tipo di schema di taglio (3 stadi)	0,804	4,54	1,2	$1{,}8 \cdot 10^{-251}$	2,58	9,6	$5{,}7 \cdot 10^{-219}$
minimizzazione degli schemi (attivata)	0,278	1,83	9,2	$1{,}7 \cdot 10^{-231}$	1,30	24,0	$1{,}2 \cdot 10^{-130}$
gestione a pacco (5 fogli)	−0,175	0,55	86,7	$7{,}2 \cdot 10^{-176}$	0,72	67,1	$6{,}3 \cdot 10^{-48}$
sovraproduzione (attivata)	0,002	1,01	46,2	0,19	0,81	72,3	$9{,}0 \cdot 10^{-89}$
risoluzione *knapsack* (*greedy*)	−0,316	0,54	97,5	$3{,}9 \cdot 10^{-249}$	0,68	80,0	$1{,}2 \cdot 10^{-120}$

B.3. Configurazioni non dominate

Le Tabelle B.5 e B.6 elencano, in ordine di tempo crescente, le configurazioni algoritmiche non dominate nel piano scarto–tempo e nel piano scarto effettivo–tempo (Sezione 4.4.7). Per ciascuna sono riportati i livelli dei sei fattori algoritmici e i valori medi delle due metriche sulle 48 configurazioni di istanza.

Tabella B.5.: Configurazioni algoritmiche non dominate nel piano scarto–tempo.

Sforzo	Schema	Minim.	Pacco	Sovrapr.	*Knapsack*	Scarto (%)	Tempo (s)
3	2 stadi	no	5	no	*greedy*	34,84	0,052
3	2 stadi	no	5	sì	*greedy*	25,21	0,053
3	2 stadi	sì	5	no	*greedy*	25,09	0,076
3	2 stadi	sì	5	sì	*greedy*	21,00	0,083
3	2 stadi	no	assente	no	*greedy*	15,68	0,088
3	2 stadi	no	assente	sì	*greedy*	7,41	0,092
3	2 stadi	sì	assente	sì	*greedy*	6,75	0,172
3	2 stadi	no	assente	sì	esatta	3,93	0,305
3	2 stadi	sì	assente	sì	esatta	3,69	0,532
20	2 stadi	no	assente	sì	esatta	3,60	6,640
20	2 stadi	sì	assente	sì	esatta	3,38	15,067

Tabella B.6.: Configurazioni algoritmiche non dominate nel piano scarto effettivo–tempo.

Sforzo	Schema	Minim.	Pacco	Sovrapr.	*Knapsack*	Scarto eff. (%)	Tempo (s)
3	2 stadi	no	5	no	*greedy*	34,84	0,052
3	2 stadi	sì	5	no	*greedy*	25,09	0,076
3	2 stadi	no	assente	no	*greedy*	15,68	0,088
3	2 stadi	sì	assente	no	*greedy*	15,29	0,158
3	2 stadi	no	assente	no	esatta	13,20	0,303
3	2 stadi	sì	assente	no	esatta	12,97	0,492
3	3 stadi	no	assente	no	esatta	11,46	1,499
3	3 stadi	sì	assente	no	esatta	11,27	3,038
20	2 stadi	sì	assente	no	esatta	11,24	14,216
20	3 stadi	no	assente	no	esatta	9,02	20,214
20	3 stadi	sì	assente	no	esatta	8,86	40,968

Bibliografia

[1] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141:241–252, 2002.

[2] Matheus Francescatto and Alvaro Neuenfeldt Júnior. Two-dimensional bin packing, cutting stock, and open dimension problems: A survey of practical requirements. *Computers & Operations Research*, 183:107209, 2025.

[3] Óscar Oliveira, Dorabela Gamboa, and Elsa Silva. An introduction to the two-dimensional rectangular cutting and packing problem. *International Transactions in Operational Research*, 30:3238–3266, 2023.

[4] Manuel Iori, Vinícius L. de Lima, Silvano Martello, Flávio K. Miyazawa, and Michele Monaci. Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, 289:399–415, 2021.

[5] Claudio Arbib, Fabrizio Marinelli, Andrea Pizzuti, and Roberto Rosetti. A heuristic for a rich and real two-dimensional woodboard cutting problem. In *Proceedings of the 7th International Conference on Operations Research and Enterprise Systems (ICORES 2018)*, pages 31–37, 2018.

[6] Manuel Iori, Vinícius L. de Lima, Silvano Martello, and Michele Monaci. 2DPackLib: a two-dimensional cutting and packing library. *Optimization Letters*, 16:471–480, 2022.

[7] Harald Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44:145–159, 1990.

[8] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183:1109–1130, 2007.