Università degli Studi di Cagliari

Facoltà di Scienze

Corso di Laurea in Informatica

Formulazione di un problema di routing per la City Logistics

Docente di riferimento:
Prof. Massimo Di Francesco

Candidato:
Marco Florenzi
(matr. 60/61/65285)

Anno Accademico 2020–2021

Indice

1 Introduzione **1**

2 Ricerca operativa & Logistica **5**
2.1 Programmazione Lineare Intera 6
2.1.1 Modellazione di un problema di PLI generico 6
2.1.2 Modello di ottimizzazione di un Vehicle Routing Problem 7
2.2 Algoritmi Esatti 9
2.2.1 Cutting-plane 10
2.2.2 Algoritmo Branch-and-Bound 10

3 Un problema di city logistics **15**
3.1 Introduzione 15
3.2 Modellazione di un problema di city Logistics 16
3.3 Un algoritmo euristico per il problema di city-logistics 19

4 Il sotto-problema di routing nel problema di city logistics. **23**

5 Sperimentazione **25**
5.1 Istanze 25
5.2 Test di sperimentazione preliminare 26
5.3 Test di sperimentazione T1 27
5.4 Test di sperimentazione T2 29
5.5 Test di sperimentazione T3 30

6 Conclusioni **33**
6.1 Sviluppi Futuri 33

Bibliografia **37**

Capitolo 1

Introduzione

Il trasporto merci è un'attività fondamentale che consente il compimento diverse mansioni economiche e sociali all'interno delle aree urbane (ad esempio, è necessario rifornire negozi e uffici, consegnare articoli a domicilio e i cittadini devono liberarsi della spazzatura). È un collegamento cruciale tra fornitori e clienti e coinvolge diverse parti interessate (ad es. Vettori, depositi, spedizionieri, destinatari, ecc.). Tuttavia, il trasporto di merci genera disagi nelle aree urbane. I veicoli utilizzati, hanno a che fare con altri mezzi e insieme competono per la capacità stradale, generando problematiche relative alla congestione stradale (ad esempio emissioni, inquinamento e rumore). Chiaramente, queste problematiche hanno un impatto sulla vita dei cittadini, sulla produttività delle aziende, nonché sull'efficienza degli stessi servizi di trasporto merci. Si prevede che questi problemi continueranno a crescere, a causa delle pratiche logistiche basate su scorte ridotte e tempi di consegna ridotti, in particolare nel dominio del commercio elettronico da impresa a cliente. Inoltre, i disagi sono amplificati dalla crescente urbanizzazione: nel 2014, il 54 % della popolazione mondiale viveva in aree urbane questa percentuale dovrebbe aumentare fino al 66 % fino al 2050 [1].

Il trasporto merci all'interno delle aree urbane è stato oggetto di una notevole quantità di ricerche [2], [3]. L'obiettivo generale è diminuire l'impatto causato da esso sulle condizioni di vita cittadine senza penalizzare le attività sociali ed economiche. In particolare, l'obiettivo è ridurre il numero e le dimensioni dei veicoli merci e aumentare l'efficienza dei loro movimenti. L'idea chiave sta nel smettere di considerare ogni spedizione, azienda e veicolo in maniera individuale e iniziare a considerare i singoli stakeholder come parti di un sistema logistico, integrato che necessita essere ottimizzato. Il termine "City Logistics" è stato adottato per denotare la necessità di un sistema ottimizzato di carichi all'interno dei veicoli e l'instradamento di essi per il coordinamento delle attività di trasporto merci, all'interno della città [4]. I piani di City Logistics devono essere organizzati dalle autorità locali/nazionali per prendere decisioni e valutare le politiche al fine di regolari i flussi di merci urbane. A volte questi piani sono realizzati da gestori della mobilità urbana, che devono essere supportati da metodi di ottimizzazione.

La ricerca svolta su tali problematiche si basa sulla rappresentazione delle città come sistemi olistici, dove le attività di coordinamento sono svolte su strutture gerarchiche a due livelli: *terminal principali*, che si trovano ai confini della città, e *depositi (o satelliti)*, che sono punti di ridistribuzione più piccoli all'interno della città, dove le merci vengono trasbordate su veicoli più piccoli per la distribuzione finale [5]. Diverse tipologie di flotte di veicoli vengono dedicate a ciascun livello. Tuttavia, questa topologia di rete non si applica alle città portuali, in cui il porto rappresenta un terminal molto importante situato nel centro cittadino [6]. Questa tesi indaga su questa tipologia variante, che spesso si presenta sui territori insulari.

Il porto rappresenta l'ingresso principale per le merci in entrata e in uscita dalle isole. In queste aree il trasporto merci è necessariamente intermodale (o multimodale), poiché le merci vengono spostate dall'origine alla destinazione mediante una sequenza di almeno due modalità di trasporto [7]. Per migliorare l'efficienza dell'intero processo di distribuzione, i carichi (es. Pallet) vengono inseriti in container (o semirimorchi) per il trasporto marittimo a lungo raggio, sfruttando al contempo l'efficienza delle operazioni locali di ritiro e consegna su camion per la distribuzione nell'ultimo miglio. I vettori possono fornire un servizio personalizzato, in cui l'intero container è dedicato esclusivamente a un particolare cliente (servizio a pieno carico), oppure sul servizio di trasporto su camion in cui il contenuto dei container viene redistribuito a clienti diversi su percorsi con destinazioni diverse (servizio a carico del camion).

In questo studio le destinazioni dei carichi all'interno dei container sono situate nella città circostante al porto (cioè se le destinazioni sono fuori città, i container vengono spostati fuori città e non sono di interesse per il gestore della mobilità urbana. Trascuriamo anche i flussi di merci provenienti dalla città, poiché la maggior parte delle isole è dominante nelle importazioni). I container non possono essere aperti nel porto, perché esso mira a rimuovere rapidamente i container dalle banchine per creare spazio per i container che arrivano a breve dal mare e dalla riva. Le destinazioni in città non possono essere tipicamente raggiunte da veicoli che trasportano container, in quanto non è possibile l'accesso a molte strade o vi è un divieto di transito regolato dalle normative locali in modo da ridurre la congestione e le emissioni inquinanti.

Per affrontare questo problema, è necessario adottare una speciale struttura di distribuzione a due livelli. Nel primo livello, i container vengono spostati per mezzo di veicoli dal porto a strutture intermedie chiamate satelliti, dove i carichi vengono disimballati dai container e imballati in veicoli più piccoli e, possibilmente, ecologici (l'attività dei satelliti è anche chiamata cross-docking [9]). Nel secondo livello, i carichi vengono spostati dai satelliti alle loro destinazioni da veicoli più piccoli. A nostra conoscenza, uno schema di distribuzione simile è stato studiato negli studi sui *dry-ports*, che sono strutture *direttamente collegate a uno o più porti marittimi con mezzi di trasporto ad alta capacità, dove i clienti possono lasciare e ritirare le loro unità standardizzate come direttamente da un porto marittimo* [10]. Essi possono fungere da satelliti speciali per alleviare la congestione nell'entroterra causata dai porti marittimi [11].

Nelle operazioni reali, i container sono tipicamente assegnati *a priori* ai satelliti in base ai contratti esistenti indipendentemente dalla destinazione dei carichi. Chiaramente, questo *modus operandi* è spesso inefficiente, poiché si possono selezionare satelliti "migliori" se si conoscono le destinazioni dei carichi nei container. Ad esempio, un responsabile della mobilità potrebbe determinare la destinazione "media" dei carichi all'interno di un container e selezionare il satellite più vicino a questa destinazione. Ancora più importante, la composizione precisa dei carichi in ogni container può essere ricavata dai dati sulle richieste di trasporto, ma è ancora pressoché inutilizzata per la pianificazione del trasporto merci nelle città marittime. Tuttavia, il potenziale degli sviluppi tecnologici nella condivisione dei dati rappresenta prospettive percorribili per affrontare efficacemente tali inefficienze.

Questa tesi indaga il problema della progettazione (o instradamento) della rete di servizi per il trasporto merci nel secondo livello di una *city port*. Più precisamente, ipotizziamo che i container siano già assegnati ai satelliti secondo politiche predefinite (e, possibilmente, convenienti) e ci concentriamo sui container in entrata riempiti con carichi da consegnare all'interno della città. L'obiettivo è supportare un *mobility manager* nella costruzione di un piano di trasporto per servire la domanda di merci da parte dei clienti operando allo stesso tempo in modo efficiente e redditizio. Questo problema di pianificazione della rete di servizi mira a determinare i percorsi dei veicoli e gli itinerari di ogni carico dal satellite alla destinazione finale.

Questa tesi presenta una formulazione di programmazione mista intera (MIP) per la pianificazione degli itinerari dei veicoli che prelevano i pallet nei satelliti, in modo da poter servire tutti i clienti e i rispettivi carichi a loro correlati, minimizzando i costi complessivi dell'intero sistema. I costi sono descritti tramite una funzione lineare delle variabili decisionali, che sono gli archi selezionati per i pallet e i veicoli. Quindi, un problema di vehicle routing viene rappresentato tramite un problema di ottimizzazione vincolato o modello MILP (Mixed Integer Linear Programming), che può essere risolto da algoritmi in modo da determinare le variabili decisionali.

Diversi risolutori MIP incorporano algoritmi esatti che mirano a determinare la soluzione del problema a minor costo (in questo problema i percorsi a minor costo per veicoli e carichi). Tuttavia, i problemi di vehicle routing sono molto difficili da risolvere, perché nel caso peggiore il tempo di calcolo aumenta esponenzialmente con la dimensione dell'istanza (problemi chiamati $\mathcal{NP}-hard$ in quanto è molto improbabile la risoluzione in tempi polinomiali). Tuttavia, è necessario sperimentare quali sono i limiti di tali algoritmi esatti per disporre di un benchmark per futuri algoritmi euristici che, pur avendo una minore complessità e una maggiore abilità di risolvere problemi più grandi, non possono certificare l'ottimalità della soluzione trovata.

Nel lavoro di tesi vengono riportati alcuni esperimenti computazionali per discutere l'efficacia di un noto risolutore MILP, che è impiegato con diverse impostazioni.

Capitolo 2

Ricerca operativa & Logistica

La *Ricerca Operativa (RO)* è una branca della matematica che si occupa dello studio di problemi decisionali, problemi di tale tipo tendono a presentarsi nella gestione di organizzazioni e aziende che essendo ormai divenuti sistemi complessi, necessitano di un accurato studio di impostazione delle variabili che li compongono, in modo da determinare le scelte da compiere durante la gestione, fornendo lo sviluppo di metodi quantitativi in modo da produrre un approccio decisionale elaborato ed ottimizzato in base al tipo di problema presentatosi. La Ricerca operativa permette quindi di affrontare le scelte in modo ragionato, essendo fortemente legata al concetto di Management le sue applicazioni si ritrovano ormai in tutti i campi della vita odierna (ad esempio umano, politico, economico e sociale).

In questa tesi, si rivolge lo sguardo ai problemi di Logistica, in particolare il Vehicle Routing Problem. La suddetta classe di problemi ha come obbiettivo lo stabilire itinerari ottimali, percorribili da flotte di veicoli in modo facilitare le decisioni di instradamento, necessarie a minimizzare i costi di trasporto derivanti dall'immissione di veicoli con varia capacità di trasporto, su differenti possibili rotte. Un problema di tale tipo, presenta una notevole quantità di potenziali soluzioni e allo stesso tempo, un altrettanto vasto insieme di algoritmi e metodi di ottimizzazione, che rendono la ricerca della soluzione ottimale un'impresa ardua. Tuttavia grazie al miglioramento delle prestazioni dei calcolatori elettronici e delle tecniche di ottimizzazione sviluppate nel corso degli anni, si è giunti alla risoluzione di istanze trattanti ingenti quantitativi di dati con tempistiche relativamente brevi.

In questo lavoro di tesi verrà affrontato un problema che si colloca nell'ambito della logistica urbana in una città marittima. Tale problema prende in considerazione dei clienti, la domanda di pallet di ognuno di essi, i veicoli, la capacità di trasporto di ognuno di essi, i costi unitari per spostare un veicolo da *cliente-i* a *cliente-j* e un insieme di vincoli da rispettare, affinché la soluzione del problema considerata possa definirsi ammissibile. La soluzione generata è l'itinerario di attraversamento dei clienti da parte dei veicoli, viene definita esatta o ottima, quando essa corrisponde a quella con il costo minimo tra tutte quelle ammissibili. Essendo il suddetto problema del tipo $\mathcal{NP} - hard$, è possibile risolvere in modo esatto solamente con istanze di dimensione limitata. Tuttavia, è cruciale determinare la loro soluzione ottima per fornire un benchmark rispetto a cui sviluppare algoritmi euristici, con cui si possono risolvere istanze di dimensione maggiore. In questo capitolo verrà discusso come modellare un problema di Vehicle Routing, attraverso la Programmazione Intera. Successivamente verranno viste delle note tecniche di risoluzione esatta per tale classe di problemi.

2.1 Programmazione Lineare Intera

La *Programmazione Lineare (PL)* rappresenta una delle più importanti classi di problemi di ottimizzazione appartenenti alla Ricerca Operativa. Tale classe di problemi si caratterizza per il fatto che le variabili decisionali sono legate linearmente ai dati. In molte situazioni si può osservare come i componenti di un sistema reagiscano in maniera lineare o approssimativamente lineare, alle decisioni prese, per questo la programmazione lineare trova ampio interesse a livello pratico nella vita di tutti i giorni. Quando alcune variabili che interessano il problema assumono la restrizione di essere intere, si parla di *PL Intera mista* se invece le variabili possono assumere soltanto valori interi si tratta di *PL Intera pura.* Per queste classi di problemi esistono due algoritmi di risoluzione molto importanti, Branch-and-Bound e Branch-and-Cut per i problemi di Programmazione Lineare Intera pura e mista. Questi algoritmi devono la loro importanza al fatto che sono metodi di risoluzione esatti, questo significa che permettono di trovare la miglior soluzione ammissibile (detta ottimo), qualora essa esista, appartenente al problema studiato. Tuttavia molti dei problemi, essendo non risolvibili in tempi polinomiali, il *Branch-and-Bound* e *Branch-and-cut* sono adatti a risolvere istanze ridotte, per quelle con notevole mole di dati è preferibile la risoluzione tramite algoritmo euristico.

2.1.1 Modellazione di un problema di PLI generico

Un problema di programmazione lineare Intera è modellabile matematicamente, per fare ciò avremmo bisogno di una notazione:

- Sia J, l'insieme delle variabili decisionali da determinare;
- Sia I, l'insieme dei vincoli;
- Sia x_j, tale che $j \in J$, la *j-esima* variabile decisionale;
- Sia c_j, tale che $j \in J$, il costo unitario della *j-esima* variabile decisionale;
- Sia u_j, tale che $j \in J$, il valore massimo che può assumere la *j-esima* variabile decisionale;
- Sia b_i la proprietà minima richiesta nel vincolo $i \in I$;
- Sia a_{ij} la quantità di proprietà $i \in I$ presente in una quantità unitaria della variabile decisionale $j \in J$.

Con la medesima notazione è possibile modellare un problema di programmazione lineare, di seguito si riporta il modello di un generico problema di *PL*:

$$z_{PL} = \min \sum_{j \in J} c_j \; x_j \quad \textit{oppure} \quad z_{PL} = c' \; x \tag{2.1}$$

$$\sum_{j \in J} a_{ij}\ x_j \geq b_i \quad oppure \quad Ax \leq b, \quad \forall i \in I \tag{2.2}$$

$$0\ \leq x_j \leq u_j, \quad \forall j \in j \quad oppure \quad 0 \leq x \leq u \tag{2.3}$$

- L' equazione 2.1 rappresenta *la funzione obbiettivo* (o *funzione di costo*), mappa i valori di una o più variabili, su un numero reale associato all'evento. Nei problemi di ottimizzazione vi è la necessità di minimizzare o massimizzare tale funzione;
- L' equazione 2.2 descrive l'uso di vincoli legati ai requisiti detti anche vincoli tecnologici;
- L'equazione 2.3 presenta ulteriori vincoli legati esclusivamente alle variabili decisionali;

Un problema di *PLI* come accennato in precedenza presenta variabili, legate al modello, con vincolo di interezza, nella sua forma canonica è definito come:

$$\begin{cases} z_{PLI} := min & c'\ x \\ & Ax\ \geq\ b \\ & x\ \geq\ 0 \quad intero \end{cases} \tag{2.4}$$

Una volta introdotti, questi concetti fondamentali possiamo modellare un problema di programmazione lineare intero, prendiamo come esempio un semplice problema intero appartenente alla classe dei Vehicle Routing Problem (VRP).

2.1.2 Modello di ottimizzazione di un Vehicle Routing Problem

In questo paragrafo verrà modellato un problema di logistica urbana, il *Vehicle Routing Problem (VRP)*, uno dei problemi di ottimizzazione combinatoria più importanti e studiati, che ha come obbiettivo la determinazione dell'insieme di percorsi ottimale, ovvero l'itinerario di attraversamento ideale, che dovrà essere eseguito da una flotta di veicoli, al fine di servire i pallet richiesti da un dato insieme di clienti.

La rete stradale, utilizzata per il trasporto delle merci, viene generalmente descritta tramite un grafo, i cui archi rappresentano i tratti stradali ed i cui vertici corrispondono alle ubicazioni di depositi e clienti. Ad ogni arco è associato un costo, che generalmente ne rappresenta la lunghezza e un tempo di percorrenza, eventualmente dipendente dal tipo di veicolo o dal periodo durante il quale l'arco viene attraversato. I percorsi effettuati per servire i clienti iniziano e terminano in uno o più depositi, posti ai vertici del tracciato stradale. Ogni deposito è caratterizzato dal numero e dalla tipologia di veicoli ad esso associati e dalla quantità complessiva di merci che può trattare, inoltre il problema presenta varie restrizioni *(vincoli)*, necessari alla determinazione delle possibili soluzioni al fine di stabilire quella ottimale.

Utilizzando le tecniche di base della programmazione lineare viste in precedenza possiamo costruire il modello matematico rappresentativo del problema.

Notazione:

- Sia N, l'insieme dei clienti e dei satelliti
- Sia V, l'insieme dei veicoli , |V| = m;
- Sia c_{ij}, il costo associato all'$arco_{ij}$;
- Sia x_{ij}, la variabile booleana tale che se $x_{ij} = 1$, l'arco è attraversato, 0 altrimenti;
- Sia d, il deposito;
- Sia r_i, la quantità di pallet demandata dal $cliente_i$;
- Sia b_v, la capacità di carico del $veicolo_i$.

Funzione obbiettivo:

$$\min z_{vrp} = \sum_{v \in V} \sum_{(i,j) \in N} c_{ij}\ x_{ijv} \tag{2.5}$$

Vincoli:

$$\sum_{v \in V} \sum_{(i,j) \in N} x_{ijv} = 1, \quad i \neq d, j \neq d \tag{2.6}$$

$$\sum_{v \in V} \sum_{j \in N} x_{djv} = m \tag{2.7}$$

$$\sum_{v \in V} \sum_{j \in N} x_{jdv} = m \tag{2.8}$$

$$\sum_{(i,j) \in N} r_j\ x_{ijv} \leq b_v \tag{2.9}$$

$$x_{ijv} \in \{0, 1\} \tag{2.10}$$

La (2.5) , mostra l'impostazione della funzione z_{vrp}, essa rappresenta una relazione tra costo di attraversamento e arco percorso, le cui somme, per ogni veicolo vanno a formare il costo complessivo relativo alla soluzione del problema. Si può notare che la funzione del problema è corredata di vincolo di minimizzazione, ovvero vi è la necessita di trovare il valore minimo di tutte le funzioni possibili (il valore ottimo). Il problema presenta cinque vicoli:

- il primo vincolo indicato dall'equazione (2.6), va ad applicare ai veicoli v_n la restrizione che una volta attraversato un nodo, non vi è la possibilità di poter tornare verso lo stesso nodo, si deve quindi a forza di modo, procedere verso un cliente successivo;

- La (2.7), impone l'obbligo al veicolo in uso di partire dal deposito, mentre la (2.8) di farvi rientro una volta terminato in servizio;
- La (2.9) impone che la quantità di pallet consegnata al cliente sia minore o uguale alla capacità di trasporto del veicolo in servizio;
- infine l'ultima equazione (2.10) impone alla variabile decisionale x_{ijv} (se settata a 1 l'arco è attraversato, 0 altrimenti), di assumere soltanto valori booleani.

Una volta risolto un problema VRP, gli archi attivati (quelli con $x_{ijv=1}$) , andranno a formare le rotte dell'itinerario, i percorsi con costo associato minimo, che soddisfano tutti i vincoli saranno la soluzione ottima del problema di minimizzazione VRP, Nel paragrafo successivo verranno presentate le tecniche di risoluzione esatta per problemi di programmazione intera come questo.

2.2 Algoritmi Esatti

Una volta modellato il problema, si ricerca la sua soluzione ottima. Questa può essere certificata solo con algoritmi esatti, che sono sinteticamente introdotti in questa sezione: il cutting-plane e il branch-and-bound. Essi prendono come punti di partenza la soluzione del Rilassamento Continuo, che è il problema continuo ottenuto ignorando la condizione di interezza sulle variabili decisionali.

$$\begin{cases} z_{PLI} := min \quad c'\,x \\ \qquad\qquad Ax \ \geq \ b \\ \qquad\qquad x \ \geq \ 0, \ intero \end{cases} \longrightarrow \begin{cases} z_{PL} := min \quad c'\,x \\ \qquad\qquad Ax \ \geq \ b \\ \qquad\qquad x \ \geq \ 0 \end{cases} \tag{2.11}$$

Si può notare che il vincolo di interezza forma un reticolo di punti all'interno di R^n. I punti che soddisfano i vincoli $Ax \geq b$ e $x \geq 0$, costituiscono la regione ammissibile del problema di PLI. Indichiamo la regione ammissibile del rilassamento continuo con P. Inoltre definiamo X l'insieme discreto e finito delle soluzioni ammissibili del problema di PLI.

$$P := \{x \geq 0 : Ax \geq b\}, \quad X := P \cap Z^n \tag{2.12}$$

Risolvendo il problema nella sua forma rilassato, otteniamo il valore ottimo del suo rilassamento continuo z_{PL}, essendo $X \subset P$ è facile dimostrare che il valore z_{PL} corrisponderà a un limite inferiore della soluzione ottima del problema intero z_{PLI}

$$z_{PL} := min\{c'x : x \in P\} \ \leq \ min\{c'x : x \in X\} =: z_{PLI}, \tag{2.13}$$

Inoltre, se la soluzione del rilassamento continuo è intera x^* (ovvero $x^* \in X$), la soluzione rilassata z_{PL}, sarà ottima anche per il problema intero, in quanto in questo caso si avrà:

$$z_{PL} = c'x^* = z_{PLI} \tag{2.14}$$

2.2.1 Cutting-plane

In generale, la soluzione x^* del rilassamento continuo presenta tipicamente una o più componenti frazionarie, per ovviare al problema si utilizza la tecnica dei Piani di taglio conosciuta come Cutting-Plane. L'idea di base del metodo del piano di taglio è quella di tagliare parti della regione ammissibile del rilassamento LP, in modo che la soluzione intera ottimale diventi un punto estremo e quindi può essere trovata con il metodo del simplesso. Il Cutting-Plane in sostanza attua il perfezionamento dell'insieme ammissibile mediante l'utilizzo disuguaglianze lineari, chiamate tagli, comunemente utilizzate per trovare soluzioni intere in problemi di tale genere.

Definizione di taglio:

Si consideri un problema di PLI in forma standard:

$$min\{c'x : Ax = b, x \geq 0, x \in N\}$$

e sia $P'' := \{x \geq 0 : Ax = b\}$, il politopo associato al suo rilassamento continuo, dato $x^* \in P''$ possiamo definire il taglio come:

$$a'x \leq a_0, \quad \forall x \in X := P'' \cap \ Z^n \tag{2.15}$$

$$a'x^* \geq a_0 \tag{2.16}$$

Le condizioni (2.15) e (2.16) esprimono il fatto che la disuguaglianza $a'x \leq a_0$ risulta valida per X, ma violata dal punto x^*. Dato x^*, il problema di individuare un taglio del tipo $a' \leq a_0$ viene detto problema di separazione di x^* da X. Procedendo iterativamente con tagli di questo tipo è possibile avvicinarsi alla soluzione ottimale del problema.

L'algoritmo Cutting-Plane nella sua forma generale può quindi essere formulato come segue:

1. Risolvi il rilassamento LP. Ottieni x^*;
2. Se x^* è intero, fermati. Altrimenti trova una disequazione valida (taglio) che escluda x^*;
3. Vai allo step 1.

2.2.2 Algoritmo Branch-and-Bound

Il Branch-and-Buond è un algoritmo di enumerazione implicita che, nella soluzione del rilassamento continuo, individua una variabile frazionaria e le fissa ai valori interi più vicini, generando due sottoproblemi lineari, che possono essere risolti con un algoritmi di programmazione. Iterando questo procedimento (detto branching sulla variabile frazionaria), il problema di PLI si presenta come un insieme di sottoproblemi di programmazione lineare in una struttura ad albero. L'albero non è visitato esplicitamente, infatti, rilevanti parti dell'albero possono essere ignorate, prevenendo la visita di un numero esponenziale di foglie. Il tempo di calcolo può essere ridotto a valori accettabili, scartando a priori spazi in cui son presenti soluzioni "non migliorative", anche se, in presenza di problemi $\mathcal{NP} - hard$, non dobbiamo aspettarci di risolvere problemi troppo grandi. Verranno forniti i principi generali su cui è basata il metodo di enumerazione implicita.

Per via della struttura ad albero, ogni nodo rappresenta un sotto-problema (il nodo radice è il rilassamento lineare) e ogni arco dell'albero rappresenta la relazione di discendenza, che consente di trasferire ai figli i vincoli presenti nel nodo padre.

Branching dei nodi

L'algoritmo Branch-and-Bound è basato sul principio divide-et-impera, che consiste nel suddividere la regione ammissibile X in sotto-insiemi di dimensione minore. Applicando la procedura in maniera ricorsiva si ottiene un albero di problemi lineari che, talvolta, hanno soluzione intera.

Le operazioni ricorsive di branching da problema a sotto-problemi causano la produzione esponenziale di sotto-problemi. Poiché l'enumerazione esplicita è inefficiente e impraticabile a livello risolutivo, è necessario un sistema di esclusione dei sotto-problemi a priori non ottimali. Detta x^* la soluzione ottima frazionaria in un nodo dell'albero, se questa è intera, è anche ammissibile per il problema di PLI. Tra tutte le soluzione intere trovate nell'albero, si ricerca la migliore.

In generale, sia $z_{PLI} = \min\{c'x : Ax = b, x \geq 0, intero\}$, il problema di PLI originale. Si risolve il rilassamento continuo corrispondente a $z_{PL} = \min\{c'x : Ax = b, x \geq 0\}$, se x^* è intero, non occorre procedere ulteriormente, altrimenti occorre scegliere un valore x_h^* frazionario e si esegue il Branch dei due sotto-problemi così definiti:

$$z_{PL2} = \min\{c'x : Ax = b,\ x_h \leq, \lfloor x_h^* \rfloor,\ x \geq 0, intero\} \tag{2.17}$$

$$z_{PL3} = \min\{c'x : Ax = b,\ x_h \geq, \lfloor x_h^* \rfloor,\ x \geq 0, intero\} \tag{2.18}$$

A questo punto si risolvono i due sotto-problemi generati, ognuno di loro verrà risolto in sequenza e se necessario, iterando il procedimento, alla fine di ciò si avrà come risultato la soluzione al problema di PLI.

Criteri di esclusione dei sotto-problemi generati

L'operazione di Branch comporta la generazione dell'albero decisionale, che nel caso peggiore produrrà un numero di foglie 2^n per un albero di profondità n, quindi all'aumentare della grandezza dell'istanza, si andrà a generare una vastità di nodi, di dimensione potenzialmente incalcolabile, sono quindi necessari dei metodi di esclusione dei sotto-problemi da analizzare in modo da abbattere notevolmente i tempi di calcolo già di natura troppo elevati.

In sostanza vi son tre criteri da seguire per evitare l'analisi di molti sotto-problemi generati:

- Vi è la possibilità che la soluzione del rilassamento continuo di alcuni dei nodi foglia dell'albero decisionale corrente sia impossibile da calcolare. Questo accade quando i vincoli iniziali $Ax = b, x \geq 0$, risultano incompatibili con i vincoli di branching relativi al percorso che dal nodo radice conduce al nodo foglia in questione, essi infatti non daranno vita a nodi figli [12].

- Lo stesso accade quando il rilassamento continuo PL_t del $nodo_t$ corrente ha soluzione ottima $x^*_{PL_t}$ intera. In questo caso come detto in precedenza, corrisponderebbe alla soluzione ottima del nodo attuale, in quanto intera è essendo il rilassamento continuo un lower-bound e in questo caso conduce a soluzione intera, occorrerebbe soltanto aggiornare il costo relativo alla soluzione ottima attuale con $c'x^*_{PL_t}$, senza procedere con ulteriore analisi del $nodo_t$ in questione.

- Il terzo criterio è conosciuto come **bounding** consente spesso di eliminare l'analisi di svariati nodi, risolvendo il rilassamento continuo del nodo associato, si ottiene una stima ottimistica del nodo su cui viene calcolato, sia z_{opt} la soluzione ottima attuale e sia $LB = \min\{c'x^* : Ax = b, x \geq 0\}$ se $LB \geq z_{OPT}$, non occorrerà procedere con l'analisi del nodo corrente in quanto il lower-bound sul nodo corrente è maggiore della soluzione ottima attuale.

Implementazione del Branch-and-Bound

Per ottenere un implementazione dell'algoritmo Branch-and-Bound si deve per prima cosa stabilire la regola di selezione della variabile di branching x^*_h, tale regola ha un notevole impatto sul numero di nodi complessivamente generati. Una metodologia molto semplice consiste nel scegliere una variabile la cui componente frazionaria sia il più vicina possibile a 0.5, in modo che la condizione di branching agisca in modo significativo su entrambi i nodi figli. L'efficacia di tale regola è però opinabile: studi computazionali recenti hanno infatti mostrato la regola non dà risultati significativamente diversi da quelli ottenibili scegliendo la variabile di branching completamente a caso (fra quelle frazionarie)[12].

In pratica vengono preferite regole che stimano l'incremento di lower-bound ottenibile ai due nodi figli, preferendo la variabile in cui il prodotto degli incrementi è massimo. Si dovrà inoltre stabilire la regola con cui selezionare il nodo da elaborare nell'iterazione corrente. Vi sono due tecniche principali per la visita dell'albero decisionale:

Tecnica depth first

Si sceglie sistematicamente il nodo non ancora elaborato più profondo, simulando un procedimento di tipo ricorsivo: dato un nodo padre si considera immediatamente il primo dei sui due figli, finchè una delle condizioni di fathoming non costringe l'algoritmo a risalire di livello.

Vantaggi:

- La possibilità di sfruttare le tecniche ricorsive del linguaggio di programmazione, è possibile implementare tale tecnica con semplicità.

Svantaggi:

- Se la scelta ricade su un nodo errato si torna indietro dopo avere attraversato tutto il sotto-albero.

Tecnica best-bound first

La scelta del nodo da elaborare viene fatta scegliendo il nodo t non ancora elaborato con LB_t minimo, e quindi verosimilmente più vicino alla soluzione ottima.

Vantaggi:

- tipicamente vengono generati meno nodi.

Svantaggi:

- l'elaborazione tende a rimanere ai livelli poco profondi dell'albero decisionale, e quindi i problemi sono poco vincolati e difficilmente portano ad un aggiornamento della soluzione ottima corrente. Questo svantaggio può essere ridimensionato individuando in modo "euristico" una buona soluzione intera, da usare per inizializzare la soluzione ottima corrente x_{OPT}.

Formalizzazione dell'algoritmo

Notazione:

- $z_{OPT} = c'x_{OPT}$ = costo associato alla soluzione ottima corrente;
- Q = coda dei nodi foglia attivi, ovvero i nodi finali che possono generare figli;
- $padre[t] = \pm p$, p rappresenta il nodo padre di t (+ se figlio di sinistra, - se figlio di destra);
- $LB[t]$ = lower-bound di t;
- $vbranch[t]$ = indice h della variabile di branching x_h;
- $valore[t]$ = valore x_h^* della variabile di branching.

Pseudo-codice algoritmo:

Algorithm 1: $Branch - and - Bound$

```
begin
    m := 1
    padre[1] := 0
    Q := ∅
    z_opt := valore soluzione euristica(eventualmente + ∞)
    x* := rilassamentoContinuo(min{c′x : Ax = b, x ≥ 0})
    LB[1] := c′x*
    if  x* ∈ Z and c′x* < z_opt then
        x_opt := x*
        z_opt := c′x*
    end
    if LB[1] < z_OPT then
        scegli x*_h
        vbranch[1] := h
        valore[1] := x*_h
        Q := {1}
    end
    while Q ≠ ∅ do
        scegli un nodo t ∈ Q, e poni Q = Q\{t} h := vbranch[t]
        val := valore[t]
        for figlio = 1 to 2 do
            m := m + 1
            if figlio = 1 then
                padre[m] := t
            else
                padre[m] := −t
            end
            definisci il problema PL_m associato al nodo m
              (vincoli di PL_t pi x_h ≤ ⌊val⌋ se figlio = 1
              oppure x_h ≥ ⌈val⌉ se figlio = 2 )
            risolvi il problema PL_m , e sia x*lasoluzioneottimatrovata
            LB[m] := c′x*
            if  x* ∈ Z and c′x* < z_opt then
                x_opt := x*
                z_opt := c′x*
                Q := Q\{j ∈ Q : LB[j] ≥ z_OPT}
            end
            if LB[m] < z_OPT then
                scegli la variabile frazionaria x*_k
                vbranch[m] := k valore[m] := x*_k Q := Q ∪ {m}
            end
        end
    end
end
```

Capitolo 3

Un problema di city logistics

3.1 Introduzione

In un porto è presente una flotta di container arrivata via mare. Ogni container è riempito con pallet, che hanno diverse destinazioni (o clienti) a terra. I container non possono essere aperti nel porto, che ha l'obiettivo di rimuovere rapidamente i container dalle banchine per creare spazio per i container in arrivo nel breve periodo. I container non possono essere trasportati direttamente nelle destinazioni in città, in quanto ciò non è possibile (ad esempio alcune strade potrebbero essere troppo strette) e / o vietate dalle normative locali, in modo da ridurre la congestione, le emissioni e l'inquinamento.

Ad ogni cliente finale è associata una nota richiesta di pallet imballati in container. Poiché la domanda di pallet è altamente personalizzata, i clienti pagano costi diversi per il loro trasporto in ciascuna destinazione. Di conseguenza, sebbene ogni domanda è associata a una richiesta diversa di merce e può essere identificata dalla sua destinazione.

Per far fronte a questo problema, si adotta una struttura di distribuzione a **due livelli**: nel primo livello, i container vengono spostati dai veicoli dal porto ai satelliti, dove i pallet vengono disimballati dai container e trasbordati in veicoli più piccoli ed ecologici, che movimentano i pallet alle loro destinazioni finali nel secondo livello. La selezione dei satelliti ha un costo fisso, indipendente dal carico di lavoro svolto.

Veicoli specifici sono impiegati a ciascun livello: sono indicati come veicoli compatibili con container (CCV) nel primo livello e veicoli compatibili con pallet (PCV) nel secondo livello. Ogni veicolo di entrambi i livelli è distinto e la sua selezione si traduce in un costo fisso (indipendente dalla distanza percorsa dopo la selezione). Tuttavia, è necessario pagare costi aggiuntivi per ogni veicolo e pallet servito in ogni satellite selezionato. Inoltre, il numero di veicoli, container e pallet serviti in ciascun satellite deve essere inferiore alle capacità dipendenti dal satellite in termini di container, PCV e pallet. Queste capacità dipendono dalla disposizione dei satelliti e dalla loro organizzazione interna.

In questa tesi i CCV dovrebbero trasportare un container alla volta nel primo livello e ogni container può essere scaricato solo su un satellite. Inoltre, il riposizionamento dei container vuoti disimballati sui satelliti viene ignorato e non viene considerata la sincronizzazione temporale delle operazioni del veicolo. Le rotte dei PCV sono aperte, ovvero i veicoli non sono tenuti a tornare su alcun satellite o porto dopo aver assistito l'ultimo cliente sulla rotta. Inoltre, la suddivisione è consentita per tutte le destinazioni del secondo livello.

Per riassumere, in questo problema è necessario determinare:

- la selezione dei satelliti;
- la selezione di CCV e PCV;
- le rotte di PCVS dai satelliti ai clienti di secondo livello.
- il trasporto di container dal porto ai satelliti di primo livello;
- il trasporto di pallet dai satelliti ai clienti di secondo livello.

3.2 Modellazione di un problema di city Logistics

Sia n_0 il porto, S l'insieme dei satelliti e Γ l'insieme dei clienti. Si prende in considerazione un grafo $G = (N, A)$, dove l'insieme N di nodi è definito come $\{n_0\} \cup S \cup \Gamma$ e l'insieme A di archi consiste nell'unione di due sottoinsiemi A_1 and A_2, che sono associati rispettivamente al primo e al secondo scaglione:

$$\begin{aligned} A_1 = &\ \{(n_0, s)\ :\ s \in S\} \\ A_2 = &\ \{(i, j)\ :\ i \in S \cup \Gamma, j \in \Gamma,\ i \neq j\}\,. \end{aligned}$$

Sia C be l'insieme dei containers e P_c^γ l'insieme dei pallet $c \in C$ aventi destinazione $\gamma \in \Gamma$. Sia K_1 e K_2 l'insieme dei CCVs e dei PCVs . Le seguenti capacità sono definite come:

- u_k^2: numero massimo di pallet che possono essere trasportati dal veicolo $k \in K_2$ nel secondo livello;
- u_s: numero massimo di container che entrano nel satellite $s \in S$;
- v_s: numero massimo di PCV serviti da satellite $s \in S$;
- π_s:numero massimo di pallet movimentati nel satellite $s \in S$.

Si definiscono i seguenti costi:

- f_s: costo fisso di selezione del satellite $s \in S$;
- λ_{ks}^1:costo di selezione CCV $k \in K_1$ più il costo unitario di assegnazione al satellite $s \in S$;
- λ_{ks}^2: costo di selezione PCV $k \in K_2$ più il costo unitario di assegnazione al satellite $s \in S$;
- β_s: costo unitario per pallet servito dal satellite $s \in S$;
- α_{ksc}^1: costo di trasporto del container $c \in C$ spostato al satellite $s \in S$da un veicolo $k \in K_1$ nel primo livello;
- $\alpha_{k\gamma(i,j)}^2$: costo di trasporto per pallet con destinazione $\gamma \in \Gamma$ attraverso l'arco $(i, j) \in A_2$ dal veicolo $k \in K_2$;

- $\mu_{k(i,j)}$: costo di routing del camion $k \in K_2$ attraverso l'arco $(i,j) \in A_2$nel secondo livello.

Le seguenti variabili decisionali sono definite come:

y_s: Variabile di selezione del satellite, che assume valore 1 se il satellite $s \in S$ è selezionato, 0 altrimenti;

x^1_{ksc}: Variabile di assegnazione del container, che assume valore 1 se il container $c \in C$ è spostato dal $k \in K_1$ dal porto verso il satellite $s \in S$ nel primo scaglione, 0 altrimenti;

$x^2_{k\gamma}(i,j)$: Variabile di trasporto pallet, che rappresenta il numero di pallet spediti lungo l'arco $(i,j) \in A_2$ verso il cliente γ dal veicolo $k \in K_2$;

$w^2_k(i,j)$: Variabile di routing, che assume valore 1 se il veicolo $k \in K_2$ attraversa l'arco $(i,j) \in A_2$, 0 altrimenti.

Di seguito si riporta il modello di ottimizzazione:

$$z = \min \sum_{s\in S} \left(f_s y_s + \beta_s \sum_{k\in K_2} \sum_{\gamma\in\Gamma} \sum_{j:(s,j)\in A_2} x^2_{k\gamma}(s,j) \right) +$$
$$+ \sum_{k\in K_1} \sum_{s\in S} \sum_{c\in C} (\lambda^1_{ks} + \alpha^1_{ksc}) x^1_{ksc} + \sum_{k\in K_2} \sum_{s\in S} \sum_{j\in\Gamma} \lambda^2_{ks} w^2_k(s,j) +$$
$$+ \sum_{k\in K_2} \sum_{(i,j)\in A_2} \mu_{k(i,j)} w^2_k(i,j) + \sum_{k\in K_2} \sum_{\gamma\in\Gamma} \sum_{(i,j)\in A_2} \alpha^2_{k\gamma(i,j)} x^2_{k\gamma}(i,j) \tag{3.1}$$

$$\sum_{k\in K_1} \sum_{s\in S} x^1_{ksc} = 1 \ , \ \forall c \in C \tag{3.2}$$

$$\sum_{k\in K_1} \sum_{c\in C} |P^\gamma_c| x^1_{ksc} = \sum_{k\in K_2} \sum_{j:(s.j)\in A_2} x^2_{k\gamma}(s,j) \ , \ \forall s \in S \, , \forall \gamma \in \Gamma \tag{3.3}$$

$$\sum_{k\in K_2} \sum_{i:(i,\gamma)\in A_2} x^2_{k\gamma}(i,\gamma) = \sum_{c\in C} |P^\gamma_c| \ , \ \forall \gamma \in \Gamma \tag{3.4}$$

$$\sum_{i:(i,\gamma')\in A_2} x^2_{k\gamma}(i,\gamma') -$$
$$- \sum_{i:(\gamma',i)\in A_2} x^2_{k\gamma}(\gamma',i) = 0 \ , \ \forall k \in K_2 \ , \ \forall \gamma', \gamma \in \Gamma \, , \ \gamma' \neq \gamma \tag{3.5}$$

$$\sum_{s\in S} \sum_{c\in C} x^1_{ksc} \leq 1 \ , \ \forall k \in K_1 \tag{3.6}$$

$$\sum_{s\in S} \sum_{j\in\Gamma} w^2_k(s,j) \leq 1 \ , \ \forall k \in K_2 \tag{3.7}$$

$$\sum_{k\in K_1} \sum_{c\in C} x^1_{ksc} \leq u_s y_s \ , \ \forall s \in S \tag{3.8}$$

$$\sum_{k\in K_2} \sum_{\gamma\in\Gamma} \sum_{j\in\Gamma} x^2_{k\gamma}(s,j) \leq \pi_s y_s \ , \ \forall s \in S \tag{3.9}$$

$$\sum_{k\in K_2} \sum_{j\in\Gamma} w^2_k(s,j) \leq v_s y_s \ , \ \forall s \in S \tag{3.10}$$

$$\sum_{i:(i,j)\in A_2} w^2_k(i,j) \geq \sum_{l:(j,l)\in A_2} w^2_k(j,l) \ , \ \forall k \in K_2 \, , \forall j \in \Gamma \tag{3.11}$$

$$\sum_{\gamma\in\Gamma} x^2_{k\gamma}(i,j) \leq u^2_k w^2_k(i,j) \ , \ \forall k \in K_2 \, , \, (i,j) \in A_2 \tag{3.12}$$

$$y_s \in \{0,1\} \ , \ \forall s \in S \tag{3.13}$$

$$x^1_{ksc} \in \{0,1\} \ , \ \forall k \in K_1 \ , \ \forall s \in S \ , \ \forall c \in C \tag{3.14}$$

$$x^2_{k\gamma}(i,j) \in \mathbb{Z}_+ \ , \ \forall k \in K_2 \ , \ \forall \gamma \in \Gamma \ , \ \forall (i,j) \in A_2 \tag{3.15}$$

$$w^2_k(i,j) \in \{0,1\} \ \ , \ \forall k \in K_2 \ , \ \forall (i,j) \in A_2 \tag{3.16}$$

Nella (3.1) si minimizzano i costi di selezione dei satelliti, quelli di selezione dei veicoli e la loro assegnazione ai satelliti, i costi delle operazioni per ogni pallet movimentato ai satelliti, i costi di trasporto di container e pallet e i costi di routing dei veicoli nel secondo livello.

Il vincolo(3.2) impone che ogni container $c \in C$ venga prelevato dal porto e spostato su un satellite $s \in S$ da un CCV $k \in K_1$.

Il vincolo (3.3)garantisce il transito al satellite $s \in S$ dei pallet richiesti da ogni cliente $\gamma \in \Gamma$.

Il vincolo (3.4) garantisce che ogni cliente $\gamma \in \Gamma$ riceva la quantità di pallet richiesta.

Il vincolo (3.5) è il bilancio di flusso dei pallet con la destinazione $\gamma \in \Gamma$ al cliente $\gamma' \in \Gamma$ visitato prima di $\gamma \in \Gamma$.

Secondo i vincoli(3.6) e (3.7), ogni CCV e PCV possono essere assegnati al massimo a un satellite, rispettivamente.

I vincoli (3.8) e (3.9), garantiscono che il numero di container in entrata e di pallet in uscita sia satellite $s \in S$ non possano essere maggiori di u_s e π_s, rispettivamente, se il satellite s è selezionato.

Il vincolo (3.10) impone che il numero di PCV che servono il satellite $s \in S$ non possano essere maggiori di v_s, se il satellite s è selezionato. Questo vincolo non è considerato nel primo livello, poiché è ridondante w.r.t. (3.8) grazie all'assunzione di un container per CCV.

Il vincolo (3.11) impone la conservazione del flusso per ogni PCV in ogni nodo nel secondo livello o la conclusione del suo percorso, se l'ultimo cliente è stato visitato.

Il vincolo (3.12) collega il flusso e le variabili di routing e impone la capacità dei PCV $k \in K_2$ nel secondo livello.

Infine, il dominio delle variabili decisionali è descritto dai restanti vincoli.

3.3 Un algoritmo euristico per il problema di city-logistics

Il metodo di risoluzione che verrà presentato in questo capitolo per il sotto-problema2, si basa sull'idea di scomporre il problema via satellite. Una volta selezionati satelliti e veicoli, i container vengono assegnati ai satelliti, i percorsi dei PCV e dei pallet possono essere determinati separatamente per ciascun satellite risolvendo sotto-problemi più semplici. Pertanto, proponiamo un metodo di soluzione iterativo in cui il problema complessivo può essere suddiviso in due sotto-problemi:

1. Viene presentato un location-allocation problem per determinare i satelliti, assegnare i container ai satelliti, selezionare e assegnare CCVS e PCV ai satelliti. Questo

problema riduce al minimo i costi associati a queste decisioni, in modo tale che ogni container sia assegnato solo a un satellite, ogni veicolo venga selezionato una sola volta, i vincoli di capacità si mantengano in ogni satellite in termini di container, pallet e PCV, che sono garantiti per avere un trasporto sufficiente capacità nel secondo scaglione. Questo problema è indicato come $Prob_1$.

2. Per ogni satellite selezionato nella soluzione di $Prob_1$ viene presentato un problema di progettazione della rete con vincoli di routing, al fine di determinare i percorsi di PCV e pallet dai satelliti ai clienti e ridurre al minimo i costi di instradamento e trasporto. Il problema di progettazione della rete del satellite $s \in S$ è indicato come $Prob_{2s}$.

Le soluzioni di questi sotto-problemi sono integrate per determinare una soluzione fattibile per il problema generale. Il valore della funzione obiettivo è ottenuto dalla somma dei costi (di rotta e di trasporto) di ogni $Prob_{2s}$ e i costi determinati da $Prob_1$. Nella fase successiva, i costi di selezione e assegnazione dei PCV ai satelliti vengono modificati in $Prob_1$ secondo la precedente soluzione di $Prob_{2s}$. Si risolve quindi nuovamente $Prob_1$, per ottenere eventualmente una diversa selezione e assegnazione dei PCV ai satelliti e / o una diversa assegnazione dei container ai satelliti. Questi problemi secondari vengono risolti in sequenza fino a quando le condizioni di arresto non vengano soddisfatte.

La seguente notazione viene presentata per illustrare il metodo di soluzione:

- $\bar{S}$ è il sottoinsieme di satelliti selezionati nella soluzione di $Prob_1$;

- C_s è l'insieme dei container $c \in C$ assegnati al satellite $s \in \bar{S}$ secondo la soluzione di $Prob_1$;

- K_{1s} è l'insieme di CCV selezionato e assegnato al satellite $s \in \bar{S}$ secondo la soluzione di $Prob_1$;

- K_{2s} è l'insieme di PCV selezionato e assegnato al satellite $s \in \bar{S}$ secondo la soluzione di $Prob_1$

- δ_{ks} è la somma dei costi di routing e trasporto di qualsiasi PCV $k \in K_2$ che può essere eventualmente assegnato al satellite $s \in S$; in una fase generica il metodo aggiorna solo i valori di δ_{ks}secondo la soluzione di $Prob_{2s}$, se PCV $k \in K_2$ è stato assegnato al satellite $s \in S$ in $Prob_1$ (i.e. se $k \in K_{2s}$ and $s \in \bar{S}$). Per motivi di rappresentazione, i valori di δ_{ks} possono essere rappresentati come le voci di una matrice δ con $|K_2|$ righe e $|S|$ colonne. La s-esima colonna di δ è indicata da δ_s. Sia anche $\bar{\delta}_s$ una colonna di δ, che viene aggiornata nell'iterazione corrente perché $s \in \bar{S}$.

- z_1^* è la somma dei costi di selezione dei satelliti e CCV, assegnazione dei container ai satelliti e movimentazione dei pallet ai satelliti. Questi costi sono determinati in $Prob_1$, che minimizza anche i costi λ_{ks}^2 di selezione e assegnazione di PCV e quelli δ_{ks} di utilizzo di PCV nel secondo scaglione;

- z_{2s}^* è il costo di routing e trasporto dei pallet da parte di tutti i PCV assegnati al satellite $s \in \bar{S}$ nell'iterazione corrente; questo costo viene calcolato nella soluzione di $Prob_{2s}$;

- z^* è il valore dell'attuale soluzione ammissibile determinata nell'iterazione corrente per il problema originale (3.1)-(3.16); più precisamente, $z^* = z_1^* + \sum_{s\in\bar{S}} z_{2s}^* + \sum_{s\in\bar{S}} \sum_{k\in K_{2s}} \lambda_{ks}^2$, come z_1^* non tiene conto dei costi di selezione e assegnazione dei PCV ai satelliti;

- z_{best} è il valore della migliore soluzione determinata finora nelle precedenti iterazioni del metodo risolutivo.

Questo è il modello del metodo di soluzione:

Set t_{max} and it_{max}; $z_{best} = \infty$, $it = 0$, $t_{exe} = 0$ e inizializza δ;
do
$z_1^*, \bar{S}, K_{1s}, K_{2s}, C_s \forall s \in \bar{S} \leftarrow Prob_1(\delta)$;
for each $s \in \bar{S}$ do
$z_{2s}^*, \bar{\delta}_s \leftarrow Prob_{2s}(K_{2s}, C_s)$;
$\delta_s = \bar{\delta}_s$ in $\delta = (\delta_1, \ldots, \delta_{|S|})$ if $s \in \bar{S}$;
$z^* = z_1^* + \sum_{s \in \bar{S}} z_{2s}^* + \sum_{s \in \bar{S}} \sum_{k \in K_{2s}} \lambda_{ks}^2$;
if $z^* < z_{best}$, then $z_{best} = \bar{z}$ and $it = 0$, else $it = it + 1$;
update t_{exe};
while ($it \leq it_{max}$ and $t_{exe} \leq t_{max}$);

- All'inizio, il numero massimo di iterazioni è impostato su it_{max} e il tempo di esecuzione massimo è impostato su t_{max}.

- Successivamente, si inizializza il valore di z_{best} a ∞; il numero di iterazione it e il tempo di esecuzione t_{exe} sono impostati a zero. I valori del vettore δ vengono inizializzati e saranno utilizzati come dati per risolvere $Prob_1$ per la prima volta.

- Determina nell'iterazione corrente il satellite selezionato $s \in \bar{S}$, i veicoli selezionati K_{1s} e K_{2s} e i container C_s assegnati a ciascun satellite. Pertanto, si conoscono anche i pallet movimentati in ciascun satellite selezionato e si possono determinare i percorsi dei PCV e i percorsi dei pallet da $Prob_{2s}$. La formulazione matematica del sotto-problema $Prob_{2s}$ è riportata nella Sezione **??**.

- Se z^* è inferiore al valore z_{best} della migliore soluzione determinata finora, aggiorna z_{best} con il valore di $\bar{z}$.

- Questi sotto-problemi vengono risolti in sequenza fino a quando il tempo di esecuzione t_{exe} è inferiore o uguale al tempo di esecuzione massimo t_{max} o il numero di iterazioni senza miglioramento it è inferiore o uguale al numero massimo di iterazioni it_{max}.

Capitolo 4

Il sotto-problema di routing nel problema di city logistics.

In questo capitolo si presenta la formulazione matematica del sotto-problema $Prob_{2s}$ introdotto nel capitolo precedente. Tale sotto-problema è formulato come segue:

$$\min z_{2s} = \sum_{k \in K_{2s}} \sum_{(i,j) \in A_2} \left(\mu_{k(i,j)} w_k^2(i,j) + \sum_{\gamma \in \Gamma_s} \alpha^2_{k\gamma(i,j)} x^2_{k\gamma}(i,j) \right) \tag{4.1}$$

$$\sum_{k \in K_{2s}} \sum_{j \in \Gamma_s} x^2_{k\gamma}(s,j) = |P_s^\gamma| \ , \ \forall \gamma \in \Gamma_s \tag{4.2}$$

$$\sum_{k \in K_{2s}} \sum_{i:(i,\gamma) \in A_2} x^2_{k\gamma}(i,\gamma) = \sum_{c \in C(s)} |P_c^\gamma| \ , \ \forall \gamma \in \Gamma_s \tag{4.3}$$

$$\sum_{i:(i,\gamma') \in A_2} x^2_{k\gamma}(i,\gamma') -$$

$$- \sum_{i:(\gamma',i) \in A_2} x^2_{k\gamma}(\gamma',i) = 0 \ , \ \forall k \in K_{2s} \ , \ \forall \gamma', \gamma \in \Gamma_s \ , \ \gamma' \neq \gamma \tag{4.4}$$

$$\sum_{j \in \Gamma_s} w_k^2(s,j) \leq 1 \ , \ \forall k \in K_{2s} \tag{4.5}$$

$$\sum_{i:(i,j) \in A_2} w_k^2(i,j) \geq \sum_{l:(j,l) \in A_2} w_k^2(j,l) \ , \ \forall k \in K_{2s} \, , \, \forall j \in \Gamma_s \tag{4.6}$$

$$\sum_{\gamma \in \Gamma_s} x^2_{k\gamma}(i,j) \leq u_k^2 w_k^2(i,j) \ , \ \forall k \in K_{2s} \, , \, (i,j) \in A_2 \tag{4.7}$$

$$x^2_{k\gamma}(i,j) \in \mathbb{Z}_+ \ , \ \forall k \in K_{2s} \ , \ \forall \gamma \in \Gamma_s \ , \ \forall (i,j) \in A_2 \tag{4.8}$$

$$w_k^2(i,j) \in \{0,1\} \ \ , \ \forall k \in K_{2s} \ , \ \forall (i,j) \in A_2 \tag{4.9}$$

$$x^2_{k\gamma}(i,j) \leq min(\sum_{c \in C(s)} |P_c^\gamma|, u_k^2) w_k^2(i,j) \ , \ \forall k \in K_{2s} \ , \ \forall \gamma \in \Gamma_s \ , \ \forall (i,j) \in A_2 \tag{4.10}$$

- Nella (4.1) si minimizzano i costi di routing dei PCV e i costi di trasporto dei pallet.

- La (4.2) garantisce che i pallet con destinazione $\gamma \in \Gamma_s$ partano dal satellite $s \in S$.
- Secondo la (4.3), ogni cliente riceve la quantità di pallet richiesta.
- I vincoli (4.4) sono il bilancio di flusso dei pallet con la destinazione $\gamma \in \Gamma_s$ al cliente $\gamma' \in \Gamma_s$ visitato prima $\gamma \in \Gamma_s$.
- Secondo i vincoli (4.5), ogni PCV viene utilizzato al massimo una volta.

  I vincoli (4.6) garantiscono che lo stesso PCV entri e lasci un determinato cliente o la conclusione del percorso se l'ultimo cliente è servito.
- I vincoli (4.7) impongono la capacità di trasporto dei PCV e le variabili di instradamento e flusso dei collegamenti.
- i vincoli (4.10) sono detti "forcing constraints". Questo tipo di vincoli è oggetto della sperimentazione di questa tesi: si tratta di un taglio finalizzato a produrre dei bounds migliori, permettendo al branch-and-bound di esplorare un numero di nodi inferiore, rispetto alla formulazione descritta in precedenza. La sua efficacia verrà verificata con dei test i cui risultati sono riportati nel capitolo 5. Tale vincolo impone che il flusso di pallet per ogni destinazione su ogni arco non sia maggiore tra la domanda totale di quella destinazione e la capacità del veicolo impiegato.
- I vincoli (4.8) e (4.9) riportano i domini delle variabili decisionali.

Capitolo 5

Sperimentazione

La fase di sperimentazione rivolge la sua attenzione al sotto-problema $Prob_{2s}$ presentato nel capitolo 4. Il modello di ottimizzazione è stato implementato in OPL e in C++ al fine di condurre il test. La sperimentazione è incentrata sull'efficacia degli effetti prodotti dal vincolo aggiuntivo (4.10) mediante il risolutore cplex. Sono stati effettuati 3 test più uno preliminare (al fine di controllare la correttezza delle implementazioni successive):

1. **Test preliminare**, eseguito con il programma implementato in OPL ed eseguito con ILOG CPLEX Optimization Studio di IBM sulle prime 15 istanze (eseguito su personal computer);

2. Test **T1**, eseguito su tutte le istanze, con implementazione in C++, con libreria concert/cplex (il test è stato eseguito sui server presenti al Palazzo delle Scienze dell'Università degli studi di Cagliari);

3. Test **T2**, eseguito su tutte le istanze con implementazione in C++, con libreria concert/cplex, in questo test viene aggiunto il vincolo 4.10, si analizzeranno gli effetti prodotti da tale vincolo sul Branch-and-Bound.

4. Test **T3**, in sostanza è simile al **T2**, con la differenza che i vincoli sono stati implementati con la modalita *lazy*. Ciò significa che i vincoli 4.10 sono inizialmente ignorati e, se violati, sono reintrodotti automaticamente da cplex.

Si riporta per comodità il vincolo (4.10):

$$x^2_{k\gamma}(i,j) \leq min(\sum_{c \in C(s)} |P^{\gamma}_c|, u^2_k) w^2_k(i,j) \ , \ \forall k \in K_{2s} \ , \ \forall \gamma \in \Gamma_s \ , \ \forall (i,j) \in A_2$$

5.1 Istanze

Nella fase di sperimentazione sono state usate delle istanze create per esperimenti precedenti a questo. Gli elementi caratterizzanti sono suddivisi secondo la notazione del sotto-problema $Prob_{2s}$:

- P, rappresenta il numero di sequenza dell'istanza testata;

- $|\Gamma|$, è il numero di clienti da servire relativo all'istanza considerata;

- $|K|$, è il numero di veicoli relativo all'istanza in questione;
- $\sum_{c \in C(s)} |P_c^\gamma|$, è la domanda complessiva di pallet demandati dai clienti;
- $\sum_{k \in K} \mu_k^2$, rappresenta la capacità totale dei veicoli inseriti nell'istanza;

Si riporta la tabella contenente le istanze testate:

ISTANZE				
P	$\|\Gamma\|$	$\|K\|$	$\sum_{c \in C(s)} \|P_c^\gamma\|$	$\sum_{k \in K} \mu_k^2$
1	6	2	70	80
2	7	5	74	85
3	8	10	67	70
4	9	4	78	80
5	10	8	113	120
6	15	2	117	120
7	16	3	176	195
8	17	2	181	200
9	18	2	207	220
10	19	3	212	250
11	20	3	202	210
12	30	4	389	400
13	40	4	370	375
14	50	3	533	550
15	60	3	649	675
16	70	4	736	810
17	80	2	820	850
18	90	4	924	925
19	100	6	1030	1050

Tabella 5.1: Istanze utilizzate nelle fasi di testing

5.2 Test di sperimentazione preliminare

Nella fase di testing, si è partiti con un test preliminare utilizzando il software ILOG CPLEX Optimization Studio di IBM, con cui è stato implementato il sotto-problema $Prob_{2s}$ presente nel capitolo 4 senza il vincolo aggiuntivo (4.10). Sono state testate le prime 11 istanze, il tempo di risoluzione limite è impostato a 3600 secondi (1 ora), entro cui il risolutore interrompe l'esecuzione e mostra il risultato ottenuto (per il test preliminare è stato deciso di riportare in tabella la dicitura TIMEOUT). Il programma si arresta quando lo scarto tra upper bound e lower bound diventa minore o uguale a 0.01%. La piattaforma HW utilizzata è un personal computer munito di Intel Core I7 e 8 GB di ram.

Vengono valutati i seguenti parametri:

- Tempo di risoluzione complessivo (secondi);

- Best Integer (valore ottimo trovato);
- GAP (valore percentuale)

Il gap rappresenta il divario tra Best Integer ottenuto è il Best Bound (Non inserito nellasuccessiva tabella), viene calcolato secondo la seguente formula:

$$GAP\% = \frac{|BESTBOUND - BESTINTEGER|}{1 * 10^{-10} + |BESTINTEGER|} \tag{5.1}$$

I risultati, sono stati utilizzati per verificare la correttezza delle successive implementazioni, di seguito viene riportata una tabella con i risultati ottenuti:

ILOG CPLEX Optimization Studio					
P	$\|\Gamma\|$	$\|K\|$	TIME RESOLUTION	*BEST INTEGER*	*GAP*(%)
1	6	2	0,14	30.412,1	0,00
2	7	5	3,19	51.754,4	0,01
3	8	10	2.660,89	122.473,3	0,01
4	9	4	111,52	62.494,6	0,01
5	10	8	649,70	72.533,5	0,01
6	15	2	92,19	60.024,5	0,01
7	16	3	533,41	65.501,9	0,01
8	17	2	127,59	52.605,9	0,01
9	18	2	383,34	56.263,0	0,01
10	19	3	TIMEOUT	*	*
11	20	3	TIMEOUT	*	*

Tabella 5.2: Risultati test preliminare

5.3 Test di sperimentazione T1

In modo da aumentare le prestazioni del sistema risolutivo è stato deciso di uscire dall'ambiente di sviluppo del programma ILOG CPLEX Optimization Studio, e si passati allo sviluppo di un programma basato su C++ facente uso delle librerie CPLEX/Concert, in modo da permettere una modellizzazione del problema più duttile e prestante. Il test è impostato con time-limit a 3600 secondi e precisione decimale sui GAP e i tempi di calcola a 0,01. La piattaforma HW utilizzata è il server della facoltà di scienze dell'Università degli studi di Cagliari dotato di un processore Intel Xeon Gold 6136 3.00GHz.

Per questo test è stato calcolato:

- tempo di risoluzione complessivo (T.MIP) in secondi (s);
- GAP % (GAP BI BB) tra Best Integer e Best Bound calcolato come nell' equazione (4.1);

- tempo di risoluzione del rilassamento continuo (LP), questo valore è stato calcolato a parte;
- Il GAP %(GAP BI LP) tra Best Integer e rilassamento continuo si riporta la formua nell'equazione (5.2);
- LP il rilassamento continuo;
- Best Bound (BB), il valore migliore con componenti frazionarie;
- Best Integer il valore migliore trovato con componente intera come da vincoli di dominio di appartenenza.

$$GAP\%BI\ LP = \frac{|LP - BESTINTEGER|}{1 * 10^{-10} + |BESTINTEGER|} \tag{5.2}$$

Confrontando la tabella T1 con i risultati del test preliminari condotto sulle prime 10 istanze si evince che il programma che è effettua il test T1 è implementato correttamente, si riporta in seguito la tabella T1:

DATA			C++ / CPLEX / CONCERT TEST senza vincolo aggiuntivo T1						
P	$\|\Gamma\|$	$\|K\|$	T. MIP (s)	GAP BI BB (%)	T. LP (s)	GAP BI LP (%)	LP	BB	BI
1	6	2	0,64	0,00%	0,00	31,82%	20.735,12	30.412,10	30.412,10
2	7	5	5,97	0,00%	0,01	18,38%	42.239,60	51.752,40	51.754,40
3	8	10	401,59	0,01%	0,03	15,24%	103.802,91	122.461,00	122.473,00
4	9	4	78,93	0,01%	0,05	14,80%	53.245,49	62.488,40	62.494,60
5	10	8	367,82	0,01%	0,08	12,20%	63.686,84	72.526,30	72.533,50
6	15	2	57,31	0,01%	0,10	60,13%	23.933,20	60.019,80	60.024,50
7	16	3	769,76	0,01%	0,16	36,64%	41.498,86	65.495,40	65.501,90
8	17	2	144,33	0,01%	0,16	47,39%	27.673,52	52.599,80	52.605,00
9	18	2	217,16	0,01%	0,31	43,02%	32.058,86	56.258,10	56.263,50
10	19	3	3600,09	0,06%	0,40	48,41%	31.428,05	60.877,90	60.916,90
11	20	3	3600,09	2,37%	0,17	52,75%	34.789,33	71.886,10	73.627,50
12	30	4	974,63	0,01%	1,77	38,86%	37.569,80	61.447,50	61.453,00
13	40	4	3600,15	70,61%	4,45	78,95%	39.047,63	54.521,40	185.524,00
14	50	3	3600,19	73,31%	12,08	80,31%	26.811,53	36.349,40	136.184,00
15	60	3	3600,43	95,50%	51,34	95,77%	25.134,60	26.763,10	594.774,00
16	70	4	3600,48	97,52%	204,60	97,57%	26.332,19	26.823,80	1.081.680,00
17	80	2				UNSOLVED			
18	90	4				UNSOLVED			
19	100	6				UNSOLVED			

Tabella 5.3: T1 test senza vincolo

Interpretazione risultati T1

Dalla seguente tabella si evince:

- un tempo di risoluzione medio $T.MIP_{avg} = 1.538,72$ secondi sulle istanze risolte;
- un tempo di risoluzione medio del rilassamento lienare $T.LP_{avg} = 17,23$ secondi sulle istanze risolte;

- un divario medio tra il Best Integer e Best Bound $GAP\%_{avg}BIBB = 21,22$ sulle istanze risolte;
- un divario medio tra il Best Integer e il rilassamento lineare $GAP\%_{avg} = 48,27$ sulle istanze risolte.

5.4 Test di sperimentazione T2

Il test T2 è molto simile al test T1, la differenza sta nel fatto che è stato aggiunto il vincolo (4.10) nel sotto-problema2 presente nel capitolo 4. L'implementazione del test è sempre svolta in C++ tramite librerie CONCERT/CPLEX, il test è impostato con time-limit a 3600 secondi e precisione decimale sui risultati a 0,01. La piattaforma HW utilizzata è il server della facoltà di scienze dell'Università degli studi di Cagliari dotato di un processore Intel Xeon Gold 6136 3.00GHz la stessa del test T1. I parametri di valutazione sono gli stessi del test T1 calcolati sono gli stessi del test T1, si riporta la tabella con i risultati del test:

DATA			C++ / CPLEX / CONCERT TEST con vincolo aggiuntivo T2						
P	$\|\Gamma\|$	$\|K\|$	T. MIP (s)	GAP BI BB (%)	T. LP (s)	GAP BI LP (%)	LP	BB	BI
1	6	2	9,55	0,00%	0,00	31,82%	20.735,12	30.412,10	30.412,10
2	7	5	12,93	0,01%	0,01	18,38%	42.239,60	51.749,30	51.754,40
3	8	10	816,06	0,01%	0,12	15,24%	103.802,91	122.461,00	122.473,00
4	9	4	52,26	0,01%	0,15	14,80%	53.245,49	62.488,40	62.494,60
5	10	8	403.61	0,01%	0,24	12,20%	63.686,84	72.526,30	72.533,50
6	15	2	51,77	0,01%	0,68	60,13%	23.933,20	60.019,20	60.024,50
7	16	3	1.250,50	0,01%	0,56	36,64%	41.498,86	65.495,40	65.501,90
8	17	2	141,63	0,01%	0.49	47,39%	27.673,52	52.601,90	52.605,00
9	18	2	315,00	0,01%	1,02	43,02%	32.058,86	56.258,20	56.263,50
10	19	3	797,13	0,01%	1.76	48,41%	31.428,05	60.911,20	60.916,90
11	20	3	3.600,06	2,96%	0.90	53,09%	34.789,33	71.973,70	74.169,60
12	30	4	388,25	0,01%	18,55	38,86%	37.569,80	61.446,90	61.453,00
13	40	4	3.600,97	9,80%	33,96	58,19%	39.047,63	84.233,70	93.387,00
14	50	3	3.601,23	5,58%	77,86	64,45%	26.811,53	71.204,20	75.412,90
15	60	3	3.600,19	93,08%	305,74	97,70%	25.134,60	75.742,50	1.094.620,00
16	70	4				UNSOLVED			
17	80	2				UNSOLVED			
18	90	4				UNSOLVED			
19	100	6				UNSOLVED			

Tabella 5.4: T2 test con vincolo (4.10) sotto-problema2

Interpretazione risultati T2 e confronto

Dalla seguente tabella si evince:

- un tempo di risoluzione medio $T.MIP_{avgT2} = 1.302,78$ secondi sulle istanze risolte;
- un tempo di risoluzione medio del rilassamento lienare $T.LP_{avgT2} = 36,57$ secondi sulle istanze risolte;
- un divario medio tra il Best Integer e Best Bound $GAP\%_{avgT2}BIBB = 7,43$ sulle istanze risolte;

- un divario medio tra il Best Integer e il rilassamento lineare $GAP\%_{avgT2} = 42,69$ sulle istanze risolte.

Il test T2 ha dimostrato risultati in linea generale, leggermente migliori rispetto al T1 ma non a livello significativo, inoltre nel T2 vi è un'istanza in meno risolta, per l'esattezza la $P16$. Confrontando i valori sulle istanze risolte da T1 e T2 si ottengono i seguenti risultati per il test T1:

- $T.MIP_{avgT1} = 1.401,27$ secondi sulle istanze risolte da T1 e da T2;
- un tempo di risoluzione medio del rilassamento lineare $T.LP_{avgT1} = 4,74$ secondi sulle istanze risolte T1 e da T2;
- un divario medio tra il Best Integer e Best Bound $GAP\%_{avgT1}BIBB = 16,13$ sulle istanze risolte T1 e da T2;
- un divario medio tra il Best Integer e il rilassamento lineare $GAP\%_{avgT1} = 44,98$ sulle istanze risolte da T1 e da T2.

Confronto T1 e T2

Considerando le istanze calcolate da entrambi (T1 e T2) si evince che il Test T2 mostra un miglioramento nel tempo medio di risoluzione complessivo per l'esattezza di: $T.MIP_{avgT1-avgT2} = T.MIP_{avgT1} - T.MIP_{avgT2} = 1.401,27 - 1.302,78 = +98,49 secondi$.

Vi è un peggioramento del tempo di risoluzione medio del rilassamento lineare in T2: $T.LP_{avgT1-avgT2} = T.LP_{avgT1} - T.LP_{avgT2} = 4,74 - 36,57 = -31,83 secondi$.

I due GAP ($GAP\%_{avgT2}BIBB$ e $GAP\%_{avgT2}$) del T2 mostrano un abbassamento del $8,7\%$ e del $2,29\%$ rispetto a T1.

5.5 Test di sperimentazione T3

Il test T3 si caratterizza per il fatto che il vincolo (4.10) è stato inserito nell'implementazione tramite una modalità denominata *lazy*, in modo da verificarne il funzionamento tramite questa modalità. In sostanza l'implementazione risulta uguale ai test T1 e T2 se non per questa differenza, il time-limit dell'esecuzione dell'algoritmo di risoluzione rimane sempre fissato a 3600 secondi, la precisione dei risultati è fissata a 0,01 e infine la piattaforma HW rimane sempre un server dotato di Intel Xeon Gold 6136 3.00GHz. Si riporta in seguito la tabella T3 contenente i risultati dell'esperimento:

Dal report dei risultati si evince un netto peggioramento rispetto ai Test T2 e T1, le istanze dalla $|P12|$ in poi non producono risultati, si riportano i seguenti risultati sulle istanze calcolate:

- un tempo di risoluzione medio $T.MIP_{avgT3} = 633,39$ secondi sulle istanze risolte;

DATA			C++ / CPLEX / CONCERT TEST con vincolo aggiuntivo Lazy mod T3						
P	$\|\Gamma\|$	$\|K\|$	T. MIP (s)	GAP BI BB (%)	T. LP (s)	GAP BI LP (%)	LP	BB	BI
1	6	2	1,16	0,00%	0,00	31,82%	20.735,12	30.412,10	30.412,10
2	7	5	9,49	0,00%	0,01	18,38%	42.239,60	51.752,60	51.754,40
3	8	10	1.695,10	0,01%	0,12	15,24%	103.802,91	122.461,00	122.473,00
4	9	4	48,79	0,01%	0,15	14,80%	53.245,49	62.488,40	62.494,60
5	10	8	212,98	0,01%	0,24	12,20%	63.686,84	72.526,30	72.533,50
6	15	2	38,66	0,00%	0,16	60,13%	23.933,20	60.021,80	60.024,50
7	16	3	401,67	0,01%	0,29	36,64%	41.498,86	65.495,40	65.501,90
8	17	2	77,27	0,01%	0.40	47,39%	27.673,52	52.601,80	52.605,00
9	18	2	116,10	0,00%	0,57	43,02%	32.058,86	56.261,60	56.263,50
10	19	3	764,53	0,01%	0,77	48,41%	31.428,05	60.911,10	60.916,90
11	20	3	3.601,59	1,72%	0,75	52,91%	34.789,33	72.611,90	73.885,50
12	30	4	UNSOLVED						
13	40	4	UNSOLVED						
14	50	3	UNSOLVED						
15	60	3	UNSOLVED						
16	70	4	UNSOLVED						
17	80	2	UNSOLVED						
18	90	4	UNSOLVED						
19	100	6	UNSOLVED						

Tabella 5.5: T3 test con vincolo (4.10) sotto-problema2 in modalità lazy

- un tempo di risoluzione medio del rilassamento lienare $T.LP_{avgT3} = 0,31$ secondi sulle istanze risolte;
- un divario medio tra il Best Integer e Best Bound $GAP\%_{avgT3}BIBB = 0,16$ sulle istanze risolte;
- un divario medio tra il Best Integer e il rilassamento lineare $GAP\%_{avgT3} = 34,63$ sulle istanze risolte.

Tabella del numero di nodi esplorati dal Branch-and-Bound

Si riporta inoltre il report di un ulteriore verifica eseguita separatamente, condotta al fine di quantificare il numero dei nodi esplorati dall'algoritmo Branch-and-Bound i test (condotti separatamente) hanno le stesse specifiche dei precedenti sia HW che software, tali test vengono denominati come $T1_2,\ T2_2,\ T3_2$, se ne riportano successivamente i risultati.

Alla luce dei test condotti e dei risultati ottenuti nel capitolo successivo verranno formulate le conclusioni finali.

Nr. Nodi esplorati dal BB					
DATA			$T1_2$	$T2_2$	$T3_2$
P	$\|\Gamma\|$	$\|K\|$	N.nodi BB	N.nodi BB	N.nodi BB
1	6	2	0	108	227
2	7	5	4.411	8.104	8.110
3	8	10	556.594	653.616	2.320.913
4	9	4	28.304	31.271	46.112
5	10	8	119.755	186.193	141.670
6	15	2	7.522	7.328	8.982
7	16	3	33.078	94.623	37.549
8	17	2	6.552	6.799	6.648
9	18	2	7.562	10.680	4.742
10	19	3	45.034	12.860	20.664
11	20	3	91.239	50.811	131.053
12	30	4	7.751	4.460	UNSOLVED
13	40	4	0	115	UNSOLVED
14	50	3	0	17	UNSOLVED
15	60	3	0	0	UNSOLVED
16	70	4	0	UNSOLVED	UNSOLVED
17	80	2	UNSOLVED	UNSOLVED	UNSOLVED
18	90	4	UNSOLVED	UNSOLVED	UNSOLVED
19	100	6	UNSOLVED	UNSOLVED	UNSOLVED

Tabella 5.6: Nr. Nodi esplorati dal Branch-and-Bound

Capitolo 6

Conclusioni

In questa tesi è stato proposto un modello di programmazione in tera per un problema di routing motivato da un problema più complesso di City Logistics. Il modello è stato implementato con la Ilog Concert Technology del solver Cplex ed è stato impiegato sotto differenti condizioni sperimentali. In particolare è stato analizzato il beneficio prodotte dall'inserimento di disuguaglianze valide.

La sperimentazione ha mostrato che tale inserimento non migliora la soluzione del rilassamento continuo e rallenta la risoluzione delle istanze più piccole. Tuttavia, alcune istanze medie sono risolte meglio, migliorando sia il lower che l'upper bound alla conclusione della computazione. Non è stato riscontrato al miglioramento nel passaggio alla gestione dei vincoli in modalità *lazy*.

6.1 Sviluppi Futuri

Sono attualmente in fase di sviluppo metodologie basate su euristiche per generare soluzioni ammissibili per il sotto-problema $Prob_{2s}$. In particolare si implementerà un algoritmo basato sull'euristica Local-Search in grado di migliorare una soluzione iniziale ammissibile. Tale algoritmo può essere inserito una metaeuristica per prevenire la stagnazione della soluzione in un ottimo locale. Un ulteriore sviluppo futuro sarà l'implementazione del modello di City Logistics in c++ tramite la concert technology.

Ringraziamenti

Mi è doveroso dedicare questo spazio del mio elaborato alle persone che hanno contribuito, con il loro instancabile supporto, alla realizzazione dello stesso.

In primis, un ringraziamento speciale va al mio relatore, prof. Massimo Di Francesco e al prof. Enrico Gorgone che con estrema pazienza mi hanno seguito costantemente nell'apprendimento della materia a livello sia teorico che pratico.

Ringrazio inoltre la mia famiglia e i miei amici che mi hanno supportato fino alla conclusione del percorso universitario.

Bibliografia

[1] United Nations (2014). World urbanization prospects, the 2014 revision. Technicalreport, United Nations, Department of Economic and Social Affairs, PopulationDivision, New York, USA (2014).

[2] Bektas, T., Crainic, T.G. and Van Woensel, T. From managing urban freight tosmart city logistics networks. In Gakis, K., Pardalos, P. Networks Design and Op-timization for Smart Cities, volume 8 of Series on Computers and Operations Re-search, 143–188. World Scientific Publishing (2017).

[3] Savelsbergh, M. and Van Woensel, T. City logistics: Challenges and opportunities.Transportation Science,50(2), 579–590 (2016).4.

[4] Crainic, T. G., Ricciardi, N., and Storchi, G. Models for evaluating and planningcity logistics systems. Transportation science,43(4) 432–454 (2009).

[5] Paddeu, D. Sustainable solutions for urban freight transport and logistics: an anal-ysis of urban consolidation centers. In Sustainable Freight Transport, 121–137.Springer, Cham (2018).

[6] Ignaccolo, M., Inturri, G., Giuffrida, N., Torrisi, V., Cocuzza, E. Sustainability of Freight Transport through an Integrated Approach: the Case of the Eastern SicilyPort System. Transportation Research Procedia,45, 177-184 (2020).

[7] Crainic, T. G., and Kim, K. H. Intermodal transportation. Handbooks in operationsresearch and management science 14: 467–537 (2007).

[8] Calabrò, G., Torrisi, V., Inturri, G., Ignaccolo, M. Improving inbound logistic plan-ning for large-scale real-world routing problems: a novel ant-colony simulation-basedoptimization. European Transport Research Review12, 1-11 (2020).

[9] Wen, M., Larsen, J., Clausen, J., Cordeau, J. F., and Laporte, G. (2009). Vehiclerouting with cross-docking. Journal of the Operational Research Society,60(12),1708-1718.

[10] Roso, V., Woxenius, J., and Lumsden, K. The dry port concept: connecting con-tainer seaports with the hinterland. Journal of Transport Geography:17(5), 338–345(2009).

[11] Crainic, T. G., Dell'Olmo, P., Ricciardi, N., and Sgalambro, A. Modeling dry-port-based freight distribution planning. Transportation Research Part C: EmergingTechnologies,55, 518–534 (2015)

[12] Fischetti, M. (2018). Lezioni di Ricerca Operativa (Italian Edition).