Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi Civita"

Corso di laurea in Informatica

Applicazione di un sistema di raccomandazione in ambito BTB

Relatore
Aiolli Fabio
Università di Padova

Laureando
Daniel Rossi

Ringraziamenti

Ecco raggiunto un altro traguardo molto importante carico di felicità che vorrei condividere con coloro che mi sono stati vicino durante questi anni.

Prima di tutto vorrei porgere i miei ringraziamenti al Professor De Giovanni per il supporto fornitomi durante il periodo di stage e durante la stesura della tesi.

Ringrazio l'azienda Trans-Cel per l'opportunità di stage offertomi, grazie a Filippo, Nicola e Beatrice per le tante ore passate insieme.

Ringrazio la mia famiglia ed in particolare mia madre per lo sforzo fatto nello spronarmi a ricercare un futuro in cui mi sentissi realizzato.

Ai miei compagni di Università, per le giornate passate insieme in Torre Archimede, nostra seconda casa, per le risate, le giornate brutte e quelle belle, grazie di avermi fatto sentire apprezzato e coinvolto, grazie per il tempo dedicatomi, in particolare grazie a Victor e Mihai per gli importanti momenti passati insieme a studiare, sarà difficile trovare migliori compagni di corso di voi.

Infine voglio ringraziare Anna, non ultima per importanza, per avermi sempre sostenuto durante tutti questi anni, per avermi ascoltato nei momenti di difficoltà e aver condiviso con me la felicità dei momenti belli, non penso potrò mai ringraziarti abbastanza per l'aiuto nella revisione di questo documento.

Padova, Dicembre 2018 *Daniel Rossi*

Abstract

Il seguente documento vuole essere un report della collaborazione svoltasi tra l'Università di Padova, nella persona del Professor Aiolli, e l'azienda Estilos SRL nel periodo intercorso tra marzo e agosto 2021.
Lo scopo della collaborazione è stato quello di applicare un sistema di raccomandazione ad un e-commerce BTB, dove i clienti sono intermediari che poi rivendono i prodotti a clienti terzi.
Il tipo di prodotti venduti nell'e-commerce non segue le classiche logiche di vendita in famosi e-commerce come può essere Amazon, quindi si è reso necessario ragionare sui diversi tipi di prodotto disponibili. Nello specifico i dati su cui abbiamo lavorato provengono da un'e-commerce di un'azienda cliente di Estilos e non sono basati, come nei classici problemi di raccomandazioni, su recensioni o valutazioni, bensì sullo storico vendite dell'azienda nel canale e-commerce.
Gli obiettivi del progetto prevedevano di rielaborare lo storico vendite in modo d'avere i dati in forme più usuali, a cui poi applicare gli approcci più popolari al momento nell'ambito dei sistemi di raccomandazione, come il collaborative filtering e il content based filtering.
Più nello specifico si vuole raccomandare a ciascun cliente una lista di prodotti che si ritiene possano interessarlo.
Ci si proponeva inoltre di trovare prodotti simili e correlati dato uno di partenza.
Infine ci si proponeva di vagliare approcci ibridi che permettessero di combinare informazioni che descrivono l'interesse di un cliente verso un prodotto rispetto diverse prospettive quali per esempio la quantità totale acquistata e la recentezza dell'acquisto.

Indice

Lista delle figure

Lista delle tabelle

1 Introduzione

1.1 Contesto progetto

Nel mondo dei software ERP *(Enterprise Resource Planning)*, ossia prodotti software pensati per le aziende che permettono la gestione e il controllo dei procesi e delle funzioni aziendali, uno dei più famosi è di certo il gestionale SAP, il quale è sviluppato in moduli integrabili e a seconda delle esigenze dell'azienda utilizzatrice se ne possono attivare in qualunque combinazione.
Uno di questi moduli è l'e-commerce hybris, utilizzato dalle aziende come canale di vendita online e alcune delle sue potenzialità sono che è altamente personalizzabile e perfettamente integrato con i sistemi SAP, come per esempio con il modulo CRM (*(Customer Relationship Management)*), il quale si occupa di tutte le modalità di gestione delle relazioni con il cliente.

1.2 L'idea di progetto

L'idea nasce, in un'ottica di innovazione del prodotto, all'interno di un progetto aziendale che mira all'ampliamento e miglioramento delle funzionalità di hybris, dove uno degli aspetti su cui si vuole lavorare è quello della personalizzazione dei prodotti mostrati agli utenti dell'e-commerce, si vuole sperimentare racco-

mandazioni sui prodotti basate sullo storico vendite e non sui feedback lasciati dall'utente in quanto la loro raccolta non è prevista dal sistema. Partendo quindi dallo storico vendite di un'azienda Cliente, con hybris configurato in versione BTB, l'obiettivo era quello di utilizzare i dati disponibili per raccomandare a ciascun cliente una lista di prodotti $TopN$ che gli risultassero interessanti e per ciascun prodotto una lista di prodotti simili ad esso, sempre interessanti per il cliente a cui viene mostrato quello specifico prodotto.
Come detto solitamente si parte da feedback impliciti / espliciti dati dagli utenti ai prodotti, ma non essendo disponibili si cercherà di estrarre informazioni relative l'interesse del cliente verso un prodotto rispetto diversi punti di vista, quali può essere la quantità acquistata, la recentezza dell'acquisto, il numero di fatture in cui compare o la spesa totale per quello specifico articolo.
Una volta organizzate le informazioni in delle matrici cliente-prodotto grezze, si voleva eseguire una sorta di preprocessing su di esse, andando a trasformarle in dei rating rispetto una scala comune che fornisse una misura d'interesse del cliente rispetto il prodotto. Si è voluto applicare le tecniche più popolari usate nei sistemi di raccomandazione, quali il collaborative filtering alle rating matrix ottenute dal preprocessing descritto precedentemente e il content-based filtering ai dati descrittivi dei prodotti. Data la non disponibilità di rating si è poi pensato di considerare il problema anche come una next basket recommendation, dove si vanno a vedere le sessioni d'acquisto e in base a queste si predice quella finale, questo approccio potrebbe funzionare nel caso i clienti acquistino spesso gli stessi prodotti.

1.3 Organizzazione del testo

Di seguito viene riportata per ogni capitolo una piccola descrizione delle tematiche trattate:

- **Capitolo 2**: viene mostrato come sono inizialmente organizzati i dati, come sono stati trattati e quali informazioni si è potuto ricavare;
- **Capitolo 3**: viene fatto un breve riepilogo della teoria sui sistemi di raccomandazione, spiegando meglio gli approcci del collaborative filtering e del content based filtering, oltre che spiegando il funzionamento degli algoritmi utilizzati e delle metriche;

- **Capitolo 4**: vengono spiegate le diverse tecniche di preprocessing utilizzate per trasformare i dati grezzi in valutazioni.
- **Capitolo 5**: viene riportata una descrizione della libreria Cornac, dove sono implementati modelli e metriche per l'esecuzione di test;
- **Capitolo 6**: vengono mostrati i risultati delle metriche rispetto i diversi algoritmi applicati al preprocessing dei dati;
- **Capitolo 7**: vengono mostrati i risultati delle metriche rispetto i diversi algoritmi applicati al preprocessing dei dati nella loro versione combinata;
- **Capitolo 8**: vengono mostrati i risultati delle metriche considerando il problema come un next basket recommendation;
- **Capitolo 9**: vengono riportate le conclusioni del lavoro svolto, andando a delineare problemi risolti, criticità e sviluppi per il futuro;

1.4 Convenzioni tipografiche

Il testo adotta le seguenti convenzioni tipografiche:

- ogni acronimo, abbreviazione, parola ambigua o tecnica viene spiegata e chiarificata alla fine del testo;
- ogni parola di glossario alla prima apparizione verrà etichetta come segue: $parola^{[g]}$;
- ogni riga di un elenco puntato terminerà con un ; a parte l'ultima riga che si concluderà con un punto.

2
Analisi dei dati

2.1 Principali tabelle

Come detto precedentemente i dati sono lo storico vendite dell'azienda Cliente estratti dal modulo hybris, questi sono organizzati secondo le tabelle SAP, avremo quindi lo storico delle fatture, dove avremo una tabella per la testata della fattura, ossia la parte descrittiva dove viene riportato l'acquirente, e una tabella per le posizioni della fattura, ossia la parte dovve vengono riportati i materiali acquistati. Abbiamo inoltre due tabelle che riportato rispettivamente l'anagrafica cliente e materiali, dove possiamo trovare informazioni aggiuntive che li descrivono.
Mi è stata inoltre fornita anche una tabella glossario che riportava per ciascuna tabella una breve spiegazione di ogni campo.
Tutte queste tabelle sono state fornite in formato Excel.
Quindi ricapitolando le principali tabelle disponibili sono le seguenti:

- **VBAK**: testata della fattura;
- **VBAP**: posizioni della fattura;
- **MARA**: anagrafica prodotto;
- **KNA1**: anagrafica materiali.

Andiamo ora a vedere per ciascuna di queste tabelle i campi annessi e alcune informazioni di natura statistica:

2.1.1 Tabella VBAK

La tabella VBAK contiene la testata di circa 35000 fatture, datate dall'anno 2016 fino a maggio 2021.
Ciascuna riga della tabella è la testata di una fattura e ne riporta il suo codice identificativo (**VBELN**) insieme con il codice del cliente a cui è associata (**KUNNR**). Inoltre ciascuna fattura riporta data e ora (**ERDAT**, **ERZET**) in cui è stata emessa e l'importo totale e valuta corrispondente (**NETWR**, **WAERK**).

2.1.2 Tabella VBAP

La tabella VBAP è la tabella che contiene le posizioni delle fatture, riporta per ogni fattura il numero prodotti acquistati riportando diverse informazioni, ci sono circa 250000 posizioni. Ciascuna riga della tabella riporta quindi il codice identificativo (**VBELN**) della fattura e il codice identificativo del prodotto acquistato (**MATNR**), poi vengono riportati per quel prodotto il prezzo unitario (**NETPR**), la quantità acquistata (**KWMENG**), la spesa totale con la valuta (**NETWR**, **WAERK**) e il codice gerarchia prodotto storico (**PRODH**), ossia quello salvato in MARA al momento del'emissione della fattura.

2.1.3 Tabella KNA1

La tabella KNA1 riporta l'anagrafica cliente, abbiamo salvati circa 3000 clienti, per ciascuna riga abbiamo il codice cliente (**KUNNR**), il codice paese d'origine (**LAND1**), il nome dell'azienda (**NAME1**), la località (**ORT01**) e la regione (**REGIO**).

2.1.4 Tabella MARA

Come detto la tabella MARA è quella che riporta l'anagrafica dei materiali, nella nostra futura trattazione considereremo questi materiali come prodotti in quanto sono acquistabili all'interno dell'e-commerce.

Ciascuna riga della tabella riporta quindi un prodotto univoco composto dal suo usuale codice identificativo (**MATNR**), dal codice della gerarchia prodotto (**PRODH**) e del gruppo merceologico (**MATKL**) e un breve descrizione testuale (**MAKTX**), poi abbiamo delle informazioni su dimensioni, volumi e peso, per le dimensioni abbiamo il campo (**GROES**) che fornisce le dimensioni nel formato (lunghezza X larghezza X altezza), oppure possiamo trovarli nei campi (**LAENG**, **BREIT**, **HOEHE**), rispettivamente lunghezza, larghezza e altezza con l'unità di misura delle dimensioni (**MEABM**), poi abbiamo per il volume il campo (**VOLUM**) con l'unità di misura (**VOLEH**), e per il peso il campo (**NTGEW**) con l'unità di misura (**GEEWI**).

2.2 Prodotti

Da questo momento in poi faremo riferimento ai materiali chiamandoli prodotti come detto in precedenza.
In totale nella tabella anagrafica materiali (MARA) sono presenti circa 75000 prodotti diversi, mentre i prodotti effettivamente venduti risultano essere molti meno attestandosi all'incirca verso gli 8000.
Abbiamo però due campi interessanti che riguardano la gerarchia prodotto (PRODH) e il gruppo merceologico (MATKL), questi due campi ci permettono di studiare la similarità dei prodotti.

2.2.1 Gerarchia prodotto (PRODH)

Il campo gerarchia prodotto (PRODH) è un campo numerico di 18 cifre utile per separare i prodotti rispetto le diverse categorie su più livelli. Nella tabella secondaria T179 vengono definiti i livelli di gerarchia e le diverse categorie. Vediamoli di seguito:

- **PRODH**: codice gerarchia prodotto;
- **STUFE**: livello gerarchia;
- **VTEXT**: descrizione testuale;

Ciascun codice PRODH contenuto nella tabella T179 avrà rispettivamente il seguente numero di cifre in base al livello di gerarchia (STUFE):

- **STUFE = 1**: 1° livello della gerarchia, il codice sarà di 5 cifre.
- **STUFE = 2**: 2° livello della gerarchia, il codice sarà di 10 cifre, dove le prime 5 identificano la categoria di 1° livello a cui appartengono mentre le restanti 5 indentificano la sotto-categoria di 2° livello.
- **STUFE = 3**: 3° livello della gerarchia, il codice sarà di 18 cifre, dove le prime 10 identificano la categoria di 2° livello a cui appartengono mentre le restanti 8 indentificano la sotto-categoria di 3° livello.

Ciascun prodotto sarà quindi provvisto di un codice di 18 cifre che identificherà una categoria per ogni livello.
Nella tabella VBAP ci sono alcune posizioni dove a parità di codice prodotto (MATNR) si hanno codici PRODH diversi, questo è dovuto al diverso momento temporale in cui sono stati acquistati. Infatti nella tabella VBAP il codice PRODH è storico, ho provveduto per semplicità ad aggiornarli tutti al codice PRODH più recente riportato nella tabella anagrafica materiali MARA. Il numero di prodotti interessati sono circa 100 su 10000 posizioni.

Selezione categorie di 1° livello

PRODH	#tot	#sold	titolo
00010	28	0	categoria1
00020	0	0	categoria2
00030	0	0	categoria3
00040	5	0	categoria4
00050	2	0	categoria5
00090	2	0	categoria6
00100	1117	173	CATEGORIA1
00200	645	130	CATEGORIA2
00250	31	11	CATEGORIA3
00300	405	92	CATEGORIA4
00400	525	36	CATEGORIA5
00500	1715	70	CATEGORIA6
00600	334	6	CATEGORIA7
00700	1	0	CATEGORIA8
00900	70441	7702	CATEGORIA9
00950	28	1	CATEGORIA10
09999	215	0	ALTRO

Nella tabella vengono mostrati i codici PRODH delle categorie di 1° livello, nella colonna #tot il numero di prodotti diversi per quella categoria, nella colonna #sold il numero di prodotti diversi acquistati almeno una volta appartententi a quella categoria ed infine il titolo della categoria. Come possiamo vedere le prime sei categorie con titolo in minuscolo hanno pochi prodotti catalogati in MARA e nessun prodotto venduto.

Chiaramento il fatto che siano tutti a zero è dovuto all'aggiornamento dei codici PRODH di cui abbiamo parlato precedentemente, a prescindere da ciò il numero di posizioni che prima riportavano codici appartenenti alle categorie prese in considerazione non superava la decina, quindi non considerare queste categorie in quanto si è smesso di usarle sembra la scelta più logica.
La categoria ALTRO (09999) non è stata considerata in quanto riporta prodotti

che non sono disponibili sull'e-commerce. Inoltre le categorie SISTEMI RICICLO (00700) e CHEMICALS (00950), dato il basso numero di prodotti presenti in MARA e le basse vendite, si è preferito non considerarle.

Overview categorie di 1° livello

PRODH	*#posizioni*	$\sum_{KWMENG}$	$\mathbb{E}_{KWMENG}$	$\mathbb{E}_{NETPR}$ (€)	$\sum_{NETWR}$ (€)	$\mathbb{E}_{NETWR}$ (€)	*titolo*
00100	5908	15461	2.61	1613.44	3865.97	22840167.61	CATEGORIA1
00200	1936	3219	1.66	5898.09	8640.59	16745496.87	CATEGORIA2
00250	333	2949	8.85	552.99	3772.04	1377422.72	CATEGORIA3
00300	745	1390	1.86	4353.24	5364.34	3996430.94	CATEGORIA4
00400	389	1651	4.24	708.17	2404.47	940777.84	CATEGORIA5
00500	1133	12984	11.46	175.75	1034.63	1172236.58	CATEGORIA6
00600	153	494	3.23	448.52	1338.80	83501.04	CATEGORIA7
00900	239740	1070334	4.46	26.67	66.61	15968039.56	CATEGORIA9
	250339	1108493.51	4.72	1706.86	3225.75	63124073.16	valori riassuntivi

Nella tabella per ogni categoria $PRODH$ di 1° livello possiamo vedere:

- $\#posizioni$: numero di posizioni in cui compaiono prodotti di quella categoria in fattura;
- $\sum_{KWMENG}$: quantità totale di prodotti acquistati appartenenti a quella categoria;
- $\mathbb{E}_{KWMENG}$: quantità media per fattura di prodotti acquistati appartenti a quella categoria;
- $\mathbb{E}_{NETPR}$: prezzo medio per fattura di prodotti acquistati di quella categoria;
- $\sum_{NETWR}$: spesa totale per prodotti di quella categoria;
- $\mathbb{E}_{NETWR}$: spesa totale media per fattura di prodotti di quella categoria;

Dalla tabella possiamo vedere come la categoria RICAMBI & ACCESSORI riporti un prezzo medio per fattura molto più basso rispetto alle altre categorie, questo è dovuto al fatto che i pezzi di ricambio ed accessori non sono macchine o sistemi da usare per fornire un servizio quanto un prodotto per riparare quanto già si possiede, possiamo vedere che in termini di posizioni l'acquisto di pezzi di ricambio copra una cospicua parte delle posizioni in fattura, oltre che valere un importante parte del fatturato per l'azienda. Le altre categorie vendono macchine e sistemi per la pulizia quindi i prezzi medi per prodotti sono molto maggiori e per i clienti finali questi prodotti rappresentano un investimento. Da quanto detto finora si vengono a creare due macro categorie di prodotti:

- **Macchine**: questa macro categoria racchiude le seguenti categorie di 1° livello: 00100, 00200, 00250, 00300, 00400, 00500, 00600;
- **Ricambi**: questa macro categoria invece racchiude la sola categoria 00900.

Dobbiamo dare un'ultima precisazione infine, la maggior parte delle categorie di 2° e 3° livello risultano essere di prodotti della macro categoria delle macchine, quindi la gerarchia è molto più densa orizzontalmente per le macchine rispetto che i pezzi di ricambio, infatti per le macchine le categorie risultano essere i diversi modelli di macchinari disponibili e i prodotti a catalogo di quella categoria sono le varianti dello stesso macchinario. Per i pezzi di ricambio abbiamo solo poche categorie contenitore che li raggruppano tutti.

2.2.2 Gruppo merceologico (MATKL)

Il gruppo merceologico (MATKL) non è organizzato come una gerarchia, come lo è invece la gerarchia prodotto (PRODH), bensì come un insieme di prodotti, in totale abbiamo circa 160 gruppi, dove uno di questi contiene tutti i prodotti che prima abbiamo classificato come macchine. Rispetto il codice PRODH, il gruppo merceologico è più divisivo rispetto i ricambi, questo ci può aiutare in quanto ora siamo in grado di categorizzare anche i ricambi.

2.2.3 Dimensione, volume e peso

I campi rigurdanti dimensione, volume e peso potrebbero essere utili per ricercare una similarità tra i prodotti.
Le informazioni sulle dimensioni, come lunghezza, larghezza e altezza sono praticamente ridondanti nei campi GROES e (LAENG, BREIT, HOEHE) se non per alcuni prodotti dove le informazioni sono esclusive di uno dei due campi.
Per peso e volume abbiamo i rispettivi campi numerici e altri due campi che riportano le unità di misura, per il volume possono essere i metri cubi o i millimetri cubi, per il peso o i kilogrammi o i grammi. La criticità riguardo queste misure sono sulla loro scarsità, infatti su 75000 prodotti avremo informazioni su volume e peso rispettivamente solo sul 20% e 39%, mentre sui prodotti acquistati almeno una volta sul 19% e 5%.

3

Sistemi di raccomandazione

3.1 Introduzione

Uno dei campi più popolari al momento verso cui si rivolge una particolare attenzione è quello dei sistemi di raccomandazione, da ora in poi RS, in quanto l'attività online sta aumentando sempre più e nascono sempre più spesso nuovi servizi che permettono di scegliere oggetti, siano questi prodotti, video, musica, film o molto altro, da cataloghi vastissimi. I sistemi di raccomandazione permettono di navigare questi cataloghi andando a cercare gli oggetti che risultino più interessanti per l'utente.

3.2 Preliminari

In generale possiamo dire che un RS si compone di diversi elementi, in primo luogo abbiamo i cosidetti "attori" del problema, gli user, ossia gli utenti del sistema, e gli item, ossia gli oggetti che si vuole consigliare. Abbiamo a disposizione inoltre informazioni riguardo l'interazione tra user e item solitamente sotto forma di feedback implicito o esplicito, questa misura viene definito rating. Questi vengono utilizzati dal RS, insieme con eventuali dati legati al contesto di user e item, per effettuare raccomandazioni.

3.2.1 Feedback impliciti / espliciti

Solitamente le informazioni che legano user e item, ossia i rating, possono essere di due tipi:

- Implicito: 1 se c'è stata interazione tra lo user e l'item, 0 se non c'è stata;
- Esplicito: valutazione numerica intera in una scala da 1 a N, 0 se non c'è stata interazione.

Nel nostro caso di studio però ci ritroviamo a metà strada in quanto, la quantità per esempio potrebbe essere considerata come un dato esplicito ma non è definita su una scala discreta, mentre se lo considerassimo implicito trascureremmo delle informazioni che possono in qualche modo fornire una misura di interesse.

3.3 User-item interaction matrix

		Items					
		1	*2*	...	*i*	...	*m*
Users	*1*	5	3		1	2	
	2		2				4
	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		

I rating sono organizzati in matrici, dette appunto user-item interaction matrix, dove sulle righe abbiamo gli user mentre sulle colonne abbiamo gli item, nell'incrocio abbiamo riportato il rating. Può essere di rating sia impliciti che espliciti e le celle vuote corrispondono allo 0.

3.4 Task

L'obiettivo del sistema può essere quello di consigliare ad uno user una lista di N item, detta *TopN* che si ritiene possano interessargli, oppure dato un item si può trovare una lista di item che si considerino simili allo stesso.

3.5 Approcci

Definito quindi il task abbiamo diversi modi per poter soddisfare il nostro obiettivo, in generale abbiamo due principali categorie di RS:

- **Non Personalizzato**: andiamo a consigliare i prodotti che globalmente risultano più popolari, ossia che abbiamo complessivamente ricevuto più valutazioni, o quelli con rating più alto. Questo approccio non va a considerare le informazioni relative il singolo user;

- **Personalizzato**: ci sono diversi approcci che vedremo nelle sezioni successive, in generale si fanno raccomandazioni basate sulle informazioni dello user.

I due approcci più famosi negli RS sono il collaborative filtering, dove si cerca di consigliare item ad uno user basandosi su user simili, mentre nel content-based filtering si cerca di raccomandare item simili a quelli con cui si ha già interagito;

3.5.1 Collaborative filtering

Collaborative filtering è un approccio agli RS basato sulla similarità, raccomandiamo ad uno user item interessanti per altri user simili ad esso, e viceversa item simili ad altri item per cui ha dimostrato interesse. La similarità può essere quindi di due tipi: item-based, basata quindi sulla similarità tra prodotti o user-based ossia su quella tra user. Ci sono due approcci possibili al collaborative filtering:

- Memory-based: utilizziamo la matrice dei rating per calcolare la similarità tra user e item, metodi basati sull'algoritmo K nearest neighbour;

- Model-based: utilizziamo dei modelli che attraverso degli algoritmi permettono di predire il rating su item non valutati.

UserKnn

UserKnn è un metodo memory-based che fa uso della matrice dei rating, ogni user avrà quindi un proprio "profilo", ossia la sua riga nella matrice dei rating. L'idea è quella di calcolare la similarità tra tutti gli user, e per ciascuno di essi selezionare i K più simili. Una volta fatta questa operazione è possibile calcolare il rating di un prodotto andando

ItemKnn

Matrix Factorization (MF)

Variational Auto-Encoder for CF (VAECF)

3.5.2 Content-based filtering

4
Test e validazione

Una attività di fondamentale importanza è stata quella di testing dei modelli.

4.1 Istanze

Diamo prima una definizione di istanza:

Una **istanza** è un insieme formato dai pacchi da disporre nel contenitore.

Diamo inoltre la definizione di gruppo di istanze:

Un **gruppo di istanze** è un insieme di istanze accomunate tra loro dalle simili dimensioni dei pacchi o dal numero di oggetti.

La prassi generale nell'eseguire il testing dei modelli è stata quella di creare un centinaio di istanze casuali, eseguire il modello sulle istanze generate e verificare la correttezza delle soluzioni fornite, con l'obiettivo di individuare imperfezioni e testare la tenuta dei vincoli. La verifica della correttezza delle soluzioni veniva fatta attraverso funzioni realizzate appositamente che verranno illustrate nelle sezioni successive. Questa procedura richiedeva circa una ventina di minuti di esecuzione ed è stata utilizzata solo dopo che i modelli avessero espresso, attraverso i loro vincoli, correttamente tutti i requisiti di partenza. La loro esecuzione è

avvenuta su istanze con pochi oggetti per risparmiare tempo e individuare subito eventuali errori.
Dopo la realizzazione di tutti i modelli, sicuri delle soluzioni fornite grazie alla validazione effettuata, si è passati all'esecuzione di 100 test per ciascun gruppo di istanze, riportate di seguito per ottenere maggiori informazioni in merito, il tutto eseguito per ciascun modello.

#ist	width_a	width_b	depth_a	depth_b
0	0.5	2.45	0.5	2.45
1	0.5	1.50	0.5	4.00
2	1.5	2.45	0.5	4.00
3	0.5	1.50	3.0	4.00
4	1.5	2.45	3.0	4.00
5	0.1	1.00	0.1	1.00
6	0.1	1.00	3.0	4.00
7	2.0	2.45	3.0	4.00
8	2.0	2.45	2.0	2.45
9	0.1	1.00	0.1	4.00

Table 4.1: Le dimensioni dei gruppi di istanze eseguite

Nella Tabella 4.1 l'indice #ist è un codice identificativo del gruppo di istanze. Vengono riportati anche gli intervalli entro cui si è deciso debbano rientrare le dimensioni casuali dei pacchi per ogni gruppo di istanze, gli intervalli sono stati così definiti:

- **Larghezza**: [width_a, width_b]
- **Profondità**: [depth_a, depth_b]

4.2 Numero di oggetti

Fin dal modello 2D abbiamo capito che il numero di pacchi per ogni istanza sarebbe dovuto essere limitato in quanto già con sette pacchi i tempi di esecuzione

aumentavano vertiginosamente, abbiamo quindi deciso di scegliere in modo casuale in un intervallo [3,10] il numero di pacchi da assegnare a ciascuna istanza, imponendo un *time limit*[g] di 300 secondi. Questo portava ad ottenere soluzioni ottime e soluzioni *best bound*[g], l'analisi dei risultati sarà divisa in queste due categorie.

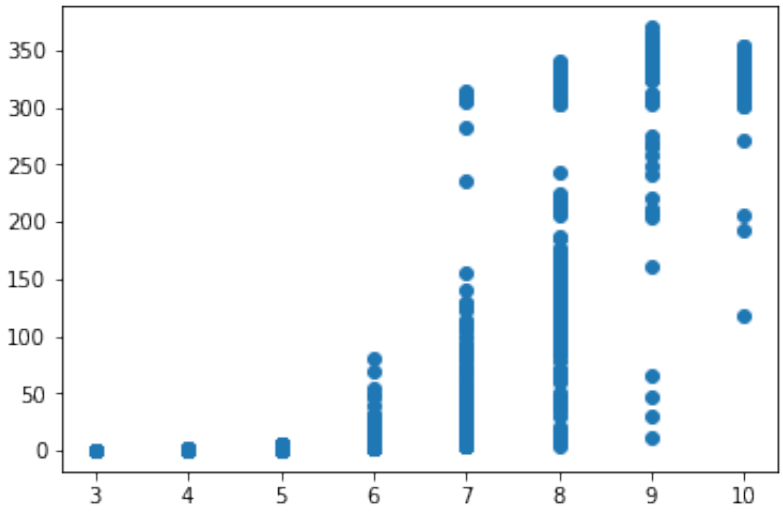

Figure 4.1: Grafico tempi esecuzione modello con diversi oggetti

Nel grafico in Figura 4.1 viene riportato come ascissa il numero di oggetti e come ordinata il tempo espresso in secondi, mostrando che aumentando il numero di oggetti cresca notevolmente il tempo di esecuzione, passando da pochi secondi a minuti.
Risultano molto interessanti quelle istanze che nonostante abbiano molti oggetti richiedano meno tempo per essere eseguiti, forse questo dovuto alle particolari dimensioni degli oggetti.

4.3 Validazione soluzioni fornite

Di fondamentale importanza per il confronto è stato che le soluzioni fornite dal modello fossero corrette, questo ha richiesto lo sviluppo di alcune funzioni che controllassero la validità delle stesse.

Per fare ciò sono state realizzate le seguenti funzioni:

- **CheckFeasible**: utilizzata per controllare che non vi fossero pacchi intersecati tra loro;
- **CheckSequence**: realizzata in due versioni:
  - **2D**: considerando che tutti gli oggetti fossero alla stessa altezza controllava che ciascun pacco avesse almeno una via attraverso cui essere scaricato;
  - **3D**: considerava anche la sovrapposizione tra pacchi e oltre a controllare di avere almeno una delle tre direzioni libere rispetto l'altezza a cui si trovava, controllava anche se sopra ciascun pacco al momento dello scarico vi fossero altri pacchi;

- **CheckStable**: utilizzata per controllare la stabilità degli oggetti, verificava che l'area di base inferiore poggiasse completamente sulle basi superiori degli oggetti sottostanti;
- **ProjectionCommonArea**: utilizzata per verificare se due parallelepipedi avessero intersezioni tra le loro proiezioni rispetto ad una coppia di assi come altezza e profondità.

Inoltre per natura stessa dei modelli essi ricercano la soluzione ottima. Si può quindi dire che se le soluzioni trovate vengono individuate entro il tempo limite e vengono verificate dalle funzioni precedentemente riportate, allora queste possono dirsi effettivamente ottime e reali, perfette candidate per un confronto con le soluzioni corrispettive fornite dall'euristica.

5 Tecnologie e strumenti

Questo capitolo riporta le tecnologie e gli strumenti utilizzati durante il corso dello stage.

5.1 Tecnologie

5.1.1 Python

Figure 5.1: Logo Python

Python[?] , il cui logo riportato in Figura 5.1, è un linguaggio di programmazione tra i più famosi attualmente disponibili, è un linguaggio ad oggetto, sintatticamente semplice e intuitivo che permette di avere un approccio ai problemi molto più operativo. Python[?] è stata la base tecnologica su cui si è basato lo stage e gli strumenti utilizzati. Oltre ad aver usato questo linguaggio abbiamo utilizzato alcune librerie fornite dalla libreria standard.

5.1.2 Google Or-Tools

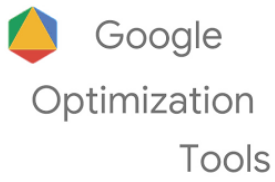

Figure 5.2: Logo Or-Tools

Google Or-Tools[?] , il cui logo riportato in Figura 5.2, è una libreria open source sviluppata da Google che permette di risolvere problemi di ottimizzazione potendo scegliere se utilizzare solver commerciali come CPLEX[?] o Gurobi[?] oppure open source come SCIP[?] o Cbc[?] . Inoltre viene fornita la libreria per diversi linguaggi come:

- C++
- Java
- Python
- C#

Nel nostro caso dato si è deciso di utilizzare la libreria per il linguaggio Python[?] , utilizzandola per realizzare dei modelli di programmazione lineare intera. La libreria è stata imposta dall'azienda dopo il primo periodo di studio degli strumenti e delle tecnologie.

5.1.3 Cbc

Figure 5.3: Logo Cbc

Cbc[?] , il cui logo riportato in Figura 5.3, è un solver di programmazione lineare intera scritto in C++[?] , il motivo per cui si è scelto questo è la compatibilità con la libreria Google Or-Tools e la sua natura open source.

5.1.4 Boost

Figure 5.4: Logo Boost

Boost[?] , il cui logo riportato in Figura 5.4, è un insieme di librerie C++[?] open source che permettono di aumentare le capacità applicative del linguaggio. Viene utilizzato per progetti open source e commerciali. Il suo utilizzo si è reso necessario in quanto le librerie fornite dall'azienda dipendevano da esso, in particolare grazie alla capacità di esporre classi C++[?] al linguaggio Python[?] .

5.1.5 Pandas

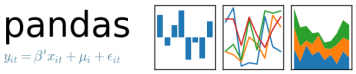

Figure 5.5: Logo Pandas

Pandas[?] , il cui logo riportato in Figura 5.5, è una libreria open source utilizzata con il linguaggio Python[?] , essa è fondamentale per eseguire analisi dei dati, inoltre offre delle funzionalità per la lettura dei dati da file xlsx,csv,txt e la manipolazione degli stessi.
Inoltre fornisce dei comodi metodi di scrittura su file anche verso LaTeX, libreria consigliata dall'azienda.

5.1.6 Matplotlib

Figure 5.6: Logo Matplotlib

La libreria Matplotlib[?] , il cui logo riportato in Figura 5.6, è stata utilizzata durante lo stage per la realizzazione dei grafici e su di essa si basano le immagini 2D e 3D dei pacchi precedentemente mostrate, è stato fondamentale per poter eseguire una verificare quantitativa delle soluzioni. Inoltre è stata utilizzata per realizzare grafici importanti grazie ai quali spiegare in modo efficace alcuni concetti.

5.1.7 Multiprocessing

La libreria Multiprocessing[?] è stata utilizzata per seguire un numero maggiore di test in modo concorrente superando gli impedimenti del *GIL*[g]. Consigliata dall'azienda in quanto i tempi di esecuzione dei test non utilizzando la concorrenza si attestavano alle 10 ore circa, con la suddetta libreria ci attestiamo circa sulle 2 ore.

5.2 Strumenti

5.2.1 Dropbox

Figure 5.7: Logo Dropbox

Dropbox[?] , il cui logo riportato in Figura 5.7, è un servizio di archiviazione condivisa in cloud molto famoso e utilizzato nel mondo, permette di condividere file e documenti di vario genere con altri collaboratori, utile per la documentazione.

Durante lo stage è stato utilizzato per condividere il diario e le presentazioni con i colleghi.

5.2.2 GitHub

Figure 5.8: Logo GitHub

GitHub[?] , il cui logo riportato in Figura 5.8, è un servizio di versionamento basato su repository git[?] . Questo servizio permette di creare repository pubbliche e private con cui versionare il proprio progetto, aspetto improntante nello sviluppo software.

5.2.3 Visual Studio Code

Figure 5.9: Logo Visual Studio Code

Visual Studio Code[?] , il cui logo riportato in Figura 5.9, è un editor open source che grazie alle estensioni installabili permette di utilizzare lo stesso editor per moltissimi linguaggi di programmazione. Inoltre grazie all'estensione per git[?] è possibile versionare direttamente dall'editor, le estensioni permettono anche di eseguire una verifica della sintassi del codice su cui si lavora a livello statico in modo da individuare eventuali errori ed è integrato in esso il terminale con cui lanciare gli script Python[?] .

5.2.4 Jupyter Notebook

Figure 5.10: Logo Jupyter Notebook

Jupyter notebook[?] , il cui logo riportato in Figura 5.10, è applicazione web open source che permette di creare e condividere documenti che condividono del codice. Supporta molti linguaggi di programmazione tra i quali Python[?] e permette una programmazione interattiva generalizzata dal tool IPython[?] . Fondamentale per semplicità di utilizzo e per la possibilità di esportare gli script in molti formati.

5.2.5 PIP

Figure 5.11: Logo Pip

PIP[?] , il cui logo riportato in Figura 5.11, è un gestore di pacchetti Python[?] fondamentale per l'installazione di moduli non presenti nella libreria standard, di facile utilizzo ha permesso di installare velocemente moduli indispensabili per il proseguimento dello stage.

5.2.6 Taiga

Figure 5.12: Logo Taiga

Taiga[7] , il cui logo riportato in Figura 5.12, è una piattaforma di project management utilizzata durante il progetto per la gestione delle task da soddisfare in un certo periodo di tempo.

6 Risultati

Nel seguente capitolo riporteremo i risultati ottenuti dai confronti tra le soluzioni fornite dai modelli e le corrispettive fornite dall'euristica.
Per ciascun modello verranno riportate due tabelle con i risultati ottenuti, queste verranno divise nelle due categorie di soluzioni ottenute:

- ottime;
- best bound.

Per best bound viene intesa una soluzione fornita da un modello dopo che si è raggiunto il suo time limit, definito come tempo massimo di esecuzione e fissato per convenzione a trecento secondi.
Queste soluzioni best bound non sono necessariamente ottime ma possono essere considerate come un buon risultato.
Le Tabelle contengono una riga per ogni gruppo d'istanze numerate da 0 a 9, esse riporteranno il numero d'istanze per gruppo, errore relativo ϵ_r nel rapporto tra i metri lineari ottenuti e errore assoluto ϵ_a espresso in metri, infine il tempo medio di esecuzione di quel gruppo d'istanze, il tempo non sarà riportato per le soluzioni best bound in quanto le soluzioni stesse sono riferite al time limit di trecento secondi.
Successivamente verranno riportate alcune osservazioni sui risultati di alcuni gruppi d'istanze e un rapporto generale sulla bontà delle soluzioni dei modelli

aldilà dei gruppi d'istanze, ricordiamo che sia il modello che l'euristica venivano eseguite sulle stesse istanze e con le stesse caratteristiche, permettendo così un confronto oggettivo.

6.1 Errori

L'obiettivo sia del modello che dell'euristica è quello di minimizzare i metri lineari occupati dai pacchi, per convenzione vengono così definiti:

- Metri lineare modello: Obj_m
- Metri lineare euristica: Obj_h

Gli errori che verranno riportati successivamente vengono calcolati come segue:

$$\epsilon_a = Obj_h - Obj_m$$

$$\epsilon_r = \frac{100(Obj_h - Obj_m)}{Obj_m}$$

6.2 Risultati modello 2DR

Di seguito verranno riportati i risultati ottenuti dal confronto tra soluzioni fornite dal modello e dall'euristica:

	#ist	ϵ_r	ϵ_a	Time
0	64.0	3.89	0.23	40.95
1	73.0	11.90	0.81	31.51
2	76.0	0.94	0.10	19.76
3	84.0	12.29	1.26	19.79
4	75.0	0.00	0.00	27.69
5	73.0	14.17	0.11	12.58
6	78.0	6.60	0.47	20.95
7	76.0	0.00	0.00	36.62
8	81.0	0.00	0.00	23.70
9	81.0	10.34	0.45	10.60

Table 6.1: Risultati 2DR ottimi

	#ist	ϵ_r	ϵ_a
0	36.0	7.35	0.59
1	27.0	15.98	1.48
2	24.0	0.91	0.17
3	16.0	17.26	2.62
4	25.0	0.00	0.00
5	27.0	16.16	0.21
6	22.0	19.40	1.70
7	24.0	0.00	0.00
8	19.0	0.00	0.00
9	19.0	20.58	1.10

Table 6.2: Risultati 2DR best bound

Le prime osservazioni che mi sento di condividere sono sul perché gruppi d'istanze come il 4,7 e 8 riportino errori ϵ_r e ϵ_a uguali a 0, questo perché mediamente i pacchi di tali gruppi d'istanze avevano dimensioni che non permettevano di essere affiancati tra loro, quindi per ogni istanza il modello e l'euristica si sono ridotti ad allineare i pacchi uno davanti all'altro, ottenendo sempre lo stesso risultato. Sembra interessante anche il gruppo d'istanze 9 che riporta mediamente un tempo di esecuzione minore, questo gruppo d'istanze è formata da mix di pacchi stretti e corti e stretti e lunghi, questo porta gli errori ϵ_r per quanto riguarda le soluzioni ottime ad un valore del 10%, mentre con le soluzioni best bound questo aumenta fino al 20%. Considerazione interessante inoltre è come ci si aspetterebbe di veder diminuire gli errori con le soluzioni best bound, questo invece non accade e anzi aumenta il distacco dalle soluzioni fornite dall'euristica, segno che le soluzioni best bound fornite dal modello sono comunque valide per il confronto.

6.3 Risultati modello 2DRS

Di seguito verranno riportati i risultati ottenuti dal confronto tra soluzioni fornite da euristica e modello 2DRS:

	#ist	ϵ_r	ϵ_a	Time
0	76.0	4.74	0.26	41.72
1	75.0	12.44	0.83	27.96
2	82.0	0.56	0.07	20.97
3	72.0	14.59	1.62	29.67
4	81.0	0.00	0.00	28.00
5	75.0	14.87	0.12	17.89
6	78.0	9.18	0.68	17.22
7	85.0	0.00	0.00	13.99
8	79.0	0.00	0.00	25.25
9	89.0	7.69	0.32	12.87

Table 6.3: Risultati 2DRS ottimi

	#ist	ϵ_r	ϵ_a
0	24.0	5.84	0.52
1	25.0	14.26	1.29
2	18.0	0.99	0.18
3	28.0	15.82	2.40
4	19.0	0.00	0.00
5	25.0	16.20	0.20
6	22.0	20.83	1.97
7	15.0	0.00	0.00
8	21.0	0.00	0.00
9	11.0	23.35	1.06

Table 6.4: Risultati 2DRS best bound

Le prime osservazioni da riportare riguardano la sequenza di scarico implementata sia dall'euristica che dal modello. La correttezza della sequenza è garantita dall'euristica attraverso l'ordinamento dei pacchi in base al loro ordine di scarico, insiemi di pacchi appartenenti ad ordini di scarico successivi verranno inseriti per primi, raggruppando così pacchi di ordini uguali in strati, questa tecnica permette di garantire la stabilità generale delle merci, la quale non è invece garantita dal modello che si limita a controllare solo che ogni pacco abbia almeno una delle tre vie di scarico libere come mostrato nella Figura **??** e **??**. Dobbiamo però ricordare come i test eseguiti lavorassero su un numero molto limitato di oggetti, ciò ha limitato molto il problema della stabilità generale permettendo così un confronto tra le coppie di soluzioni. Possiamo osservare come un aumento della complessità del problema non abbia portato però ad un aumento degli errori generalizzato ma anzi in alcuni gruppi d'istanze sia diminuito, mentre i tempi d'esecuzione siano aumentati generalmente. Le istanze con gli errori maggiori restano sempre le

stesse del precedente modello, in particolare salta all'occhio come l'errore ϵ_r del gruppo d'istanze 3 si attesti al 20%, questo gruppo è formato da pacchi stretti e lunghi, valore importante che porta ad uno scarto tra modello e euristica di quasi 3 metri.

6.4 Risultati modello 3D

Di seguito verranno riportati i risultati ottenuti dal confronto tra soluzioni fornite da euristica e modello 3D con rotazione e sovrapposizione:

	#ist	ϵ_r	ϵ_a	Time
0	76.0	5.05	0.25	39.15
1	77.0	11.11	0.67	31.87
2	84.0	0.84	0.10	33.15
3	72.0	9.83	0.96	34.07
4	79.0	0.00	0.00	36.59
5	65.0	9.90	0.08	13.53
6	76.0	6.33	0.45	26.50
7	76.0	0.00	0.00	39.09
8	78.0	0.00	0.00	25.45
9	76.0	10.73	0.37	20.24

Table 6.5: Risultati 3D ottimi

	#ist	ϵ_r	ϵ_a
0	24.0	5.24	0.49
1	23.0	12.87	1.22
2	16.0	0.80	0.11
3	28.0	21.01	2.95
4	21.0	0.00	0.00
5	35.0	14.47	0.17
6	24.0	16.85	1.55
7	24.0	0.00	0.00
8	22.0	0.00	0.00
9	24.0	13.96	0.59

Table 6.6: Risultati 3D best bound

Le istanze che sono state generate non contenevano solo oggetti *stackable*$^{[g]}$, ma contenevano in numero maggiore anche oggetti *fragili* che sopra di sé non potevano averne nessun altro.
Nell'ordinare gli oggetti l'euristica inseriva nel container arbitrariamente oggetti *fragili* e non comportando così che al momento dell'inserimento di un oggetto stackable magari non vi fossero più oggetti compatibili da potervi mettere sopra. Il modello in questione risulta essere più vincolato dell'euristica, in quanto non è stato possibile esprimere il concetto di stabilità degli oggetti nella sua totalità.

Il concetto è stato semplificato in modo che, se un oggetto si trova al di sopra di un altro, allora la sua base inferiore deve essere completamente poggiata alla faccia superiore di quello sotto. L'euristica aveva invece la possibilità che lo stesso pacco potesse poggiare su più pacchi, nei test abbiamo tenuto conto del problema inserendo solo alcuni pacchi stackable.

7 Conclusioni

Nel seguente capitolo verranno riportate le conclusioni che si è potuto trarre alla fine di questo stage.

7.1 Consuntivo finale

Prima di iniziare lo stage, si era concordato insieme con l'azienda la pianificazione dello stesso, definendo obiettivi da raggiungere e calendarizzandoli sulle 320 ore disponibili al fine di poterli soddisfare a pieno. La pianificazione iniziale è stata rispettata nella sua visione generale andando a modificarne invece le scadenze orarie come mostrato nella Tabella 7.1. La formazione è stata più breve del previsto dato che la scelta del software di modellazione algebrica è ricaduta sullo stesso Google Or-Tools[?] , per la realizzazione dei modelli in generale la prototipazione e la scrittura del modello in Google Or-Tools[?] è risultata meno dispendiosa del previsto, questo ha permesso di realizzare un modello non preventivato su consiglio dell'azienda e del tutor accademico. Si è deciso di eseguire i test di confronto con l'euristica nella parte finale dello stage invece che durante la loro realizzazione, questo perché l'intero sistema usato per testare i modelli fosse unico, concentrandosi di più nell'esecuzione di test di validazione che permettessero di verificare le soluzioni fornite dai modelli.

Durata in ore		**Descrizione dell'attività**
32		**A**: Formazione
	8	• Ricerca *framework* di modellazione algebrica
	24	• Studio Google OR - Tools
64		**B**: Versione algoritmo 2D
	8	• Individuazione ed analisi *constraints*
	8	• Prototipazione modello
	24	• Traduzione in Python - Google OR - Tools
	24	• Test validazione
64		**C**: Versione algoritmo 2DR
	8	• Individuazione ed analisi *constraints*
	8	• Prototipazione modello
	24	• Traduzione in Python - Google OR - Tools
	24	• Test validazione
72		**C**: Versione algoritmo 2DRS
	8	• Individuazione ed analisi *constraints*
	16	• Prototipazione modello
	16	• Traduzione in Python - Google OR - Tools
	32	• Test validazione
72		**D**: Versione algoritmo 3D
	8	• Individuazione ed analisi *constraints*
	16	• Prototipazione modello
	16	• Traduzione in Python - Google OR - Tools
	32	• Test validazione
16		**E**: Test confronto modello ed euristica
	4	• Confronto modello 2D-euristica
	4	• Confronto modello 2DR-euristica
	4	• Confronto modello 2DRS-euristica
	4	• Confronto modello 3D-euristica
Totale: 320		

Table 7.1: Ripartizione reale delle ore di stage

7.2 Raggiungimento degli obiettivi

Gli obiettivi concordati prima dell'inizio dello stage e riportati nel **Capitolo 2**, prevedevano il soddisfacimento di otto obiettivi obbligatori, quattro desiderabili e uno opzionale. Tutti gli obiettivi concordati sono stati soddisfatti a pieno e anzi è stato realizzato un modello non preventivato.
L'obiettivo opzionale *op01*: evoluzione euristica attraverso integrazione di nuove funzionalità non è stato inserito nella pianificazione in quanto si è svolto in contemporanea alla realizzazione dei modelli, le nuove funzionalità sono le funzioni di verifica con cui validare le soluzioni, riportate nel **Capitolo 5**.

7.3 Conoscenze acquisite

Le conoscenze acquisite durante il corso dello stage sono state le seguenti:

- Linguaggio Python[?] ;
- Framework Google Or-Tools[?] ;
- Jupyter notebook[?] ;
- Librerie Matplotlib[?] ;
- Librerie Pandas[?] ;
- Modulo Multiprocessing[?] ;
- Package management attraverso PIP[?] ;

7.4 Valutazione personale

La valutazione dello stage svolto presso Trans-Cel non può che dirsi positiva, ho trovato un ambiente accogliente e dei colleghi che sono poi diventati amici. Tecnicamente parlando affrontare un problema così complesso è stato formativo, poiché ho imparato ad usare strumenti utili ed un linguaggio molto popolare ed usato al giorno d'oggi.

I lunghi confronti con il tutor aziendale sono stati importanti per capire le meccaniche dell'euristica. Lo stage è stato la conclusione di un percorso di tre anni molto intensi e credo si possa dire abbia coronato gli sforzi fatti sia dal punto di vista tecnico che umano.

Glossario

- **Agile**: modello di ciclo di vita nato per sopperire alla rigidità dei modelli precedenti, caratterizzato da rilasci rapidi e incrementali, usato per rispondere velocemente alle richieste dei clienti in termini di nuovi requisiti.

- **Best Bound**: è una soluzione fornita da un modello dopo che si è raggiunto il tempo limite di esecuzione, questa soluzione non è ottima ma rappresenta il miglior risultato disponibile.

- **Brainstorming**: incontri di gruppo creativi utili per risolvere problemi o individuare nuove idee. Servono più di due partecipanti in modo che vi sia una discussione arbitraria e quindi utile.

- **Ciclo Di Vita**: insieme degli stati che il prodotto assume dal concepimento al ritiro.

- **Euristica**: algoritmo progettato per risolvere un problema velocemente, spesso una strada obbligata per risolvere problemi molto difficili.

- **GIL**: meccanisco usato dagli interpreti dei linguaggi di programmazione per sincronizzare i thread in modo che ve ne sia in esecuzione in coda.

- **Modello Incrementale**: modello di ciclo di vita che prevede rilasci multipli e successivi, dove ciascuno di essi realizza un incremento di funzionalità. I requisiti vengono trattati per importanza, prima quelli di maggior importanza in modo che possano stabilizzarsi con il rilascio delle versioni fino a quelli minori.

- **Milestone**: punto nel tempo al quale associamo un insieme di stati di avanzamento.

- **Open Source**: termine utilizzato per riferirsi ad un software di cui i detentori dei diritti sullo stesso ne rendono pubblico il codice sorgente.

- **Project Management**: gestione delle attività di analisi, progettazione, pianificazione e realizzazione degli obiettivi di un progetto, compito svolto dal project manager di un'azienda attraverso anche strumenti idonei.

- **Slack**: tempo aggiuntivo ad un'attività che ha lo scopo di evitare ritardi nella produzione del prodotto.
- **Soluzioni**: costrutto software e termine usato per indicare l'insieme di pacchi da disporre e le loro coordinate che permettono di collocarli nel contenitore ed il contenitore stesso..
- **Solver**: software commerciale o open source che permette di risolvere problemi di programmazione lineare.
- **Stackable**: termine utilizzato per indicare se un pacco può avere sopra di sé altri pacchi.
- **Time Limit**: termine usato per indicare un tempo limite entro il quale può essere ricercata una soluzione dal solver.
- **Vehicle Routing**: famiglia di problemi che trattano tutti gli aspetti della gestione di una flotta di veicoli nell'ambito della logistica.

Lista degli acronimi

2D ..... Modello 2D

2DR ..... Modello 2D con rotazione

2DRS ..... Modello 2D con rotazione e sequenza di scarico

3D ..... Modello 3D

CBC ..... Coin-or branch and cut

GIL ..... Global Interpreter Lock