Indice

Introduzione

In questa tesi sono stati esaminati diversi problemi di logistica del trasporto merci indotti da casi reali di studio, definiti nell'ambito di progetti di ricerca industriale e trasferimento tecnologico. Le problematiche affrontate si riferiscono sia al trasporto merci in container, nell'ambito della gestione di terminali marittimi di accumulo e rinvio, sia al trasporto di merci sfuse ma consolidate, nell'ambito della pianificazione della distribuzione e della gestione delle scorte in una catena logistica a due livelli (magazzino centrale e magazzini periferici) con un unico distributore. I modelli decisionali risultanti, di tipo combinatorio e di dimensione elevata, hanno motivato lo sviluppo di procedure euristiche ad hoc. Al fine di valutare numericamente la robustezza e la validità degli approcci proposti, a fronte dell'incertezza presente nei domini reali di riferimento, è stato valutato il comportamento di tali approcci in ambienti stocastici anche attraverso l'integrazione con strumenti di simulazione ad eventi.

La tesi è stata articolata in cinque capitoli.

Capitolo 1. È stato riportato uno studio di alcune meta-euristiche sviluppate negli ultimi anni (Simulated Annealing, Iterated Local Search, Nested Partition, Local Branching), mirato ad evidenziare punti di forza e punti di debolezza delle varie proposte in letteratura, nell'ipotesi di una loro specializzazione ai problemi di logistica focalizzati nella tesi.

Capitolo 2. È stato esaminato un problema di "*Retangular Bin Packing*" in quanto può esser visto come una semplificazione del problema di logistica portuale noto come "*Berth Allocation Problem*" (BAP). Per tale problema è stato sviluppato un nuovo metodo risolutivo, denominato "*Two Stage Local Branching*" (TSLB), che si configura come un'evoluzione della metodologia del *Local Branching*. Il metodo sviluppato è stato confrontato con un'implementazione diretta della tecnica di *Local Branching*, per individuarne le potenzialità rispetto alla possibile scelta di basare sul TSLB una nuova classe di algoritmi per il BAP.

Capitolo 3. È stato riconsiderato il problema del BAP sia a livello tattico sia a livello operativo, a partire da una formulazione di IP consolidata in letteratura, dopo aver aggiunto ulteriori vincoli dettati dall'esperienza sul campo a Gioia Tauro (vincoli che limitano la scelta delle posizioni di ormeggio di una nave). L'integrazione dei due livelli decisionali è stata realizzata con due algoritmi ad hoc (uno per l'ottimizzazione a livello tattico e l'altro per la riottimizzazione a livello operativo) combinati con un simulatore ad eventi che riproduce la dinamica del processo logistico in questione, con tutti gli elementi di incertezza tipici dell'ambiente reale. I risultati numerici su istanze reali hanno confermato l'importanza di riottimizzare la soluzione del BAP fornita dal livello tattico, soprattutto a fronte di ritardi sui tempi delle operazioni di carico e scarico.

Capitolo 4. L'attenzione è stata rivolta alla modellazione e soluzione di un problema di logistica di retroporto, dove occorre pianificare in maniera congiunta la distribuzione delle merci sfuse accumulate in un'area di stoccaggio verso una serie di depositi di secondo livello dispersi sul territorio. In generale, tale problema è denominato *Inventory Routing Problem* (IRP) e, in questa tesi, un metodo euristico di soluzione è stato proposto e specializzato ad un caso reale in cui occorre gestire le scorte di centinaia di prodotti (a tasso di consumo costante) e le rotte di consegna degli stessi verso i depositi sul territorio.

Capitolo 5. È stato affrontato lo studio dell'IRP nell'ipotesi di tassi di consumo aleatori, esaminando gli approcci risolutivi noti dalla letteratura, ma riferiti al caso di un singolo tipo di merce. Sono stati evidenziati i limiti applicativi rispetto ad una possibile estensione al caso multi prodotto trattato in precedenza e si è scelto di sviluppare un nuovo algoritmo basato sul metodo euristico proposto nel capitolo precedente. Attraverso un approccio di tipo *roll-out*, è stato possibile immergere l'algoritmo sviluppato per il caso deterministico in un ambiente di modellazione dove la domanda di ogni singolo prodotto è stata modellata attraverso una opportuna funzione di probabilità.

Metaeuristiche Per Problemi Di Logistica Del Trasporto Merci

1.1. Introduzione

Nelle decisioni tattiche ed operative di Logistica del trasporto merci, possono essere formulati spesso problemi di ottimizzazione che riguardano la selezione della migliore configurazione tra un insieme di variabili al fine di perseguire gli obiettivi prefissati. Essi sembrano dividersi naturalmente in due categorie: quelli in cui le soluzioni sono codificate con variabili a valori reali, e quelli in cui le soluzioni sono codificate con variabili a valori discrete. A questi'ultima categoria afferiscono una classe di problemi di Ottimizzazione Combinatoria (OC). Secondo Papadimitriou and Steiglitz 1982, in problemi di OC, l'obiettivo di ricerca si sviluppa all'interno di un insieme finito o infinitamente numerabile. In genere, l'oggetto della ricerca è un numero a valore intero, un sottoinsieme, una permutazione, o una struttura a grafo.

Definizione 1.1 Un problema di Ottimizzazione Combinatoria $P = (S, f)$ è definito da:

- un insieme di variabili $X = (x_1,..., x_n)$;
- i domini delle variabili $D_1,..., D_n$;
- i vincoli sulle variabili;
- una funzione obiettivo da minimizzare[1], dove $f : D_1 \times ... \times D_n \to \Re^+$.

L'insieme di tutti possibili assegnamenti è:

$$S = \{s = \{(x_1, v_1),..., (x_n, v_n)\} | \, v_i \in D_i, \, s \; soddisfa \; tutti \; i \; vincoli\}.$$

S è solitamente definito spazio di ricerca (o delle soluzioni), ove ciascun elemento dell'insieme è una soluzione ammissibile. Per risolvere un problema di ottimizzazione combinatoria occorre trovare una soluzione $s^* \in S$ con valore minimo di funzione obiettivo, cioè, $f(s^*) \leq f(s) \, \forall s \in S$. s^* è detta soluzione globale ottima di (S, f) e l'insieme $S^* \subseteq S$ è l'insieme delle soluzioni ottime globali.

Esempi di problemi di OC sono il problema del commesso viaggiatore (TSP), il problema dell'assegnamento quadratico (QAP), problemi di *timetabling* e di *scheduling*. Data l'importanza pratica di detti problemi, per affrontarli sono stati sviluppati numerosi algoritmi. Questi algoritmi possono essere classificati in algoritmi esatti o algoritmi approssimati. Per ogni istanza finita di un problema di OC, gli algoritmi esatti garantiscono l'ottenimento di una soluzione ottima in tempo finito (Nemhauser and Wolsey 1988). Eppure, per i problemi di OC che sono NP-hard [Garey and Johnson 1979], non esiste nessun algoritmo in tempo polinomiale, assumendo che $P \neq NP$. Pertanto nel caso peggiore i metodi esatti potrebbero richiede un tempo di calcolo esponenziale. Questo sovente induce a sostenere tempi di calcolo troppo alti per impieghi pratici. Pertanto, l'uso di metodi approssimati per risolvere problemi di OC ha ricevuto attenzione crescente negli ultimi 30 anni. Nei metodi approssimati viene sacrificato l'obiettivo di trovare soluzioni ottime con quello di trovare buone soluzioni in un tempo significativamente inferiore.

Tra i metodi approssimati di base si è soliti distinguere tra metodi costruttivi e metodi di ricerca locale. Gli algoritmi costruttivi generano soluzioni a partire da zero aggiungendo via via componenti ad una soluzione parziale inizialmente vuota, fino a completamento della soluzione. In genere, i metodi approssimati sono più veloci, eppure spesso restituiscono soluzioni di qualità inferiore rispetto agli algoritmi di ricerca locale. Gli algoritmi di ricerca locale partono da una soluzione iniziale e in modo iterativo provano a sostituire alla soluzione corrente una soluzione migliore appartenente al "vicinato" (i.e. *neighborhood*) della soluzione corrente, in cui il vicinato è formalmente definito come segue:

[1] La massimizzazione di una funzione obiettivo f equivale a minimizzare la funzione $-f$. In questa rassegna, senza perdita di generalità, si considerano problemi di minimizzazione.

Definizione 1.2. Una struttura di vicinato è una funzione $N : S \to 2^S$ che assegna ad ogni $s \in S$ un insieme di vicini $N(s) \subseteq S$. $N(s)$ è detto vicinato di s.

L'introduzione di una struttura di vicinato permette di definire il concetto di soluzione di minimo locale.

Definizione 1.3. Una soluzione di minimo locale (o minimo locale) con riferimento ad una struttura di vicinato $N(s)$ è una soluzione per cui $\forall s \in N(\hat{s}) : f(\hat{s}) \leq f(s)$. $\hat{s}$ è detta di minimo locale stretto se $f(\hat{s}) < f(s) \forall s \in N(\hat{s})$.

Negli ultimi 20 anni, è emerso un nuovo tipo di algoritmo approssimato che cerca in sostanza di combinare metodi euristici di base in *framework* di più alto livello volte a esplorare in maniera efficace ed efficiente lo spazio di ricerca. Questi metodi sono oggi comunemente chiamate metaeuristiche[2]. Il termine metaeuristica, introdotto per prima in Glover 1986, deriva dalla composizione di due parole greche. Euristica deriva dal verbo *heuriskein* che significa "trovare", mentre il suffisso *meta* significa "oltre" ovvero a livello superiore. Prima che detto termine venisse ampiamente adottato, le metaeuristiche sono state spesso chiamate euristiche moderne Reeves 1983.

Questa classe di algoritmi comprende[3], ma non si limita a, Iterated Local Search (ILS), Simulated Annealing (SA), e Tabu Search (TS). Ad oggi non si è giunti ad una definizione comunemente accettata per il termine metaeuristica. È solo negli ultimi anni che alcuni ricercatori del campo hanno cercato di proporre una definizione. Nel seguito ne vengono riportate alcune.

Una metaeuristica è formalmente definita come un processo iterativo di generazione delle soluzioni che guida un'euristica subordinata, combinando diversi concetti in maniera intelligente per esplorare e valutare lo spazio di ricerca. Strategie di apprendimento vengono utilizzate per strutturare le informazioni in modo da trovare soluzioni quasi-ottime Osman and Laporte 1996.

Una metaeuristica è un processo iterativo maestro che guida e modifica le operazioni di euristiche subordinate per produrre in modo efficiente soluzioni di alta qualità. Può manipolare una singola soluzione finale (o parziale) o un insieme di soluzioni ad ogni iterazione. L'euristica subordinata può essere una procedure di alto (o basso) livello, o una semplice ricerca locale, o solo un metodo di costruzione Van Laarhoven et al. 1982.

Le metaeuristiche sono generalmente strategie di alto livello che guidano una euristica sottostante più specifica al problema, per aumentare le loro prestazioni. L'obiettivo principale è quello di evitare gli svantaggi legati al miglioramento di tipo iterativo, permettendo al processo di ricerca locale di uscire da ottimi locali. Questo risultato è ottenuto o consentendo mosse peggiorative o mediante la generazione di nuove soluzioni di partenza per la ricerca locale in un modo più intelligente rispetto alla semplice fornitura di una soluzione iniziale generata in modo casuale. Molti di questi metodi possono essere visti come mezzo per introdurre un bias in modo che le soluzioni di alta qualità vengano prodotte in fretta. Detto bias può essere di vario tipo e può essere inquadrato come bias discendente (in base alla funzione obiettivo), bias di memoria (sulla base di decisioni prese in precedenza) o bias di esperienza (in base alle prestazioni precedenti). Molti degli approcci metaeuristici si basano su decisioni probabilistiche effettuate durante la ricerca. Ma, la differenza principale rispetto alla ricerca puramente casuale è che negli algoritmi metaeuristici l'aleatorietà non viene utilizzata ciecamente, ma in maniera intelligente, sulla base della forma di bias appunto Stűtzle 1999.

Una metaeuristica è un insieme di concetti che possono essere utilizzati per definire i metodi euristici che possono essere applicati ad una vasta gamma di problemi diversi. In altre parole, una metaeuristica può essere vista come un *framework* algoritmico di tipo generale che può essere applicato a problemi di ottimizzazione diversi con relativamente poche modifiche da apportare per renderli adeguati ad uno specifico problema.

Riassumendo, si riportano le principali proprietà che caratterizzano le metaeuristiche:

- le metaeuristiche sono strategie che guidano il processo di ricerca;
- l'obiettivo è quello di esplorare in maniera efficiente lo spazio di ricerca al fine di trovare soluzioni (quasi)ottime;

[2] L'importanza crescente delle metaeuristiche è evidenziata dalla conferenza biennale Metaheuristics International Conference (MIC).

[3] In ordine alfabetico.

- le tecniche che costituiscono algoritmi metaeuristici spaziano da semplici procedure di ricerca locale a processi di apprendimento complessi;
- algoritmi metaeuristici sono approssimati e di solito non deterministici;
- essi possono includere meccanismi per evitare di rimanere intrappolati in aree confinate dello spazio di ricerca;
- i concetti base delle metaeuristiche permettono un livello di descrizione di tipo astratto;
- le metaeuristiche non sono specifiche per singoli problemi;
- le metaeuristiche possono avvalersi di conoscenza legata allo specifico dominio sotto forma di euristiche controllate dalla strategia di livello superiore;
- le metaeuristiche moderne più avanzate utilizzano l'esperienza di ricerca (registrata in una qualche forma di memoria) per guidare la ricerca.

In sintesi si può affermare che le metaeuristiche sono strategie di alto livello per esplorare gli spazi di ricerca utilizzando metodi diversi. Vale la pena osservare che è importante l'equilibrio dinamico tra la *diversificazione* e l'*intensificazione*. Il termine diversificazione si riferisce generalmente all'esplorazione dello spazio di ricerca, mentre il termine intensificazione si riferisce alla valorizzazione delle esperienze di ricerca accumulate. Questi termini nascono nell'ambito della Tabu Search Glover and Laguna 1997, ed è importante chiarire che i termini di *esplorazione* e di *valorizzazione* sono a volte utilizzati, invece, per esempio nel campo del calcolo evolutivo [Eiben and Schippers 1998], con un significato più limitato. Infatti, le nozioni di valorizzazione e di esplorazione si riferiscono spesso a strategie di breve termine legate alla casualità, mentre l'intensificazione e la diversificazione fanno riferimento anche alle strategie di medio e lungo termine basate sull'uso della memoria. Rispetto al loro significato iniziale, l'uso dei termini di diversificazione e intensificazione diventa sempre più accettato dall'intero campo delle metaeuristiche.

Le strategie di ricerca delle diverse metaeuristiche sono fortemente dipendente dalla filosofia del medesima metaeuristica. Ci sono diverse filosofie alla base delle metaeuristiche esistenti. Alcune di esse possono essere viste come estensioni intelligenti degli algoritmi di ricerca locale. L'obiettivo di questo tipo di metaeuristica è quello di uscire da minimi locali al fine di procedere nell'esplorazione dello spazio di ricerca e di continuare per trovare (si spera) minimi locali migliori. Questo è, per esempio, il caso del Tabu Search, della Iterated Local Search, della Variable Neighborhood Search (VNS), GRASP e Simulated Annealing. Queste metaeuristiche (chiamate anche metodi di traiettoria) lavorano su una o più strutture di vicinato imposte ai membri (le soluzioni) dello spazio di ricerca.

Appare quasi impossibile realizzare una rassegna completa ed accurata sulle metaeuristiche da tutti i punti di vista. Inoltre, una rassegna su una così vasta area, come quella delle metaeuristiche deve concentrarsi su alcuni aspetti e, quindi, trascurarne purtroppo degli altri. Pertanto, a questo punto è bene specificare che detta rassegna è redatta dal punto di vista concettuale. Ci si vuole soffermare sui concetti che vengono utilizzati nelle diverse metaeuristiche al fine di analizzarne le similitudini e le differenze.

1.2. Classificazione Delle Metaeuristiche

Ci sono diversi modi per classificare e descrivere gli algoritmi di tipo metaeuristico. A seconda delle caratteristiche selezionate per differenziare tra loro le classificazioni, è possibile redigerne diverse, ognuna delle quali è il risultato di un punto di vista specifico. In seguito si riassumono i più importanti modi di classificazione delle metaeuristiche.

Ispirati dalla natura o non ispirati dalla natura. Forse, il modo più intuitivo per classificare le metaeuristiche è basata sulle origini dell'algoritmo. Esistono algoritmi ispirati alla natura, come algoritmi genetici e gli Ant Algorithm, e quelli non ispirati alla natura, come Tabu Search e Iterated Local Search. Questa classificazione non sembra molto significativa per i seguenti due motivi. In primo luogo, molti algoritmi ibridi recenti non rientrano in una classe (o, in un certo senso, si adattano ad entrambi). In secondo luogo, a volte è difficile attribuire con chiarezza un algoritmo ad una delle due classi. Così, per esempio, ci si potrebbe domandare se l'uso della memoria nel Tabu Search non sia ispirato anch'esso alla natura.

Ricerca basata sulla popolazione o sui singoli. Un'altra caratteristica che può essere utilizzata per classificare le metaeuristiche è il numero delle soluzioni utilizzate al contempo: ad un qualsivoglia istante, l'algoritmo lavora su una

popolazione o su una singola soluzione? Gli algoritmi che lavorano sulle singole soluzioni sono chiamati metodi di traiettoria e comprendono metaeuristiche basata sulla ricerca locale, come il Tabu Search, la Iterated Local Search e Variable Neighborhood Search. Tutti condividono la caratteristica di descrivere una traiettoria nello spazio di ricerca durante il processo di ricerca. Le metaeuristiche basate sulla popolazione, al contrario, eseguono processi di ricerca che descrivono l'evoluzione di un insieme di punti nello spazio di ricerca.

Funzioni obiettivo dinamiche e statiche. Le metaeuristiche possono anche essere classificati in base al loro modo di fare uso della funzione obiettivo. Mentre alcuni algoritmi mantengono la funzione obiettivo data dalla rappresentazione del problema come originariamente proposta, altri, come la Guided Local Search (GLS), la modificano durante la ricerca. L'idea alla base di questo approccio è quello di uscire dai minimi locali modificando la regione di ricerca. Pertanto, durante il processo di ricerca la funzione obiettivo è alterata, cercando di integrare le informazioni raccolte durante il processo di ricerca.

Strutture a vicinato singolo o multiplo. La maggior parte degli algoritmi di tipo metaeuristico lavorano su una struttura a vicinato singolo. In altre parole, la topologia della regione non cambia nel corso dell'algoritmo. Altre metaeuristiche, come la Variable Neighborhood Search, utilizzano un insieme di strutture di vicinato che dà la possibilità di diversificare la ricerca mediante lo scambio tra regioni diverse.

Uso di metodi basati sulla memoria e senza memoria. Una caratteristica molto importante per classificare le metaeuristiche è l'uso che fanno della storia di ricerca, a prescindere che vengano basate o meno sulla memoria[4]. Algoritmi non basati sulla memoria riproducono un processo di Markov, in quanto l'informazione che utilizzano per determinare l'azione successiva è data dallo stato attuale del processo di ricerca. Ci sono diversi modi per fare uso della memoria. Di solito si distingue tra l'uso della memoria a breve termine e l'uso di quella a lungo termine. Il primo tipo di solito tiene traccia dei movimenti effettuati di recente, le soluzioni visitate o, in generale, le decisioni prese. Il secondo è di solito un accumulo di parametri di sintesi sulla ricerca. L'utilizzo della memoria è oggi riconosciuta come uno degli elementi fondamentali di una metaeuristica potente.

Nel seguito si descrivono le metaeuristiche più importanti in accordo alla classificazione sulla ricerca basata sulla popolazione o sui punti singoli, che divide le metaeuristiche in metodi di traiettoria e metodi basati sulla popolazione. Questa scelta è motivata dal fatto che tale classificazione permette una definizione più chiara degli algoritmi. Inoltre, una tendenza attuale è l'ibridazione di metodi nella direzione della integrazione degli algoritmi di ricerca a punto singolo con quelli basati sulla popolazione. Le due sezioni danno una descrizione dettagliata delle meta euristiche più importanti.

1.3. Metodi di Traiettoria

In questa sezione si affrontano metaeuristiche note col nome di metodi di traiettoria. Questa espressione viene utilizzata in quanto il processo di ricerca svolto da questi metodi è caratterizzato da una traiettoria nello spazio di ricerca.

Una soluzione successiva potrebbe appartenere o non al vicinato della soluzione corrente. L'algoritmo parte da uno stato iniziale (la soluzione iniziale) e descrive una traiettoria nello spazio degli stati. La dinamica del sistema dipende dalla strategia utilizzata; algoritmi semplici generano una traiettoria composta da due parti: una fase *transitoria* seguita da un *attrattore* (un punto fissato, un ciclo o un attrattore complesso). Algoritmi dotati di strategie avanzate sono in grado di generare traiettorie più complesse le quali non possono essere suddivise nelle suddette due fasi. Le caratteristiche della traiettoria forniscono informazioni circa il comportamento dell'algoritmo e la sua efficacia rispetto alla misura considerata. Si osserva che la dinamica è il risultato della combinazione di algoritmi, la rappresentazione del problema e l'istanza del problema. In effetti, la rappresentazione del problema insieme alle strutture del vicinato definiscono l'ambito di ricerca; l'algoritmo descrive la strategia utilizzata per esplorare lo spazio ed, infine, le caratteristiche effettive dello spazio di ricerca vengono definite dall'istanza di problema da risolvere.

[4] In questa sede si fa riferimento alla memoria adattiva, in contrasto alla memoria rigida, come utilizzata, per esempio, nel Branch & Bound.

Prima di procedere con la rassegna sulle strategie di maggiore complessità, verranno descritti gli algoritmi di ricerca locale di base. Infine, verranno considerati gli algoritmi che sono strategie esplorative di tipo generale e che possono incorporare metodi di traiettoria quali loro componenti.

1.3.1 Basic Local Search: Iterative Improvement

Nella ricerca locale di base con miglioramento iterativo ogni "mossa"[5] viene eseguita solo se la soluzione risultante è migliore rispetto alla soluzione corrente. L'algoritmo si arresta appena giunge ad un minimo locale. L'algoritmo di alto livello è rappresentato nella Figura 1.1.

$s \leftarrow$ GeneraSoluzioneIniziale()
repeat
$s \leftarrow$ Migliora $(s \leftarrow N(s))$
until nessun ulteriore miglioramento è possibile

Figura 1.1 – Algoritmo di Miglioramento Iterativo

Di fatto, la funzione Migliora($N(s)$) può riferirsi ad un *primo miglioramento*, a quello *migliore* o ad una qualsiasi opzione intermedia. Nel primo caso viene esplorato il vicinato $N(s)$ e viene scelta la prima soluzione che risulta migliore di s; nel secondo caso si esplora in modo esaustivo il vicinato e viene restituita una delle soluzioni con il valore di funzione obiettivo più basso. Entrambi i metodi vengono arrestati in corrispondenza di minimi locali. Pertanto, la loro performance dipende fortemente dal definizione di S, f e N. L'adozione di procedure di miglioramento iterativo per la risoluzione di problemi di OC è di solito abbastanza insoddisfacente. Pertanto, diverse tecniche sono state sviluppate per prevenire che gli algoritmi rimangano intrappolati in minimi locali, il che è ottenuto con l'aggiunta di meccanismi che consentono loro di saltarne fuori. Questo implica anche che le condizioni di arresto degli algoritmi metaeuristici siano più complesse del semplice raggiungimento di un minimo locale. Infatti, tra le condizioni di arresto si considerano: tempo massimo di CPU, numero massimo di iterazioni, l'ottenimento di una soluzione s con valore di $f(s)$ minore di un valore di soglia predefinito, o raggiungimento del massimo numero di iterazioni senza miglioramenti.

1.3.2 Simulated Annealing

La Simulated Annealing (SA) è comunemente nota come la più antica tra le metaeuristiche e sicuramente è uno dei primi algoritmi ad aver avuto una strategia per evitare di rimanere intrappolati in minimi locali. Le origini di questo algoritmo sono rinvenibili nella meccanica statistica (algoritmo di Metropolis) ed è stato presentato originariamente come un algoritmo di ricerca per i problemi di OC in Kirkpatrick et. al. 1983. L'idea alla base dell'algoritmo consiste nell'accettare soluzioni di qualità peggiore rispetto alla soluzione corrente (i.e. si muove in salita), per uscire da un minimo locale. La probabilità di intraprendere una simile mossa decresce all'avanzare della ricerca. L'algoritmo di alto livello è descritto nella Figura 1.2.

$s \leftarrow$ GeneraSoluzioneIniziale()
$T \leftarrow T_0$

[5] Una mossa consiste nel selezionare una soluzione s' dal vicinato $N(s)$ di una soluzione s.

while not(criterio di arresto) **do**

$s' \leftarrow$ ScegliACaso $(N(s))$

if($(f(s') < f(s))$ **then**

$s \leftarrow s'$ ovvero s' sostituisce s

Else

Accetta s' quale nuova soluzione corrente con probabilità $p(T, s', s)$

Endif

Aggiorna (T)

Endwhile

Figura 1.2 – Algoritmo di Simulated Annealing (SA)

L'algoritmo inizia con la generazione di una soluzione iniziale (in modo casuale o costruita mediante euristica) e inizializzando il cosiddetto parametro di controllo o temperatura T. Quindi, ad ogni iterazione una soluzione $s' \in N(s)$ viene campionata in modo casuale, ed è accettata come nuova soluzione corrente in base ai valori di $f(s)$, $f(s')$ e T. s' si sostituisce ad s se $f(s') < f(s)$ oppure, nel caso in cui $f(s') \geq f(s)$, con probabilità calcolata in funzione di T e $f(s') - f(s)$. La probabilità è generalmente calcolata in base alla distribuzione di Boltzmann

$$exp\left(-\frac{f(s') - f(s)}{T}\right)$$

La temperatura T decresce[6] durante il processo di ricerca, quindi, all'inizio della ricerca la probabilità di accettare mosse in salita è elevata e diminuisce gradualmente, convergendo così ad un semplice algoritmo iterativo di miglioramento. Questo processo è analogo al processo di ricottura dei metalli e del vetro, i quali assumono una configurazione a bassa energia quando vengono raffreddati con un opportuno schema di raffreddamento. Per quanto riguarda il processo di ricerca, l'algoritmo è il risultato di due strategie combinate: *random walk* e miglioramento iterativo. Nella prima fase della ricerca, il bias nel miglioramento è basso e permette di esplorare lo spazio di ricerca; questa componente irregolare diminuisce lentamente portando così la ricerca a convergere ad un minimo (locale). La probabilità di accettare mosse in salita è controllata da due fattori: la differenza di valore delle funzioni obiettivo e la temperatura. Da un lato, a temperatura fissata, al crescere della differenza $f(s') - f(s)$, decresce la probabilità di accettare una mossa da s a s'. D'altra parte, a valori elevati di T, corrispondono maggiori probabilità di effettuare mosse in salita.

La scelta di un opportuno schema di raffreddamento è di cruciale importanza per la prestazione di un algoritmo. Lo schema di raffreddamento definisce il valore T a ciascuna iterazione k, $T_{k+1} = Q(T_k, k)$, dove $Q(T_k, k)$ è funzione della temperatura e del numero di iterazione. Risultati teorici sulle catene di Markov non omogenee (Aarts et. al. 1997) affermano che in particolari condizioni dello schema di raffreddamento, l'algoritmo converge in probabilità ad un minimo globale per $k \to \infty$. Più precisamente:

$$\exists \Gamma \in \Re \quad s.v. \quad \lim_{k \to \infty} p(\text{minimo globale raggiunto dopo } k \text{ iterazioni}) = 1 \quad \text{se e solo se} \quad \sum_{k=1}^{\infty} \exp\left(\frac{\Gamma}{T_k}\right) = \infty$$

Un particolare schema di raffreddamento che rispetta l'ipotesi per la convergenza è quello che segue una legge logaritmica:

$$T_{k+1} = \frac{\Gamma}{\log(k + k_0)}$$

[6] T non decresce necessariamente in modo monofonico. Schemi di raffreddamento complessi comprendono anche un incremento occasionale della temperatura.

(dove k_0 è una costante). Purtroppo, l'uso di schemi di raffreddamento che garantiscono la convergenza ad un ottimo globale non è ammissibile nelle applicazioni, perché detti schemi sono troppo lenti per essere utilizzati a fini pratici. Pertanto, nelle applicazioni vengono adottai schemi di raffreddamento più veloci. Uno di quelli più utilizzati segue una legge geometrica $T_{k+1} = \alpha T_k$ dove $\alpha \in (0,1)$, che corrisponde ad un decadimento esponenziale della temperatura.

Lo schema di raffreddamento può variare nel corso della ricerca, con l'obiettivo di bilanciare tra la diversificazione e l'intensificazione. Per esempio, all'inizio della ricerca, T potrebbe essere costante o caratterizzata da una riduzione lineare, al fine di campionare lo spazio di ricerca; quindi, T potrebbe seguire una legge quale la geometrica, per convergere ad un minimo locale alla fine della ricerca. Sono varianti di maggiore successo gli schemi di raffreddamento *non monotoni* (e.g., si veda Osman 1983, Lundy and Mees 1986). Quest'ultimi sono caratterizzati da fasi alterne di raffreddamento e di riscaldamento, fornendo così un equilibrio oscillante tra diversificazione e intensificazione.

Lo schema di raffreddamento e la temperatura iniziale dovrebbero essere adattate alla particolare istanza del problema, poiché il prezzo da sostenere per uscire da un minimo locale dipende dalla struttura dello spazio di ricerca. Un modo semplice per determinare in modo empirico la temperatura di partenza T_0 è quello di campionare lo spazio di ricerca con un modalità *random walk* per valutare approssimativamente la media e la varianza dei valori della funzione obiettivo. Comunque, anche schemi più elaborati possono essere implementati in Ingber 1996.

Il processo dinamico descritto dalla SA è una catena di Markov (Feller 1968), in quanto segue una traiettoria nello spazio degli stati in cui lo stato successivo viene scelto in funzione solo dello stato incombente. Questo significa che la SA di base è dotata di assenza di memoria. Tuttavia, l'utilizzo di memoria può essere un beneficio per approcci basati sulla SA (vedere per esempio in Chardaire et. al. 1995).

La SA è stata applicata a diversi problemi di OC, quali il problema dell'assegnamento quadratico (QAP) ed il problema del job shop scheduling (JSS). Oggi ls SA è utilizzata come elemento componente nelle metaeuristiche, piuttosto che applicata come algoritmo di ricerca *stand-alone*. Esistono varianti della SA note come algoritmi con "accettazione di soglia" e l'algoritmo del "grande diluvio".

1.3.4 Metodi di ricerca locale esplorativi

In questa sezione vengono presentati i metodi di traiettoria più recenti. Trattasi della Greedy Randomized Adaptive Search Procedure (GRASP), la Variable Neighborhood Search (VNS), la Guided Local Search (GLS) e la Iterated Local Search (ILS).

1.3.4.1 Iterated Local Search

La presente trattazione sulle strategie esplorative si conclude con la cosiddetta ricerca locale iterata (ILS), lo schema più generale tra le strategie esplorative. Da un lato, le sue generalità la rendono un framework di riferimento per altre metaeuristiche (come la VNS); d'altro canto, altre metaeuristiche possono essere facilmente incorporate come subcomponenti. La ILS è un algoritmo metaeuristico semplice, ma molto potente (Stűtzle 1999). Applica la ricerca locale ad una prima soluzione finché non trova un ottimo locale; poi perturba la soluzione e riavvia la ricerca locale. L'importanza della perturbazione è evidente: se troppo piccola potrebbe non consentire al sistema di uscire dal bacino di attrazione dell'ottimo locale appena trovato; d'altra parte, se troppo grande renderebbe l'algoritmo simile ad un riavvio di ricerca locale di tipo casuale.

Una ricerca locale è efficace se è in grado di trovare minimi locali buoni, cioè, se si può trovare il bacino di attrazione di questi stati. Quando lo spazio di ricerca è ampio e/o quando i bacini di attrazione di ottimi locali buoni sono di

piccole dimensioni[7], un semplice algoritmo multi-start è quasi inutile. Un processo di ricerca efficace potrebbe essere progettato come una traiettoria a partire dal solo insieme di ottimi locali $\hat{S}$, invece dell'insieme S di tutti gli stati. Purtroppo, nella maggior parte dei casi non è possibile introdurre una struttura di vicinato per $\hat{S}$. Di conseguenza, viene eseguita una traiettoria lungo gli ottimi locali $\hat{s}_1, \hat{s}_2, ..., \hat{s}_t$, senza l'introduzione di una struttura di vicinato, applicando il seguente schema:

1. esegui la ricerca locale (LS) da uno stato iniziale s fino a quando non viene trovato un minimo locale $\hat{s}$;
2. perturba $\hat{s}$ ed ottieni s';
3. esegui la LS a partire da s' fino a quando non viene raggiunto un minimo locale $\hat{s}'$;
4. sulla base di un criterio di accettazione decidere se impostare $\hat{s} \leftarrow \hat{s}'$;
5. vai al passo 2.

$s_0 \leftarrow$ GeneraSoluzioneIniziale()
$\hat{s} \leftarrow$ RicercaLocale(s_0)
while not condizione di arresto **do**
    $s' \leftarrow$ Pertubazione$(\hat{s}, storia)$
    $\hat{s}' \leftarrow$ RicercaLocale(s')
    $\hat{s} \leftarrow$ ApplicaCriterioAccettazione$(\hat{s}, \hat{s}', storia)$
Endwhile

Figura 1.3 – Algoritmo di Iterated Local Search (ILS)

Il requisito per la perturbazione di $\hat{s}$ è quello di produrre un punto di partenza per la ricerca locale in modo tale che un minimo locale diverso da $\hat{s}$ venga raggiunto. Tuttavia, questo nuovo minimo locale dovrebbe essere più vicino a $\hat{s}$ rispetto ad un minimo locale prodotto da un riavvio casuale. Il criterio di accettazione funge da contrappeso, dal momento che filtra e fornisce feedback per l'azione delle perturbazioni, in base alle caratteristiche del nuovo minimo locale. Una descrizione di alto livello della ILS per come riportata in Lourenço et. al. 2002, è data in Figura 1.3.
La progettazione di algoritmi ILS ha diversi gradi di libertà nella scelta della soluzione iniziale, della perturbazione e dei criteri di accettazione. Un ruolo fondamentale è svolto dalla storia della ricerca che può essere sfruttata in forma di memoria a breve e a lungo termine.
La costruzione di soluzioni iniziali dovrebbe essere veloce (e computazionalmente poco costoso), e le soluzioni iniziali dovrebbero essere un buon punto di partenza per la ricerca locale. Il modo più rapido per produrre una prima soluzione è quella di generarla a caso, tuttavia, questo risulta essere il modo più semplice per problemi che non sono vincolati, mentre in altri casi, la costruzione di una soluzione ammissibile richiede anche il controllo dei vincoli. Possono anche essere adottati anche i metodi costruttivi guidati da euristiche. Vale la pena sottolineare che una prima soluzione è considerata un buon punto di partenza a seconda della particolare LS applicata e della struttura del problema, quindi l'obiettivo del progettista dell'algoritmo è quello di trovare un compromesso tra velocità e qualità delle soluzioni.
La perturbazione di solito è non-deterministica, per evitare il ciclaggio. La sua caratteristica più importante è la *forza*, più o meno definita come il numero di modifiche effettuate sulla soluzione corrente. La forza può essere fissa o variabile. Nel primo caso, la distanza tra $\hat{s}$ e s' viene mantenuta costante, indipendentemente dalla dimensione del problema. Tuttavia, una forza variabile è in generale più efficace, poiché sperimentalmente è stato accertato che, nella maggior parte dei problemi, al crescere della dimensione del problema, cresce la forza. Sono possibili altri sistemi sofisticati sono possibili, ad esempio, la forza può essere adattiva: aumenta quando è necessaria maggiore

[7] La dimensione del bacino di attrazione di un punto s (in uno spazio finito) è definita come la frazione di stati iniziali delle traiettorie che convergono al punto s.

diversificazione e diminuisce quando l'intensificazione sembra preferibile. La VNS e le sue varianti appartengono a questa categoria. Una seconda scelta è il meccanismo per produrre le perturbazioni. Questo può essere un meccanismo casuale, o la perturbazione può essere prodotto da un metodo (seme) deterministico (ad esempio, una LS diversa da quella utilizzata nell'algoritmo principale).
La terza componente principale è il criterio di accettazione. Due esempi estremi consistono in: (1) accettare il nuovo ottimo locale solo in caso di miglioramento e (2) accettare sempre la nuova soluzione. In mezzo, ci sono diverse possibilità. Ad esempio, è possibile adottare una sorta di schema di annealing: accettare tutte i nuovi ottimi locali migliorativi e accettare anche quelli non-migliorativi, in accordo ad una probabilità che è funzione della temperatura T e della differenza dei valori della funzione obiettivo. In formule:

$$\mathbf{p}\left(Accetta\left(\hat{s},\hat{s}',storia\right)\right)=\begin{cases}1 & \text{se } f\left(\hat{s}'\right)<f\left(\hat{s}\right)\\ \exp\left(-\dfrac{f\left(\hat{s}'\right)<f\left(\hat{s}\right)}{T}\right) & \text{altrimenti}\end{cases}$$

Lo schema di raffreddamento può essere monotono (non crescente nel tempo) o non monotono (adattato per ottimizzare l'equilibrio tra diversificazione ed intensificazione). Lo schema non monotono è particolarmente efficace se si sfrutta la storia della ricerca, in modo simile alla Tabu Search Reattiva Taillard 1991 riportata alla fine della sezione dedicata alla TS. Quando l'intensificazione non appare più efficace, è necessaria una fase di diversificazione e viene incrementata la temperatura.
Esempi di applicazioni di successo della ILS riguardano il TSP, il QAP ed il problema della tardiness pesata totale con macchina singola (SMTWT). Riferimenti ad altre applicazioni possono essere trovate in Lourenço et. al. 2002.

1.4 Il Metodo delle Partizioni Innestate

I problemi di ottimizzazione discreta di grandi dimensioni riguardano numerose applicazioni reali. Essi, però, sono difficili da risolvere specialmente quando sono caratterizzati da particolari vincoli che ne complicano ulteriormente la struttura.
In letteratura, per risolvere problemi di ottimizzazione discreta, sono descritti, in genere, due tipi di approcci: quello esatto e quello approssimato. Come è noto, il primo ricava la soluzione ottima del problema tramite l'individuazione di una sequenza di soluzioni ammissibili. Tra i più comuni approcci esatti si annovera il *branch and bound*, il *branch and cut* e la *column generation*. Il secondo tipo di approccio, tra cui si annovera la *local search*, la *nested partitions* e la *programmazione dinamica approssimata*, cerca di generare buone soluzioni (appunto approssimate) in maniera più efficiente possibile (anche in funzione dei tempi computazionali complessivamente richiesti). Infatti, mentre l'approccio esatto, soprattutto su problemi di grandi dimensioni, risulta impraticabile a causa degli elevati tempi computazionali richiesti, quello approssimato, invece, risulta essere più efficiente, sebbene alcuni algoritmi risentano molto della tipologia di vincoli trattati. In genere, maggiore è il numero di vincoli del problema oppure complessa è la loro struttura, elevati potrebbero essere anche i tempi computazionali richiesti dall'approccio approssimato. Ma questi ultimi, comunque, rimangono più bassi rispetto a quelli richiesti dall'approccio esatto, a parità di condizioni.
Tra i possibili metodi approssimati, sopra citati, sta assumendo sempre più interesse quello noto come *Nested Partitions* (NP), Shi and Olafsson 2000. In particolare, tale metodologia può essere applicata a problemi che presentano una regione ammissibile finita, di tipo combinatorio. Ma, con opportune estensioni, essa può essere applicata anche ai problemi con regione ammissibile numerabile ma infinita o limitata ma non numerabile. Questo evidenzia la sua struttura completamente generale e soprattutto giustifica l'interesse dei ricercatori nel definire differenti versioni, adattabili a diversi contesti operativi.
La versione più generale di NP si caratterizza dei seguenti 4 passi principali:

1. *Partizionamento*: si partiziona la regione che correntemente è la più promettente in un certo numero fissato di sotto-regioni e si aggregano le rimanenti in un'unica regione detta "regione circostante"(anche *surrounding region*).
2. *Campionamento casuale*: si campionano sia le sotto-regioni che quella circostante, individuate al passo 1, tramite una specifica procedura. In genere, tale procedura viene scelta in maniera da garantire che tutti i campioni, in una data regione, abbiano la stessa probabilità di selezione (campioni equiprobabili). Un classico metodo usato è noto come quello del campionamento pesato.
3. *Calcolo della regione più promettente*: si deriva, per ciascuna regione, un cosiddetto *indice di performance*, utilizzato per determinare quale è la più promettente.
4. *Backtracking*: tale passo non è necessariamente applicato ad una generica iterazione del metodo. Infatti, esso è solo richiamato quando la regione più promettente, scelta all'iterazione successiva, è la circostante. Si ricorda, a tal proposito, che, ad ogni iterazione del metodo, la nuova regione più promettente potrebbe essere o figlia della regione correntemente più promettente o quella circostante. Il *backtracking* consente, quindi, alla NP di ritornare o al nodo radice (regione di partenza) oppure a qualsiasi altro nodo (regione) lungo il percorso che ha condotto verso la regione correntemente più promettente.

1.4.1 Il metodo per l'ottimizzazione combinatoriale

In tale sezione del documento, si descrive l'approccio NP applicato al contesto dei problemi di ottimizzazione combinatoriale. A tal proposito, si consideri il seguente problema:

$$\vartheta^* \in \arg\min_{\vartheta \in \Theta} f(\vartheta)$$

dove Θ è l'insieme finito delle soluzioni e $f : \Theta \to \Re$ è la funzione obbiettivo da ottimizzare (nel caso specifico da minimizzare).
Un possibile approccio risolutivo di un tale problema consiste nell'enumerazione totale (cioè nella generazione di tutte le possibili soluzioni) e nella selezione della soluzione migliore. Pur essendo sicuramente un metodo molto semplice da implementare, l'elevato numero di possibili soluzioni da generare e valutare lo rende impraticabile, soprattutto per i problemi in contesti applicativi reali.
Alcuni problemi di ottimizzazione si caratterizzano di regioni ammissibili che possono essere esplorate per trovare la soluzione, senza dover controllare tutte quelle possibili. Molti altri problemi, però, soprattutto quelli estrapolati da contesti reali, non godono di tale proprietà e per essi l'unico metodo che garantisce di selezionare l'ottimo globale è quello di valutare tutte le soluzioni possibili. Un esempio di questi problemi sono tutti quelli che afferiscono alla classe dei problemi NP completi. Come già evidenziato nella sezione precedente, anche per questa classe di problemi di ottimizzazione, la NP, ad ogni iterazione, assume la conoscenza della regione più promettente, che coincide con un sottoinsieme di Θ. Essa è partizionata in M sottoregioni; le restanti soluzioni compresi in Θ, che non appartengono alla regione più promettente, vengono aggregati in un'unica regione che prende il nome di *regione circostante*. Quindi ad ogni iterazione dell'algoritmo si avranno M+1 sottoregioni distinte e disgiunte tra loro, che creano una partizione di Θ. Ognuna di queste M+1 regioni deve essere campionata, per esempio tramite un *random sampling* o tecniche maggiormente evolute, mentre il valore della funzione obbiettivo (o indice di performance) di ogni singolo campione (punto all'interno della singola sottoregione) è usato per determinare *l'indice della regione più promettente*. Tale indice determina quale sottoregione diventerà la più promettente all'iterazione successiva. Se dovesse accadere che la regione più promettente sia quella *circostante*, l'algoritmo fa *backtracking* alla regione più grande che contiene la vecchia regione più promettente. La nuova regione più promettente all'iterazione successiva sarà partizionata e trattata nel modo descritto in precedenza.

Nella prima iterazione dell'algoritmo si suppone che la regione più promettente sia l'intera regione ammissibile e non si ha alcuna regione circostante. Ciò si ripeterà ogni volta che si considererà l'intera regione ammissibile come quella più promettente.
È chiaro che essendo Θ finita ci saranno delle regioni *singleton*, cioè regioni che contengono solo un punto al loro interno. Esse saranno denominate *regioni massima profondità*. La profondità di una regione è definita iterativamente, partendo dal fatto che Θ ha la profondità di lunghezza 0.
Si assuma, inoltre, di avere a disposizione uno schema di partizionamento per la regione ammissibile e di aver fissato il valore di M. Ogni regione così costruita con tale schema di partizionamento sarà denominata *regione valida*. Se una *regione valida* σ è stata ottenuta partizionando la regione η, allora σ è una *sottoregione* di η mentre η è la *superregione* di σ. Non è, quindi, necessario conoscere tutte le regioni valide. Ciò che è invece indispensabile è la conoscenza delle regole usate per ottenere le regioni valide, nota una regione più promettente. Il partizionamento della regione ammissibile in un insieme di M sottoregioni influenza l'efficienza dell'algoritmo.
Se la struttura imposta alla partizione è in grado di *clustrerizzare* le soluzioni migliori, la NP concentrerà la sua ricerca in tale regione e convergerà, quindi, velocemente. Per una regione ammissibile finita, una buona partizione esiste sempre e si può ottenere enumerando tutte le possibili soluzioni, che potrebbero non essere note a priori. Alla luce di quanto affermato in questa sezione, la strategia di partizionamento è la componente principale della NP, al fine di ricercare l'ottimo globale.

1.4.2 Schema algoritmico della NP

Prima di fornire i dettagli implementativi dell'algoritmo, in questa sezione, vengono presentate alcune notazioni introduttive:

1. Θ = regione ammissibile;
2. Σ ={ σ ($\sigma \subseteq \Theta$) regione valida fissata in una partizione};
3. Σ_0 ={ $\sigma \in \Sigma$ σ ha lunghezza massima} ;
4. $\sigma(k)$ = regione più promettente all'iterazione k-esima;
5. $d(\sigma)$ = lunghezza della regione $\sigma \in \Sigma$;
6. $s(\sigma)$ = super-regione di $\sigma \in \Sigma$

In particolare, la funzione $s : \Sigma \to \Sigma$ tiene traccia della super-regione di ogni regione più promettente e si può definire formalmente nel seguente modo: sia $\sigma \in \Sigma \setminus \Theta$ e $s(\sigma) = \eta \in \Sigma$ se e solo se $\sigma \subset \eta$ e se $\sigma \subseteq \xi \subseteq \eta$ allora $\xi = \sigma$ o $\xi = \eta$. Inoltre, per completezza, si definisce $s(\Theta) = \Theta$. Tale funzione è necessaria per il *backtracking*. Si deve tener traccia,infatti, di questi valori solo per un insieme limitato di regioni valide. Si noti che in ogni iterazione la sequenza $\{\sigma(k)\}_{k=0}^{\infty}$ è una stima della migliore regione valida. La regione $\sigma(k+1)$ dipende dalle stime degli indici di performance alla k-esima iterazione. Tali indici a loro volta dipendono dai campioni selezionati. Quindi $\{\sigma(k)\}_{k=0}^{\infty}$ è un processo stocastico

Algoritmo NP

Step 1. ***Partizionamento***: sia $M_{\sigma(k)}$ il numero di sottoregioni $\sigma(k)$ della corrente regione più promettente. Tale numero dipende solo dalla regione più promettente corrente ma non dall' iterazione. Si partizioni $\sigma(k)$ in $M_{\sigma(k)}$ sottoregioni $\sigma_1(k),\ldots,\sigma_{M_{\sigma(k)}}(k)$ e si aggreghi le sottoregioni $\Theta \setminus \sigma(k)\ \sigma_{M_{\sigma(k)}+1}(k)$ in una unica regione

Step 2. ***Campionamento Casuale***: sia N_j il numero di campioni della regione $\sigma_j(k)$. Il numero di punti da campionare può dipendere sia dalla regione corrente più promettente che dall'iterazione. Si usi il campionamento casuale per selezionare N_j punti da ogni sottoregione $\sigma_j(k)$ con $j=1,2,\ldots,M_{\sigma(k)}+1$:

$$\theta^{j1},\theta^{j2},\ldots,\theta^{jN_j},\ j=1,2,\ldots,M_{\sigma(k)}+1$$

e si calcoli il corrispondente indice di performance:

$$f(\theta^{j1}), f(\theta^{j2}),\ldots, f(\theta^{jN_j}),\ j=1,2,\ldots,M_{\sigma(k)}+1$$

L'unico requisito richiesto al campionamento casuale è che, ogni regione, i campioni siano equiprobabili e con probabilità di selezione positiva.

Step 3. ***Stima dell' indice di performance più promettente***: sia data una funzione per il calcolo dell'indice di performance, $I:\Sigma\to\Re$. Si stimi l'indice di performance di ogni regione utilizzando tale funzione. Si assuma che la scelta della migliore regione sia fatta nel seguente modo:

$$I(\sigma)=\min_{\theta\in\sigma} f(\theta),\quad \sigma\in\Sigma$$

Ciò implica che, per ogni regione $\sigma_j(k)$ con $j=1,2,\ldots,M_{\sigma(k)}+1$ si stima l'indice $I(\sigma_j)$ usando i campioni ottenuti nel passo precedente:

$$I(\sigma_j)=\min_{j\in\{1,2,\ldots,N_j\}} f(\theta^{ji})\quad j=1,2,\ldots,M_{\sigma(k)}+1$$

dove $\hat{I}(\sigma_j)$ è una variabile casuale.

Step 4. ***Backtracking***: si determini la regione successiva più promettente $\sigma_{\hat{j}_k}$, dove

$$\hat{j}_k\in\arg\min_{j\in\{1,\ldots,M_{\sigma_k(k)}+1\}}\hat{I}(\sigma_j)$$

Se due o più regioni hanno lo stesso valore, se ne sceglie una arbitrariamente. Se tale indice corrisponde ad una sottoregione di $\sigma(k)$, allora essa sarà la prossima regione promettente. Se, invece, tale indice corrisponde alla regione circostante, si effettua *backtracking* verso la superregione della precedente regione più promettente. Ciò implica la necessità di tenere traccia della sequenza di regioni generate prima della precedente regione più promettente. In alternativa il *backtracking* può essere fatto direttamente alla regione ammissibile iniziale. In questo modo la struttura che si tiene in memoria è più snella poiché non si deve tenere traccia del percorso che ha condotto a tale sottoregione più promettente.

Il primo modo di operare il *backtracking* non continua a cercare nelle prossimità della zona che è fallita. Ciò significa che, nel caso di minimo locale, l'algoritmo non ne uscirebbe immediatamente ma rimarrebbe a cercare ancora nelle prossimità di tale regione.
Invece, la seconda tecnica effettua un cambiamento drastico poiché, tornando direttamente ad esplorare l'intera regione ammissibile, ricomincia una nuova ricerca e quindi più facilmente può uscire dalle situazioni di minimo locale.
Una terza tecnica, alternativa alle suddette, è quella in cui si effettua un *backtracking* verso la più piccola regione, già generata, che contiene la soluzione ottima corrente. Tale tecnica viene chiamata *breadth-first* (o *Best Bound* (BB)). Essa presenta molti vantaggi:

1. non richiede una soluzione iniziale poiché possiede una visione globale dell'intera situazione;

2. combina gli elementi di una ricerca globale (partizione) con gli elementi euristici di ricerca locale in un modo del tutto naturale.

3. si presta molto ad una possibile implementazione parallela. Infatti, la valutazione di ogni sottoregione, aspetto, in genere, più *time consuming* dell'intera metodologia, è indipendente dalle altre ed è quindi possibile parallelizzarla. Alla fine della computazione, si collezionano i vari indici di performance ottenuti per le diverse sotto regioni per selezionare la regione più promettente.

In generale la NP favorisce le regioni che posseggono molte buone soluzioni e nessuna cattiva soluzione rispetto a regioni in cui esiste un'unica buona soluzione e tutte le altre sono cattive. Ciò implica che esiste la possibilità, seppur remota, che l'algoritmo potrebbe non raggiungere la soluzione ottima in un tempo finito, perché questa è circondata da molte cattive soluzioni. Questo significa anche che molte cattive soluzioni cadono nella stessa regione valida della soluzione ottima e quindi anche che la lunghezza di tali soluzioni è vicina a quella massima, a cui corrisponde la soluzione ottima.
In genere, quando la regione ammissibile del problema è discreta ed illimitata:

1. è selezionato un grande valore di K e la regione è partizionata in due insiemi S={1,2,…K} e T={K+1,….}. Il campionamento in T dovrebbe essere fatto tramite un' opportuna funzione di densità di probabilità (ad esempio Poisson).

2. Se la regione S è la più promettente, NP si comporta come già descritto.

3. Se la regione più promettente è T allora essa viene partizionata in due insiemi {K+1,….2K} e {2K+1,…} e l'algoritmo segue la classica procedura.

1.4.3. Il criterio di arresto dell'algoritmo NP

Uno degli aspetti più critici di tale approccio euristico è il criterio di arresto. Da esso, infatti, dipende fortemente la qualità della soluzione ottenuta in output. Di seguito, vengono proposti due criteri d'arresto. Si supponga che il minimo globale sia raggiunto in un unico punto e ad esso corrisponde il seguente valore dell'indice di performance:

$$f^* \equiv \min_{\theta \in \Theta} f(\theta).$$

Prima di arrestare l'algoritmo, è necessario controllare, come prima condizione, che lo stato corrente, che contiene la migliore soluzione ammissibile finora determinata, sia a massima profondità (cioè, regione di singleton). A tale scopo, si deve tenere traccia della migliore soluzione $\hat{\theta}(k)$,trovata all'iterazione k-esima. La condizione, quindi, necessaria per arrestare l'algoritmo in uno stato $\eta \in \Sigma$ alla k-esima iterazione è:

$$\eta = \left\{\hat{\theta}(k)\right\} \in \Sigma_0 \quad (1.1)$$

Un miglioramento in termini di tempi computazionali richiesti dal metodo, consiste nell'utilizzo di una tecnica di campionamento che focalizza la ricerca all'interno della regione più promettente.

Esiste, comunque, uno schema approssimato di campionamento. In molte iterazioni, infatti, la regione circostante $\Theta \setminus \sigma(k)$ contiene al più tutte le soluzioni ammissibili. Quindi, invece di campionare direttamente la regione circostante, si campiona tutta la regione ammissibile Θ e si usano i soli punti che cadono nella regione $\Theta \setminus \sigma(k)$, cioè si utilizza uno schema di campionamento di tipo accettazione/rigetto. Inoltre i punti che non cadono nella regione circostante devono per forza cadere all'interno della regione $\sigma(k)$ e quindi possono essere utilizzati in tale regione. Questa strategia fornisce uno schema costante di campionamento, indipendente dall'iterazione. Per semplicità si può assumere che tale schema di campionamento sia uniforme in tutta la regione ammissibile. Ulteriori schemi di campionamento, anche più generali e alternativi, possono essere utilizzati all'interno delle regole d'arresto e per essi si indica con $Q(k)$ il numero totale di campioni all'iterazione k-esima.

1.4.4.1 Ordinal Stopping Rule

La prima regola d'arresto si basa sulla *Ordinal Optimization*. Essa non si basa su un confronto cardinale delle diverse soluzioni ma compara o il valore ordinale della soluzione o il suo rango. Quindi il metodo si arresta quando raggiungiamo una delle *l* migliori soluzioni $\left(l < |\Theta|\right)$. Queste *l* soluzioni possono essere considerate come le migliori soluzioni della regione ammissibile. In altre parole, non si è interessati alle diverse performance che si trovano ad una determinata distanza dalla migliore, bensì solo a trovare le soluzioni che appartengono ad un certo percentile rispetto a tutte le altre. Si assuma, quindi, un campionamento uniforme.

La probabilità di selezionare una delle *l* migliori soluzioni è $\alpha = l/|\Theta|$, che corrisponde alla probabilità di successo. Si noti che $(1-\alpha)$ è il percentile desiderato e la distribuzione del numero di volte che una delle soluzioni buone in $Q(k)$ tentativi è selezionata (cioè il numero di successi) segue quella binomiale con parametri $(\alpha, Q(k))$. La probabilità di selezionare almeno una buona soluzione in $Q(k)$ prove è data da 1- la probabilità di non selezionarne mai una. La probabilità di includere una buona soluzione in un *batch* di $Q(k)$ campioni casuali selezionati è data dalla seguente relazione:

$$p \equiv 1 - \binom{Q(k)}{0} \alpha^0 (1-\alpha)^{Q(k)} = 1 - (1-\alpha)^{Q(k)} \quad (1.2)$$

Perché spesso si tiene traccia della migliore soluzione e si campiona la regione più promettente non con campionamento uniforme ma in modo più intensivo, la (1.2) è un *lower bound* per la probabilità di avere selezionata una soluzione buona con il percentile $(1-\alpha)$ all'iterazione k-esima.

Si assuma, di seguito, di voler trovare una soluzione con un percentile $(1-\alpha)$ con almeno la probabilità p. L'algoritmo terminerà allo stato η, se η soddisfa l'equazione (1.1) e

$$Q(k) \geq \frac{\log(1-p)}{\log(1-\alpha)}. \quad (1.3)$$

Si noti che poiché tale regola non ingloba informazioni ottenute durante l'esecuzione dell'algoritmo, essa può essere utilizzata per pianificare lo sforzo computazionale in anticipo.

1.4.4.2 Cardinal Stopping Rule

E', in genere, possibile usare la costante approssimata per lo schema di campionamento uniforme al fine di costruire un intervallo di confidenza (one-side) e calcolare il valore ottimo dell'indice di performance nella regione ammissibile. In tale sezione si assume che $\Theta \subseteq \Re^n$. A tale scopo, inoltre, si può introdurre il concetto delle *statistiche di ordinamento*. Dati *N* punti ed i corrispondenti indici di performance

$$f(\theta_1), f(\theta_2),..., f(\theta_n)$$

il relativo rango determina l'ordine seguente $f_{[1]} \leq f_{[2]} ... \leq f_{[n]}$.

L'indice i-esimo di performance identifica l'i-esimo miglior valore $f_{[i]}$. Chiaramente, l'indice con rango 1 rappresenta l'ottimo globale trovato finora. Assumendo che l'equazione (1.1) sia soddisfatta ed utilizzando un risultato sugli intervalli di confidenza nelle statistiche di ordinamento, è possibile rilassare la regione ammissibile approssimandola ad un sottoinsieme di $\Re^n$. Tale regione rilassata è indicata con $\tilde{\Theta} \subseteq \Re^n$ e tale rilassamento lo si può ottenere, per esempio, imponendo che $\tilde{\Theta}$ sia una regione convessa includente la regione ammissibile Θ. Si definisce anche un indice di performance su tale regione $\tilde{f} : \tilde{\Theta} \to \Re$ tale che $\tilde{f}(\theta) = f(\theta)$, per tutti $\theta \in \Theta$ e sia $\tilde{f}$ misurabile. Inoltre è anche richiesto che $\tilde{f}$ e *f* posseggano lo stesso minimo. Nella regione $\tilde{\Theta} \setminus \Theta$ la funzione rilassata $\tilde{f}$ può avere qualsiasi valore purché sia misurabile e maggiore del minimo globale della funzione originale *f*. Se tale rilassamento esiste, è sempre possibile costruire tale funzione poiché l'unico vincolo imposto è che essa sia misurabile. Si noti che la regione ammissibile rilassata è sempre limitata e che $f^* \equiv \min_{\theta \in \Theta} f(\theta)$ e coincide con l'unico valore ottimo. Inoltre, $f^* = \tilde{f}^* \equiv \min_{\theta \in \tilde{\Theta}} \tilde{f}(\theta)$ perché le tue funzioni hanno lo stesso minimo. Sia, quindi, X una variabile casuale uniformemente distribuita e definita su $\tilde{\Theta}$, sia:

$$F(y) = P\left[\tilde{f}(X) \leq y\right].$$

Se $\tilde{f}$ è misurabile e $\tilde{\Theta}$ è limitata ed esiste una costante $v > 0$ tale che:

$$\lim_{\delta \to 0} \frac{F\left(\tilde{f}^* + c\delta\right)}{F\left(\tilde{f}^* + \delta\right)} = c^v, \ \forall c > 0$$

Allora in De Haan (1981) è stato dimostrato che con la probabilità $1 - p$ l'intervallo di confidenza per la funzione $\tilde{f}^*$ è il seguente:

$$\left[f_{[1]} - \frac{\left(f_{[2]} - f_{[1]}\right)}{(1-p)^{(-1/v)} - 1}, f_{[1]} \right]. \quad (1.4)$$

Le precedenti condizioni, imposte sulla $\tilde{f}$ (sempre misurabile) e su $\tilde{\Theta}$ (limitata), devono essere sempre soddisfatte. Poiché la funzione rilassata deve possedere sempre lo stesso minimo della funzione originale, questa equazione fornisce una base per una regola d'arresto.

Tuttavia per applicarla il valore di *v* deve essere noto, cosa che non accade sempre. Quindi, sono necessari alcuni metodi per stimarlo. Si potrebbe usare, per esempio, il metodo del *maximum likelihood* (Zhigljavsky 1991), o il metodo proposto da De Haan (1981). Sia $\{n_k\}_{k=1}^{\infty}$ una sequenza di numeri tale che $\lim_{k \to \infty} n_k = \infty$ e $\lim_{k \to \infty} n_k / k = 0$, si

può dimostrare che asintoticamente si può avere un intervallo di confidenza per f^*, quando *v* incognito sostituendo nella (1.4) al posto di $1/v$ la seguente equazione

$$\log\left(\frac{f_{[Q(k)]}-f_{[3]}}{f_{[3]}-f_{[2]}}\right)\Big/\log n_{Q(k)}\,.$$

Si assuma di voler trovare, con probabilità *p*, una soluzione il cui indice di performance cada all'interno di una distanza *d* da quello ottimo. L'algoritmo si arresta allo stato η, se η soddisfa l'equazione (1.1) e

$$\frac{\left(f_{[2]}-f_{[1]}\right)}{(1-p)^{-x}-1}\le d$$

Dove

$$x=\frac{1}{\log n_{Q(k)}}\left(\frac{f_{[Q(k)]}-f_{[3]}}{f_{[3]}-f_{[2]}}\right)$$

e sono soddisfatte i seguenti limiti:

$$\lim_{i\to\infty} n_i=\infty \text{ e } \lim_{i\to\infty} n_i/i=0\,.$$

Poiché per applicare questa regola di arresto, è necessario calcolare solo 4 quantità ($f_{[1]}$, $f_{[2]}$, $f_{[3]}$ e $f_{[Q(k)]}$), essa non richiede eccessivi overhead computazionali.

1.4.5. Tecnica per il calcolo dell'indice di performance

Come è stato già notato nella sezione 4, l'indice di perfomance consente di stabilire e selezionate, ad ogni iterazione del metodo NP, la migliore soluzione e quindi stabilisce la nuova regione corrente e la regione circostante. Diverse possono essere le tecniche da applicare per tale calcolo ma in questo documento si presenta la cosiddetta tecnica OCBA, come descritto nella sezione successiva.

1.4.5.1 La tecnica OCBA

Il passo dell'algoritmo, in cui è previsto il calcolo dell'indice più promettente, potrebbero essere effettuate delle simulazioni. Ciò implica che esso sia il passo più oneroso dal punto di vista computazionale e rappresenta quindi un collo di bottiglia per la procedura NP.
Per ovviare a tale problema, una possibile tecnica da utilizzare è quella nota come *Optimal Computation Budget Allocation* OCBA (Chen et al. (1996, 1999)). Più in dettaglio, si suppone di aver selezionato una soluzione θ_a:

$$\theta_a\equiv\arg\min_{\theta}\hat{J}(\theta)\left(\equiv\frac{1}{t}\sum_{i=1}^{t}L(\theta,\xi_i)\right)$$

Si definisce la probabilità di selezione corretta (*probability of correct selection)*:

$$P(CS) \equiv P\{\text{la soluzione corrente top - ranking } \theta_b \text{ è attualmente la migliore soluzione scelta}\}.$$

Sia t_θ il tempo di simulazione della soluzione θ, se le simulazioni sono realizzati su una macchina sequenziale e la differenza del costo di computazione delle simulazioni delle differenti soluzioni è trascurabile, il tempo computazionale totale approssimato è $\Sigma_{\theta\in\Theta} t_\theta$.

L'obbiettivo è scegliere t_θ per tutte le θ in modo che il costo computazionale totale sia minimizzato. Soggetto a queste restrizioni, il livello di confidenza definito da $P(CS)$ è maggiore del livello richiesto:

$$\min_{t_\theta} \sum_{\theta\in\Theta} t_\theta$$
$$s.t. \quad P(CS) \geq P^*.$$

Dove P^* è il livello di confidenza richiesto dall'utente che corrisponde al criterio di arresto in ogni iterazione si NP. Chen et al. (1999) approssima la $P(CS)$ usando il *bound* di Chernoff (Ross 1994) e il modello Bayesiano Chen et al. (1996) ed offre una soluzione asintotica, come riassunto nel seguente teorema. Dato il numero totale del budget di simulazione T e allocato un numero finito di soluzioni alternative, la $P(CS)$ può essere asintoticamente massimizzata quando:

(a) $$\frac{t_a}{t_b} \to \frac{s_a}{s_b}\left[\sum_{\substack{i=1 \\ 1\neq a}}^{k}\left(\frac{\delta_{a,b}^2}{\delta_{a,i}^2}\right)\right]^{1/2}$$

(b) $\frac{t_i}{t_b} \to \left(\frac{\sigma_i / \delta_{a,i}^2}{\sigma_b / \delta_{a,b}^2}\right)^2$ per $\theta\in\Theta$ e $\theta_i \neq a \neq b$,

dove *a* è la soluzione con la media più grande, *b* è la soluzione con la seconda media più elevata, e

$$\delta_{i,j} = \frac{1}{t_i}\sum_{u=1}^{t_i} L(i,\xi_u) - \frac{1}{t_j}\sum_{u=1}^{t_j} L(j,\xi_u), \text{ per ogni } i, j \in \Theta.$$

1.4.6. Convergenza del metodo NP

1.4.6.1 Teorema di convergenza globale di NP

L'algoritmo NP con un valido schema di partizionamento converge quasi sicuramente ad un minimo globale in tempo finito.

Proposizione 1: il processo stocastico $\{\sigma(k)\}_{k=1}^{\infty}$ è una catena di markov con $\sum$ spazio degli stati.

Proposizione 2: Uno stato $\eta \in \sum$ è uno stato assorbente per la catena di markov $\{\sigma(k)\}_{k=1}^{\infty}$ se e solo se

$\eta \in \sum_0$ e $\eta = \theta^*$ dove θ^* è un minimo globale per il problema originario.

DIMOSTRAZIONE

→Se

$$Assume: \begin{cases} \eta = \sigma(k) \in \sum_0 \\ \eta = \{\theta^*\} \in \arg\min_{\theta \in \phi} f(\theta) \end{cases} \Rightarrow P_{\eta,\eta} = P[PI(\eta) \le PI(\phi \setminus \eta)] = P[f(\theta^*) \le PI(\phi \setminus \eta)] =$$

→solo se

$$Assume: \begin{cases} \eta = \sigma(k) \in \sum_0 \\ \eta = \{\theta\} \notin \arg\min_{\theta \in \phi} f(\theta) \end{cases} \Rightarrow \exists \{\theta^*\} = \{\theta^* | \theta^* \in \phi \setminus \eta, f(\theta^*) < f(\theta)\}$$

$$\Rightarrow P_{\eta,\phi\setminus\eta} = P[PI(\phi \setminus \eta) < PI(\eta)] = P[PI(\phi \setminus \eta) < f(\theta)] =$$

$$= P[the\_event\_that\_\{\theta^*\}\_will\_be\_selected\_at\_random\_from\_\phi \setminus \eta] > 0$$

1.4.6.2 Bounds sul numero atteso di iterazioni prima della convergenza

Definizioni:

Y_η denota il numero atteso di iterazioni spese nello stato $\eta \in \sum$. Si divide lo spazio degli stati in tre sotto-insiemi disgiunti

$\{\sigma_{opt}\}$ il gruppo delle regioni a singolo punto , ciascuno contenente una soluzione ottima globale;

$$\sum_1 = \{\eta \in \sum \setminus \{\sigma_{opt}\}, \exists \alpha \in \{\sigma_{opt}\} : \alpha \subseteq \eta\}$$

$$\sum_2 = \{\eta \in \sum \setminus \{\sigma_{opt}\}, \forall \alpha \in \{\sigma_{opt}\} : \alpha \not\subset \eta\}$$

$$T_\eta = \min(k > 0 | \sigma_k = \eta)$$

E' possibile, quindi, enunciare il seguente teorema:

Il numero atteso di iterazioni fino a che la catena di Markov raggiunga uno stato assorbente è dato da

$$E[Y] = 1 + \sum_{\eta \in \sum_1} \frac{1}{P_\eta[T_{\sigma_{opt}} < T_\eta]} + \sum_{\eta \in \sum_2} \frac{P_\phi[T_\eta < \min(T_\phi, T_{\sigma_{opt}})]}{P_\eta[T_\phi < T_\eta] * P_\phi[T_{\sigma_{opt}} < \min(T_\phi, T_\eta)]}$$

Al fine di rendere più chiara possibile la trattazione di questa metodologia, di seguito, si propone uno schema algoritmo costituito di 5 *step* principali:

1. Si partiziona la regione Θ in M sotto-regioni;

2. Si campiona ciascuna sotto regione e si sceglie quella con il miglior indice che diventa quindi la regione più promettente σ
3. Si partiziona σ in altre M sotto-regioni e si aggregano le restanti nella regione circostante così da avere in totale M+1 regioni candidate all'iterazione successiva;
4. Si campiona ogni sotto-regione e si stima il suo indice. Si aggiorna quindi σ con la regione scelta come quella più promettente. Se la regione circostante risulta essere quella più promettente allora si effettua il backtracking.
5. Fino a quando non è soddisfatto il criterio di arresto scelto, si ritorna al punto 4.

Come sarà anche più chiaro in seguito, il punto critico di un tale algoritmo risulta anche essere lo step 5, ovvero il criterio di terminazione.

1.5 Local branching

Un secondo metodo euristico basato sulla metodologia del Branch & Bound è stato proposto, più recentemente, in Fischetti e Lodi 2003. La tecnica di base si configura come un metodo esatto ed è basata sull'utilizzo di condizioni di branching espresse attraverso disuguaglianze lineari (local branching cuts). Essi considerano un generico MIP con variabili 0-1 nella seguente forma:

$$(P)\ \min c^T x \qquad (1.5)$$

$$Ax \geq b \qquad (1.6)$$

$$x_j \in \{0,1\} \quad \forall j \in B \neq 0 \qquad (1.7)$$

$$x_j \geq 0, integer \quad \forall j \in G \qquad (1.8)$$

$$x_j \geq 0 \quad \forall j \in C \qquad (1.9)$$

dove l'insieme delle variabili $\mathrm{N} := \{1,\cdots,\mathrm{n}\}$ è partizionato in $(\mathrm{B},\mathrm{G},\mathrm{C})$. $\mathrm{B} \neq \emptyset$ è l'insieme delle variabili 0-1, mentre gli insiemi G e C che possono essere anche vuoti corrispondono rispettivamente alle variabili intere e alle variabili continue. Data un soluzione ammissibile di riferimento $\bar{x}$ di (P) e un parametro intero non negativo k, il vicinato k-OPT $\bar{x}$ di (P) è definito come l'insieme delle soluzioni ammissibili di (P) che soddisfano il vincolo addizionale di local branching:

$$\Delta(x,\bar{x}) := \sum_{j\in B:\bar{x}_j=1}(1-x_j) + \sum_{j\in B:\bar{x}_j=0} x_j \leq k \qquad (1.10)$$

dove i due termini nella parte sinistra tengono conto del numero die variabili binarie che possono mutare il loro valore (rispetto a $\bar{x}$) o da 1 a 0 oppure da 0 a 1. Il local branching è utilizzato in Fischetti e Lodi 2003 come un criterio di branching all'interno di un sistema enumerativo per (P). Infatti, data la soluzione incombente storico $\bar{x}$, lo spazio delle soluzioni associate al nodo corrente può essere partizionata mediante le seguenti disgiunzione:

$$\Delta(x,\bar{x}) \leq k \text{ o } \Delta(x,\bar{x}) \geq k+1 \qquad (1.11)$$

dove il parametro k determina la dimensione del vicinato ed è scelto in modo appropriato. L'approccio alterna fasi strategiche di alto livello in cui sono utilizzati i local branching cuts per definire delle regioni in cui le soluzioni sono più 'promettenti' e le fasi tattiche di basso livello in cui queste regioni sono elencate implicitamente, attraverso una

branching classico sulle variabili. Il risultato è uno schema esatto del tutto generale che ha lo scopo di favorire i primi aggiornamenti della soluzione incombente, e quindi produrre soluzioni di alta qualità nelle fasi iniziali del calcolo. Nelle osservazioni conclusive del lavoro di Fischetti e Lodi si accenna alla possibilità di utilizzare il loro metodo per progettare un autentico framework meta euristico per MIP simile ad una Tabu Search (TS) (Glover and Laguna 1997) o Variable Neighborhood Search (VNS) (Mladenovìc and Hansen 1997). Infatti, tutti gli ingredienti principali di queste metaeuristiche (che definisce il vicinato di una soluzione, considerando soluzioni tabu o mosse, imponendo una diversificazione adeguata, ecc) possono essere facilmente modellati in termini di tagli lineari che possono essere dinamicamente inseriti e rimossi dal modello. Lo schema metaeuristica è poi costruito in cima ad un risolutore general-purpose MIP, che viene utilizzato come una scatola nera per esplorare le soluzioni contenute nel vicinato che viene definito dal modello originale MIP a cui vengono aggiunti dei local branching cuts. Questo porta naturalmente a uno schema completamente generare di TS o di VNS per MIPS, che è facile da implementabile e, si spera anche più efficace (e talvolta anche meglio) delle ad-hoc TS o dei metodi VNS sviluppati per gli specifici problemi. Nel lavoro di Fischetti et. al. 2004 è stata introdotta ed analizzata una specifica implementazione dell'idea sovra esposta. L'obiettivo è quello di mostrare come il paradigma di local branching può essere specializzato per importanti classi di problemi, in modo da ottenere un (ancora piuttosto generale) sistema euristico con prestazioni migliorate. In particolare, si propone una variante del classico schema VNS che viene chiamato Diversification, Refining, and Tight-refining (DRT). Il nuovo approccio è particolarmente adatto per MIP in cui l'insieme delle variabili binarie può essere partizionato in due insiemi (chiamati livelli), con la proprietà che il fixing del valore delle variabili di primo livello, produce un sottoproblema più facile da risolvere (ma non banale).

1.5.1 The DRT method

Consideriamo il problema MIP (P) definita in precedenza, e assumiamo che l'insieme delle variabili binarie B è stato partizionato in (B1, B2), dove gli insiemi B_1 e B_2 corrispondono alle variabili di primo e secondo livello, rispettivamente. Data una soluzione corrente $\bar{x}$ del problema (P), ci proponiamo di esplorare nel modo più efficacemente possibile il vicinato della soluzione ottenuta fissando (o quasi) le variabili di primo livello per il loro valore corrente in $\bar{x}$. In questa fase, denominata refining, aggiungiamo, semplicemente, il seguente vincolo:

$$\Delta_1(x, \bar{x}) \coloneqq \sum_{j \in B_1 : \bar{x}_j = 1} (1 - x_j) + \sum_{j \in B_1 : \bar{x}_j = 0} x_j \leq k_1 \qquad (1.12)$$

al MIP corrente, dove è fissato il parametro k_1 a un valore molto piccolo (ad esempio, $k_1 = 2$ o anche $k_1 = 0$). Abbiamo utilizzato un risolutore MIP general-purpose nel tentativo di risolvere il modello risultante in ingresso, inoltre abbiamo fissato l'upper bound al valore della migliore soluzione conosciuta. Se il modello non è risolto all'ottimo entro un tempo limite, si entra in una fase euristica di tight-refining in cui le variabili di primo livello sono ancora vincolati da (8), ma si limita anche la variazione delle variabili di secondo livello attraverso un local-branching constraints del seguente tipo:

$$\Delta_2(x, \bar{x}) \coloneqq \sum_{j \in B_2 : \bar{x}_j = 1} (1 - x_j) + \sum_{j \in B_2 : \bar{x}_j = 0} x_j = k_2 \qquad (1.13)$$

Per valori crescenti del vicinato $k_2 = 0, k_2^{step}, 2\, k_2^{step}, \ldots, k_2^{max}$, dove k_2^{step} e k_2^{max} sono parametri di input per la procedura di DRT (tipicamente, $k_2^{step} = 2$ e $k_2^{max} = 10$). In questo modo possiamo esplorare attraverso il risolutore MIP general-purpose (con un adeguato tempo limite) una serie di vicinati di secondo livello, che sono tutti contenuti nel vicinato di primo livello definito dalla (1.12). La fase di tight-refining termina quando k_2 raggiunge k_2^{max} (il valore massimo consentito), o quando il limite di tempo complessivo per questa fase è raggiunto. Quando rimuoviamo i vincoli (1.12) e (1.13) dal MIP corrente, e proseguiamo considerando differenti settaggi per le variabili di primo livello. Questa fase di diversificazione è attuata secondo l'idea della VNS. Per essere precisi, si aggiunge il seguente vincolo di diversificazione

$$k_1^{min} \leq \Delta_1(x, \bar{x}) \leq k_1^{max} \quad (1.14)$$

al modello corrente, cercando qualsiasi (in linea di principio casuale) soluzione ammissibile contenuta in tale regione. Come in Fischetti e Lodi 2003, questa fase è ottenuta attraverso l'utilizzo di un risolutore MIP come un black-box, che viene eseguito senza inserire alcun limite superiore e viene interrotta non appena la prima soluzione euristica viene trovata. Ciò produce una buona soluzion da utilizzare come soluzione corrente ($\bar{x}$) di riferimento all'interno del processo per l'iterazione successiva. Qualora non si trovi una soluzione entro un tempo limite, i parametri k_1^{min} e k_1^{max} sono modificati in modo da definire un vicinato sempre più ampio. Si deve rilevare che per come è descritto l'algoritmo fino ad ora non si può escludere che durante la fase di diversificazione noi possiamo riottenere la stessa soluzione di riferimento ottenuta in precedenza e quindi avere sempre la stessa struttura per le variabili di primo livello. Per evitare questo rischio, prima di sostituire la soluzione di riferimento attuale $\bar{x}$ con quela nuova, si procede inserendo, secondo la filosofia TS , il seguente vincolo tabù:

$$\Delta_1(x, \bar{x}) \geq 1 \quad (15)$$

al modello MIP corrente, in modo statico (cioè, questo vincolo non sarà mai rimosso dal modello). Il metodo DRT richiede una scelta intelligente delle variabili di primo livello. Questa scelta può essere lasciata al progettista del modello in grado di fornire in modo esplicito l'elenco delle variabili di primo livello, basato sulla sua conoscenza del problema formulato come un MIP. Inoltre, si deve osservare che il nostro metodo DRT funziona correttamente anche nel caso in cui nessuna variabile di primo livello esiste (cioè, quando $B_1 = \emptyset$). Il metodo globale DRT è descritto nell'algoritmo sottostante. In ogni passo, la milgior soluzione possibile è implicitamente aggiornata. Alla prima esecuzione del ciclo repeat-until, una fase di diversification (piuttosto che un processo di refining) è immediatamente eseguito, in cui la soluzione di riferimento è $\bar{x}$, trovate da qualche euristica è probabile che sia di buona qualità, quindi, il fissaggio delle sue variabili di primo livello in genere non si traduce in un miglioramento significativo. Il nostro modo di scegliere i parametri della DRT può portare a risultati diversi., in particolare il limite di tempo. Impostando un limite temporale grande per la fase d'intensificazione ci porterà a esplorare in modo approfondito solo alcuni configurazioni che riguardano un intorno delle variabili di primo livello. In alcuni casi, i risultati migliori sono attenuti, esplorando con un numero maggiore di fasi di diversificazione, anche se un limite temporale per la fase d'intensificazione è breve.

Algorithm 1.1 : The overall DRT method

find heuristically a "good" starting reference solution $\bar{x}$, e.g., by applying the MIP solver with a short time limit;

repeat

add statically the tabu constraint (1.15) to the current MIP;

add temporarily the diversification constraint (1.14) to the current MIP, and apply the MIP solver (with no upper bound) to find a first feasible solutionthat replaces $\bar{x}$;

remove the diversification constraint, and (almost) fix the first-level variables to their value in $\bar{x}$, by adding temporarily constraint (1.12) to the current MIP;

solve heuristically the resulting MIP through the MIP solver, possibly adding a sequence of temporary local branching constraints (1.13) for nested-neighborhood exploration;

remove all temporary constraints

until the overall time limit is exceeded

Si deve di evitare di spendere troppo tempo computazionale per esplorare delle configurazioni di primo livello che non conducanmo ad un miglioramento significativo della soluzione corrente. A tal fine, non appena si rileva il rischio di stallo si forza l'algoritmo a compiere un passo diversificazione in cui è richiesto ad un numero maggiore di variabili di

primo livello a flippare. I parametri di questa strategia sono: (a) la percentuale di miglioramento utilizzato per rilevare la situazione di stallo, (b) il numero massimo iterazioni senza miglioramenti prima di una diversificazione, (c) il numero delle variabili di primo livello per il flipping in una fase di diversificazione, e (d) il numero massimo di diversificazioni di grossa taglia.

Local Branching a due stadi

In questo capitolo verrà presentata una metodologia, denominata "*two stage local branching*", che sarà applicata ai problemi di ottimizzazione combinatoria classificabili come "*rectangle packing problems*". Tali problemi classici si configurano come una possibile base a partire dalla quale indagare la possibilità di formulazioni alternative per il problema di logistica portuale noto come *Berth Allocation Problem* (BAP), del quale ci si occuperà nel capitolo successivo. La differenza maggiore tra il *rectangle packing problem* e il BAP è costituita dall'assenza del vincolo sul tempo di rilascio per ogni oggetto rettangolare che, nel BAP corrisponde all'occupazione spazio-temporale della banchina da parte della nave ormeggiata. Dopo aver illustrato alcune differenti formulazioni matematiche per il *rectangle packing problem*, presenti nella letteratura internazionale, saranno proposti alcuni tagli di *local branching* per quelle formulazioni e verrà discussa e sperimentata la metodologia di *two stage local branching* per le stesse formulazioni. I risultati numerici e le prestazioni della metodologia a due stadi saranno confrontati contro l'implementazione classica del *local branching*; questo al fine di stabilire se si possa disporre, almeno per sottoproblemi del BAP, di una efficace metodologia di soluzione in presenza di dimensioni reali del problema formulato nell'ambito della logistica portuale.

2.1 Stato dell'arte

La maggior parte degli algoritmi per *two dimensional bin packing problem* (2BP) sono di natura "greedy". Gli elementi vengono inseriti uno ad uno, ed inseguito non vengono più riconsiderati. In letteratura sono presenti algoritmi ad una o a due fasi (Lodi et. al. 2002). Nella prima fase dell'algoritmo, gli elementi vengono inseriti direttamente nei bin, mentre negli algoritmi a due fasi prima gli elementi vengono partizionati in livelli ed in seguito vengono poi assegnati nei bin, risolvendo un bin packing monodimensionale. Berkey e Wang 1987 hanno descritto l'euristica classica denominata First Fit, che è un algoritmo "greedy" di tipo monofase. Gli elementi sono ordinati per altezze non crescenti: il primo elemento inizializza il primo livello nel bin e definisce l'altezza del livello; ogni ulteriore elemento è aggiunto nel primo livello tenendo conto della larghezza residua nel livello. Se non esiste spazio nel livello in uso, un nuovo livello viene inizializzato e ciò può essere ripetuto fino a quando il bin corrente non viene saturato. All'interno di ogni livello, gli oggetti non devono essere sovrapposti. L'approccio di formulare un packing or cutting problem come un problema di set covering fu per la prima volta esposto in Gilmore e Gomory 1961. Essi proposero questa tecnica per il problema del cutting stock monodimensionale. La formulazione adottata introduceva una variabile per ogni possibile schema di taglio, per un unico bin. Poiché, in generale, il numero di queste variabili aumenta in modo esponenziale rispetto alla dimensione del problema, non tutte le variabili venivano esplicitamente considerate e quindi si adottava uno schema di risoluzione denominato generazione di colonne. In Gilmore e Gomory 1965, gli autori applicano la stessa tecnica di base al stock cutting bidimensionale. La differenza principale risiede nel metodo per risolvere il problema del pricing, vale a dire, in che modo determinare i modelli/variabili promettenti che possano migliorare la soluzione corrente. Una variante più veloce del metodo di Gilmore e Gomory fu proposta da Oliveira e Ferreira 1994, dove anche i three-stage e general multi-stage cutting stock problems sono considerati. Per risolvere il problema del pricing, un'euristica greedy viene prima adottata, nella speranza che trovi rapidamente una variabile adatta. Solo se questo metodo fallisce, si utilizza un algoritmo esatto che risulta essere più lento rispetto al precedente. Monaci e Toth 2006 presentano una approccio euristico basato sul set covering per problemi di bin packing. In una prima fase, le colonne (cioè i modelli) sono generati usando euristiche costruttive ma veloci; in una seconda fase il relativo set-covering è risolto per mezzo di un Lagrangian-based heuristic algorithm. Più di recente il 2-stage 2BP è stato considerato in Lodi et al.2004. Hanno introdotto il primo modello ILP compatto che coinvolge solo un numero polinomiale di variabili e vincoli. Un' applicazione reale del three-stage cutting problem, con specifiche proprietà aggiuntive, è stata trattata in Vanderbeck 1998 e poi ripresa in Puchinger et al.2004. Vanderbeck risolve un

three-stage two-dimensional cutting stock problem. L'obiettivo principale è quello di minimizzare gli sfridi, tenendo conto dell'invecchiamento dei pezzi e considerando pure ordini urgenti e i costi (fissi) di set-up. L'approccio di risoluzione utilizza un metodo ricorsivo di generazione di colonne. Puchinger et al 2004 considerano un problema a 3 stadi 2BP (originariamente proposto per problemi di taglio del vetro) in cui vengono imposti vincoli specifici aggiuntivi rispetto all'ordine degli elementi. Il problema è risolto utilizzando un algoritmo euristico di tipo evolutivo (EA). Una panoramica più generale degli algoritmi evolutivi per il problema di cutting and packing si trova in Hopper 2000. I seguenti due algoritmi esatti risolvono il problema generale del 2BP in cui sono consentite anche profili (pattern) di oggetti rettangolari che non possono essere tagliati col metodo a ghigliottina. Il primo è dovuto a Martello e Vigo 1998, che descrivono un algoritmo di branch-and-bound a due livelli. Gli elementi sono assegnati ai bin da un albero decisionale. Possibili modelli di packing per i bin sono generati prima utilizzando un'euristica e poi, se questa fallisce, si tenta di trovare uno schema di taglio ammissibile attraverso un metodo enumerativo. Un algoritmo ibrido di branch-and-price/constraint programming è stato proposto da Pisinger e Sigurd 2006. Qui si utilizza il metodo della generazione di colonne di Gilmore e Gomory, mentre si risolve il problema specifico del pricing utilizzando la metodologia di constraint programming. Lodi et al. 1999 descrivono uno schema euristico abbastanza generale, applicabile a diverse varianti di 2BP. Una tabu search viene utilizzata per assegnare gli elementi ai bin ed i modelli di taglio per ogni bin sono ottenuti attraverso diverse euristiche. Infine e più recentemente, appare degno di nota un algoritmo di branch and price per il problema del bin packing dovuto a Puchinger and Raidl 2007.

2.2 Formulazioni: rectangular packing

Il primo modello che sarà illustrato è il modello denominato *two-level strip packing* (2LSP), introdotto da Lodi et al. 2004, che fanno uso di variabili d'inizializzazione e d'inserimento (y_i: vale 1 se e solo se l'oggetto *i* inizializza il livello *i*; x_{ij}: vale 1 se e solo se l'oggetto *j* è inserito nel livello *i*):

$$\min \sum_{j \in N} h_i y_i, \qquad (2.1)$$

s.v.

$$\sum_{i=1}^{j-1} x_{ik} + y_k = 1, \qquad \forall k \in N \qquad (2.2)$$

$$\sum_{i \le k} w_j x_{ik} \le (W - w_k) y_k, \qquad \forall k \in N \qquad (2.3)$$

$$x_{ik} \in \{0,1\}, \qquad \forall i = 1 \dots n-1, \forall k > i \qquad (2.4)$$

$$y_k \in \{0,1\}, \qquad \forall k = 1 \dots n, \qquad (2.5)$$

Tutti gli oggetti sono ordinati per altezza non crescente. Ogni variabile x_{ij} indica se l'oggetto *i* è assegnato al livello inizializzato dall'oggetto *j*, quindi ogni variabile x_{jj} indicata se l'oggetto *j* ha inizializzato il livello. Grazie all'ordinamento degli oggetti, possono essere fissate a zero le variabili x_{ij} con *i>j* e dal modello. Il vincolo (2.2) impone che ogni oggetto deve essere assegnato ad un livello. Il vincolo (2.3) impone che la somma delle larghezze degli oggetti assegnati alla strip non ecceda la larghezza di quest'ultima. L'obbiettivo di tale modello è quello di minimizzare l'altezza complessiva delle strisce inizializzate. Qui di seguito viene presentato un modello che trae ispirazione da quello presentato da Puchinger and Raidl 2007 per il problema del bin packing. Di fatto, è stato esteso al problema del rectangle packing eliminando alcune restrizioni inserite dai suddetti autori. Le notazioni su parametri e variabili del modello sono le seguenti:

Parametri

n, cardinalità degli oggetti.

W, larghezza del bin.

w_i, larghezza dell'oggetto i-esimo.

h_i, altezza dell'oggetto i-esimo.

l, Strips

Variabili

H: Altezza massima dell'intero bin.

h^k: Altezza dell'k-esima Strip.

α_{ji}: vale 1 se l'oggetto i è contenuto nello stack j, 0 altrimenti.

β_{kj}: vale 1 se lo stack j è contenuto nella k-esima Strip, 0 altrimenti.

δ^k: set to 1 if and only if strip k is used

$$\alpha_{ji} = \begin{cases} 1, & se\ l'oggetto\ i\ è\ contenuto\ nello\ stack\ j \\ 0, & altrimenti \end{cases}$$

$$\beta_{kj} = \begin{cases} 1, & se\ lo\ stack\ j\ è\ contenuta\ nella\ strip\ k \\ 0, & altrimenti \end{cases}$$

La nuova formulazione del modello, in seguito denominato stack packing model (SPM) è la seguente:

$$\min H, \tag{2.6}$$

s.t.

$$\sum_{j=1}^{i} \propto_{ji} = 1, \qquad \forall\ i = 1, \dots, n \tag{2.7}$$

$$\propto_{ji} = 0, \qquad \forall\ i > j\ \wedge\ w_i \leq w_j \tag{2.8}$$

$$\sum_{k=1}^{l} \beta_{kj} = \propto_{jj}, \qquad \forall\ j = 1, \dots, n \tag{2.9}$$

$$\sum_{i=j+1}^{n} \propto_{ji} \leq (n-j) \propto_{jj}, \qquad \forall\ j = 1, \dots, n-1 \tag{2.10}$$

$$\sum_{i=j}^{n} h_i \propto_{ji} \leq h^k + M\left(1 - \beta_{kj}\right), \qquad \forall\ k = 1, \dots, l;\ \forall\ j = 1, \dots, n \tag{2.11}$$

$$h^k \leq M\delta_k, \qquad \forall\ k = 1, \dots, l \tag{2.12}$$

$$\sum_{j=1}^{n} w_j \beta_{kj} \leq W, \qquad \forall\ k = 1, \dots, l, \tag{2.13}$$

$$\sum_{k=1}^{l} h^k \leq H \tag{2.14}$$

$$\sum_{j=1}^{n} \beta_{kj} \leq M\delta_k \qquad \forall\ k = 1, \dots, l \tag{2.15}$$

$$\delta_k \geq \delta_{k+1} \qquad \forall\ k = 1, \dots, l-1, \tag{2.16}$$

$$H \in I \tag{2.17}$$

$h^k \in I$ $\forall\, k = 1, \dots, l$ (2.18)

$\propto_{ji} \in \{0,1\},$ $\forall\, i = 1, \dots, n, \forall\, i = 1, \dots, n$ (2.19)

$\beta_{kj} \in \{0,1\},$ $\forall\, k = 1, \dots, l, \forall\, j = 1, \dots, n$ (2.20)

Rammentando che gli oggetti sono ordinati per altezza non crescente, i vincoli possono essere spiegati come segue. Il vincolo (2.7) implica che ogni oggetto è assegnato ad uno *stack*. Il vincolo (2.8) implica che gli oggetti inseriti nello *stack* hanno un indice crescente rispetto all'elemento che lo ha inizializzato, inoltre le larghezze degli oggetti che appartengono al medesimo *stack* devono essere minori o uguali alla larghezza dell'oggetto che ha inizializzato lo *stack*. Il vincolo (2.9) implica che ogni *stack* inizializzato deve essere assegnato ad una striscia. Il vincolo (2.10) implica che gli oggetti assegnabili alla strip *j* possono essere solo gli oggetti che hanno un id maggiore di j. Il vincolo (2.11) calcola l'altezza della striscia k, se tale striscia è inizializzata. Il vincolo (2.12) azzera l'altezza della striscia se questa non è inizializzata. Il vincolo (2.13) implica che la larghezza complessiva di tutti gli *stack* inseriti nella striscia *k* non eccede la larghezza massima della striscia. Il vincolo (2.14) calcola l'altezza complessiva di tutte le strisce. Il vincolo (2.15) abilita l'assegnamento degli *stack* alla striscia se questa è inizializzabile. Il vincolo (2.16) identifica le strisce inizializzabili, poiché non si può utilizzare una striscia k se la striscia k-1 non è stata ancora utilizzate, questo vincolo elimina delle soluzioni simmetriche.

2.3 Il taglio nella tecnica del local branching

Per ogni modello illustrato in precedenza (LSP & SPM) sono stati formulati dei di tagli per il local branching sia di primo livello che di secondo. Per il modello LSP sono stati formulati i seguenti due tagli

$\Delta(y_i, \bar{y}_i) \geq 2$ (2.21)

$\Delta(x_{ij}, \bar{x}_{ij}) \geq 2$ (2.22)

mentre i tagli di local branching (sia di primo che di secondo livello) per la fase di diversificazione sono i seguenti:

$\Delta(y_i, \bar{y}_i) \geq 2$ (2.23)

$\Delta(x_{ij}, \bar{x}_{ij}) \leq 2 * (iter)$ (2.24)

Per il modello di SPM sono stati formulati i seguenti tagli, rispettivamente di primo e di secondo livello:

$\Delta(\propto_{ji}, \bar{\propto}_{ji}) \geq 2$ (2.25)

$\Delta(\beta_{kj}, \bar{\beta}_{kj}) \geq 2$ (2.26)

mentre i tagli di local branching (sia di primo che di secondo livello) per la fase di diversificazione sono i seguenti:

$\Delta(\propto_{ji}, \bar{\propto}_{ji}) \geq 2$ (2.27)

$\Delta(\beta_{kj}, \bar{\beta}_{kj}) \leq 2 * (iter)$ (2.28)

In generale, nella fase di diversificazione sia il vincolo di primo livello che quello di secondo livello possono dipendere dall'iterazione corrente di diversificazione.

2.4 Approccio Risolutivo: 2-Stage local branching

L'approccio risolutivo proposto si compone di 3 fasi, la prima delle quali ha il compito di trovare una soluzione iniziale mediante una qualsiasi euristica. In questa sede è stata approfondita e sperimentata la tecnica euristica della Feasibility Pump (FP), proposta da Lodi and Fischetti (2005). Dovendo determinare una soluzione iniziale "feasible", nel passo 1 dell'algoritmo si risolve il rilassato lineare del problema in esame. Tale risoluzione fornisce, in generale, una soluzione con componenti frazionarie (che, però, rispetta tutti gli altri vincoli del problema). Nel passo successivo (il 2) ogni componente della soluzione viene arrotondata ad un valore intero (ad es.: se il valore della variabile è minore di 0.5 la variabile assume il valore zero altrimenti assume il valore 1). Come gli stessi autori suggeriscono, per accelerare il processo di risoluzione, il passo 1 può essere sostituito con la generazione di una soluzione completamente casuale, dove ogni componente viene inizializzata con un numero casuale compreso tra zero ed uno che, in seguito, dovrà essere arrotondato (step 2), per ottenere una soluzione intera di riferimento, $\tilde{x}$, con tutte le componenti intere. A questo punto entra in gioco la funzione $\Delta(x, \tilde{x})$, chiamata anche distanza di Hamming, che è definita come segue:

$$\Delta(x_j, \bar{x}_j) = \sum_{j \in I} |x_j - \bar{x}_j|$$

Tale espressione sarà la funzione obbiettivo nel problema che dovrà essere risolto ad ogni iterazione, nella euristica FP. Il problema avrà la stessa regione ammissibile del problema originario, i vincoli d'interezza saranno sempre rilassati mentre l'obbiettivo diventerà quello di trovare una soluzione frazionaria, x, ammissibile e che minimizzi la distanza dalla soluzione $\tilde{x}$ (che non è ammissibile, perché in generale non rispetta i vincoli imposti dal problema, pur avendo tutte componenti intere). Dopo aver determinato una nuova soluzione, se dovesse accadere che le sue componenti non siano intere occorrerà ripetere l'operazione di arrotondamento. L'algoritmo sarà arrestato quando la distanza di Hamming diventerà nulla, ovvero quando la soluzione di riferimento, $\tilde{x}$, risulterà non solo intera ma pure ammissibile. Potrebbero verificarsi alcuni piccoli inconvenienti durante la determinazione di una soluzione iniziale per il problema originario, poiché il processo potrebbe entrare in un ciclo e rimanerci all'infinito. Sono due le cause che potrebbero determinare un ciclo infinito: la prima è legata ad un non variazione della soluzione di riferimento, $\tilde{x}$, cioè la soluzione di riferimento, $\tilde{x}$, è la stessa del passo precedente; la seconda causa è legata al verificarsi di un ciclo di soluzioni, ovvero alla riproposizione delle stesse soluzioni negli ultimi passi dell'algoritmo. La prima situazione di ciclo è facilmente identificabile e l'azione che si adopera per uscire da una simile condizione di stallo è quella di perturbare la soluzione di riferimento facendo cambiare il valore di una o più variabili da zero ad uno; oppure è possibile compiere un'azione di restart del processo, che consiste nel generare una soluzione completamente casuale e poi successivamente arrotondarla, facendo così ripartire il processo di ricerca. La seconda condizione di stallo viene controllata euristicamente, cioè se nelle ultime KK (gli autori suggeriscono 70) iterazioni il valore H della distanza di Hamming non è migliorato allora viene eseguita un'azione di restart come descritto in precedenza. L'algoritmo proposto dagli autori è completamente generale ed indipendente dal modello da risolvere. Dopo aver trovato una soluzione iniziale con la feasibility pump (o attraverso un'altra euristica), è stato sviluppato un algoritmo di local branching indipendente dal problema da risolvere. Tale algoritmo viene iterato fintanto che la soluzione corrente può essere migliorata. In tale algoritmo nessuna fase di intensificazione viene intrapresa poiché è stato osservato sperimentalmente che l'intensificazione non apporta, quasi sempre, alcun miglioramento alla soluzione corrente. Se il processo principale non riesce a trovare una soluzione migliore di quella corrente (che fornisce un UB al nostro problema) viene compiuta una fase di diversificazione, che si compone di un certo numero di iterazioni fissato dall'utente (nel nostro caso è uguale a 3), nella quale si esplorano differenti partizioni della regione ammissibile in modo da focalizzare la ricerca in determinate aree che potrebbero essere più promettenti. Quando, in una delle iterazioni della diversificazione, si riesce a trovare una soluzione migliore l'algoritmo si arresta e continua ad esplorare la regione ammissibile con le stesse regole usate prima nella fase di diversificazione, riprendendo la sua nomale esecuzione; altrimenti, se nessuna iterazione nella fase di diversificazione trova una soluzione migliore, l'algoritmo si arresta. Ogni nodo dell'albero del *local branching* viene risolto entro un certo limite di tempo ed imponendo che la funzione obbiettivo sia compresa tra un *lower bound* ed un *upper bound*:

$$LB \leq f.o. \leq UB$$

Al fine di non riscontrare soluzioni peggiori rispetto a quella corrente, grazie al *lower bound,* si restringe la regione da esplorare. Una delle varianti proposte è sperimentate in questo lavoro è stata quella di usare la migliore soluzione ottenuta dal processo FP+LB come soluzione di riferimento per un ulteriore algoritmo di LB specializzato ad una formulazione matematica diversa da quella originaria, nel senso che presenta una regione ammissibile più ampia di quella precedente e la ingloba totalmente. Lo pseudo codice dell'algoritmo per il *local branching* implementato è il seguente:

```
(1)   Modello m=createModel();
(2)   sol=inizialize(m);
(3)   while (true){
(4)       addToModelFirstCut(m,sol);
(5)       addToModelSecondCut(m,sol);
(6)       addToModel_LB_&_UB(m,lowerBound,objectiveValue(sol));
(7)       if(solve(m)){
(8)           sol=getCurrentSolution(m);
(9)       }else{
(10)          boolean noImprovements=true;
(11)          for(int iter=0;iter<maxDivesificationIter;iter++){
(12)              addToModelFirstDiversificationCut(m,sol,iter);
(13)              addToModelSecondDiversificationCut(m,sol,iter);
(14)              addToModel_LB_&_UB(m,lowerBound,objectiveValue(sol))
(15)              if(solve(m)){
(16)                  sol=getCurrentSolution(m);
(17)                  noImprovements=false;
(18)                  break;
(19)              }
(20)          }
(21)          if(noImprovements){
(22)              break;
(23)          }
(24)      }
(25)  }
```

Dopo aver formulato il modello matematico su cui sarà applicata la tecnica del *local branching* (passo 1), occorre determinare una soluzione ammissibile per inizializzare l'euristica basata sul *local branching*. E si hanno tre possibilità. La prima, come già anticipato, prevede una chiamata ricorsiva ad un ulteriore metodo di *local branching* strutturato su un diverso problema le cui soluzioni sono un sotto insieme del problema che si vuole risolvere. La seconda alternativa è quella di risolvere il problema in modo euristico/meta euristico ed usare tale soluzione per inizializzare il *local branching*. La terza alternativa, che si può usare in qualsiasi circostanza e soprattutto quando non si dispone di un'euristica per il problema, prevede l'utilizzo della *feasibility pump* . L'algoritmo di *local branching* continua ad iterare (passo 3) fino a che a soluzione corrente sarà migliorata, altrimenti l'algoritmo si arresta grazie ai passi 21-23. Nel passo 4 si applica, al problema originario, il taglio di primo livello costruito sulla base della soluzione corrente, mentre nel passo 5 si applica al modello il taglio di secondo livello costruito sempre sulla base della soluzione corrente. Infine, al passo 6, si persegue la ricerca di una soluzione migliore utilizzando il *lower bound* e l'*upper bound* a disposizione per il problema, in modo da restringere la regione d'interesse. Al passo 7 viene risolto il modello derivante dal problema originale più i tagli di *local branching.* Se l'esito è positivo allora la soluzione corrente viene aggiornata e l'algoritmo continua con i passi da 4 a 7. Se al passo 7 non si dovesse trovare una soluzione migliorativa rispetto a quella corrente allora si deve diversificare la ricerca. I passi della diversificazione sono un numero finito che coincide con maxDiversificationIter (imposto dall'utilizzatore); inoltre la regione d'interesse viene sempre ristretta applicando i

migliori *lower bound* e l'*upper bound* conosciuti. Se in una delle fasi della diversificazione viene trovata una soluzione migliore la fase di diversificazione si arresta e l'algoritmo procede con i passi da 4 a 7. Come detto in precedenza, se alla fine della fase di diversificazione non è stata trovata una soluzione migliore allora l'algoritmo di *local branching* si arresta.

2.5 Risultati Numerici

Prima verrà illustrata la prestazione del *local branching* sui modelli LSP e SPM ed il *2 Stage local branching* sulle cosiddette istanze mal condizionate per il problema di *Packing*, come definite da Lodi et al. (2004).

	Local Branching LSP			Local Branching SPM			2 Stage Local Brnaching		
Instanza	time	fo	%LB	time	fo	%LB	time	fo	%LB
CGCUT02	0,5	78,0	21,88%	1158,4	70,0	9,38%	1711,8	70,0	9,38%
CGCUT03	574,4	711,0	7,56%	2519,8	743,0	12,41%	4176,5	700,0	5,90%
GCUT02	0,1	1262,0	4,64%	2066,8	1249,0	3,57%	1017,7	1262,0	4,64%
GCUT03	1,0	1810,0	1,74%	2500,4	1866,0	4,89%	1224,7	1810,0	1,74%
GCUT04	34,3	3126,0	3,99%	4742,3	3395,0	12,94%	3138,8	3112,0	3,53%
HT04	0,2	16,0	6,67%	1523,2	16,0	6,67%	533,2	16,0	6,67%
HT05	0,2	19,0	26,67%	1011,7	16,0	6,67%	1238,7	16,0	6,67%
HT06	0,1	16,0	0,00%	11,8	16,0	0,00%	0,1	16,0	0,00%
NGCUT12	0,1	102,0	21,43%	1555,4	87,0	3,57%	1968,6	87,0	3,57%
BENG02	304,1	62,0	8,77%	2030,5	62,0	8,77%	1518,6	62,0	8,77%
BENG03	606,8	92,0	9,52%	2500,5	94,0	11,90%	2108,2	92,0	9,52%
BENG04	609,0	114,0	5,56%	4001,1	124,0	14,81%	2182,5	114,0	5,56%
BENG05	951,8	141,0	4,44%	3001,4	156,0	15,56%	2521,5	141,0	4,44%
BENG06	300,8	42,0	16,67%	2000,5	38,0	5,56%	2319,3	38,0	5,56%
BENG07	601,9	75,0	11,94%	2500,7	76,0	13,43%	2114,3	75,0	11,94%
BENG08	49,3	108,0	6,93%	6004,4	125,0	23,76%	1735,1	108,0	6,93%
BENG09	916,1	134,0	6,35%	4005,9	175,0	38,89%	2916,9	134,0	6,35%
BENG10	1725,6	164,0	5,13%	non risolto			3727,2	164,0	5,13%
media	**370,9**	448,4	**9,44%**	**2537,3**	488,7	**11,34%**	**2008,5**	445,4	**5,91%**
deviazione standard	476,8	827,9	0,07	1474,1	907,6	0,09	1056,4	825,9	0,03

Tabella 2.1: istanze "easy"

Si può riconoscere che, sul modello LSP, *local branching* ha un comportamento migliore (in termini di distanza dal *lower bound*) rispetto al *local branching* sul modello SPM. Infatti, il primo presenta una distanza media del 9,44% mentre il secondo presenta una distanza media del 11,34%. Il *2 stage local branching* presenta una distanza media dal *lower bound* del 5,91%; dunque,tale tecnica ha un comportamento medio migliore rispetto alle precedenti e anche la deviazione standard è inferiore. Per quanto riguarda il tempo di calcolo, il *local branching* sul modello SPM presenta (mediamente) il valore maggiore (2537sec), mentre il *local branching* sul modello LSP presenta (mediamente) il tempo di calcolo migliore (solo 370sec). Invece, il *2 stage local branching* presenta un tempo computazionale che è maggiore al tempo del *local branching* sul modello LSP ma inferiore al tempo del *local branching* sul modello SPM. Quindi, in tal caso, sul modello SPM (combinato con il modello LSP) il *2 stage local branching* ha un comportamento migliore

rispetto al classico *local branching* sempre sullo stesso modello, sia in termini di distanza media dal *lower bound* sia in termini di tempo computazionale. Ancora, si può notare come la BENG10, che è l'istanza che contiene il maggior numero di oggetti (n. oggetti uguale a 120), non viene risolta dal *local branching* classico a causa dell'elevato numero di variabili e vincoli creati: la tecnica risolutiva si blocca già nella fase iniziale (determinazione di una soluzione iniziale attraverso la *feasibility pump*), invece la 2 stage *local branching* (che dispone di una soluzione iniziale fornita dal *local branching* LSP) istanzia un minore numero di variabili e vincoli e questo permette la risoluzione del modello. Si osservi che quest'ultimo approccio è stato sperimentato sulle 500 istanze presentate nel lavoro di Lodi et al. 2004 (si tratta di 10 classi d'istanze, dove ogni classe contiene 50 istanze e queste 50 istanze presentano un numero differente di oggetti). Come si può vedere nelle seguenti tabelle, il trend precedentemente individuato si mantiene; il *2 stage local branching* presenta la distanza media dal *lower bound* migliore rispetto al *local branching* sui modelli LSP o SPM ed anche la deviazione standard è migliore rispetto a quella stimata per le altre due tecniche risolutive. Per quanto riguarda il tempo di calcolo, il *local branching* LSP è il più veloce ma il *2 stage local branching* ha un tempo computazionale medio inferiore al *local branching* LSP. La differenza più eclatante tra il *local branching SPM e il 2 stage local branching* consiste nella differenza di trend: mentre il primo ottiene delle prestazioni che peggiorano con l'aumentare del numero di oggetti, il *2 stage local branching* ottiene delle prestazioni che migliorano con l'aumentare del numero di oggetti. Logicamente, l'indce di prestazione che è stato preso a riferimento è la distanza media dal *lower bound*.

$$Gap\% = \frac{(bestFeasibleSolution - ContinousBound)}{ContinousBound}$$

	Local Branching 2DSSP					
	Time (sec.)		O.F. Value		GAP%	
# Items	Mean	Std	Mean	Std	Mean	Std
20	**2,67**	**14,16**	**364,39**	**328,24**	**14,64%**	**0,11**
40	**145,56**	**257,24**	**717,23**	**646,95**	**8,22%**	**0,05**
60	**299,89**	**342,23**	**1071,39**	**984,43**	**6,27%**	**0,03**
80	**353,44**	**340,29**	**1465,69**	**1326,78**	**4,83%**	**0,03**
100	**621,13**	**554,00**	**1749,59**	**1576,32**	**4,61%**	**0,02**
ALL	**284,54**	**404,68**	**1073,66**	**1177,94**	**7,71%**	**0,07**

Tabella 2.2: Local Branching su 2DSSP

	Local Branching 2DSPM					
	Time (sec.)		O.F. Value		GAP%	
# Items	Mean	Std	Mean	Std	Mean	Std
20	**1256,92**	**829,74**	**353,20**	**326,25**	**7,84%**	**0,06**
40	**3886,97**	**2044,49**	**716,50**	**649,70**	**6,77%**	**0,04**
60	**3692,72**	**1172,44**	**1113,71**	**1006,90**	**10,30%**	**0,06**
80	**3914,02**	**1341,13**	**1560,94**	**1361,12**	**12,98%**	**0,07**
100	**5594,52**	**3000,44**	**1931,57**	**1662,14**	**17,30%**	**0,09**
ALL	**3669,03**	**2304,82**	**1135,18**	**1242,12**	**11,04%**	**0,08**

Tabella 2.3: Local Branching su 2DSPM

	Two Stage Local Branching					
	Time (sec.)		O.F. Value		GAP%	
# Items	Mean	Std	Mean	Std	Mean	Std
20	**1091,73**	**858,17**	**353,19**	**326,26**	**7,84%**	**0,06**
40	**2968,70**	**1462,95**	**704,26**	**644,73**	**5,13%**	**0,04**
60	**3341,46**	**1247,50**	**1061,20**	**981,36**	**4,99%**	**0,03**
80	**3612,34**	**1520,97**	**1456,57**	**1325,25**	**3,95%**	**0,03**
100	**3692,26**	**1374,18**	**1742,97**	**1574,41**	**4,13%**	**0,02**
ALL	**2941,30**	**1623,00**	**1063,64**	**1177,00**	**5,21%**	**0,04**

Tabella 2.4: Two Stage Local Branching

Un ambiente di Simulazione-Ottimizzazione per un berth planning robusto

In questo capitolo si farà riferimento alla logistica portuale, dove un certo numero di processi complessi e interdipendenti devono necessariamente essere descritti in un ambiente dinamico e aleatorio, governato dal verificarsi di eventi. Tali processi possono essere ben modellati come sistemi ad eventi discreti (DES). Al fine di valutare le prestazioni ed ottimizzare uno specifico processo della logistica portuale, può essere perseguita l'idea di inserire un motore di simulazione, che riproduce il processo logistico, in un algoritmo di ottimizzazione delle decisioni di allocazione delle risorse e di schedulazione delle attività del processo stesso. Il metodo di ottimizzazione risultante è di tipo iterativo, nel senso che l'algoritmo di ottimizzazione genera una configurazione (decisionale) iniziale ammissibile e poi procede ad esplorare l'intera regione ammissibile (processo di ricerca) fino a quando sarà capace di individuare delle soluzioni con performance migliori della configurazione iniziale, oppure, in alternativa, fino a quando il tempo di calcolo raggiungerà un prefissato tempo limite. Il ruolo del motore di simulazione è fondamentale (nel processo di valutazione) in quanto effettua una stima della funzione obiettivo che si vorrebbe ottimizzare con la configurazione decisionale ma che, per ipotesi, non può essere restituita dalla semplice valutazione di un'espressione algebrica in forma chiusa.
La metodologia accennata è nota come Simulation Optimization (SO) (Andradottir 2007). Il compromesso tra la quantità di tempo di calcolo necessario per trovare soluzioni alternative migliorative, a fronte dello sforzo computazionale necessario nella stima delle prestazioni tramite la simulazione per una specifica soluzione, è sempre stato un elemento chiave nella maggior parte delle tecniche di SO (Lee et al. 2006).
(Fu 2001) ha diviso le tecniche di S&O nelle seguenti categorie principali:

- procedure statistiche (ad esempio ranking & selection e multiple comparison per il confronto di due o più configurazioni alternativo del sistema);
- metaeuristiche (metodi di ricerca adottati nell'ottimizzazione deterministica);
- ottimizzazione stocastica ((random search, stochastic optimization);
- altri, che includono ordinal optimization e sample path optimization.

In questo capitolo si proporrà un ambiente integrato di Simulation Optimization per gestire il problema fondamentale del berth allocation problem (BAP), tenendo in considerazione sia gli aspetti tattici sia quelli operativi. L'obbiettivo del ambiente di SO è quello di effettuare il raffinamento della soluzione tattica per il BAP, quando le condizioni impreviste e/o indesiderati si manifestano nella fase operativa. L'ambiente SO sviluppato, per ridurre il tempo computazionale richiesto, adotta uno stimatore a media mobile per calcolare la media campionaria usata per comparare le diverse soluzioni alternative per il BAP.
L'ambiente sviluppato, infatti, si basa su una procedura che appartiene alla prima categoria (ovvero: ranking & selection) per stimare la migliore tra una serie di soluzioni alternative per il berth allocation, così come le metaeuristiche che modificano la generazione delle soluzioni sia a livello tattico sia a livello operativo.

3.1. Il problema del Berth Allocation

La logistica in un terminal container è definita attorno a tre processi principali (Vis e De Koster 2003): accettazione delle navi, manovre e ormeggio, caricamento e scaricamento dei container e prelievo ed immagazzinamento dei container dal piazzale. Questi processi implicano l'uso di risorse costose per la movimentazione dei container e l'organizzazione degli stessi può senz'altro beneficiare del supporto fornito da modelli matematici e procedure informatiche basate sulla valutazione e l'ottimizzazione delle prestazioni nell'allocazione delle risorse e nella pianificazione delle attività. Nel seguito ci si concentrerà sul berth allocation problem (BAP), dove una banchina di lunghezza fissata deve essere gestita in maniera ottimale nello spazio e nel tempo in modo da accogliere in maniera adeguata le navi che devono essere ormeggiate lungo di essa.
Come è stato ben sottolineato da (Moorty e Teo 2006), il BAP può essere opportunamente considerato sia a livello tattico sia a quello operativo. Nel primo caso si definisce un " weekly plan ", cioè un modello di ormeggio dove le navi

in arrivo sono ormeggiate in alcuni segmenti di ormeggio preferenziali, sotto l'ipotesi che (1) le navi in arrivo entrano nel porto ad una determinato istante di tempo deterministico e (2) uno specifico, ma fisso, tasso medio di servizio deve essere garantito durante le operazioni di carico-scarico di ogni nave attraccata. Viceversa, a livello operativo, al gestore del terminale viene chiesto di gestire:

- i ritardi con cui le navi arrivano al porto,
- disponibilità in tempo reale di ogni gru e la manodopera da assegnare,
- i ritardi che possono generarsi durante le fasi della lavorazione,
- ostacoli fisici come il pescaggio e limiti fisici sulla banchina che limitano le posizioni di ormeggio
- ecc.

Tutto ciò causa delle correzioni in tempo reale nel weekly plan. A questo livello del processo di pianificazione, sarebbe necessario considerare una rappresentazione più fine dei mezzi e della banchina (ad esempio la posizione delle gru in banchina, i vincoli di spostamento, ecc), oltre ad una riproduzione più fine del processo di carico-scarico per mezzo di un simulatore ad eventi discreti.

Tornando alle decisioni a livello tattico, l'intera banchina di ormeggio può essere vista come un insieme discreto di segmenti di attracco di piccole dimensioni o come un continuo, unico, lungo segmento dove ogni nave è rappresentata come un rettangolo spazio-tempo che riflette lo spazio-tempo occupato all'interno del schema proposto. Qualunque sia la rappresentazione della banchina, l'obiettivo prefissato è quello di raggiungere un buon abbinamento tra le posizioni di stoccaggio container nel piazzale con la posizione di ormeggio della nave in cui vengono effettuate le operazioni di carico-scarico. Nella letteratura precedente, entrambi gli approcci di modellazione della banchina (discreto e continuo) sul piano tattico sono stati perseguiti (Lim 1998, Imai et al. 2001, Kim e Luna 2003, Guan e Cheung 2004, Imai et al. 2005, Cordeau et al. 2005) sotto l'ipotesi comune che sia l'ora di arrivo sia il tempo di elaborazione della nave sono noti a priori e su cui non vi è alcuna fluttuazione dovuta all'incertezza. Lavori più recenti si concentrano sul miglioramento delle prestazioni computazionali e dei metodi euristici proposti per la risoluzione del problema (Wang e Lim 2007, Hansen et al. 2008, Lee e Chen 2009, Buhrkal et al. 2009), mentre altri sono dedicati all'integrazione dell'allocazione della nave, con la successiva decisione di assegnare il giusto numero di gru, ora per ora, ad ogni nave durante le operazioni di carico-scarico (vedi Bierwith e Meisel 2010 per una vasta indagine). Si osservi che solo un paio di paper (Zhou e Kang 2008, Hendriks et al 2010) si sono concentrati sul problema di gestire l'incertezza nei tempi di arrivo della nave e nei tempi di servizio. Precisamente, Zhou e Kang adotta una rappresentazione discreta del berth, con un dato numero di punti di servizio lungo la banchina, e propone un modello integrato di ormeggio e gru di banchina, utilizzando un modello di programmazione stocastica 0-1 con l'obiettivo di minimizzare i tempi di attesa sia per l'ormeggio sia per l'assegnazione delle gru. Viceversa, Hendriks et al. 2010 sottolineano che gli operatori del terminal container e le compagnie di navigazione possono concordare delle finestre temporali in cui le navi sono attese presso il porto. Sviluppano un modello di pianificazione che tiene conto di queste finestra. L'idea di robustezza nella pianificazione del berth è pensato come la capacità di restituire una soluzione ammissibile per ogni scenario possibile d'arrivo in cui ogni nave arriva all'interno della sua finestra temporale. Questa finestra si ottiene semplicemente spostando l'orario di arrivo di ogni nave con l'obiettivo di minimizzare il numero massimo di gru riservate. Si osservi che il BAP è un problema fondamentale per tutte le altre problematiche all'interno del terminal. Infine, per completezza, si osservi che il problema del BAP è stato immerso in modelli di simulazione ad eventi discreti della logistica portuale per valutare le prestazioni globali del terminal in termini di produttività del terminale e di tempo di soggiorno delle navi nel porto (Yun e Choi 1999 , Legato e Mazza 2001, Bielli et al. 2006).

In questo capitolo, si prenderà il terminal container nel porto di Gioia Tauro come riferimento e, dunque, la pratica dell'azienda terminalista sul problema in questione. La generazione del piano settimanale degli ormeggi (weekly plan) è supportato dal software Calema: un ambiente di simulazione progettato per riprodurre l'attracco delle navi, con un focus particolare sulla contesa del canale d'ingresso e la gestione dei punti di attracco (Canonaco et al 2007).

3.2. Il Framework di Simulazione Ottimizzazione

Per modellare e risolvere il problema di berth allocation proponiamo un frame work di simulazione-ottimizzazione illustrato nella Figura 3.1 che colma il divario naturale tra la soluzione tattica e quella operativa.

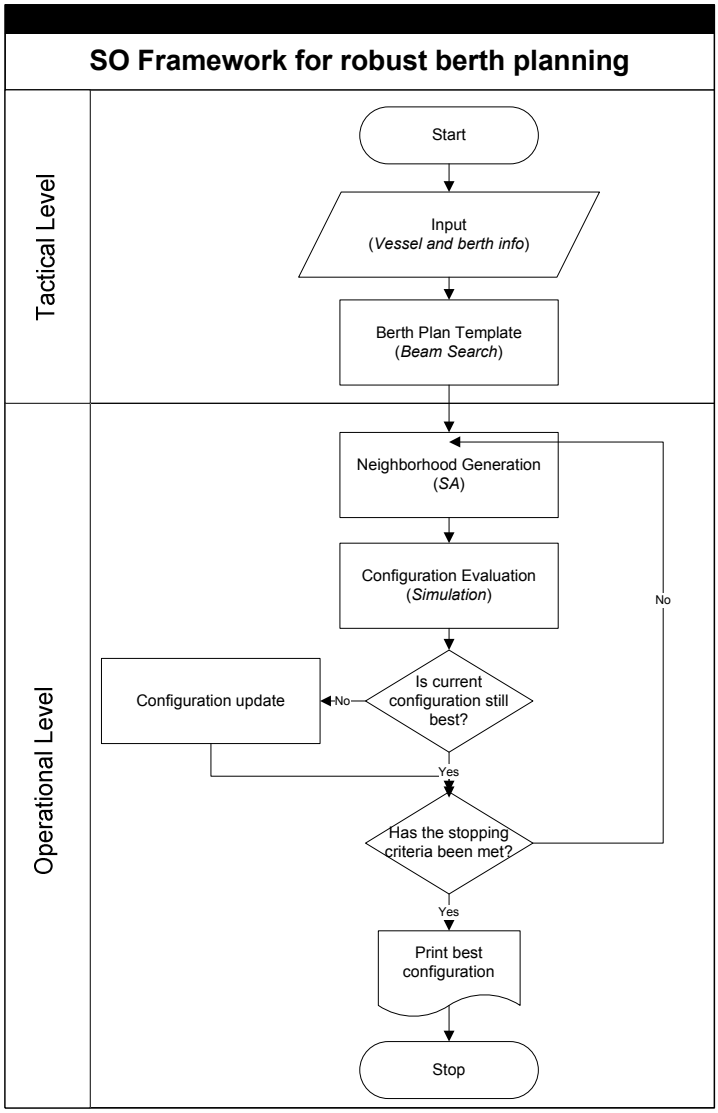


Figure 3.1 –l'ambiente di SO per un ormeggio robusto

Per quanto riguarda il livello tattico, l'ambiente cerca di produrre un modello di ormeggio mediante l'applicazione di un algoritmo costruttivo. Questo modello di ormeggio specifica la posizione nel tempo e nello spazio della singola nave, tenendo conto sia degli accordi contrattuali definiti tra la società terminalista e le compagnie di navigazione sia dei vincoli fisici imposti dalle navi (pescaggio, lunghezza, ecc).

Anche se questa soluzione è ottenuta in modo rapido ed accurato, essa non tiene in conto, in alcun modo, dell'aleatorietà contenuta nel sistema e causata dal processo di arrivo della nave e/o dal carico-scarico, nonché dell'effettiva disponibilità delle risorse necessarie per effettuare tali processi.

Le prestazioni di questa soluzione iniziale devono essere testate a livello operativo tenendo conto di una vasta gamma di condizioni supplementari che, a nostro avviso, offrono una misura della cosiddetta robustezza della soluzione trovata. In altre parole, nell'ambiente proposto valuta la bontà della soluzione trovata a livello tattico ed in seguito la confrontata con le soluzioni alternative sul piano operativo generate da un algoritmo euristico che esplorazione il vicinato. L'obiettivo generale dell'ambiente di SO consiste nel ridurre al minimo il tempo di attesa subito dalle navi a causa dei ritardi negli arrivi, tempi di servizio non deterministici e/o risorse non disponibili.

3.2.1. Il Livello Tattico

Costruire una soluzione, a livello tattico, per il BAP può essere differente da terminal a terminal. Presso il terminal container di Gioia Tauro, una costruzione preliminare del piano di ormeggio settimanale si basa sulla cosiddetta "*crane intensity*", cioè il numero medio di gru contemporaneamente attive su una nave ormeggiata. Fissato da un accordo formale con la compagnia di navigazione, la "*crane intensity*" permette di stimare la lunghezza della finestra di tempo da riservare per le navi che appartengono ad una compagnia specifica. Chiaramente, la durata della finestra è completamente determinata da ulteriori considerazioni: i) il numero (medio) di container da movimentare, ii) la posizione di ormeggio lungo la banchina e, infine iii) il numero di veicoli (cioè straddle carrier) dedicati al trasferimento dei container tra il punto di attracco e il punto di stoccaggio nel piazzale.
In generale, i modelli di programmazione matematica sono utilizzati per risolvere il BAP da un punto di vista tattico. Qui viene proposta un'estensione della formulazione proposto da (Kim e Moon 2003) al fine di includere alcune restrizioni dovute a vincoli fisici presenti lungo la banchina ((3.8) e (3.9)). La funzione obiettivo è differente: il nostro secondo termine rappresenta il costo di ritardare dell'attracco della nave, piuttosto che il costo risultante da una partenza ritardata della nave. Nella sua interezza, la funzione obiettivo rappresenta il costo aggiuntivo sostenuto dal gestore del terminale quando le navi sono attraccate in condizioni non ottimali (es. ritardo nell' ormeggio e lontano dalla sua posizione ideale ormeggio).
La notazione è la seguente:

N il numero di navi.

L la lunghezza della banchina.

p_i la locazione preferenziale della nave i lungo la banchina.

a_i il tempo atteso d'arrivo della nave i.

b_i il tempo necessario per il carico-scarico per la nave i.

S_i l'insieme dei segmenti della banchina lungo cui ormeggiare il vessel i.

$\inf_{si}$ la bitta più piccolo del segmento $s \in S_i$ dove il vessel i può essere ormeggiata.

$\sup_{si}$ la bitta più grande del segmento $s \in S_i$ dove la nave i può essere ormeggiata.

c_{1i} il coefficiente di penalità dovuto alla distanza dal punto ottimo di ormeggio p_i.

c_{2i} il coefficiente di penalità per una unità temporale di ritardo della nave i.

Variabili decisionali:

x_i la posizione lungo la banchina in cui ormeggiare il vessel i.

y_i l'istante temporale in cui il vessel i viene ormeggiato.

$z_{ij}^x = 1$ se la nave i è posizionato lungo la banchina alla sinistra della nave j, 0 altrimenti.

$z_{ij}^y = 1$ se la nave i è ormeggiato nel tempo prima della nave j, 0 altrimenti.

$\sigma_{si} = 1$ se la nave i è ormeggiato nel segmento s, 0 altrimenti.

$$\min \sum_{i=1}^{N} \left\{ c_{1i} \left(\alpha_i^+ + \alpha_i^- \right) + c_{2i} \left(y_i - a_i \right) \right\} \tag{3.1}$$

s.t.

$$x_i - p_i = \alpha_i^+ - \alpha_i^- \quad \text{per ogni } i \tag{3.2}$$

$$x_i + l_i \le L \quad \text{per ogni } i \tag{3.3}$$

$$x_i + l_i \le x_j + M\left(1 - z_{ij}^x\right) \quad \text{per ogni } i \text{ e } j,\ i \ne j \text{ ed un numero abbastanza grande M} \tag{3.4}$$

$$y_i + b_i \le y_j + M\left(1 - z_{ij}^y\right) \quad \text{per ogni } i \text{ e } j,\ i \ne j \text{ ed un numero abbastanza grande M} \tag{3.5}$$

$$z_{ij}^x + z_{ji}^x + z_{ij}^y + z_{ji}^y \ge 1 \quad \text{per ogni } i \text{ e } j,\ i \ne j \tag{3.6}$$

$$y_i \ge a_i \quad \text{per ogni } i \tag{3.7}$$

$-M+(\inf_{si}+M)\sigma_{si} \le x_i \le (\sup_{si}-l_i-M)\sigma_{si}+M$ for all i for all $s \in S_i$ (3.8)

$\sum_{s\in S_i} \sigma_{si} = 1$ per ogni i (3.9)

$\alpha_i^+, \alpha_i^-, x_i \ge 0$, Interi per ogni i (3.10)

$z_{ij}^x, z_{ij}^y, \sigma_{ij} \in \{0,1\}$ per ogni i e j, $i \ne j$ (3.11)

Si noti che, secondo le indicazioni ricevute dal gestore del terminale marittimo di Gioia Tauro, risulta $c_{2i} / c_{1i} = 10$ il che significa che ritardare una nave per 1 ora costa 10 volte di più che spostare una nave di 1 bitta dalla sua posizione preferenziale.

Alcuni requisiti pratici impediscono di applicare questo modello. Prima di tutto, i risolutori commerciali normalmente utilizzati per questo scopo possono risolvere solo casi di piccole dimensioni. In secondo luogo, qualunque sia la dimensione del problema, i risolutori commerciali non possono essere incorporati nelle applicazioni software già in uso presso il terminal marittimo

Di conseguenza, l'ambiente di SO in Figura 3.1 è stato progettato per utilizzare un'euristica che fornisce una soluzione tattica per il BAP. Ci rendiamo conto che in questo caso particolare per ottenere una soluzione rapida e precisa a livello tattico è più opportuno l'utilizzo di una metodologia già proposta in letteratura (un approccio costruttivo). E' stata scelta una euristica chiamata beam search (BS) (proposta da Wang e Lim 2007) che è stata naturalmente adattata al terminal portuale di interesse. Uno schema di massima di implementazione della BS è dato dallo pseudo-codice (Algoritmo 3.1) qui riportato, ma che verrà spiegato in seguito.

Algoritmo 3.1: Procedura BS

Initialization
- Set BS parameters
- Generate arrival sequence for incoming vessels
- Order vessels according to ETA (expected time of arrival)

Berth Definition and Assignment
- Extract vessel with smallest ETA and generate all possible configurations
- **For** every configuration
  - Define the n feasible berth segments $[(lower\ bollard - upper\ bollard), length, depth]$
  - Determine the ($\le 2n$) feasible berth configurations
  - Assign a feasible berth configuration to current vessel
- **End For**

Berth Definition and Assignment for the Next Δ Vessels
- **For** the $i = 1$ **to** Δ
  - Define the n feasible berth segments for vessel i $[(upper\ bollard - lower\ bollard), length, depth]$
  - Determine the ($\le 2n$) feasible berth configurations for vessel i
  - Assign a feasible berth configuration to vessel i
  - $i = i + 1$
- **End For**
- Insert all the r ($\le \Delta * i * 2n$) configurations in Set_R

Detailed Selection
- Evaluate objective function f_Δ for each configuration in Set_R
- Order solutions in Set_R
- Select a partial solution in Set_R according to the top m criterion

Exit Condition
- **If** all incoming vessels have been berthed **Then**
  - Return a final solution
- **Else**

Return to *Berth Definition and Assignment*

End If

Nell'algoritmo appena riportato, tutte le navi che arrivano al porto (nell'orizzonte temporale considerato) devono essere ormeggiate e nessuna può rimanere esclusa, per cui il problema è altamente combinatorio. Per fortuna, il numero limitato di possibili posti d'attracco riduce il numero di combinazioni possibili; infatti, una nave può essere ormeggiata lungo un determinato segmento solo se la dimensione del segmento corrisponde alla lunghezza della nave misurata in bitte, tenendo sempre conto delle obbligatorie bitte di sicurezza. Inoltre, come accennato in precedenza, la profondità dell'acqua nelle diverse zone di ormeggio (non occupate da alcuna nave) deve rispettare il pescaggio della nave.

Per meglio cogliere il funzionamento dell'algoritmo di allocazione, si consideri quanto segue. Se entrambe le condizioni sulla lunghezza della nave e sul pescaggio sono soddisfatte, allora una delle due seguenti situazioni si può manifestare: la nave può essere ormeggiata in uno dei due angoli (superiore o inferiore) di una zona libera come illustrato per la nave n°4 nei fotogrammi A e B di Figura 3.2, oppure, la nave può essere ormeggiata nelle stesse posizioni delle navi precedentemente ormeggiate una volta che queste hanno completato il loro processo di carico-scarico, come mostrato nel quadro C della Figura 3.2.

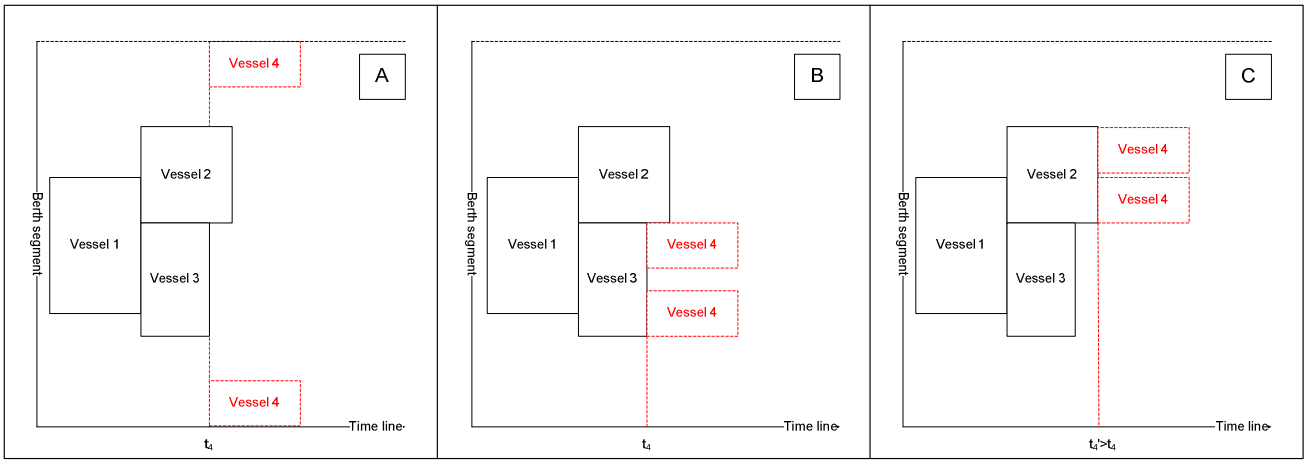


Figure 3.2 – Possibili posizioni di ormeggio per la vessel n°4

La stima della funzione obiettivo f_Δ associata ad una qualsiasi delle suddette decisioni di allocazione viene eseguita nella sezione *Detailed Selection* tenendo conto di due contributi. Un contributo è rappresentato dal costo della nave che sono già sono stati ormeggiati e, quindi, restituisce una valutazione immediata sulla base delle precedenti assegnazioni. L'altro contributo consiste in una stima dei costi necessari per l'attracco dei successivi Δ vessel ($i+1, i+2, \ldots, i+\Delta$), allocando ogni nave nella migliore posizione possibile (logica greedy). Grazie a tale stima, è possibile selezionare (secondo il criterio top m) le m migliori soluzioni parziali che successivamente verranno completate. L'intero meccanismo verrà iterato fintanto che tutte le navi in arrivo non saranno ormeggiate e, quindi, la procedura BS restituisce una soluzione finale per il BAP.

E' il caso di sottolineare in conclusione che ognuno dei nodi finali dell'albero di ricerca ha la stessa probabilità di essere selezionato come la soluzione finale restituita a livello tattico. Anche se un solo nodo sarà effettivamente scelto per questo scopo, gli altri saranno ugualmente utilizzato per studiare la robustezza della soluzione tattica. Questo consente all'intero framework di verificare se la miglior soluzione operativa è la stessa della soluzione tattica o coincide con uno qualsiasi dei nodi BS finale. In tal caso, la soluzione tattica si dice che sia robusta grazie alla sua capacità di preservare la migliore deviazione dall'ottimalità anche sotto l'incertezza del livello operativo. Al contrario, la soluzione tattica sarà un buon punto di partenza utilizzato dal SO basato su processo di ricerca per trovare la migliore soluzione operativa.

3.2.2. Il Livello Operativo

Il livello operativo dell'ambiente SO è concepito per testare la soluzione restituita dal livello tattico rispetto alla sua cosiddetta robustezza, con la quale s'intende la capacità della soluzione trovata a preservare, in tutti gli scenari, la migliore (peggiore) deviazione dall'ottimalità. In generale, la robustezza può essere valutata prendendo in considerazione a livello operativo la variabilità di un diverso numero di fattori tra i quali:

- I tempi di arrivo delle navi;
- disponibilità delle Gru di banchina;
- tempi di servizio.

Per quanto riguarda il primo punto, quando si pianifica nel giro di pochi giorni prima l'ETA della nave, i tempi di arrivo delle navi possono essere considerati deterministici di fatto. Inoltre, il terminale di riferimento si trova nel Mar Mediterraneo ed il maltempo non ha effetti significativi sul tempo d'arrivo delle navi. Il porto è normalmente colpito da una media di 20 giorni di maltempo l'anno. In conclusione, se si considerano scenari a breve termine (ad esempio simulazioni di 7 giorni), i ritardi legati ai tempi di arrivo delle navi non saranno riprodotti.
La gestione delle gru di banchina è un altro aspetto fondamentale a livello operativo. L'essenza di questa fase decisionale è illustrato in Figura 3.3, tale logica è incorporata nell'ambiente SO.
Quindi, in generale, al fine di effettuare le operazioni pianificate a livello tattico, il responsabile del terminale deve prima verificare la disponibilità complessiva delle gru di banchina e delle squadre di lavoro per turno; poi deve affrontare il problema di assegnare delle specifiche gru, per un determinato intervallo di tempo, a specifiche navi: questo concetto è noto come "assignment mode" (Meisel e Bierwirth, 2006).

Time horizon (h)	Deployment of cranes C1,…,C9 (from bollard n°1 to bollard n°22)																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1			C1				C2	C3					C4	C5				C6	C7	C8	C9	
2			C1				C2	C3					C4	C5				C6	C7	C8	C9	
3			C1				C2	C3					C4	C5				C6	C7	C8	C9	
4			C1				C2					C3	C4	C5				C6	C7	C8	C9	
5			C1				C2					C3	C4	C5				C6	C7	C8	C9	
6			C1				C2					C3	C4	C5				C6	C7	C8	C9	
7		C1					C2					C3	C4	C5	C6	C7				C8	C9	
8		C1					C2					C3	C4	C5		C6		C7		C8	C9	
9		C1					C2					C3	C4	C5		C6		C7		C8	C9	
10		C1					C2					C3	C4	C5				C6	C7	C8	C9	
11		C1					C2					C3	C4	C5				C6	C7	C8	C9	
12		C1					C2		C3	C4				C5				C6	C7	C8	C9	
13		C1					C2	C3	C4	C5								C6	C7	C8	C9	
14		C1					C2	C3	C4	C5								C6	C7	C8	C9	
15		C1					C2	C3	C4	C5								C6	C7	C8	C9	
16		C1		C2	C3		C4	C5	C6	C7										C8	C9	
17		C1	C2	C3			C4	C5	C6	C7										C8	C9	
18		C1	C2	C3	C4			C5	C6	C7										C8	C9	
19		C1	C2	C3	C4			C5	C6	C7										C8	C9	
20		C1	C2	C3	C4			C5	C6	C7										C8	C9	
21		C1	C2	C3	C4			C5	C6	C7										C8	C9	
22		C1	C2	C3	C4			C5	C6	C7										C8	C9	
23		C1	C2	C3	C4			C5	C6	C7										C8	C9	
24		C1	C2	C3	C4			C5	C6	C7										C8	C9	

Figure 3.3 – L'assegnamento delle gru di piazzale ed il posizionamento lungo la banchina

Infine, parlando del caricamento e scaricamento dei container, i tempi di servizio connessi rappresentano il reale e, il più delle volte, un'incontrollabile fonte d'incertezza. Il diverso numero e la varietà di risorse che vengono utilizzate in modo condiviso sia sul lato banchina che sul lato piazzale sono alla base delle difficoltà di sincronizzazione dei mezzi di *handling* che movimentano i container. Di conseguenza, i tempi reali che si verificano nel processo di carico-scarico sono quasi certamente non deterministici e dunque il simulatore gioca un ruolo importante a livello operativo, visto che la soluzione restituita dall'ottimizzazione effettuata a livello tattico non tiene conto di aspetti aleatori.
Alla procedura di SO viene chiesto di confrontare delle soluzioni, il che significa effettuare delle stime del valore atteso della funzione obiettivo per diverse soluzioni restituite dal modulo di simulazione ad ogni iterazione. Per garantire la

corretta selezione del campione di "migliore" media ad un livello fissato d'incertezza, un grande sforzo computazionale può essere richiesto in termini di numero di osservazioni per ogni replica su cui calcolare la media del campione: maggiore è la varianza della media campionaria e maggiore sarà il numero di osservazioni necessarie.
Le procedure più comune di ranking and selection (R & S) lavorano con la classica media standard del campione a disposizione (Kim e Nelson 2006), che, nel contesto di riferimento è calcolata attraverso la replica degli esperimenti di simulazione (dimensione del campione). In questa sede si propone un diverso modo di calcolare la media campionaria, ovvero di usare una logica a finestra mobile molto simile alla procedura di Welch per stimare la durata del transitorio nella simulazione. Così, per prima cosa, si devono organizzare *n* indipendenti osservazioni d'uscita in *b* gruppi e quindi si deve calcolare il valore medio delle *i-esima* osservazioni in questi gruppi a seconda della larghezza w della finestra mobile. Sia Y_{ji} l'osservazione *i-esima* all'interno del gruppo b, allora:

$$\overline{Y}_i = \frac{1}{b}\sum_{j=1}^{b} Y_{ji}. \tag{3.12}$$

Quindi l'insieme dei valori, $\overline{Y}_1, \overline{Y}_2, \ldots, \overline{Y}_m$, è usato per definire la media mobile $\overline{Y}_i(w)$ con una larghezza della finestra di w allora si ha:

$$\overline{Y}_i(w) = \frac{\sum_{s=-w}^{w} \overline{Y}_{i+s}}{2w+1} \qquad i = w+1,\ldots,m-w. \tag{3.13}$$

Si osservi che la media mobile dei vicini (cioè $\overline{Y}_i(w)$ e $\overline{Y}_{i+1}(w)$) è ancora uno stimatore della media delle osservazioni di uscita, però le medie mobili sono (positivamente) correlate a causa delle osservazioni comuni condivise. Di conseguenza, la varianza della media mobile $\overline{Y}_i(w)$ è più piccola della varianza associata allo stimatore standard (con $w = 0$) e, quindi, lo sforzo computazionale richiesto è inferiore. Si osservi che lo stimatore a media mobile può essere utilizzato in qualunque tipo di procedura di R&S, che si tratti di un processo ad una fase (Bechhofer 1954), o a due stadi (Rinott 1978) o n-stadi (Goldsman et al. 2002).
Adesso l'attenzione verrà focalizzata sulla parte di ottimizzazione della procedura di SO.
Si osservi che la componente principale a livello operativo si basa su una struttura di generazione del vicinato che permette di passare da una soluzione del BAP ad un'altra. In particolare, una nave viene selezionata dalla soluzione corrente e si prova a scambiare la posizione di quest'ultima con quella di un'altra nave. Lo scambio è considerato fattibile se la futura posizione della nave è conforme alla dimensione della nave ed inoltre si deve tenere conto che deve esistere una certa compatibilità tra i tempi di ormeggio delle due navi. Ovviamente, per le altre navi vicine l'attività di scambio può richiedere un "aggiustamento" della posizione lungo il berth.
Tutte le navi che soddisfano le condizioni di cui sopra sono inserite nell'insieme delle navi spostabili e le corrispondenti soluzioni BAP rappresentano nuove soluzioni vicine al piano di ormeggio corrente. Una (o più di una) soluzione sarà scelta da questo insieme e verrà valutata attraverso la simulazione. Al momento di decidere quale soluzione scegliere tra l'attuale e le nuove soluzioni BAP, verrà utilizzata una procedura di *simulated annealing* (SA) (Kim e Luna 2003).
Lo pseudo-codice che descrive questa parte dell'ambiente di SO è riportato di seguito.

Algoritmo 3.2: Procedura di ricerca e procedura di valutazione

Set SA parameters
While $T > threshold$ **Do**
Read current BAP solution
Select (primary) vessel to be swapped
Create set of (secondary) swappable vessels based on $(arrival\ \ time,\ \ departure\ \ time)$
Select a (secondary) vessel from the above set
Perform vessel swap and vessel adjustment
Simulate this new BAP configuration
If $f_{BAP}(new) < f_{BAP}(current)$ **Then**
$current = new$

Else

$current = new$ with probability $e^{-\Delta f / T}$

End if

Decrease T

If no improvements have occurred in the last n iterations **Then** exit

End while

Return $current$

3.3. Risultati numerici

In questa sezione l'ambiente di SO è stato testato considerando un orizzonte temporale di una settimana per i 3 chilometri di banchina del porto di Gioia Tauro, dove la profondità dell'acqua per i 3 segmenti principali di ormeggio varia da 11 a 16 metri. Le 65 istanze realistiche che abbiamo scelto possono essere divise in tre categorie: i) istanze di piccole dimensioni in cui sono previste un massimo di 20 navi, ii) istanze di medie dimensioni in cui il numero delle di imbarcazioni varia tra 30 e 40; iii) istanze di grandi dimensioni in cui il numero minimo di navi è 50. In ogni caso, gli arrivi delle navi si verificano con e senza finestre temporali, come accade per le navi oceaniche e le *feeders*, rispettivamente

Lo scopo degli esperimenti è duplice: da un lato, ci aspettiamo di dimostrare, che a causa delle condizioni precedentemente descritte di incertezza, la soluzione tattica restituita per il BAP richiede una raffinamento a livello operativo, dall'altro, si vuole verificare in che misura il framework di SO proposto può beneficiare di un minore sforzo computazionale dovuto all'uso di un *moving-average estimator* per le medie campionarie all'interno delle procedure R&S, piuttosto che la semplice media campionaria.

Per quanto riguarda il primo obiettivo, si può osservare che in tutti i test numerici riportato il valore della funzione obiettivo è fortemente influenzato dall'incertezza quando si passa dal livello tattico a quello operativo. La spiegazione pratica è semplice: un ritardo su uno specifico vessel può causare un effetto domino su le altre navi il cui ormeggio è stato pianificato nella stessa posizione della nave in ritardo.

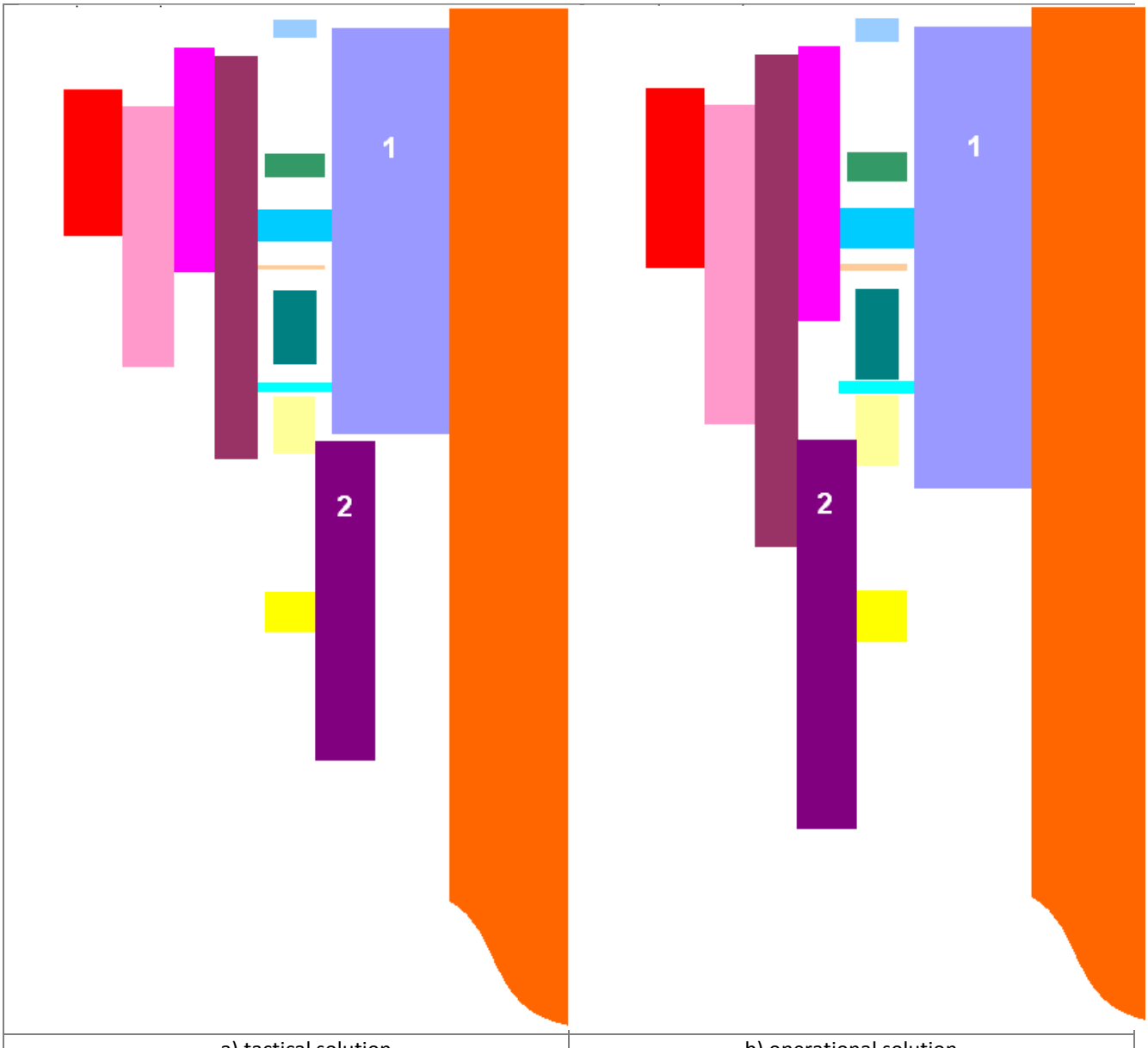
Risultati numerici per le istanze di piccole dimensioni sono illustrate nella Tabella 3.1, dove, in particolare, abbiamo contrassegnato con una stella le istanze per le quali la versione 10,1 di CPLEX ha trovato la soluzione ottimale in 300.

		Tactical solution			**Operational solution**	
Instance #	**Vessels #**	**OF**	**OF by Sim**	**Time (s)**	**OF by Sim**	**Time (s)**
*1	13	10.40	23.13	3.69	17.02	13.03
2	19	29.40	81.80	1.56	50.13	95.62
*3	13	16.20	59.86	1.16	22.68	8.10
*4	16	55.80	71.21	54.62	68.61	31.20
*5	13	10.50	11.62	1.27	11.62	10.36
6	17	19.90	53.07	0.51	29.86	28.48
7	17	77.90	96.60	0.70	92.07	26.94
8	20	16.90	116.84	1.22	35.16	70.41
9	19	88.30	98.03	1.57	97.77	47.97
*10	15	9.20	29.88	3.69	15.19	18.52
*11	17	259.20	310.08	46.85	302.01	20.42
*12	12	16.40	32.28	5.66	26.29	12.78
13	19	23.60	42.32	1.24	33.57	44.69
*14	18	175.30	231.33	261.94	207.36	32.66
*15	16	40.60	149.01	49.25	67.08	25.56

16	18	20.90	45.70	1.02	27.42	36.41
*17	16	13.90	37.19	36.31	17.80	20.11
18	18	69.80	144.03	0.97	88.02	43.01
*19	18	105.60	150.14	231.43	128.28	62.71
20	20	78.50	112.88	0.79	96.26	69.79
21	20	78.50	121.47	0.89	102.69	37.23
*22	15	10.80	31.29	10.19	22.34	19.60
*23	19	11.50	54.90	58.95	36.75	37.50
*24	15	8.50	29.20	6.59	24.39	16.16
*25	16	13.30	29.18	224.99	26.68	26.39

Tabella 3.1 – Risultati numerici per istanze di piccole dimensioni

Mentre i tempi di calcolo sono dello stesso ordine (≈ 40s), il valore medio della funzione obiettivo restituito per la soluzione tattica è 50,44 contro il valore medio della funzione obiettivo restituita per la soluzione operativa, 57,81. Il valore della funzione obiettivo restituita per la soluzione operativa supera il valore della funzione obiettivo restituita per il livello tattico 22 volte su 25, con un miglioramento medio del 27,85%.



a) tactical solution | b) operational solution

Figura 3.5 – Confronto tra la soluzione tattica e quella operativa del weekly plan

Per una maggiore comprensione di questi risultati, verrà illustrata la soluzione dell'istanza n°10 della Tabella 3.1. Il weekly plan è cambiato a livello operativo grazie ai tempi di servizio inflazionati, ciò è legato all'incertezza (Figura 3.5.b). Questa inflazione è particolarmente evidente per il tempo di servizio della nave n°1 che si andrebbe a sovrapporre con quella della nave n°2 e, quindi, ciò causerebbe un ritardo nella seconda nave. Pertanto, la nave n°2 è stato fatta traslare a sinistra lungo la banchina.

Per le 25 istanze medie nella Tabella 3.2, i tempi di elaborazione delle soluzioni tattiche e operative mostra una variazione più o meno regolare, mentre i loro valori medi differiscono notevolmente (9.14s e 551.52s, rispettivamente). Ancora una volta, il valore medio della funzione obiettivo restituito per la soluzione tattica è inferiore al valore medio della funzione obiettivo restituito per le soluzioni operative (120,89 <149,65). Il valore della funzione obiettivo ritornata per le soluzioni operative supera il valore della funzione obiettivo restituito per la soluzione tattica 25 volte su 25, con un miglioramento medio del 17,06%.

		Tactical solution			**Operational solution**	
Instance #	**Vessels #**	**OF**	**OF by Sim**	**Time (s)**	**OF by Sim**	**Time (s)**
26	34	36.00	43.81	6.61	38.76	340.09
27	40	81.60	150.19	9.33	126.39	574.00
28	36	232.70	271.55	9.30	266.96	466.91
29	33	37.70	68.18	4.22	40.71	321.25
30	37	143.70	217.24	6.99	184.60	470.01
31	40	43.00	79.84	12.41	63.03	759.19
32	33	60.30	69.86	5.79	67.85	309.19
33	39	43.40	71.20	5.26	49.81	494.33
34	37	37.40	79.74	6.99	56.56	602.45
35	38	160.30	222.93	10.44	202.38	878.47
36	34	40.10	60.44	8.87	57.50	391.21
37	40	389.20	478.30	13.18	464.50	731.14
38	36	180.20	238.67	4.84	224.27	278.78
39	38	51.50	93.64	9.53	78.83	457.12
40	36	36.20	58.46	8.21	58.46	350.26
41	35	274.60	401.24	7.72	345.72	422.33
42	38	171.50	244.32	20.11	221.57	715.28
43	38	36.90	75.86	6.93	44.22	790.89
44	39	173.00	218.99	8.24	207.07	445.83
45	39	51.00	80.69	14.70	65.78	759.14
46	34	210.50	262.18	7.01	246.69	418.23
47	38	260.00	371.76	11.70	238.77	477.93
48	39	46.60	218.52	14.24	111.73	1414.74
49	34	122.30	166.63	6.86	149.35	452.35
50	37	102.50	150.02	9.02	129.70	466.89

Tabella 3.2 – Risultati Numerici per istanze di medie dimensioni

Per le 15 istanze di grandi dimensioni nella Tabella 3.3, i tempi medi di calcolo sono estremamente diversi: 33.98s per le soluzioni tattiche contro 1903.98s per le soluzioni operative. Questa differenza è presente nei valori medi della funzione obiettivo restituita per la soluzioni tattiche e le soluzioni operative e (915,83 e 1248,39, rispettivamente. Il valore della funzione obiettivo restituita dalla soluzione operativa supera sempre il valore della funzione obiettivo restituito per la soluzione tattica, con un miglioramento medio del 39,06%.

		Tactical solution			Operational solution	
Instance #	**Vessels #**	**OF**	**OF by Sim**	**Time (s)**	**OF by Sim**	**Time (s)**
51	55	657.00	2050.22	41.42	1217.20	1913.19
52	53	671.00	2060.19	23.16	1100.33	1716.33
53	53	1831.70	2478.99	39.46	2100.99	1429.45
54	55	891.90	3207.53	30.37	1395.22	2158.84
55	54	1189.70	2478.59	30.72	2190.12	1786.32
56	51	523.80	1175.77	28.13	963.60	1252.77
57	50	314.00	1887.23	27.77	1167.77	1338.00
58	50	283.90	1984.75	27.32	849.24	1341.18
59	51	1966.10	2826.04	34.41	2581.92	1425.89
60	58	1201.80	3026.21	33.71	1691.36	2023.14
61	66	1530.40	2311.52	59.09	1726.31	4003.69
62	57	630.40	3645.66	36.45	1597.67	1977.74
63	58	386.60	2758.97	37.75	1125.62	2145.85
64	50	284.90	3414.21	20.23	939.27	1347.07
65	57	1374.20	5221.53	39.73	3343.12	2700.18

Tabella 3.3 – Risultati Numerici per le istanze di grandi dimensioni

Per quanto riguarda il secondo obiettivo degli esperimenti numerici, che consiste nell'indagare l'effetto della procedure *moving-average* nella R&S, piuttosto che lo stimatore standard per la media campionaria, abbiamo organizzato n osservazioni simulate in gruppi *b*, ciascuno di dimensione *m*, e poi abbiamo calcolato il valore medio dell'*ith* osservazione con larghezza della finestra mobile pari a *w*. Ci siamo quindi concentrati sui risultati computazionali restituiti quando si aumenta il valore di *w* da 1 a 5.

width of moving window (w)	**estimator/ instance #**	**#1**	**#2**	**#3**	**#4**	**#5**
1	MA	110	283	459	195	145
	S	828	3771	6835	2571	1952
2	MA	104	192	233	133	117
	S	847	3840	7026	2448	1952
3	MA	104	142	195	119	103
	S	834	3836	7025	2544	1964
4	MA	100	139	184	119	101
	S	832	3925	7124	2641	1963
5	MA	100	118	141	105	100
	S	829	3765	7107	2507	1975
indifference zone parameter		$\delta = 1‰$				

Tabella 3.4 – Numero medio di run di simulazione richiesti dai differenti stimatori della media campionaria nella procedura di R&S.

Nella Tabella 3.4 per cinque istanze selezionate abbiamo calcolato il numero di osservazioni aggiuntive (*runs*) richiesti sia dalla media mobile (MA), sia della semplice media campionaria Negli esempi sopra riportati, entrambi gli stimatori sono stati utilizzati per determinare il numero di *run* aggiuntivi richiesti con una probabilità fissa a di corretta selezione $1-\alpha = 0.95$ e $\delta = 1$ ‰. A questo proposito, segnaliamo l'efficacia dello stimatore MA in quanto riduce la dimensione del campione da intollerabile a tollerabile, soprattutto se si considera il numero di run cumulati su tutte le soluzioni da paragonare (si consideri che in queste cinque istanze il vicinato può raggiungere al più il valore 100). Inoltre, la MA offre anche un crescente effetto di riduzione della varianza all'aumentare della larghezza della finestra (*w*). Nella Tabella 3.4, questo può essere apprezzato leggendo i valori MA verticali per ogni istanza.

Gestione Combinata delle scorte e dell'instradamento dei veicoli

La logistica distributiva ha da sempre catturato l'interesse della comunità scientifica della Ricerca Operativa attraverso lo studio e l'analisi di problemi legati all'instradamento dei veicoli (problemi di *routing*). La ragione di questo profondo e costante interesse sta nelle numerose applicazioni che da sempre sono state collegate a questa classe di problemi, nonché allo studio teorico di questi problemi, il più classico dei quali, il problema del commesso viaggiatore (TSP), è uno dei problemi più noti e studiati nell'ambito dell'ottimizzazione combinatoria. La letteratura relativa a problemi di *routing* è quindi cospicua: i primi lavori in merito al problema del commesso viaggiatore sono apparsi negli anni '50 e da allora il volume degli articoli apparsi sul problema e sulle sue numerose varianti è andato sempre crescendo. La letteratura scientifica classica dedicata alle applicazioni della Ricerca Operativa, così come la maggior parte della letteratura riguardante problemi classici di ottimizzazione combinatoria, si è concentrata sullo studio di tali problemi in condizioni "statiche": tutti i dati relativi al problema sono assunti noti a priori rispetto alla soluzione dello stesso. In altri termini, è implicitamente assunto che tutte le informazioni sull'istanza siano note quando l'ottimizzazione è eseguita e che la soluzione ottenuta non debba essere modificata successivamente. Con l'avvento dell'era dell'Information Technology si è assistito ad un mutamento profondo dello scenario economico di riferimento. La disponibilità di nuove tecnologie ha portato innovazione nei modelli di business e ha permesso lo sviluppo negli ultimi anni dei cosiddetti problemi di routing dinamico, problemi di *routing* con profitti e problemi integrati di *routing* all'interno della *supply chain*. Uno degli effetti più evidenti ed importanti legato alla disponibilità delle nuove tecnologie dell'informazione è il drastico miglioramento della comunicazione intra-aziendale (tra nodi logistici diversi e geograficamente sparsi facenti parte di una medesima azienda) e inter-aziendale (tra aziende diverse facenti parte di una medesima *supply-chain*). Questo ha creato nuove opportunità per il miglioramento della gestione integrata dei nodi logistici e dell'intera *supply chain*. E' infatti noto come una gestione separata delle funzionalità di una rete logistica comporti notevoli costi aggiuntivi. L'integrazione delle diverse parti della *supply chain* risulta necessaria per una gestione ottimale, e indubbiamente gli strumenti di comunicazione attuale favoriscono e danno una notevole spinta al processo di integrazione. Nascono quindi nuove forme di relazione all'interno della supply chain. Una di queste è il cosiddetto paradigma VMI (*Vendor-Managed Inventory*). Il paradigma *Vendor-Managed* (VMI) rappresenta una nuova forma di integrazione fra funzioni logistiche diverse. In generale, in un sistema logistico che utilizza il VMI, un fornitore effettua il monitoraggio del livello di inventario dei suoi clienti e stabilisce le politiche di rifornimento. Rispetto alle politiche convenzionali, nelle quali il fornitore riceve gli ordini dai clienti ed organizza le consegne in relazione alle scelte operate dagli stessi clienti, in un sistema VMI il fornitore espleta un ruolo decisionale preminente assumendo il controllo dell'inventario dei clienti e decidendo:

a) quando rifornire i clienti,
b) di quanto rifornire i clienti e
c) in che modo rifornire i clienti.

Il vantaggio principale offerto da questo modo di operare risiede nella eliminazione delle diseconomie di scala, che incidono in modo significativo sull'efficienza del funzionamento della catena logistica. Questo trasferimento di funzioni dai clienti al fornitore permette di migliorare il coordinamento della catena logistica nella fase di organizzazione degli ordini e di schedulazione delle consegne. Nel confronto con i sistemi tradizionali basati sulle politiche *customer-oriented*, il VMI presenta i seguenti punti di forza:

- libertà del fornitore nella fase di pianificazione delle consegne;
- maggiore flessibilità nella schedulazione temporale delle consegne, al fine di garantire un coordinamento ottimale fra mantenimento dell'inventario e costi della distribuzione;
- possibilità di operare sull'anticipo o posticipo delle consegne in modo da ridurre i costi di trasporto;
- riduzione delle risorse necessarie nella fase di monitoraggio dei livelli di inventario dei clienti, e nella fase di evasione degli ordini;
- maggiore capacità di risposta alle esigenze dei clienti evitando l'assenza di prodotto nei magazzini periferici.

In questo contesto si inserisco i problemi di *Inventory Routing* (IRPs), che rappresentano una particolare implementazione del processo di integrazione nell'ambito del paradigma VMI. Nei problemi di *Inventory Routing* le decisioni combinate sulla gestione delle scorte e l'instradamento dei veicoli per la consegna dei prodotti ai clienti vengono assunte in modo integrato e simultaneo, evitando che si verifichino situazioni di assenza di prodotto nei magazzini dei clienti, e facendo in modo che le quantità consegnate siano compatibili con le capacità di stoccaggio dei magazzini e con le capacità dei veicoli disponibili per le consegne. L'obiettivo è minimizzare il costo complessivo di inventario e quello di trasporto su un prefissato orizzonte temporale. Questa classe di problemi presenta un livello di difficoltà maggiore rispetto ai classici problemi di instradamento, poiché integra le scelte connesse alla gestione delle scorte con quelle relative alla definizione della schedulazione migliore con cui rifornire un insieme di clienti, comunque distribuiti in un'area geografica limitata. La gestione delle scorte interviene nel momento in cui tutti i clienti consumano prodotti con un tasso fisso o variabile in un intervallo temporale limitato, e devono gestire le quantità di prodotto di cui essere riforniti in relazione alla capacità di stoccaggio dei loro magazzini. Questa considerazione naturalmente porta all'introduzione della dimensione temporale nell'ambito dell'instradamento classico, dove la domanda dei clienti è conosciuta a priori. Nella letteratura scientifica sono state definite diverse varianti del problema di *Inventory Routing* e, per ciascuna di esse, sono stati proposti numerosi approcci di soluzione. Il problema classico di *Inventory Routing* è definito su una rete logistica rappresentata da un grafo non orientato $G = (V, E)$, in cui $V = \{0,1, \dots, n\}$ è l'insieme dei vertici e $E = \{(i,j) \in V \times V : i < j\}$ è l'insieme degli spigoli. Il vertice 0 rappresenta il fornitore, nel quale una flotta K di veicoli omogenei con capacità Q è disponibile per le consegne. I vertici $i \in V\backslash\{0\}$ rappresentano i clienti. Ad ogni spigolo $(i,j) \in E$ è associato un costo (lunghezza) non negativo c_{ij} ed un tempo di attraversamento t_{ij}. Il consumo del prodotto in un cliente è rappresentato da q_i^t o μ_i^t, a seconda che l'orizzonte di pianificazione T sia un insieme discreto o un intervallo continuo. In particolare, se $T = \{1, \dots, H\}$, q_i^t è il consumo del prodotto nel periodo $t \in T$ nel cliente i; qualora $T = [1, H]$, μ_i^t rappresenta invece il tasso di consumo del prodotto del cliente i in t. Al fornitore, la quantità di prodotto disponibile nel periodo $t \in T$ è rappresentata da p_0^t o, più semplicemente, p^t. I_0^0 rappresenta il livello iniziale di inventario presso il fornitore, mentre I_i^0 è il livello di inventario iniziale dell'i-esimo cliente. I livelli di inventario del fornitore e del cliente i, all'inizio o alla fine del periodo $t \in T$, sono denotati rispettivamente con I_0^t e I_i^t. I livelli massimo di inventario nel fornitore e nel cliente i sono indicati rispettivamente con U_0 e U_i . infine, i costi di stoccaggio per unità di prodotto nel fornitore e nel cliente i sono rappresentati da h_0 e h_i. La sola componente d'instradamento rende il problema di *Inventory Routing* difficile da risolvere. Dal punto di vista computazionale un problema di *Inventory Routing* è NP-hard poiché esso può essere polinomialmente ridotto ad un TSP, ponendo nel problema originario l'orizzonte temporale pari a uno, azzerando tutti i costi d'inventario, assumendo infinita la capacità dei veicoli, ed imponendo che tutti i rivenditori siano serviti (visitati). L'esempio illustrato di seguito serve a comprendere la complessità nel risolvere una istanza anche piccola del problema di *Inventory Routing*.

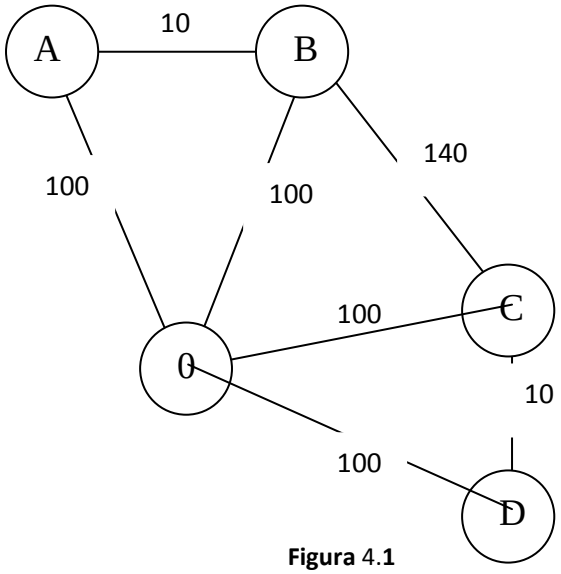


Figura 4.**1**

Si fa riferimento ad una piccola istanza del problema descritta da Bell et al. (1983) e definita sul grafo rappresentato nella Figura 4.1, nel quale sono riportati i tempi di attraversamento degli spigoli. Gli indici dei clienti sono rappresentati dall'insieme $I = \{A, B, C, D\}$. La Tabella 4.1 mostra il livello massimo di inventario U_i ed il consumo periodico q_i di ogni cliente $i \in I$. Si suppone che $I_i^0 = U_i$, e che il fornitore disponga di una flotta con un numero illimitato di veicoli di capacità $Q = 5000$. Si ipotizza, inoltre, che la disponibilità di prodotto nel fornitore sia illimitata ($U_0 = \infty$), e che la gestione dell'inventario abbia un costo irrilevante rispetto al costo della distribuzione. L'obiettivo consiste nella definizione della migliore politica di rifornimento dei clienti, in modo da evitare assenza di prodotto nei magazzini dei clienti medesimi, rispettando i vincoli sui livelli massimi di inventario e sulla capacità dei veicoli. La periodicità della politica implica che i livelli di inventario dei clienti siano uguali all'inizio ed alla fine dell'orizzonte temporale di pianificazione. Ciò permette di assumere che tale orizzonte sia infinito. Esistono 15 modi diversi per rifornire i clienti a prescindere dalle quantità da spedire (Tabella 4.2), ed un numero infinito di combinazioni relative alle quantità di prodotto che possono essere consegnate ai clienti.

Cliente (*i*)	U_i	q_i
A	5000	1000
B	3000	3000
C	2000	2000
D	4000	1500

Tabella 4.1. Livello massimo di inventario U_i e consumo periodico q_i di ogni cliente i nell'esempio di IRP

Modo di rifornimento	*Clienti serviti*
1	*A*, *B*, *C*, *D*
2	*A*, *B*, *C*
3	*A*, *B*, *D*
4	*A*, *C*, *D*
5	*B*, *C*, *D*
6	*A*, *B*
7	*A*, *C*
8	*A*, *D*
9	*B*, *C*
10	*B*, *D*
11	*C*, *D*

12	*A*
13	*B*
14	*C*
15	*D*

Tabella 4.2. Possibili modalità di rifornimento per i quattro clienti dell'esempio di IRP

Una semplice e naturale pianificazione della distribuzione giornaliera è la seguente:

- Giorno t:
  - $V_A = 1000$ unità di prodotto al cliente A e $V_B = 3000$ unità di prodotto al cliente B, con un costo di trasporto corrispondente a 210 miglia;
  - $V_C = 2000$ unità di prodotto al cliente C e $V_D = 1500$ unità di prodotto al cliente D, con un costo di trasporto corrispondente a 210 miglia.

Il costo giornaliero complessivo è pari a 420. La soluzione ottima prevede, invece, consegne differenti in giorni successivi:

- Giorno t:
  - $V_B = 3000$ unità di prodotto al cliente B e $V_C = 2000$ unità di prodotto al cliente C, con un costo di trasporto corrispondente a 340 miglia;
- Giorno $t + 1$:
  - $V_A = 2000$ unità di prodotto al cliente A e $V_B = 3000$ unità di prodotto al cliente B, con un costo di trasporto corrispondente a 210 miglia.
  - $V_C = 2000$ unità di prodotto al cliente C e $V_D = 3000$ unità di prodotto al cliente D, con un costo di trasporto corrispondente a 210 miglia.

Il costo medio giornaliero corrispondente è di 380. Adelman prima (2004), Song e Savelsbergh successivamente (2006), hanno definito un approccio risolutivo capace di determinare la soluzione ottima di questa istanza. Se si analizza in dettaglio la soluzione ottima dell'esempio di Bell si osserva che essa prevede l'attivazione di una rotta nel primo giorno di consegna, andando a servire i clienti che hanno il maggiore consumo periodico. Precisamente, vengono consegnate 3000 unità di prodotto al cliente B e 2000 unità di prodotto al cliente C, ma invece di farlo con due rotte diverse, come avverrebbe in una politica di distribuzione naturale, lo si fa con un'unica rotta. Nel giorno successivo vengono attivate due rotte: la prima rifornisce i clienti A e B, consegnando rispettivamente 2000 e 3000 unità di prodotto, mentre la seconda rotta serve i clienti C e D, consegnando rispettivamente 2000 e 3000 unità di prodotto. Come si può facilmente notare tale soluzione oltre ad essere ottima riesce ad utilizzare tutta la capacità del veicolo impiegato nelle singole rotte. Tale politica distributiva si reitera nei giorni successivi dell'orizzonte temporale. Il modello MIP (4.1)-(4.8), formulato di seguito, permette di provare l'ottimalità della soluzione dell'esempio di Bell:

$$\min \frac{1}{H} \sum_{k \in K} \sum_{t \in T} c_k y_k^t \tag{4.1}$$

$$\sum_{i \in I} x_{ik}^t \leq Q y_k^t \quad \forall k \in K \;\; \forall t \in T \tag{4.2}$$

$$x_{ik}^t \leq Q r_i^k \;\; \forall t \in T \;\; \forall k \in K \;\; \forall i \in I \tag{4.3}$$

$$I_i^t = I_i^0 + \sum_{s=0}^{t} \sum_{k \in K} x_{ik}^s - tq_i \quad \forall t \in T \quad \forall i \in I \tag{4.4}$$

$$I_i^t + q_i \leq C_i \quad \forall t \in T \quad \forall i \in I \tag{4.5}$$

$$I_i^t \geq 0 \quad \forall t \in T \quad \forall i \in I \tag{4.6}$$

$$x_{ik}^t \geq 0 \quad \forall t \in T \quad \forall i \in I \quad \forall k \in K \tag{4.7}$$

$$y_k^t \in \{0,1\} \quad \forall t \in T \quad \forall k \in K \tag{4.8}$$

In esso, H è l'ampiezza dell'orizzonte temporale finito e discreto $T = \{1, \dots, H\}$, mentre K è l'insieme di tutte le possibili rotte di consegna. Una rotta di consegna è l'insiemi di clienti visitati nella corrispondente *feasible delivery pattern*: $\delta(P_j) = \{i \in I : d_{ji} > 0\}$. Il concetto di *feasible delivery pattern* è stato introdotto da Savelsbergh e Song (2007). Un *feasible delivery pattern* $P_j = (d_{j1}, \dots, d_{jn}) \in \mathbb{R}_+^n$ è un vettore n-dimensionale delle quantità rifornite agli n clienti, tale che $\sum_{i \in I} d_{ji} \leq Q$, e $0 \leq d_{ji} \leq C_i$ per ogni $i \in I$; dove I è l'insieme di clienti da rifornite, Q è la capacità dei veicoli e C_i è la massima capacità di stoccaggio del cliente *i*-esimo. In pratica, ad ogni *feasible delivery pattern* corrisponde una ed una sola rotta di consegna. Nell'esempio di Bell le rotte di consegna sono: k_1={A}, k_2={B}, k_3={C}, k_4={D}, k_5={A, B}, k_6={B, C}, k_7={C, D}. Il costo di ogni singola rotta è rappresentato da c_k, e corrisponde al costo del tour ottimo del TSP definito sulla rotta k, mentre r_i^k è un parametro booleano che vale 1 se il cliente i è rifornito nella rotta k, 0 altrimenti. Si tratta di un parametro e non di una variabile dal momento che la risoluzione del modello (4.1)-(4.8) implica una fase di preprocessamento in cui sono determinati tutti i *feasible delivery patterns*, dai quali scaturiscono le rotte di consegna; pertanto si conosce già quali clienti devono essere visitati in ciascuna di esse. La funzione obbiettivo (4.1) minimizza il costo totale medio di trasporto, il vincolo (4.2) è un classico vincolo di zaino sulla capacità del veicolo. Il vincolo (4.3) impone che la quantità consegnata al cliente i con la rotta di consegna k al tempo t (x_{ik}^t), se tale cliente è servito da k (se $r_i^k = 1$), non ecceda la capacità del veicolo, il vincolo (4.4) calcola il livello d'inventario di ogni cliente in ogni periodo dell'orizzonte temporale, essendo q_i la quantità di prodotto consumata in ogni periodo dell'orizzonte temporale ed assunta costante su tutto T, il vincolo (4.5) garantisce che la capacità di stoccaggio in ogni cliente non venga superata. I vincoli (4.6) e (4.7) implicano che le variabili del modello siano non negative, mentre il vincolo (4.8) stabilisce che le variabili y_k^t siano booleane. La Figura 4.2 mostra le soluzioni ottime ottenuti in due differenti situazioni. La prima riga corrisponde al caso in cui il vincolo sulla capacità massima di stoccaggio C_i deve essere rispettato. La seconda riga corrisponde al caso in cui la capacità massima di stoccaggio nei clienti non è presente nel modello, vale a dire, $C_i = \infty$ per ogni $i \in I$. In quest'ultimo caso viene studiato l'effetto del rilassamento del vincolo (5). Nel primo caso il valore ottimo della funzione obiettivo è 380, mentre nel secondo caso il valore ottimo è 340. Questo si spiega perché il modello rilassato tende a consegnare una quantità maggiore di prodotto ai clienti, rispetto alla soluzione ottima, percorrendo un numero inferiori di miglia. In entrambe le situazioni, la capacità dei veicoli è pienamente utilizzata in tutte le rotte di consegna. Un'altra analisi interessante è quella finalizzata a capire cosa accade quando la capacità dei veicoli viene aumentata in maniera progressiva. I valori delle soluzioni ottime sono riportati nella Tabella 4.2 per $Q = \{5000, 10000, 15000, 20000\}$. Nel caso in cui il vincolo (5) debba essere rispettato, il costo della soluzione ottima passa da 380 a 377,5. Questo accade per $Q \geq 7000$. Nel caso in cui il vincolo (4.5) venga rilassato, il costo della soluzione ottima passa da 340 a 105. Questo decremento si percepisce a partire da $Q \geq 16000$. Come previsto, l'impatto della capacità dei veicoli sul costo ottimo è notevolmente superiore nel secondo caso. Il rilassamento del vincolo (4.5) fa sì che la consegna del fabbisogno dei clienti su tutto l'orizzonte di pianificazione avvenga nei primi giorni, in modo da attivare il minore numero di rotte tale da fare fronte alle esigenze dei singoli clienti per tutto il periodo di riferimento, e quindi minimizzare il costo complessivo del trasporto.

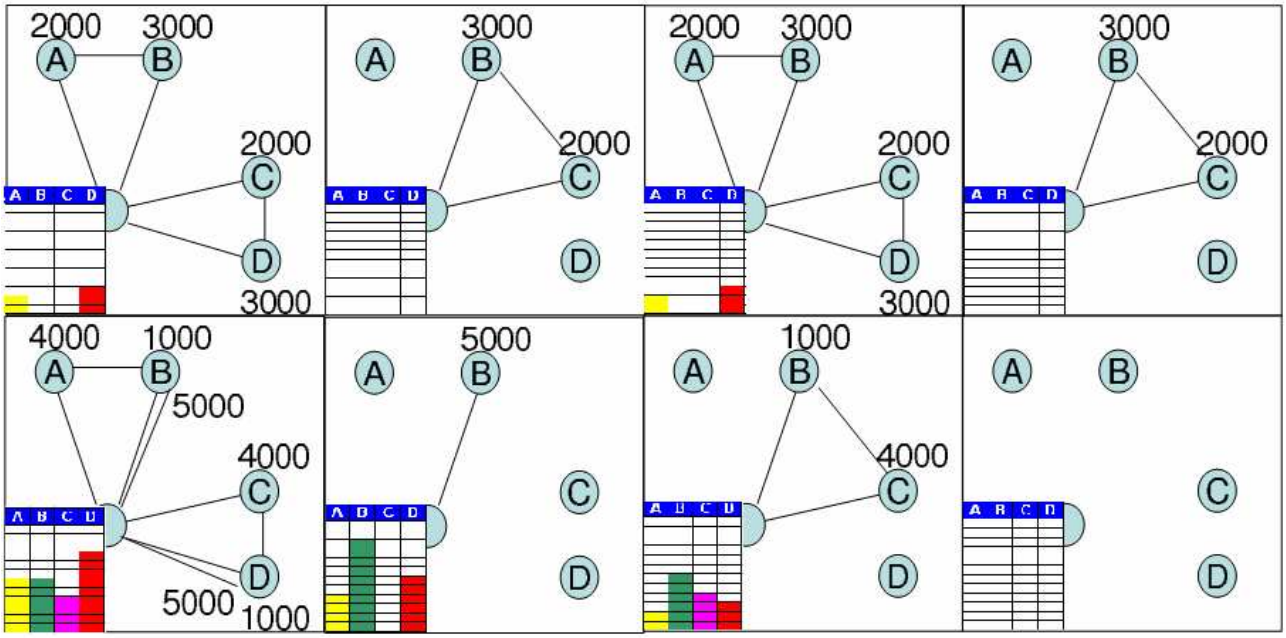


Figure 4.2: Minimizzazione dei Costi di trasporto. a) Con limite sulla massima capacità di stoccaggio: FO = 380. b) Senza alcun vincolo sul livello massimo di stoccaggio: FO = 340.

Q	C_i given	$C_i = +\infty$
5000	380.0	340.0
10000	377.5	190.0
15000	377.5	152.5
20000	377.5	105.0

Tabella 4.3:Costi ottimi di trasporto con o senza il vincolo (4.5), ed al variare della capacità del veicolo.

La Figura 4.3 mostra le soluzioni in cui ottenute con una funzione obiettivo in cui sono presenti differenti costi di stoccaggio. La prima riga mostra la soluzione nel caso in cui il costo di stoccaggio è piccolo, h_i=0,01 per tutti i clienti $i \in I$, mentre la seconda riga mostra la soluzione nel caso in cui il costo di magazzino è grande, h_i=0,1 per tutti i clienti $i \in I$. E interessante osservare che, l'incremento dei costi di magazzino determina un aumento della frequenza delle visite ai clienti nella soluzione ottima. Inoltre, la capacità del veicolo a disposizione non è più utilizzata completamente. Negli ultimi anni è emerso che i sistemi produttivi in grado di operare con minori quantitativi di scorte di magazzino beneficiano di un vantaggio competitivo. Per tenere conto di questi aspetti si è pensato di introdurre nella funzione obbiettivo dei modelli matematici di Inventory Routing un'ulteriore termine di costo relativo al costo di stoccaggio per unità di prodotto. La f.o. del modello (4.1)-(4.8) si trasforma pertanto nel modo seguente:

$$\min \frac{1}{H}\left(\sum_{i\in I}\sum_{t\in H} c_k y_k^t + \sum_{i\in I}\sum_{t\in T} h_i I_i^t\right)$$

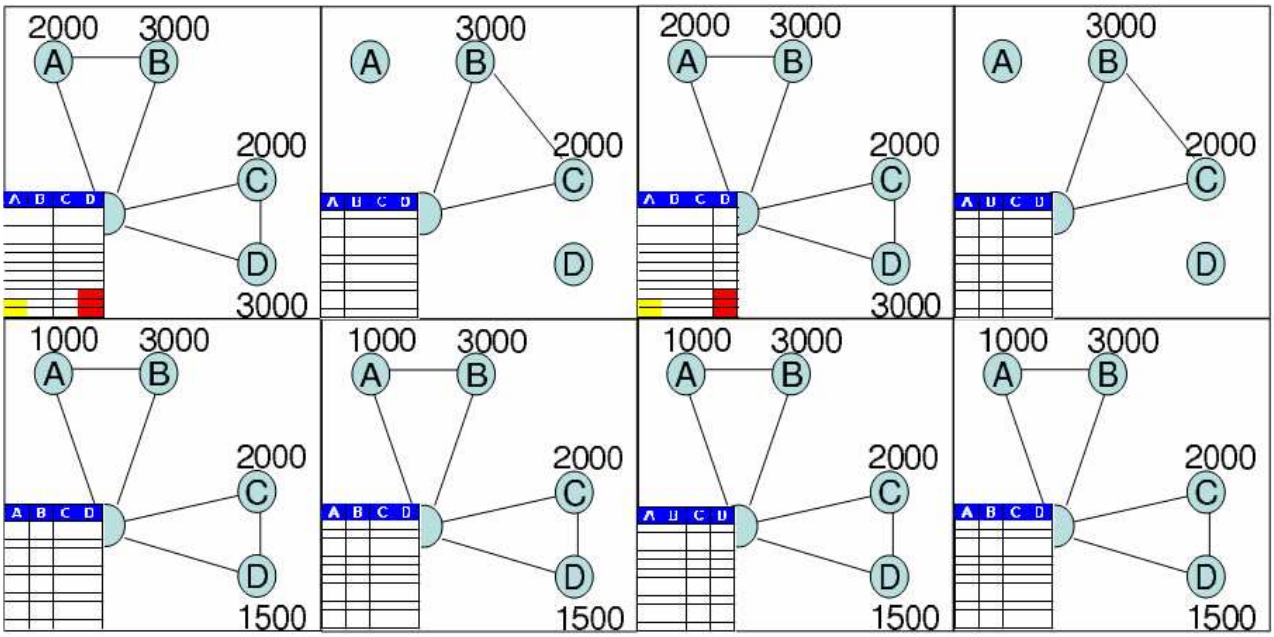


Figura 4.3: Minimizzazione dei costi di trasporto e di stoccaggio presso ogni cliente: a) Con h_i=0.01: fo=380 +12.5 = 392.5. b) con h_i=0.1: fo=420 + 0 = 420.

I primi ad introdurre i costi di stoccaggio in un modello di *Inventory Routing* a tempo discreto sono stati Speranza e Ukovich nel 1994. Gli effetti dovuti all'introduzione di tale termine nella f.o. dipendono dal valore che assume h_i. Infatti, se questi risultano essere bassi, la soluzioni non si discosta da quella ottima descritta in precedenza, mentre se sono alti il sistema tende a visitare ogni giorno i singoli clienti consegnando la quantità di prodotto che viene consumata quotidianamente dai clienti.

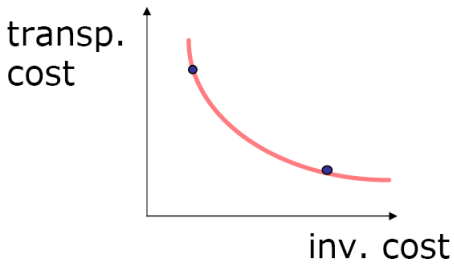


Figura 4.4: Relazione tra i costi di trasporto e i costi di stoccaggio

La minimizzazione dei due costi presenta degli effetti contrastanti poiché minimizzare i costi di stoccaggio significa visitare ogni giorno i clienti aumentando inevitabilmente i costi di trasporto; dall'altro lato minimizzare i costi di trasporto vuol dire consegnare ad ogni cliente la massima quantità di prodotto compatibile con la sua capacità di stoccaggio, in modo da dover visitare ogni cliente il minor numero di volte nell'orizzonte temporale. Sarà quindi compito del modello trovare il giusto compromesso tra i due costi presi in esame. Il modello (4.1)-(4.8), appena descritto, non può essere usato in casi reali per diversi motivi, il primo dei quali è che esso è computazionalmente difficile da risolvere. Inoltre, in esso è stata fatta implicitamente una ipotesi importante: l'istante di consegna del prodotto al cliente deve avvenire prima che inizi il consumo dello stesso. Un ulteriore difficoltà relativa alla risolvibilità del modello riguarda l'enumerazione di tutte le possibili rotte di consegna, che è di per se un problema combinatorio difficile. Ancora, il modello non prevede che i veicoli possano essere utilizzati per più di un viaggio in ogni periodo dell'orizzonte temporale, ed inoltre non viene distinto il momento preciso del periodo temporale, all'interno del quale il prodotto è realmente disponibile presso il cliente. In pratica si dice che il prodotto viene consegnato al cliente nell'intervallo $[t, t+1)$, senza specificare la frazione di t in cui la consegna ha luogo. I modelli discussi fin qui sono a tempo discreto, ma potrebbero essere non adatti a rappresentare alcune situazioni reali, nelle quali i prodotti sono

consumati in modo continuo nel tempo. In questi contesti l'istante temporale delle consegne e la capacità di stoccaggio disponibile giocano un ruolo fondamentale. Pertanto, la quantità massima di prodotto consegnabile dipende dall'istante di tempo in cui avviene la consegna. Se il tasso di consumo del prodotto e la capacità di stoccaggio variano in modo significativo tra i vari clienti, sarà inevitabile avere un insieme di rotte di consegna lunghe e brevi, quindi l'assunzione di attivare una singola rotta per veicolo in ogni periodo temporale non può essere più valida. Nei sistemi in cui il consumo dei prodotti avviene in modo continuo, la scelta del momento esatto in cui realizzare la consegna avviene all'interno di una finestra temporale; tanto più l'istante temporale della consegna risulta vicino al valore massimo della finestra temporale, tanto maggiore è la quantità di prodotto di cui è possibile rifornire il cliente, per via del tasso di consumo continuo. Dall'altro lato, le consegne non sono istantanee ma dipendono dalla quantità da consegnare; pertanto, più tardi arriva il veicolo nel cliente e minore risulta il tempo disponibile per consegnare la quantità trasportata. Questi due contrastanti aspetti rendono non così semplice, come potrebbe apparire, la determinazione della schedulazione ottima delle consegne ad una sequenza di clienti in una rotta. In questa direzione, un contributo significativo su come massimizzare la quantità di volume consegnata con una rotta, nell'ambito dei problemi di Inventory Routing, è quello dovuto a Campdell e Savelsbergh (2004).

4.1 La letteratura

Lo scopo di questo paragrafo è quello di offrire una descrizione sintetica dei contributi scientifici più significativi nell'ambito dei problemi di Inventory Routing. Esistono numerose survey sui problemi di Inventory Routing, alle quali è possibile rinviare per trovare informazioni più dettagliate. Di seguito un elenco di quelle più recenti in ordine cronologico: Federgruen e Simchi—Levi (1995), Campbell et al. (1998), Cordeau et al. (2007), Moin e Salhi (2007), Bertazzi et al. (2008), Andersson et al. (2010). Il criterio seguito nell'analisi della letteratura scientifica è stato quello di partire dai lavori pioneristici, definiti su orizzonti temporali molto limitati e con domanda deterministica dei clienti, per poi passare in rassegna i contributi più recenti relativi a problemi definiti su orizzonti temporali di medio e lungo periodo. Infine, si è deciso di elencare i lavori più significativi nell'ambito dei problemi di Inventory Routing con domanda stocastica e si è fatto riferimento a lavori apprezzabili relativi a problemi di Inventory Routing di grandi dimensioni. Il primo lavoro che analizza un problema di Inventory Routing su un orizzonte di pianificazione di un solo giorno è stata proposto da Beltrami nel 1974. Nel 1984, Assad et al. hanno studiato lo stesso problema proposto da Beltrami, applicandolo alla distribuzione di gas propano ad un insieme di clienti. Federgruen e Zipkin (1984) hanno preso in esame il problema tattico relativo alla gestione delle risorse ed alla pianificazione dei rifornimenti ad un insieme di clienti. L'obiettivo è stato quello di determinare una politica di rifornimento tale da ottimizzare le rotte di consegna. La domanda dei clienti è stata assunta stocastica ed i costi di stoccaggio sono stati ipotizzati non lineari. Inoltre, sono stati presi in considerazione anche i costi dovuti allo stock-out presso i clienti. Un problema di IR definito su un periodo di programmazione settimanale è stato proposto da Dror et al. (1985). Nel problema studiato da Dror, l'obiettivo è stato quello di definire una politica ottimale per il rifornimento settimanale di un insieme di clienti utilizzando veicoli con capacità limitata. E' stata assunto che il consumo giornaliero seguisse una legge di distribuzione continua di probabilità. Il problema è stato trattato con un modello di programmazione intera a due stadi per la selezione dei clienti da servire e per programmare i percorsi delle rotte di consegna. Dror e Ball (1987) hanno proposto una tecnica di riduzione di un problema di distribuzione multi-periodo ad un problema singolo periodo. L'obiettivo del lavoro è stato quello di definire dei sottoinsiemi di clienti da servire in un singolo periodo dell'orizzonte di pianificazione. In questo lavoro si trova una interessante definizione della scorta di sicurezza presso i clienti. Tutti i modelli e metodi menzionati in precedenza non sono in grado di catturare gli effetti di lungo termine delle decisioni tattiche nell'orizzonte di pianificazione. Il motivo di ciò è dovuto alla limitatezza dell'orizzonte di pianificazione entro il quale le decisioni tattiche sono valutate. I primi contributi significativi relativi al cosiddetto multi-day IRP sono stati proposti da Bard et al. (1998a), Bard et al. (1998b), Bard et al. (2002). Anily e Federgruen (1990, 1991) hanno studiato una strategia di Inventory caratterizzata da lunghi lead times, dell'ordine di mesi o anche anni tra le operazione di consegna. Nel loro lavoro hanno proposto una strategia integrata di rifornimento finalizzata a soddisfare la domanda deterministica dei clienti, ed a minimizzare il costo medio di stoccaggio e di routing dovuto alle consegne, lungo un

periodo di pianificazione esteso. Gallego e Simchi-Levi (1990) hanno studiato l'incidenza percentuale delle consegne dirette (una rotta che parte dal centro di distribuzione e serve un solo cliente) nel lungo periodo. Nel loro lavoro vengono esaminate diverse strategie sia per il rifornimento che per l'instradamento, e viene analizzata l'incidenza percentuale della dimensione del lotto minimo economico rispetto alla capacità del veicolo. Bramel e Simchi-Levi (1995) hanno concentrato i loro sforzi nella definizione di strategie efficienti per il rifornimento dei clienti, introducendo la cosiddetta politica delle Partizioni Fisse (Fixed Partition Policy, FP), che consiste nel suddividere il gruppo di clienti in base alla loro vicinanza geografica, in modo tale da formare delle aree geografiche servite separatamente ed indipendentemente dalle altre aree. Si assume che in un periodo dell'orizzonte di pianificazione tutti i clienti appartenenti ad una area siano visitati allorché un solo cliente della stessa area debba essere servito. In questo contesto, Chan et al. (1998) hanno studiato la strategia ottimale di trade-off fra la politica di gestione delle scorte e delle consegne, con l'obiettivo di minimizzare i costi di distribuzione totali su un orizzonte temporale infinito. Una interessante analisi del comportamento asintotico delle politiche di rifornimento cosiddette Fixed Partition e Zero Inventory Ordering è presente in questo lavoro. Un politica di rifornimento dei clienti di tipo Order-Up-to level su un orizzonte temporale finito è stata introdotta per la prima volta da Bertazzi et al. (2002). Un approccio di decomposizione a due fasi per risolvere un problema di Inventory Routing deterministico, nel quale gli aspetti legati alla variabilità temporale del tasso di consumo dei clienti sono trattati con uno schema di tipo rolling-horizon, è stato proposto da Campbell e Savelsbergh (2004). Tale lavoro si caratterizza per l'enfasi che è stata attribuita alla valutazione dell'impatto delle decisioni tattiche assunte nella fase di pianificazione sulle scelte operative prese nella fase di organizzazione delle rotte di consegna. Tali decisioni sono indotte da un modello di programmazione lineare intera che stabilisce: a) quando effettuare le consegne, b) il volume di cui rifornire i clienti e, c) quali rotte di consegna attivare su un orizzonte di pianificazione di lungo termine. A tale fase è seguita una fase di schedulazione delle consegne nelle singole rotte, nella quale si opera anche una re-ottimizzazione dei volumi consegnati. Nello stesso lavoro, le prestazioni delle strategie di IR sono state valutate mediante diversi indicatori statistici, fra i quali è stato considerato il rapporto tra il volume totale delle consegne e la distanza complessiva (espressa in miglia o chilometri percorsi) nell'orizzonte di pianificazione. Uno analisi approfondita di questa statistica si trova nel lavoro di Savelsbergh e Song (2004). Gaur e Fisher (2004) hanno proposto una applicazione di un modello di IR alle catene di supermercati. In particolare, il problema della distribuzione all'interno di una catena di supermercati è stato studiato come un problema di IR periodico, in cui lo scopo è stato quello di definire il migliore giorno in cui effettuare la consegna di un singolo prodotto da un venditore centrale ad una serie di negozi su un orizzonte di pianificazione settimanale, con l'obiettivo di minimizzare il costo dovuto alle rotte di consegna. Bertazzi et al. (2005) hanno studiato un sistema complesso di produzione-distribuzione, in cui un impianto produce diversi prodotti che sono distribuiti a un gruppo di rivenditori con una flotta di veicoli. Tale problema è stato studiato proponendo due diverse modalità di decomposizione in sotto-problemi più semplici da risolvere, e sviluppando procedure ottimali e/o euristiche per la risoluzione dei sotto-problemi stessi. Un algoritmo esatto di tipo branch and cut è stato proposto da Archetti et al. (2007) per risolvere istanze di piccole e medie dimensioni di problemi di Inventory Routing, ciascuno dei quali è caratterizzato da politiche diverse di rifornimento dei clienti. Savelsbergh e Song (2007, 2008) hanno studiato varianti particolari dei problemi classici di IR. In particolare, essi hanno definito il cosiddetto *Inventory Routing Problem with Continuous Moves* (IRPCM), in cui un insieme di clienti ed una serie di fornitori si trovano in un'area geografica tale che, le distanze reciproche sono così elevate da non potere garantire il servizio dei clienti con rotte di consegna che abbiano una durata temporale minore o uguale della durata del giorno lavorativo. Ciò implica che le consegne possono avere una durata di diversi giorni. La procedura risolutiva che è stata proposta consiste in un algoritmo euristico goloso di tipo randomico (RGH), a cui viene agganciata una procedura di ricerca locale basata su un modello MIP (*Mixed Integer Programming*). Archetti et al. (2009) hanno proposto una procedura euristica ibrida, che si basa su un algoritmo Tabu Search in cui viene utilizzato un modello MIP in alcune fasi della ricerca. Tale algoritmo è stato applicato ad un problema di IR, in cui un fornitore deve servire un insieme di clienti in un orizzonte temporale limitato. Hemmelmayr et al. (2010) hanno studiato un problema di IR in cui occorre rifornire di sangue una serie di ospedali, a partire da una banca del sangue. L'approccio risolutivo che è stato proposto si basa sull'integrazione di un modello di programmazione lineare intera in una procedura di ricerca locale basata sull'esplorazione di vicinati di ampiezza variabile. Una procedura di ricerca locale randomizzata è stata proposta da Benoist et al. (2010) per risolvere istanze reali di un problema di IR nel quale sono stati presi in considerazione vincoli operativi di diversa natura (consegne e

prelievi, finestre temporali, vincoli contrattuali degli autisti, ecc). La ricerca locale randomizzata è stata applicata direttamente (senza decomposizione) all'intero problema. Le prestazioni della procedura euristica di risoluzione sono state valutate sia nella fase di pianificazione (determinazione dei volumi di consegna) che nella fase di instradamento. Tali prestazioni sono state misurate attraverso la definizione di un indicatore statistico denominato *rapporto logistico*, che è il rapporto fra il costo complessivo di instradamento ed il volume totale consegnato nell'orizzonte di pianificazione. Uno studio a parte merita la letteratura dei problemi di Inventory Routing in cui il rifornimento riguarda classi di prodotti soggette a politiche di rifornimento diverse. In questo caso si parla di *multi-item IRP*. La letteratura sul multi-item-IRP è sicuramente meno copiosa di quella relativa ai problemi classici di IR. I lavori pioneristici sul muti-item IRP sono quelli di Viswanathan e Mathur (1997) e di Qu et al. (1999). Viswanathan e Mathur hanno studiato per primi l'integrazione del problema di instradamento dei veicoli con quello legato alla gestione delle scorte, nell'ambito di un sistema di distribuzione multi-prodotto con domanda deterministica. Il problema viene risolto con una euristica che sfrutta una politica di rifornimento cosiddetta *nidificata, stazionaria e congiunta*. Si tratta di un strategia nella quale gli intervalli di rifornimento dei diversi prodotti sono stati espressi sotto forma di potenza del due e/o multipli dell'ampiezza di un periodo di pianificazione base, e sono stati calcolati con la formula modificata del Lotto Economico di Riordino (Economic Order Quantity). Il limite di questo approccio è che, in alcune situazioni reali, non è realistico pensare di potere esprimere gli intervalli di rifornimento dei diversi prodotti in funzione di un intervallo base, dal momento che viene a cadere l'ipotesi di stazionarietà dei rifornimenti. Si pensi, per esempio, alla grande distribuzione in ambito agroalimentare. Qu et al. (1999) hanno studiato un sistema di raccolta di materiali di diversa natura, modellato come problema di IR con un magazzino centrale e con una flotta di veicoli di capacità illimitata. I clienti sono geograficamente distribuiti ed hanno domanda stocastica. E' stata proposta una politica di rifornimento periodica modificata, nella quale i volumi consegnati sono stati tali da saturare (Order-Up to level) la capacità di stoccaggio corrente, ed in cui ciascun periodo di rifornimento è stato calcolato come un multiplo intero di un periodo base. Anche in questo caso i limiti dell'approccio proposto risiedono nell'ipotesi di periodicità dei rifornimenti. Tale approccio ha determinato la progettazione di in un algoritmo euristico in cui il problema è stato decomposto in sottoproblemi di inventario e di instradamento. L'analisi della letteratura esistente si completa con una veloce disamina dei contributi principali nell'ambito dei problemi di Inventory Routing con domanda stocastica, meglio noti come problemi di Inventory Routing Stocastico (SIRP), e dei problemi di Inventory Routing di grandi dimensioni. Il primo studio di un problema di Inventory Routing Stocastico è stato quello proposto da Minkoff (1993). Kleywegt et al. (2002, 2004) hanno studiato il SIRP come un Problema Decisionale di Markov (MDP) su un orizzonte temporale infinito. Adelman (2003, 2004) ha proposto una tecnica euristica di valutazione della stocasticità della domanda attraverso una approssimazione dei costi futuri delle scelte correnti, espressi dai valori ottimi delle variabili duali di un modello di programma lineare. Recenti lavori sul SIRP sono stati proposti da Lejeune e Ruszczynski (2007) e da Hvattum et al. (2009). Un algoritmo euristico basato sul rollout è stato proposto da Bertazzi et al. (2011) per risolvere un problema di IR con domanda dei clienti distribuita secondo una variabile aleatoria discreta su un orizzonte temporale finito, e con una politica di rifornimento dei clienti del tipo Order-Up to level. Metodologie per risolvere problemi di IR deterministico e stocastico di grandi dimensioni si trovano nei lavori di Chen et al. (2008, 2009) e di Huang et al. (2009).

4.2 Contributi

Esistono ancora molti problemi di *Inventory Routing* che sicuramente meritano di essere investigati per le ricadute di natura applicativa che essi comportano. Un primo contributo del presente lavoro di tesi è stato quello di estendere metodologie ed approcci risolutivi consolidati a problemi di *Inventory Routing* deterministico multi-prodotto. Tale contributo è risultato essere non solo interessante dal punto di vista scientifico, ma foriero altresì di ricadute applicative in alcuni settori della logistica distributiva quali quello della grande distribuzione alimentare. Un secondo contributo è stato quello di proporre modelli matematici capaci di catturare e descrivere le peculiarità del problema di *Inventory Routing multi-item*. I modelli matematici proposti si prestano ad essere risolti con algoritmi esatti, anche se limitatamente ad istanze di piccole dimensioni. Pertanto, il terzo contributo è stato quello di progettare un approccio risolutivo capace di determinare buone soluzioni ammissibili per istanze di medie e grandi dimensioni del problema. In particolare, è stato definito un approccio di decomposizione del problema in due fasi, all'interno delle quali è stato implementato uno schema euristico risolutivo in grado di coniugare efficienza computazionale e qualità della

soluzione fornita. Lo studio del modello di distribuzione preso in esame, secondo il paradigma del VMI, fa riferimento ad una catena logistica di secondo grado (*two echelon*), caratterizzata da un unico fornitore (*one vendor o supplier*) che dispone di P prodotti (*multi product*) da rifornire a M clienti (*multiple destinations*), attraverso una flotta di veicoli con capacità omogenea, su un orizzonte di pianificazione finito e discreto. L'obiettivo è quello classico di un problema di *Inventory Routing*: definire le strategie ottimali di rifornimento dei clienti per ogni classe o tipologia di prodotto consegnato, in maniera tale da evitare lo stock-out e minimizzare il costo totale di instradamento delle rotte di consegna. Nel seguito si farà riferimento a questo problema come *Multi-product* o *Multi-item* IRP (MIRP). Come già premesso, l'approccio risolutivo proposto decompone il problema in due fasi: una fase tattica nella quale si decide a) quali clienti servire, quando servirli e di quali prodotti rifornirli e, b) il volume di ciascun prodotto di cui rifornire i clienti; una fase operativa nella quale, conoscendo i periodi dell'orizzonte di pianificazione in cui servire un insieme di clienti, di quali prodotti rifornirli ed in che quantità, si decide il numero delle rotte necessarie per le consegne e, per ciascuna rotta, si definisce l'ordine con cui servire i clienti nella rotta. In questa fase le scelte operate sono condizionate da vincoli di natura operativa, quali la capacità dei veicoli e le finestre temporali entro le quali occorre effettuare le consegne ai clienti. Si assume che tali finestre siano uniche per ogni tipo di prodotto consegnato, vale a dire che esse siano fissate per ciascun cliente indipendentemente dalla tipologia di prodotto di cui viene rifornito. La decisioni tattiche scaturiscono da un modello MIP, nel quale l'attenzione è posta sul singolo cliente che deve essere rifornito di un numero elevato di prodotti differenti in un orizzonte di pianificazione esteso. L'aspetto caratterizzante del MIRP risiede nella modalità di trattare il numero elevato di prodotti coinvolti nel processo distributivo. Ad ogni prodotto può essere associata una politica di rifornimento diversa in relazione al contesto applicativo di riferimento. Esistono in letteratura approcci risolutivi per problemi di *Inventory Routing* single-item che possono essere estesi al caso *Multi-item* (o *Mult-product*), alcuni dei quali richiedono una espansione del grafo di partenza dovuta all'introduzione di tanti nodi fittizi quanti sono i prodotti trattati da ciascun cliente, aggravando quindi sensibilmente la complessità computazionale dell'approccio. Altri approcci risolutivi, progettati per problemi di *Inventory Routing Multi-item*, riducono la complessità del problema facendo riferimento a politiche di rifornimento espresse sotto forma di multipli di una stessa politica base. I limiti di tali approcci in taluni contesti applicativi sono stati già evidenziati. La novità dell'approccio proposto risiede principalmente nel focus modellistico con cui viene trattata la fase tattica. Infatti, il modello matematico proposto, denominato *Customer Model* (CM), si pone l'obiettivo di ottimizzare il processo di distribuzione dal punto di vista del cliente, in modo tale da garantire un determinato livello di servizio ed evitare situazioni di stock-out. Da questo punto di vista si può dire che il CM privilegia il punto di vista del cliente rispetto ai modelli classici di *Inventory Routing*, in cui il focus era quello del fornitore. Il MIRP è stato contestualizzato in un caso aziendale reale relativo alla distribuzione di caffè. In particolare, il caso in esame si riferisce ad un magazzino ubicato vicino il porto di Gioia Tauro, noto per essere uno dei maggiori porti del Mediterraneo per il "transhipment" di container. In tale magazzino sono stoccate, in sacchi, diverse tipologie di caffè crudo (un centinaio circa); tali tipologie devono essere consegnate ad una filiera di torrefattori, dislocati su un area geografica molto ampia, che va dalla Puglia alla Sicilia. Ogni torrefattore miscela le diverse tipologie di caffè crudo in modo differente per ottenere un caffè con un particolare aroma. La percentuale di miscelazione di ogni tipologia di caffè crudo è nota a priori. Inoltre, conoscendo quale sia il volume della miscela "finale" di caffè che deve essere prodotta alla fine del periodo di pianificazione, è possibile calcolare i tassi giornalieri di consumo di ogni tipologia di caffè crudo (in sacchi), per ogni specifica miscela prodotta. Tali tassi sono costanti sull'intero periodo di pianificazione preso in esame. Dal punto di vista modellistico il MIRP presenta una elevata complessità computazionale dovuta:

- all'elevato numero di variabili decisionali e di vincoli operativi;
- all' elevato numero di rotte ammissibili di consegna;
- all'estensione del periodo di pianificazione.

Per rendere il MIRP computazionalmente più trattabile può essere conveniente ridurre opportunamente il numero di rotte di consegna ed aggregare i periodi finali dell'orizzonte temporale. Tecniche di questo tipo sono già state utilizzate con successo da Campbell e Savelsbergh 2004a. L'ulteriore aggravio modellistico che si presenta nel MIRP risiede non solo nel numero di clienti (qualche centinaio) da servire, ma anche nel numero di prodotti di cui rifornire ciascun cliente. Il numero di tali prodotti può arrivare anche a qualche migliaio per cliente. Questi numeri fanno intuire subito come approcci risolutivi esatti per i modelli matematici del MIRP siano impraticabili. Si è valutata quindi

l'opportunità di adottare tecniche risolutive metaeuristiche. L'ultimo contributo di questa tesi, relativamente ai problemi di Inventory Routing, consiste nella progettazione ed implementazione di algoritmi di ricerca locale per il MIRP, basati su tecniche di diversificazione spaziale e temporale per sfuggire da minimi locali. Per valutare le prestazioni dell'approccio proposto, i risultati ottenuti su istanze di medie e grandi dimensioni sono stati confrontati con un semplice lower bound modellistico. L'approccio euristico presenta buone performance dal punto di vista del tempo di calcolo necessario per costruire una buona soluzione ammissibile del MIRP. L'indicatore statistico utilizzato per il confronto è stato il rapporto tra il volume totale V di prodotti consegnati e la distanza D percorsa nell'orizzonte temporale. I valori di tale statistica in corrispondenza delle soluzioni euristiche per il MIRP sono risultati prossimi ai valori del rapporto volume/distanza corrispondenti al lower bound.

4.3 Il modello di Planning

In questo paragrafo viene illustrato il modello matematico alla base della fase di pianificazione. Le ipotesi essenziali sono elencate di seguito.

1. Nella fase di pianificazione il numero dei veicoli è considerato illimitato. Questa assunzione è verosimile tenuto conto dell'elevato numero di coppie (cliente, prodotto) che occorre servire.
2. Nella fase di instradamento dei veicoli non sono state considerate le problematiche relative alla composizione degli equipaggi impegnati nella distribuzione, alla manutenzione programmata dei veicoli e alla turnazione degli addetti alla guida dei veicoli.
3. I prodotti sono assemblati in lotti e/o bancali nelle aree di spedizione del magazzino del vendor e, da qui, vengono instradati sui veicoli e distribuiti ai clienti. Il problema di caricamento sui veicoli non viene esaminato nella sua versione spaziale (2D o 3D bin packing). Questo è motivato, in parte, dal fatto che i veicoli non effettuano viaggi a pieno carico.

4.3.1 La fase di pianificazione

In questa fase sono vengono prese le decisioni relative a quali clienti rifornire, in che periodo dell'orizzonte di pianificazione, di quali prodotti rifornirli ed in che quantità. Si fa riferimento ad un orizzonte temporale finito e discreto $T=\{0,\ldots,H\}$ ed ad una rete logistica definita su un grafo non orientato Sia $G=(V,E)$, in cui $V=\{0,\ldots,n\}$ è l'insieme dei vertici ed $E\subseteq V\times V$ è l'insieme degli spigoli. Ad ogni spigolo $(i,j)\in E$ è associato un costo simmetrico non negativo $c_{ij}=c_{ji}\geq 0$ che rappresenta la distanza chilometrica del cammino minimo che separa il vertice *i* dal vertice *j* sulla rete stradale reale; inoltre, ad ogni spigolo $(i,j)\in E$ resta associato il tempo medio di percorrenza t_{ij} corrispondente al suddetto cammino. Sia 0 il nodo fornitore (vendor o supplier) del sistema di distribuzione ed *M* il numero di veicoli della flotta disponibile per le consegne nel vendor. Tutti i veicoli hanno la stessa capacità. Si definisce tour ottimo del Travelling Salesman Problem (TSP) il vettore circuitale costituito dagli indici dei vertici di V, avente come primo indice e come ultimo indice quello relativo al nodo fornitorer: $TSP=<0,\ldots,i,\ldots,j,\ldots,0>$. Il costo del TSP è la somma dei tempi medi di percorrenza degli spigoli costituiti da due vertici consecutivi di esso: $\sum_{i\in TSP,\,i+1\in TSP} t_{i,i+1}$ Il tour ottimo del TSP rappresenta la sequenza di visita di minimo costo dei vertici di V, a partire dal nodo vendor. Siano P_i l'insieme dei prodotti *p* di cui rifornire il cliente $i\in M=V\setminus\{0\}$ nel periodo T, e sia K_{pi} il numero delle consegne del prodotto *p* al cliente *i* nell'orizzonte temporale *T*, previste con il modello di gestione delle scorte adottato presso il cliente. Nella analisi proposta si assumerà che tale modello sia un EOQ (Economic Order Quantity) per tutti i clienti $i\in M$. Si denoti con π^{e}_{pik} il costo dovuto all'anticipo della *k*-esima

consegna del prodotto *p* al cliente *i*, e con π^{l}_{pik} il costo dovuto al ritardo nella *k*-esima consegna del prodotto *p* al cliente *i*, per ogni $i \in M$, $p \in P_i$, $k = 1, \ldots, K_{pi}$. Tali costi giocano un ruolo decisivo nella fase tattica, in cui il vendor assume decisioni sulle consegne ai clienti. Infatti, la possibilità di controllare gli anticipi ed i ritardi delle consegne tramite i costi definiti poc'anzi, permette di avere maggiore flessibilità nella scelta dei giorni delle consegne. In questo modo, nella successiva fase di routing potranno essere enfatizzati i vincoli operativi derivanti dalla capacità di trasporto dei veicoli e dalle finestre temporali per le consegne dei prodotti ai clienti. I valori dei suddetti costi di penalità dipendono dalla particolare applicazione prese in esame.
$\pi^{e}_{pik} = \left(h_{pi} / H\right)\left(q_{pi} l_{pik} / \mu_{pi}\right)$ il costo dovuto all'anticipo della *k*-esima consegna del prodotto *p* al cliente *i*, e $\pi^{l}_{pik} = \left(l^{s}_{pik} / 2\mu_{pi} H\right)\left(v_{pi} l^{s}_{pik} - h_{pi} q_{pi}\right) + \left(l^{S}_{pik} / H\right)\left(u_p - f_p / q_{pi} - c_p\right)$ il costo dovuto al ritardo nella *k*-esima consegna del prodotto *p* al cliente *i*, per ogni $p \in P_i$, $i \in V \setminus \{0\}$ e $k = 1, \ldots, K_{pi}$. Sono dati i seguenti parametri:

- h_{ip} := costo di stoccaggio di una unità del prodotto *p* al cliente *i* nell'unità di tempo (giorno);
- μ_{pi} := tasso di consumo del prodotto *p* presso il cliente *i*, supposto costante in *T*;
- f_p := costo fisso di riordino del prodotto *p*, supposto indipendente dai clienti;
- $q_{pi} = \sqrt{2 f_p \mu_{pi} / h_{pi}}$:= lotto economico di riordino del prodotto *p* al cliente *i*;
- t^{*}_{pik} := giorno della *k*-esima consegna della quantità q_{pi} al cliente *i*;
- $l_{pik} = \left(t^{*}_{pik} - t\right) \cdot \mu_{pi}$:= livello di inventario del prodotto *p* nel cliente *i* il giorno $t \in \left\{t^{*}_{pik-1}, \ldots, t^{*}_{pik}\right\}$, prima della *k*-esima consegna;
- $l^{s}_{pik} = \left(t - t^{*}_{pik}\right) \cdot \mu_{pi}$:= quantità della scorta di sicurezza (*safety stock*) del prodotto *p* consumata nel cliente *i* il giorno $t \in \left\{t^{*}_{pik}, \ldots, t^{*}_{pik+1}\right\}$, dopo la *k*-esima consegna;
- u_p := costo di una unità della scorta di sicurezza del prodotto *p*, supposto indipendente dal cliente;
- v_{pi} := costo di una unità della scorta di sicurezza del prodotto *p* nel cliente *i*, nell'unità di tempo;
- c_p := costo di una unità del prodotto *p*, supposto indipendente dai clienti.

L'obiettivo del Customer Model è quello di minimizzare il numero di volte che ogni cliente è visitato per essere rifornito nell'orizzonte temporale, garantendo che le consegne dei prodotti avvengano entro un tempo limite che assicuri l'assenza di stock-out. Il Customer Model è formulato e risolto per ogni cliente $i \in V \setminus \{0\}$. Pertanto, nelle variabili decisionali del modello si omette l'indice relativo al cliente. Esse sono:

Variabili booleane:

1) y^{t} vale 1 se il cliente è visitato nel periodo *t*, 0 altrimenti; per ogni $t \in T$;
2) x^{t}_{pk} vale 1 se la *k*-esima consegna del prodotto *p* avviene nel periodo *t*, 0 altrimenti; per ogni $p \in P_i$, $k = 1, \ldots, K_p$, $t \in T$.

Variabili fisiche:

d^{t}_{p} := quantità di prodotto *p* consegnata nel periodo *t*.

I vincoli caratteristici del modello esprimono le seguenti limitazioni:

- poiché K_p rappresenta il numero di consegne del prodotto *p* nell'orizzonte temporale di pianificazione, occorre garantire che ciascuna di esse abbia luogo in *T*: $\sum_{t\in T} x_{pk}^t = 1$;
- se il cliente è visitato nel periodo *t*, il numero di consegne non può superare il numero di prodotti richiesti: $\sum_{p\in P_i}\sum_{k=1}^{K_p} x_{pk}^t \le |P_i| y^t$;
- una consegna può essere anticipata o ritardata non oltre un numero prefissato di periodi, indicato con LD (*Limit Days*). I vincoli che esprimono il ritardo o l'anticipo sono: $\alpha_{pk} = \sum_{t\in T} t x_{pk}^t - t_{pk}^* \le LD$, se $t \ge t_{pk}^*$, oppure $\beta_{pk} = t_{pk}^* - \sum_{t\in T} t x_{pk}^t \le LD$, se $t \le t_{pk}^*$, dove t_{pk}^* è il periodo della *k*-esima consegna del prodotto *p*, previsto dalla politica di rifornimento (EOQ);
- la penalizzazione di un ritardo o un anticipo nella consegna di un prodotto può essere espressa da: $\sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^e \beta_{pk} \le \Pi$, se $t \le t_{pk}^*$, o

  $\sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^l \alpha_{pk} \le \Pi$, se $t \ge t_{pk}^*$,

  essendo W_{pk}^e e W_{pk}^l rispettivamente i pesi associati alle consegne avvenute in anticipo o in ritardo. Si ha $W_{pk}^e = \pi_{pk}^e$ per $t \le t_{pk}^*$, e $W_{pk}^l = \pi_{pk}^l$ per $t \ge t_{pk}^*$. Il parametro Π rappresenta la massima penalità che il cliente è disposto a pagare per avere consegne irregolari; se $W_{pk}^e = W_{pk}^l = 1$, e $\alpha_{pk} = \beta_{pk} = LD$ per ogni $p \in P_i$, e $k = 1, \ldots, K_p$, il massimo livello di penalità è $\Pi_{\max} = LD|P_i|K_p$. Il parametro Π è una percentuale di $\Pi_{\max}$: $\Pi = \gamma\Pi_{\max}$. I livelli di penalità presi in considerazioni sono tre, così identificati:

  penalità bassa: $\Pi = 0.25\Pi_{\max}$;

  penalità media: $\Pi = 0.35\Pi_{\max}$;

  penalità alta: $\Pi = 0.50\Pi_{\max}$;

- i vincoli di stock-out sono espressi dalle disuguaglianze seguenti: $LL_p^t \le \sum_{s=1}^{t} d_p^s \le UL_p^t$,

  dove $LL_p^t = max(0, t\mu_p - I_p^0)$ rappresenta il lower bound sulla quantità totale di prodotto *p* da consegnare nel periodo *t*, espresso in termini di:

  a) I_p^0 := livello di inventario iniziale del prodotto *p*;

  b) $t\mu_{pi}$:= quantità di prodotto *p* consumata nell'intervallo $[0, t]$.

  Il valore di LL_p^t deve essere tale che il cliente riesca a soddisfare la domanda del prodotto *p* almeno fino al giorno seguente, mentre $UL_{pi}^t = t\mu_{pi} + C_{pi} - I_{pi}^0$ rappresenta l'upper bound sulla quantità totale del prodotto

p che può essere consegnata nel periodo *t*. Tale upper bound dipende anche dalla capacità di stoccaggio C_{pi} del prodotto *p*. Il valore di UL_p^t non deve essere maggiore della quantità di prodotto *p* da consegnare;

- la quantità di prodotto consegnata, in anticipo o in ritardo, deve essere tale da ripristinare i livelli di inventario: $(q_p - LD\mu_p)\sum_{k=1}^{K_p} x_{pk}^t \le d_p^s \le (q_p + LD\mu_p)\sum_{k=1}^{K_p} x_{pk}^t$,

  dove q_p è il lotto economico di riordino secondo l'assegnata politica di rifornimento del cliente (EOQ), e μ_p è il tasso medio di consumo del prodotto *p*.

L'obiettivo è la minimizzazione del numero di visite e delle penalità dovute agli anticipi ed ai ritardi nelle consegne. Esso si può formulare come:

$$min\, W\sum_{t\in T} y^t + \sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^e \beta_{pk} + \sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^l \alpha_{pk} \ ,$$

dove W è un parametro che misura il peso associato al numero di visite, e serve per omogeneizzazione la funzione obiettivo. Il modello è stato implementato con un valore di W pari a 100, a cui si è giunti dopo una fase di taratura sperimentale. Il modello di programmazione lineare per la fase di pianificazione può essere formulato come segue:

$$min\, W\sum_{t\in T} y^t + \sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^e \beta_{pk} + \sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^l \alpha_{pk} \tag{4.9}$$

s. a

$$\sum_{t\in T} x_{pk}^t = 1,\ \forall p \in P_i, k = 1,\ldots, K_p\ , \tag{4.10}$$

$$\sum_{p\in P_i}\sum_{k=1}^{K_p} x_{pk}^t \le |P_i| y^t\ ,\ \forall t \in T\ , \tag{4.11}$$

$$\alpha_{pk} - \beta_{pk} = \sum_{t\in T} t x_{pk}^t - t_{pk}^*\ ,\ \forall p \in P_i, k = 1,\ldots, K_p\ , \tag{4.12}$$

$$\beta_{pk} \le LD\ ,\ \forall p \in P_i, k = 1,\ldots, K_p\ , \tag{4.13}$$

$$\alpha_{pk} \le LD\ ,\ \forall p \in P_i, k = 1,\ldots, K_p\ , \tag{4.14}$$

$$\sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^e \beta_{pk} \le \Pi\ , \tag{4.15}$$

$$\sum_{p\in P_i}\sum_{k=1}^{K_p} W_{pk}^l \alpha_{pk} \le \Pi\ , \tag{4.16}$$

$$LL_p^t \le \sum_{s=1}^{t} d_p^s \le UL_p^t\ ,\ \forall p \in P_i, t \in T\ , \tag{4.17}$$

$$\left(q_p - LD\mu_p\right)\sum_{k=1}^{K_p} x_{pk}^t \le d_p^s \le \left(q_p + LD\mu_p\right)\sum_{k=1}^{K_p} x_{pk}^t,\ \forall p \in P_i, t \in T, \tag{4.18}$$

$$x_{pk}^t \in \{0,1\},\ \forall p \in P_i, t \in T, k = 1,\ldots,K_p, \tag{4.19}$$

$$y^t \in \{0,1\},\ \forall t \in T, \tag{4.20}$$

$$\alpha_{pk} \ge 0, \text{ intere, } \forall p \in P_i, k = 1,\ldots,K_p, \tag{4.21}$$

$$\beta_{pk} \ge 0, \text{ intere, } \forall p \in P_i, k = 1,\ldots,K_p, \tag{4.22}$$

$$d_p^t \ge 0,\ \forall p \in P_i, t \in T. \tag{4.23}$$

Al fine di limitare i ritardi e gli anticipi nelle consegne, ed indirizzare le scelte del fornitore verso il massimo rispetto delle politiche di rifornimento messe in atto presso i singoli clienti, il modello (4.9)-(4.23) è risolto con una bassa penalità ($\Pi = 0.25\Pi_{\max}$), e con un ritardo massimo nelle consegne $LD = 2$. Emerge molto chiaramente la complessità computazionale del modello (4.9)-(4.23), sia in termini di numero di variabili decisionali che dal punto di vista del numero dei vincoli caratterizzanti. In un contesto applicativo reale il numero di clienti può essere anche di qualche centinaio, ed il numero di prodotti stoccati presso ogni cliente può arrivare anche a qualche migliaio. Se a ciò si aggiunge che l'orizzonte di pianificazione preso in considerazione è abbastanza esteso, si intuisce facilmente come istanze realistiche non possono essere risolte all'ottimo con un modello di programmazione lineare intera mista (Mixed Integer Program), quale quello (4.9)-(4.23). D'altro canto, l'utilità di un tale modello ricorre laddove da esso sia possibile ricavare un modello surrogato, più facile da risolvere all'ottimo dal punto di vista della complessità computazionale, e che quindi possa fornite un buon lower bound per il MIRP.

4.3.2 Un bound modellistico per il MIRP

In questo paragrafo viene descritta una procedura per determinare un bound per il MIRP. L'obiettivo è quello di valutare la minima distanza percorsa dai veicoli per consegnare la quantità richiesta di ogni prodotto ai singoli customer nell'orizzonte di pianificazione considerato. Si suppone di poter consegnare tutta la quantità richiesta nell'intero periodo di pianificazione da ciascun cliente in un unico periodo dell'orizzonte di pianificazione (per esempio il primo). Tale assunzione equivale a rilassare il vincolo sulla capacità massima di stoccaggio di ciascun prodotto; in questo modo si farà in modo di conferire maggiore enfasi alla scelta che minimizza la distanza da percorrere nell'orizzonte di pianificazione. Si evidenzia che la quantità totale di prodotto da consegnare è esattamente quella minima necessaria a soddisfare la domanda del prodotti dei clienti in T. Tale quantità è determinata dal valore ottimo del seguente modello di programmazione intera mista. I vincoli del modello garantiscono l'assenza di stock-out, ripartendo la quantità complessiva di ciascun prodotto in un numero di periodi dell'orizzonte temporale, pari al numero di consegne previsto dalla politica di rifornimento di ogni singolo prodotto. Applicando un algoritmo di instradamento nel periodo (unico) dell'orizzonte temporale in cui si suppone siano concentrate tutte le consegne, risulta possibile rifornire tutti i clienti minimizzando la distanza percorsa. Il MIP per la scelta dei volumi è formulato di seguito:

$$\min \sum_{t\in T} \sum_{p\in P_i} d_p^t \tag{4.24}$$

soggetto ai seguenti vincoli:

$$LL_p^t \le \sum_{s=1}^{t} d_p^s \le UL_p^t,\ \ \forall p \in P_i, \forall t \in T \tag{4.25}$$

$$\sum_{t\in T} z_p^t = K_p, \quad \forall p \in P_i \tag{4.26}$$

$$d_p^t \leq U_p z_p^t, \qquad \forall p \in P_i, \forall t \in T \tag{4.27}$$

$$z_p^t \in Z^+, \quad \forall p \in P_i, \forall t \in T \tag{4.28}$$

$$d_p^t \in R^+, \quad \forall p \in P_i, \forall t \in T \tag{4.29}$$

I vincoli del modello hanno il seguente significato:

- i vincoli (4.25) assicurano l'assenza di stock-out ed il rispetto dl vincolo sulla capacità di stoccaggio dei prodotti;
- i vincoli (4.26) sono una riformulazione dei vincoli (4.10) del modello (4.9)-(4.23), espressi in forma aggregata avendo: a) sommato sull'indice k per ogni $p \in P_i$ e, b) aggregato le variabili x_{pk}^t mediante l'introduzione delle variabili surrogate $z_p^t = \sum_{k=1}^{K_p} x_{pk}^t$;
- i vincoli (4.27) rappresentano un rilassamento del vincolo (4.28) ottenuto rimuovendo la parte sinistra del vincolo originale, ponendo $LD = \frac{U_p - q_p}{\mu_p}$ ed utilizzando le variabili surrogate z_p^t. Tali vincoli sono vincoli logici di priorità, nel senso che il prodotto p non può essere consegnato nel periodo t se in esso non sono schedulate consegne;
- le variabili (4.28) rappresentano il numero di consegne del prodotto p nel periodo t;
- le variabili (4.29) rappresentano la quantità di prodotto p consegnata nel periodo t. In quanto segue, la quantità minima complessiva di ogni prodotto a ciascun cliente risulterà $\underline{d}_{ip} = \sum_{t \in T} \underline{d}_p^t$, per ogni $p \in P_i$, essendo $\underline{d}_p^t$ i valori restituiti dal modello (4.24)-(4.29);
- un lower bound $\underline{D}$ sulla distanza minima percorsa, necessaria a soddisfare la domanda di prodotto di tutti i clienti, è ottenuto applicando un algoritmo di instradamento in cui i vincoli sulle finestre temporali dei clienti sono rilassati, e consegnando a ciascun cliente $i \in M$, il volume aggregato $\sum_{p \in P_i} \underline{d}_{ip}$. Il risultato è un upper bound sul valore dell'indice statistico volume consegnato per chilometri percorsi:

$$UB_{\underline{D}}^{V} = \frac{\sum_{i \in M} \sum_{p \in P_i} \underline{d}_{ip}}{\underline{D}}.$$

Da quanto fin qui esposto, si evince come la complessità del problema dal punto di vista matematico e computazionale richieda l'utilizzo di tecniche euristiche e meta euristiche per potere trattare casi reali. In quanto segue si fornisce una descrizione delle procedure di base che saranno utilizzate all'interno degli algoritmi euristici definiti nelle successive sezioni. Nella notazione che segue t_{ipk} rappresenta il periodo di *T*, in cui la *k*-esima consegna del prodotto *p* al cliente *i* ha luogo.

4.3.3 Un bound euristico per il MIRP

Anche il modello (4.24)-(4.29) soffre di una scarsa trattabilità computazionale laddove occorre ottenere un bound per istanze reali, in cui le dimensioni degli insiemi P_i sono veramente grandi (dell'ordine delle centinaia di migliaia), e l'orizzonte temporale può essere significativamente esteso (dell'ordine dei mesi). In questi casi, il volume di prodotto di cui rifornire ciascun cliente è stato valutato utilizzando una formula, in cui si ipotizza un tasso di consumo costante del prodotto in tutto l'orizzonte di pianificazione:

$$\underline{d}_{ip} = H\mu_{pi} - I_{pi}^0.$$

H è l'estensione dell'orizzonte temporale, μ_{pi} è il tasso di consumo medio del prodotto $p \in P_i$ per il cliente $i \in M$ in *T*, mentre I_{pi}^0 è il livello d'inventario iniziale del prodotto. Tale espressione restituisce la quantità minima di cui rifornire il cliente *i* del prodotto *p* nel periodo di pianificazione affinché non si verifichi stock out. Tale quantità è ripartita in K_p periodi in modo uniforme. Dopo aver partizionato la quantità da consegnare nell'orizzonte temporale

di pianificazione, viene richiamato l'algoritmo di routing per schedulare le consegne ai clienti. L'upper bound euristico sul valore dell'indicatore statistico di prestazione $\frac{V}{D}$ è dato da:

$$UB_{\frac{V}{D}}^{H} = \frac{\sum_{i\in M}\sum_{p\in P_i}\left(H\mu_{ip} - I_{ip}^{0}\right)}{\underline{D}}.$$

Procedure di base

- $d_{ip} \leftarrow Get\_Order(i;\ p;\Pi_{\mathrm{p}})$ è la procedura utilizzata per calcolare la quantità di prodotto d_{ip}, di cui rifornire il cliente *i*, nel rispetto dei vincoli (4.17); tale valore è calcolato sulla base della politica di rifornimento Π_{p} del prodotto p;
- l'insieme $S_i^t = \left\{d_{ip}: t_{ipk} = t,\ \forall p \in P_i, \forall k = 1, \dots, K_{ip}\right\}$ è l'insieme delle quantità d_{ip} che devono essere consegnate al cliente i nel periodo t. Si noti che $S^t = \bigcup_{i\in M} S_i^t$, $S_i = \bigcup_{t\in T} S_i^t$, e $S = \bigcup_{i\in M} S_i$
- l'insieme $ND_i = \{ND_i^0, \dots, ND_i^H\}$ è l'insieme delle consegne ND_i^t al cliente i nel periodo t, per ogni $t \in T$;
- $ND_i \leftarrow Get\_OrderNumber(i;\ P_i)$ è la procedura utilizzata per calcolare il numero di consegne al cliente i in T;
- $\{0,1\} \leftarrow Check\_Earlier(i;\ p;\ k;\ t_{ipk})$ è la procedura che restituisce il valore 1 se la $k-esima$ consegna del prodotto p al cliente i può essere schedulata in anticipo (nel periodo t-1) rispetto al periodo programmato t, 0 altrimenti. Il controllo che viene eseguito all'interno della procedura assicura che, nel periodo $t_{ipk} - 1$, la massima quantità di inventario $UL_{ip}^{t_{ipk}-1} = U_{ip} + \left(\left(t_{ipk} - 1\right) - t_{ipk-1}\right)\mu_{ip} - I_{ip}^{t_{ipk-1}}$ soddisfi la seguente relazione:

  $$I_{ip}^{t_{ipk}-1} + q_{ip} \le UL_{ip}^{t_{ipk}-1},$$

  essendo $I_{ip}^{t_{ipk-1}}$ il livello di inventario del prodotto p nel cliente i nel periodo t_{ipk-1}.
- $\{0,1\} \leftarrow Check\_Later(i;\ p;\ k;\ t_{ipk})$ è la procedura che restituisce il valore 1 se la $k-esima$ consegna del prodotto $p \in P_i$ al cliente $i \in M$ può essere schedulata in ritardo (nel periodo $t+1$) rispetto al periodo programmato t, 0 altrimenti. Inoltre, con riferimento alla prima consegna ($k = 1$), la procedura controlla che la differenza tra il livello di inventario nel periodo t_{ip1} e la quantità di prodotto consegnata nello stesso periodo sia maggiore o uguale della quantità di prodotto consumata nel periodo successivo, ovvero che:

  $$I_{ip}^{t_{ip1}} - q_{ip} \ge \left(\left(t_{ipk} + 1\right) + t_{ip1}\right)\mu_{ip}$$

  Viceversa, se $k > 1$, essa verifica che la somma relativa:

  - al livello di inventario nel periodo $t_{ipk} - 1$;
  - alla quantità di prodotto che dovrebbe essere consegnata nel periodo t_{ipk};
  - la quantità di prodotto consumata nel periodo successivo;

  sia minore o uguale del livello massimo di inventario nel periodo t_{ipk}, ovvero:

  $$I_{ip}^{t_{ipk}-1} + q_{ip} + \left(\left(t_{ipk} + 1\right) - t_{ip1}\right)\mu_{ip} \le UL_{ip}^{t_{ipk}-1}$$

  Si osservi che la quantità $q_{ip} + I_{ip}^{t_{ipk}-1} + \left(\left(t_{ipk} + 1\right) - t_{ip1}\right)\mu_{ip}$, che rappresenta il livello di inventario nel periodo $t_{ipk-1} - 1$, è sufficiente a soddisfare una maggiore domanda di prodotto, nel periodo $t_{ipk} + 1$, pari a $q_{ip} + \left(\left(t_{ipk} + 1\right) - t_{ip1}\right)\mu_{ip}$.

- $S_i \leftarrow Move\_Earlier(i;\ p;\ k;\ t_{ipk};\ S_i)$ è la procedura utilizzata per spostare la $k-esima$ consegna del prodotto p al cliente i dal periodo t_{ipk} al periodo $t_{ipk} - 1$; inoltre, tale procedura aggiorna l'insieme S_i:

  $$S_i = \left\{\bar{S}_i^{t_{ipk}-1}: t \in T\right\},$$

dove $\bar{S}_i^{t_{ipk}-1} = S_i^{t_{ipk}-1} \cup \{q_{ip}: Check\_Earlier(i;\ p;\ k;\ t_{ipk}) = TRUE\}$;

- $S_i \leftarrow Move\_Later(i;\ p;\ k;\ t_{ipk};\ S_i)$ è la procedura utilizzata per spostare la $k-esima$ consegna del prodotto p al cliente i dal periodo t_{ipk} al periodo $t_{ipk}+1$; inoltre, tale procedura aggiorna l'insieme S_i

$$S_i = \left\{\bar{S}_i^{t_{ipk}+1} : t \in T\right\},$$

dove $\bar{S}_i^{t_{ipk}+1} = S_i^{t_{ipk}+1} \cup \{q_{ip}: Check\_Later(i;\ p;\ k;\ t_{ipk}) = TRUE\}$.

4.4 Algoritmo Euristico per la gestione delle scorte

L'Hill Climbing heuristic per il MIRP (HC-MIRP) è una procedura di ricerca locale randomica che è impiegata come passo di intensificazione nell'ambito della meta euristica di tipo Iterated Local Search (ILS) applicata al MIRP, e di seguito descritta. Prima di tutto, si definiscono *moveable* i periodi t_m di T in cui il numero di consegne è inferiore o uguale a $\gamma \in |P_i|$, essendo γ una soglia fissata. Tutti i periodi cosiddetti "mobili" sono inseriti in un insieme T_m. Si controlla quindi che tutte le consegne che avvengono in un periodo t_m *moveable* possano essere spostate prima o dopo t_m. Qualora si verificasse che non tutte le consegne programmate in t_m possano essere spostate in un periodo precedente o successivo, si etichetta t_m come tabù e lo si rimuove da T_m. Il periodo t_m è selezionato casualmente da T_m e le sue consegne sono spostate prima o dopo, solo dopo che la verifica di ammissibilità dello spostamento risulti positiva. Tale procedura termina quando T_mè vuoto. Per potere operare la procedura necessita di una distribuzione iniziale delle quantità di prodotto d_{ip} ai clienti, ottenuta tramite la funzione INITIALIZE descritta nella Algoritmo 4.1, e della funzione CHECKSHIFT, descritta nell'outline INITIALIZE, che verifica l'ammissibilità dello spostamento dell'intero volume consegnato in t_m. Il pseudo-codice della procedura HC-MIRP è presentato in dettaglio nell'Algoritmo 4.4.

Algoritmo 4.1: INITIALIZE

1: if $S = \emptyset$; then
2: for all $i \in M$ do
3: for all $p \in P_i$ do
4: $d_{ip} \leftarrow$ Get_Order$(i;\ p; \Pi_p)$, ed inizializza i seguenti insiemi S_i^t, S^t, S_i, S
5: end for
6: end for
7: end if

Algoritmo 4.2: CHECKSHIFT

1: Select $t_p = \text{argmax}_{t\in\{t^*-1,t^*+1\}}\{I_{ip}^{t^*-1}\ \text{Check}_{\text{Earlier}(i;\ p;\ k;\ t^*)}, I_{ip}^{t^*+1}\ \text{Check\_Earlier}(i;\ p;\ k;\ t^*)\}$
2: if $t_p = t^* - 1 > 0$ then
3: $S_i \leftarrow$ Move_Earlier$(i;\ p;\ k;\ t_p;\ S_i)$
4: else if $t_p = t^* + 1 > 0$ then
5: $S_i \leftarrow$ Move_Later$(i;\ p;\ k;\ t_p;\ S_i)$
6: end if

Algoritmo 4.3: ROUTING

1: for all $t \in T$ do
2: if $S^t \neq \emptyset$; then
3: Inizializza M^te P^t
4: $R = R \cup R^t \leftarrow$ Routing$(M^t;\ P^t;\ S^t; G; m; Q)$
5: end if
6: end for

Algoritmo 4.4: HC-MIRP

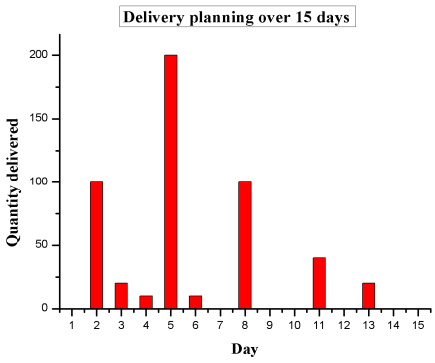
Require: Sets M, S, graph G, fleet of m vehicles with capacity Q, and planning horizon T
Ensure: Set R of delivery routes scheduled over T and the updated set S. Initially set $R = \emptyset$;
1:invoke INITIALIZE procedure
2: for all $i \in M$ do
3: Let $B_i = \emptyset$;
4: while TRUE do
5: $ND_i \leftarrow$ Get OrderNumber$(i;\ P_i)$
6: $T_m \leftarrow$ Get Moveable$(ND_i; \gamma; B_i)$
7: if $T_m = \emptyset$; then
8: break
9: end if
10: for all $t_m \in T_m$ do
11: Set MOVE = TRUE
12: for all $p \in P_i$, such that $t_{ipk} = t_m$ do
13: if Check_Earlier$(i; p; k; t_{ipk})$ = Check_Later$(i; p; k; t_{ipk})$ = FALSE then

14: MOVE = FALSE
15: $B_i = B_i \cup \{t_m\}$ and $T_m = T_m \backslash \{t_m\}$
16: break
17: end if
18: end for
19: end for
20: if $T_m \neq 0 = ;$ then
21: Let t_m be the moveable day randomly selected from T_m
22: for all $p \in P_i$, such that $t_{ipk} = t_m$ do
23: Invoke procedure CHECKSHIFT
24: end for
25: end if
26: end while
27: end for
28: Invoke procedure ROUTING

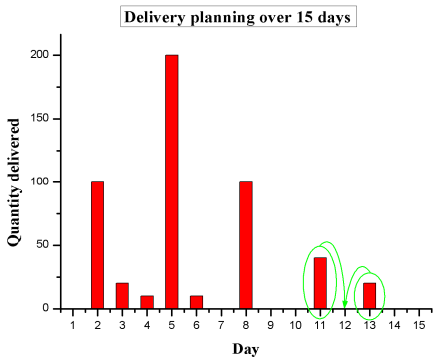
La procedura MERGE viene utilizzata all'interno della ILS-MIRP nel tentativo di ridurre il numero dei periodi delle consegne. Tale procedura opera come descritto di seguito:

1. identifica tutti i cosiddetti *empty periods* dell'orizzonte di pianificazione, ovvero tutti i periodi dell'orizzonte di pianificazione nei quali non si effettuano consegne, tali che nei periodi immediatamente precedenti e/o successivi ad essi si effettuano delle consegne;
2. verificare che tutte le consegne che hanno luogo nel periodo immediatamente antecedente o successivo ad un *empty period* possano essere spostate in esso, ed infine
3. sposta tutte le consegne nell'*empy period*.

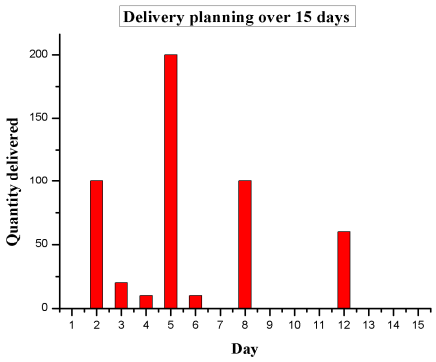
Più formalmente, sia t_e il primo giorno vuoto trovato in T, occorre quindi verificare che tutte le consegne programmate in $t_e - 1$ e $t_e + 1$ possano essere spostate in t_e e, quindi, posticiparle e anticiparle tutte nello stesso periodo t_e nel rispetto dei vincoli (4.17). Qualora lo spostamento risulti inammissibile per i vincoli (4.17), t_e è etichettato come tabù. La procedura termina quando nessun *empty period* è stato trovato.



a) Initial Delivery Planning



b) Merge Phase



c) Delivery Planning after Merge Phase

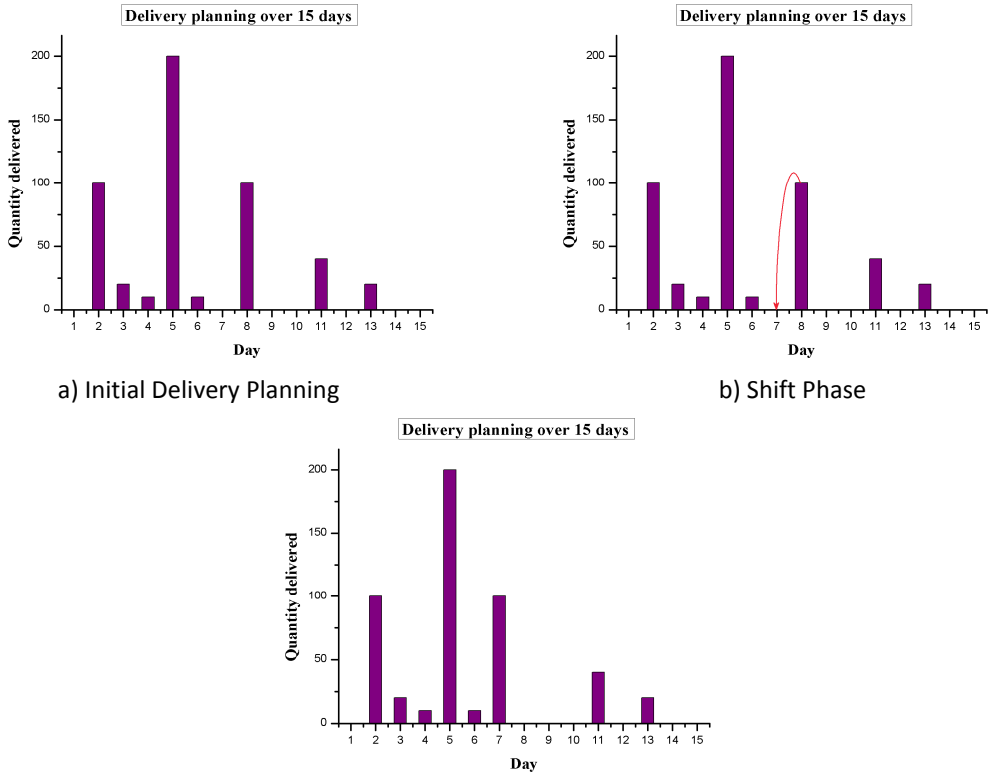
Figura 4.5 Merge Phase

La Figura 4.5 illustra il comportamento della procedura MERGE. Lo pseudo-codice è descritto nell'Algoritmo 4.5, riportato di seguito:

Algoritmo 4.5: MIRP - Merge

Require: Customer i, set S_i and planning horizon T
Ensure: Set S_i
1: Let $B_i = 0$;
2: while TRUE do
3: $ND_i \leftarrow Get\_OrderNumber(i; P_i)$
4: $t_e \leftarrow Get\_Empty(i; T; B_i)$. This procedure returns the first empty day in T, or 0 if no empty day is found.
5: if $t_e = 0$ then
6: break
7: else
8: Set MOVE = TRUE
9: for all $p \in P_i$, such that $t_{ipk} = t_e - 1$ do
10: if $Check\_Later(i; p; k; t_{ipk})$= FALSE then
11: $MOVE = FALSE$, $B_i = B_i \cup \{t_e\}$ and break
12: end if
13: end for
14: if MOVE then
15: for all $p \in P_i$, such that $t_{ipk} = t_e + 1$ do
16: if $Check\_Earlier(i; p; k; t_{ipk})$= FALSE then
17: $MOVE = FALSE$, $B_i = B_i \cup \{t_e\}$and break
18: end if
19: end for
20: end if
21: if MOVE then
22: for all $p \in P_i$, such that $t_{ipk} = t_e - 1$ do
23: $S_i \leftarrow Move\_Later(i; p; k; t_{ipk}; S_i)$
24: end for
25: for all $p \in P_i$, such that $t_{ipk} = t_e + 1$ do
26: $S_i \leftarrow Move\_Earlier(i; p; k; t_{ipk}; S_i)$
27: end for
28: end if
29: end if
30: end while

La Figura 4.6 mostra il comportamento della procedura SHIFT.



a) Initial Delivery Planning

b) Shift Phase

c) Delivery Planning after Shift Phase

Figura 4.6: Shift Phase

La procedura SHIFT è utilizzata nella fase di perturbazione dell'algoritmo ILS-MIRP. Essa mira a compattare i giorni di consegna distribuiti nell'orizzonte temporale. Tale procedura esegue una semplice mossa che consiste nell'aggregare le consegne in una serie di periodi consecutivi. I passi implementati sono descritti di seguito:

1. si determina il primo *empty day* t_e non ancora etichettato nell'orizzonte di pianificazione. Se nessun *empty day* è stato trovato la procedura si arresta;
2. si verifica che tutte le consegne che hanno luogo prima e dopo t_e possano essere spostate in t_e, nel rispetto dei vincoli (27). Ogni volta che questa condizione non è soddisfatta si etichetta t_e come tabù e si torna al passo 1), altrimenti
3. si seleziona casualmente lo spostamento da eseguire e si esegue tale spostamento: se lo spostamento scelto è quello di anticipare tutte le consegne di $t_e + 1$ in t_e , si realizza tale spostamento; in alternativa, lo spostamento, selezionato casualmente, consiste nel ritardare tutte le consegne dal periodo $t_e - 1$ a t_e. Alla fine di questo passo si torna al passo 1).

I passi della procedura SHIFT sono descritti nell'Algoritmo 4.6, riportato qui sotto:

Algoritmo 4.6: MIRP - Shift

Require: Customer i, set S_i and planning horizon T
Ensure: Set S_i
1: Let $B_i = 0$;
2: while TRUE do
3: $ND_i \leftarrow Get\_OrderNumber(i; P_i)$
4: $t_e \leftarrow Get\_Empty(i; T; B_i)$.
5: if $t_e = 0$ then
6: break
7: else
8: Set $MOVE\_EARLIER = MOVE\_LATER = TRUE$
9: for all $p \in P_i$ such that $t_{ipk} = t_e + 1$ do
10: if $Check\_Earlier(i; p; k; t_{ipk}) = FALSE$ then
11: $MOVE\_EARLIER = FALSE$
12: end if
13: end for
14: for all $p \in P_i$ such that $t_{ipk} = t_e - 1$ do
15: if $Check\_Later(i; p; k; t_{ipk}) = FALSE$ then
16: $MOVE\_LATER = FALSE$
17: end if
18: end for
19: if $MOVE\_EARLIER = MOVE\_LATER = FALSE$ then
20: $B_i = B_i \cup \{t_e\}$
21: end if
22: if $MOVE\_EARLIER = MOVE\_LATER = TRUE$ then
23: if Random > 0:5 then
24: $MOVE\_EARLIER = FALSE$
25: else
26: $MOVE\_LATER = FALSE$
27: end if
28: end if
29: if MOVE_EARLIER then
30: for all $p \in P_i$, such that $t_{ipk} = t_e + 1$ do
31: $S_i \leftarrow Move\_Earlier(i; p; k; t_{ipk}; S_i)$
32: end for
33: end if
34: if MOVE LATER then
35: for all $p \in P_i$, such that $t_{ipk} = t_e - 1$ do
36: $S_i \leftarrow Move\_Later(i; p; k; t_{ipk}; S_i)$
37: end for
38: end if
39: end if
40: end while

4.4.1 Algoritmo di ILS (Iterated Local Search)

La metaeuristica Iterated Local Search (ILS) è un metodo di ricerca locale che genera una sequenza di soluzioni di un problema di ottimizzazione combinatoria mediante l'utilizzo di una euristica di supporto, ed è in grado di produrre

soluzioni ammissibili di qualità superiore rispetto a quelle che potrebbero essere generate attraverso l'esecuzione ripetuta e casuale della medesima euristica di base. Il framework della ILS può essere sinteticamente definito in quanto segue. Sia:

- Θ un insieme di soluzioni ammissibili;
- ϑ una soluzione di Θ;
- J: $\Theta \rightarrow R$ una funzione obiettivo e $J(\vartheta)$ il valore della funzione obiettivo corrispondente alla soluzione ϑ;
- Θ^* un insieme di soluzioni ottime locali ϑ^*;
- LS: $\Theta \rightarrow \Theta^*$ una funzione di ricerca locale da Θ a Θ^*.

L'idea alla base della ILS è il concetto di ricerca diagonale in Θ. In pratica, partendo da una soluzione di ottimo locale ϑ^*, una nuova soluzione ϑ' può essere ottenuta effettuando una perturbazione di ϑ^*. Di conseguenza, una nuova soluzione ottima locale può essere ottenuta come $LS(\vartheta') = \vartheta'^*$. Infine, la migliore soluzione ottima locale, selezionata in Θ^*, è restituita come migliore soluzione corrente $\vartheta^*_{\text{best}}$.

Nel contesto del MIRP è stato progettato un algoritmo basato sulla ILS relativamente alla fase di pianificazione delle consegne e della determinazione dei volumi di cui rifornire i clienti per evitare lo stock-out. Di seguito la descrizione dei passi dell'algoritmo. Per ogni cliente:

1. Si definisce una soluzione iniziale consistente in un piano di consegne per l'intero periodo di pianificazione. Tale soluzione può essere ottenuta applicando, ad esempio, le indicazioni fornite dalle politiche standard di rifornimento dei clienti;
2. si invoca la procedura MERGE sulla base dei piani di consegna ottenuti al passo precedente con l'obiettivo di ridurre, sull'intero periodo di pianificazione, il numero di periodi in cui sono previste consegne;
3. si applica la procedura HC-MIRP per migliorare il piano delle consegne ottenuto al passo precedente;
4. si invoca la procedura SHIFT per perturbare la soluzione ottenuta al passo precedente;
5. se la procedura SHIFT genera una nuova soluzione perturbata (diversa da quella ottenuta con la procedura HC-MIRP), occorre ripetere i passi 3 e 4, altrimenti ci si arresta.

L'*outline* della procedura è descritta nell'Algoritmo 4.7.

Algoritmo 4.7: ILS-MIRP

Require: Sets M, S, graph G, fleet of m vehicles with capacity Q, and planning horizon T
Ensure: Set R of delivery routes scheduled over T and the updated set S. Initially set R =0;
1: for all $i \in M$ do
2: Let S_i be the set of reorder sizes delivered to customer i over T, and inizialized by performing procedure $d_{ip} \leftarrow Get\ Order(i;\ p; \Pi_p)$ for each $p \in P_i$and for each $t \in T$
3: Let R_i be the delivery routing for customer i. Initially set $R_i = \emptyset$;
4: Let S_i^p be the perturbed set of reorder sizes at customer i. Initially set $S_i^p = \emptyset$;
5: $S_i = Merge(i; S_i; T)$
6: while TRUE do
7: S_i = returned from step 3 to step 26 of HC-MIRP algorithm
8: S_i^p=$Shift(i; S_i; T)$
9: if $S_i^p = S_i$ then
10: break
11: else
12: $S_i = S_i^p$
13: end if
14: end while
15: end for
16: $S = S \cup \bigcup_{i \in M} S_i$
17: for all $t \in T$ do
18: if $S^t \neq 0$ then
19: Initialize M^t and P^t
20: $R = R \cup R^t \leftarrow Routing(M^t; P^t; S^t; G; m; Q)$
21: end if
22: end for

4.5 L'agoritmo di routing con finestre temporali

A valle della fase di pianificazione sono note le informazioni necessarie per organizzare le consegne ai clienti nei periodi dell'orizzonte temporale. Per fare questo in modo ottimale è necessario risolvere un problema di instradamento con vincoli di natura operativa che, nel caso preso in esame, sono vincoli di finestre temporali. L'algoritmo di instradamento che è stato proposto visita i singoli clienti in modo tale che le quantità di ciascun prodotto da consegnare non siano spacchettate su più veicoli. Lo schema generale è il seguente:

Algoritmo 4.8: Routing
While (Stopping criteria satisfied){
_r_=Create Route;
_r_=Optimize Route (r) (local search using 2-opt and/or 3 -opt)
}

Il precedente schema illustra i passi fondamentali dell'algoritmo proposto. Nella prima fase l'algoritmo determina un insieme di rotte *R*, che rispetteranno "classici" i vincoli operativi relativi a:

- capacità massima del veicolo (Q);
- finestre temporali dei singoli clienti;
- lunghezza temporale massima delle singole rotte (WT: Working Time).

Nella successiva fase interviene una procedura di ricerca locale basata sull'esplorazione di vicinati classici, ottenuti tramite mosse di tipo 2-opt e 3-opt, con l'obiettivo di ottimizzare le rotte ottenute nella prima fase. Il criterio d'arresto scelto è quello di terminare la ricerca quando, per un numero di iterazione prefissato, non si migliora la migliore soluzione corrente. La determinazione dell'insieme delle rotte *R* è realizzata, a sua volta, attraverso due macro procedure, una innestata nell'altra. Nella prima si seleziona il cliente (*c*) che deve essere servito. Di ciascun cliente è nota la schedula delle consegne dei differenti prodotti ($i \in P_c$, dove P_c è la lista dei prodotti da consegnare al cliente *c*), ognuno dei quali con un determinato volume (w_i^c)). Nella seconda procedura si determina il numero di rotte necessario ad effettuare le consegne in modo tale da minimizzare la distanza percorsa e massimizzare la capacità del veicolo. Per fare ciò, tutti i clienti sono inseriti in una lista ordinata (*OL*), secondo un certo criterio (per esempio, per ampiezza non decrescente della finestra temporale e, a parità di ampiezza, dando priorità maggiore al cliente con l'istante d'inizio della finestra temporale più piccolo). Il cliente selezionato è quello estratto in modo stocastico da tale lista (OL), alla quale resta associata una funzione di distribuzione di probabilità, lineare a tratti, tale che i clienti che stanno in cima alla lista sono quelli con la probabilità maggiore di essere estratti. Questo non esclude che anche i clienti che stanno in fondo alla lista possano essere estratti, avendo anch'essi una probabilità maggiore di zero, minore di quella dei clienti che stano in cima alla lista. La ragione prevalente per cui si è deciso di utilizzare un criterio stocastico di questo tipo è che esso garantisce un'azione di "recovery" nei confronti di un errato ordinamento dei clienti. Lo schema algoritmico della routine per la determinazione delle rotte è riportato di seguito:

Algoritmo 4.9: Create Route
While(OL is not empty){
Client c=stochastic selection of a client from OL
r = selec from R all route that satisfied the following constraint:
- **Time windows constrint of cliente c,**
- **distance traveled for the insertion of client c is less the distance traveled with a direct delivery**
- **Capacity constraint: residual capacity is greater then min(wic, $\forall\, i \in P_c$)**

R = R /{r}
r_{DD}=Calculate minimum nuber of Direc Deliver necessary to deliver all product $i \in P_c$
r= r $\cup$ r_{DD}
r = Enhance Route (r, P_c)
R = R $\cup$ r

}

Ogni volta che la sequenza di passi descritta nella routine, denominata Create Route (Algoritmo 4.9), è eseguita, si determina un insieme *R* di rotte differente, per via della selezione stocastica del cliente dalla lista ordinata (OL). La procedura con cui si cerca di compattare il numero di rotte necessarie per le consegne risolve un problema combinatorio, noto in letteratura come problema di impaccamento con dimensione variabile degli oggetti e con un costo fisso (Variable Size Bin Packing Problem with Fixed Cost), a cui si farà riferimento nel seguito utilizzando l'acronimo VSBPPFC. Il motivo per cui ci si riconduce a questo problema deriva dal fatto che le rotte selezionate r hanno una capacità residua variabile e sicuramente minore di Q, inoltre ogni Bin (veicolo) ha un costo differente da quello degli altri, che coincide con la distanza da percorrere per servire il cliente c. L'obbiettivo è quello di ottenere un insieme delle rotte che minimizzino la distanza percorsa ($d(r)$), e massimizzino la capacità del veicolo, secondo la seguente funzione obiettivo:

$$\min k_1 \sum_{r \in R} d(r) + k_2 \sum_{r \in R} \min (0, Q - Q(r)) \qquad (1)$$

dove $Q(r)$ è il volume degli oggetti trasportati nella rotta e $d(r)$ è la distanza percorsa dalla rotta r, mentre $Q - Q(r)$ è la capacità residua della rotta r.

Il VSBPPFC è stato risolto modificando l'algoritmo A-BFD di Crainic et al. (2010). Il nostro algoritmo si trova descritto nella outline riportata di seguito:

Algoritmo 4.10: Enhance Route
1: Input I: Items to accommodated into route
2: Input K: Route available to load items
3: S: Set of selected route (empty at beginning)
4: Sort the item in I according to non-increasing order of their volume
5: Sort the route in K according to non-increasing order of ratio c_j/V_j and non-increasing order of V_j when the unit costs c_j are equal.
6: $S = \{\emptyset\}$
7: for all $i \in I$ do
8: **if i can be accommodate into a route S then**
9: **Accommodate i into the best route $b \in S$**
10: **else**
11: **$S = S \cup \{b'\}$, when b' is the first route in the ordered list k**
12: **Accommodate i into b'**
13: **end if**
14: end for
15: for all $j \in S$ do
16: **for all $k \in K \backslash S$ do**
17: **$U_j = \sum_{i \text{ loaded in } j} u_i$**
18: **if $V_k \geq U_j$ and $c_k < c_j$ then**
19: **Move all the item from j to k**
20: **$S = S \backslash \{j\} \cup \{k\}$**
21: **end if**
22: **end for**
23: return S

4.6 Risultati Computazionali

Il problema reale che abbiamo preso in esame è caratterizzato da un insieme di clienti pari a qualche decina mentre il un numero di prodotti per cliente è all'incirca di un centinaio. La bontà dell'approccio proposto è stata validata su tre tipi di problemi test presenti in letteratura, con un numero di clienti variabile da 50 a 200. Le istanze più piccole con 50 clienti sono quelle definite da Bertazzi et al. [2002]. Poiché tali istanze fanno riferimento ad un contesto diverso da quello trattato nel presente capitolo (singolo prodotto ed un solo veicolo), per ciascuna di esse è stato generato un planning con 100 prodotti per ogni cliente. Per quanto riguarda le istanze con un numero di clienti maggiore di 50, è

stata condotta una vasta sperimentazione computazionale su tutti i tipi di istanze (R1, R2, C1, C2, RC1, RC2) proposte da Solomon per il problema di instradamento con vincoli capacitivi e di finestre temporali. Tali istanze fanno riferimento a grafi con un numero di clienti pari a 100. Come per le istanze di Bertazzi et al. [2002], anche per esse è stato generato per ogni cliente un planning di 100 prodotti. Infine, una sperimentazione computazionale su istanze con un numero di clienti superiore a 100 e con un planning di 100 prodotti per cliente è stata condotta a partire dalle istanze di tipo RC1 di Homberger, anch'esse generate per il problema di instradamento con vincoli capacitivi e di finestre temporali. L'orizzonte di pianificazione (*H*) che abbiamo utilizzato è di 14 giorni.

Primo di tutto vengono presentati i risultati del confronto tra il bound "euristico " e la tabu search per le split delivery proposta da Ho and Haugland 2004, tale test mira ha illustrare la bontà dell'algoritmo di routing proposto dimostrando che la distanza percorsa dal nostro algoritmo, che non effettua lo split della domanda, non è molto peggiore rispetto all'algoritmo con lo split delivery. Nella Tabella 4.4 viene riportata la sperimentazione sulle istanze con un numero di customer pari a 50.

		"Heuristic" Bound					Tabù Search "Split Delivery"				
	Name istance	Q	V	Q/V	% vehicle	time (sec.)	Q	V	%Gap (V)	% vehicle	time (sec.)
1	abs1n50,dat_P_100N_50_1,dat	910434	592889	1,535589	97,091	6,461	910434	573644	-3,35%	99,87888601	0
2	abs2n50,dat_P_100N_50_2,dat	915458	601263	1,522558	96,91263	18,713	915458	582913	-3,15%	99,91116175	0
3	abs3n50,dat_P_100N_50_3,dat	909426	469020	1,938992	96,84182	6,746	909426	453742	-3,37%	99,95801312	0
4	abs4n50,dat_P_100N_50_4,dat	914395	555153	1,647104	96,89015	10,789	914395	536904	-3,40%	99,97714708	0
5	abs5n50,dat_P_100N_50_5,dat	911756	517694	1,761187	96,83194	8,08	911756	501164	-3,30%	99,94782029	0

Tabella 4.4: Confronto tra il bound "euristico " e la tabu search, per le istanze con 50 clienti

Come si può vedere dalla Tabella 4.4, l'algoritmo con lo split delivery, che trova una soluzione "infeasible" per il nostro problema, percorre una distanza minore di circa il 3% rispetto alla distanza corrispondente alla soluzione calcolata con il nostro algoritmo, tale valore assicura che l'algoritmo proposto è in grado di determinare una buona soluzione. La colonna Q/V che riporta i valori calcolati prendendo in considerazione il volume trasportato e la distanza percorsa, ottenuta utilizzando l'algoritmo di instradamento proposto, rappresenta un "bound" per il MIRP.

		IIS+alg Routing						"Heuristic" Bound	
	Name istance	Q	V	Q/V	% vehicle	time inve	time routing	Q/V	Compare Q/V with bound
1	abs1n50,dat_P_100N_50_1,dat	1236771	830257	1,49	96,51807	0	54	1,54	-2,99%
2	abs2n50,dat_P_100N_50_2,dat	1237810	836470	1,48	96,51134	0	42	1,52	-2,81%
3	abs3n50,dat_P_100N_50_3,dat	1229727	656012	1,87	96,39187	0	40	1,94	-3,32%
4	abs4n50,dat_P_100N_50_4,dat	1240984	774671	1,60	96,50193	0	46	1,65	-2,74%
5	abs5n50,dat_P_100N_50_5,dat	1236275	724154	1,71	96,3924	0	44	1,76	-3,07%

Tabella 4.5: Confronto tra l'approccio euristico per il MIRP e il bound, per le istanze con 50 clienti

Come si evince dalla Tabella 4.5, confrontando il valore Q/V ottenuto risolvendo il MIRP con l'approccio proposto, con il bound euristico ottenuto, la distanza percentuale è intorno al 3% ciò ci garantisce che la soluzione ottenuto è piuttosto buona.

		"Heuristic" Bound						Tabù Search "Split Delivery"				
	Name istance	Q	V	Q/V	% vehicle	num Rout	time	Q	V	%Gap (V)	% vehicle	time
1	rc101,txt_P_100N_100_1,dat	1826405	111709	16,34966	96,81406	1668	20,43	1826405	108272,3	3,17%	99,9573	0
2	rc102,txt_P_100N_100_2,dat	1822059	110439,7	16,49822	97,04295	1647	25,555	1822059	107637,6	2,60%	99,95606	0
3	rc103,txt_P_100N_100_3,dat	1819902	108554,4	16,76489	96,82086	1619	13,7	1819902	106031,8	2,38%	99,96792	0
4	rc104,txt_P_100N_100_4,dat	1820187	110769,9	16,43215	96,96354	1651	25,375	1820187	108766,6	1,84%	99,99176	0
5	rc105,txt_P_100N_100_5,dat	1823627	111085,4	16,41643	96,8757	1660	27,726	1823627	108339	2,54%	99,94634	0
6	rc106,txt_P_100N_100_6,dat	1823099	111148,5	16,40237	97,02785	1654	27,121	1823099	108664,4	2,29%	99,99007	0
7	rc107,txt_P_100N_100_7,dat	1824280	109143,6	16,71449	97,00562	1624	14,07	1824280	106470,6	2,51%	99,96011	0
8	rc108,txt_P_100N_100_8,dat	1824755	108721,4	16,78377	96,89949	1622	18,113	1824755	105947,8	2,62%	99,98154	1
1	c101,txt_P_100N_100_1,dat	1826405	97335,56	18,76401	96,93028	1666	20,622	1826405	95547	1,87%	99,99124	0
2	c102,txt_P_100N_100_2,dat	1822059	96246,78	18,93112	96,98406	1648	22,783	1822059	93830,88	2,57%	99,95606	0
3	c103,txt_P_100N_100_3,dat	1819902	94639,67	19,2298	96,94061	1617	37,902	1819902	92044,22	2,82%	99,97001	0
4	c104,txt_P_100N_100_4,dat	1820187	96385,86	18,88438	97,02231	1650	23,931	1820187	94219,49	2,30%	99,99176	0
5	c105,txt_P_100N_100_5,dat	1823627	96975,02	18,80512	96,9341	1659	13,193	1823627	94567,07	2,55%	99,94634	0
6	c106,txt_P_100N_100_6,dat	1823099	96647,49	18,86339	97,02785	1654	20,646	1823099	94374,84	2,41%	99,99007	0
7	c107,txt_P_100N_100_7,dat	1824280	94979,7	19,20705	96,94593	1625	18,468	1824280	92405,9	2,79%	99,96011	0
8	c108,txt_P_100N_100_8,dat	1824755	94658,93	19,27716	96,89949	1622	25,823	1824755	92961,7	1,83%	99,98154	0
9	c109,txt_P_100N_100_9,dat	1825409	94973,84	19,22012	96,91086	1628	28,586	1825409	93241,94	1,86%	99,98154	0
1	r101,txt_P_100N_100_1,dat	1826405	84160,76	21,70138	96,87214	1667	16,343	1826405	82529,41	1,98%	99,95327	0
2	r102,txt_P_100N_100_2,dat	1822059	83249,79	21,88665	97,10191	1646	17,294	1822059	81707,47	1,89%	99,95606	0
3	r103,txt_P_100N_100_3,dat	1819902	81767,33	22,25708	96,94061	1617	29,182	1819902	80084,07	2,10%	99,95095	0
4	r104,txt_P_100N_100_4,dat	1820187	83466,71	21,80734	96,96354	1651	24,225	1820187	81833,99	2,00%	99,99176	0
5	r105,txt_P_100N_100_5,dat	1823627	83909,46	21,73327	96,99256	1658	14,713	1823627	81534,87	2,91%	99,94357	1
6	r106,txt_P_100N_100_6,dat	1823099	83693,33	21,78309	96,85218	1657	27,176	1823099	81597,02	2,57%	99,94356	0
7	r107,txt_P_100N_100_7,dat	1824280	82286,85	22,16976	96,94593	1625	36,725	1824280	80442,3	2,29%	99,95392	0
8	r108,txt_P_100N_100_8,dat	1824755	81931,22	22,27179	96,95927	1621	27,072	1824755	81016,48	1,13%	99,98154	0
9	r109,txt_P_100N_100_9,dat	1825409	82397,66	22,15365	96,85136	1629	20,501	1825409	80516,94	2,34%	99,95142	0
10	r110,txt_P_100N_100_10,dat	1815010	82832,85	21,91172	97,00276	1637	27,67	1815010	81962,8	1,06%	99,99592	0
11	r111,txt_P_100N_100_11,dat	1821538	83343,39	21,85582	96,92383	1650	23,361	1821538	81212,88	2,62%	99,9527	1
12	r112,txt_P_100N_100_12,dat	1822732	82782,32	22,01837	96,90677	1637	38,326	1822732	81088,31	2,09%	99,95991	0
1	C201,txt_P_100N_100_1,dat	1826405	100276,4	18,21371	96,87214	1667	23,301	1826405	98859,3	1,43%	99,99124	0
2	C202,txt_P_100N_100_2,dat	1822059	99077,39	18,39026	97,04295	1647	29,891	1822059	96936,69	2,21%	99,95606	0
3	C203,txt_P_100N_100_3,dat	1819902	97454,61	18,67435	97,0006	1616	14,451	1819902	94919,17	2,67%	99,97001	0
4	C204,txt_P_100N_100_4,dat	1820187	99231,44	18,34285	96,90485	1652	14,66	1820187	97019,62	2,28%	99,99176	1
5	C205,txt_P_100N_100_5,dat	1823627	99903,63	18,25386	96,99256	1658	21,314	1823627	97345,72	2,63%	99,94634	0
6	C206,txt_P_100N_100_6,dat	1823099	99622,82	18,30001	96,96922	1655	35,036	1823099	97680,84	1,99%	99,98733	0
7	C207,txt_P_100N_100_7,dat	1824280	97815,66	18,65018	97,00562	1624	13,224	1824280	95295,74	2,64%	99,96011	0
8	C208,txt_P_100N_100_8,dat	1824755	97570,45	18,70192	96,95927	1621	26,746	1824755	95653,97	2,00%	99,98154	0
1	RC201,txt_P_100N_100_1,dat	1826405	111784,4	16,33863	96,87214	1667	14,813	1826405	109929,9	1,69%	99,99124	0
2	RC202,txt_P_100N_100_2,dat	1822059	110384,8	16,50642	96,98406	1648	14,057	1822059	107887,6	2,31%	99,95606	0
3	RC203,txt_P_100N_100_3,dat	1819902	108552,9	16,76512	96,8807	1618	33,418	1819902	106079,1	2,33%	99,97001	0
4	RC204,txt_P_100N_100_4,dat	1820187	110695,3	16,44322	96,84622	1653	14,409	1820187	108766,6	1,77%	99,99176	0
5	RC205,txt_P_100N_100_5,dat	1823627	111178,6	16,40267	96,99256	1658	15,33	1823627	108484	2,48%	99,94634	0
6	RC206,txt_P_100N_100_6,dat	1823099	111044,4	16,41774	96,96922	1655	18,526	1823099	108832,1	2,03%	99,99007	0
7	RC207,txt_P_100N_100_7,dat	1824280	109071	16,72562	96,94593	1625	13,625	1824280	106705	2,22%	99,96011	0
8	RC208,txt_P_100N_100_8,dat	1824755	108747,2	16,77979	96,89949	1622	15,499	1824755	105849	2,74%	99,98154	1
1	R201,txt_P_100N_100_1,dat	1826405	84202,99	21,6905	96,87214	1667	28,439	1826405	82536,96	2,02%	99,99124	0
2	R202,txt_P_100N_100_2,dat	1822059	83335,69	21,86409	96,92525	1649	28,098	1822059	81826,23	1,84%	99,95606	0
3	R203,txt_P_100N_100_3,dat	1819902	81960,74	22,20456	96,8807	1618	16,412	1819902	79887,44	2,60%	99,97001	0
4	R204,txt_P_100N_100_4,dat	1820187	83428,91	21,81722	96,96354	1651	24,361	1820187	81672,37	2,15%	99,99176	0
5	R205,txt_P_100N_100_5,dat	1823627	83780,5	21,76672	96,9341	1659	22,495	1823627	82246,04	1,87%	99,94634	0
6	R206,txt_P_100N_100_6,dat	1823099	83714,42	21,7776	96,91067	1656	19,142	1823099	82524,23	1,44%	99,99007	0
7	R207,txt_P_100N_100_7,dat	1824280	82244,15	22,18127	96,94593	1625	20,795	1824280	80560,3	2,09%	99,96011	0
8	R208,txt_P_100N_100_8,dat	1824755	82102,82	22,22524	96,83979	1623	16,714	1824755	81116,8	1,22%	99,98154	0
9	R209,txt_P_100N_100_9,dat	1825409	82310,4	22,17714	96,91086	1628	15,386	1825409	81002,6	1,61%	99,98154	0
10	R210,txt_P_100N_100_10,dat	1815010	82765,79	21,92947	97,00276	1637	26,03	1815010	81065,51	2,10%	99,99592	1
11	R211,txt_P_100N_100_11,dat	1821538	83362,41	21,85083	96,92383	1650	35,001	1821538	81884,05	1,81%	99,9527	0

Tabella 4.6:Confronto tra il bound euristico e lo TS, per le istanze con 100 clienti

Come si può vedere dalla Tabella 4.6, l'algoritmo con lo split delivery, determina una distanza inferiore che è sempre minore del 3% (tranne in un caso) rispetto alla soluzione calcolata con il nostro algoritmo, tale valore ci garantisce che il nostro algoritmo sta producendo una buona soluzione. Per la colonna Q/V che è calcolata prendendo in considerazione il volume trasportato e la distanza percorsa utilizzando l'algoritmo di instradamento proposto rappresenta un "bound" per il MIRP.

		IIS+alg Routing						"Heuristic" Bound	
	Name istance	Q	V	Q/V	% vehicle	time inve	time routing	Q/V	Compare Q/V with bound
1	rc101,txt_P_100N_100_1,dat	2467593	157060,9	15,71106	96,11365	0	103	16,34966	-3,91%
2	rc102,txt_P_100N_100_2,dat	2469661	156026,8	15,82844	96,32589	0	105	16,49822	-4,06%
3	rc103,txt_P_100N_100_3,dat	2475846	153897,6	16,08762	96,27592	0	100	16,76489	-4,04%
4	rc104,txt_P_100N_100_4,dat	2471977	157272,1	15,71784	96,28531	0	96	16,43215	-4,35%
5	rc105,txt_P_100N_100_5,dat	2471921	156981,3	15,7466	96,32455	0	117	16,41643	-4,08%
6	rc106,txt_P_100N_100_6,dat	2471371	156936,2	15,74762	96,26118	0	92	16,40237	-3,99%
7	rc107,txt_P_100N_100_7,dat	2473521	154001,9	16,06163	96,39118	0	92	16,71449	-3,91%
8	rc108,txt_P_100N_100_8,dat	2475957	154419,9	16,03393	96,28024	0	122	16,78377	-4,47%
1	c101,txt_P_100N_100_1,dat	2467593	136936	18,02004	96,36837	0	88	18,76401	-3,96%
2	c102,txt_P_100N_100_2,dat	2469661	136196,2	18,13311	96,54052	0	112	18,93112	-4,22%
3	c103,txt_P_100N_100_3,dat	2475846	133934,7	18,48547	96,53742	0	78	19,2298	-3,87%
4	c104,txt_P_100N_100_4,dat	2471977	136357,2	18,12868	96,58473	0	90	18,88438	-4,00%
5	c105,txt_P_100N_100_5,dat	2471921	137186,5	18,01869	96,282	0	82	18,80512	-4,18%
6	c106,txt_P_100N_100_6,dat	2471371	136284,2	18,13394	96,56026	0	126	18,86339	-3,87%
7	c107,txt_P_100N_100_7,dat	2473521	134198,4	18,43182	96,30426	0	89	19,20705	-4,04%
8	c108,txt_P_100N_100_8,dat	2475957	134486	18,41052	96,36725	0	95	19,27716	-4,50%
9	c109,txt_P_100N_100_9,dat	2474025	134553,2	18,38696	96,40714	0	92	19,22012	-4,33%
1	r101,txt_P_100N_100_1,dat	2467593	118709,3	20,78686	96,11365	0	116	21,70138	-4,21%
2	r102,txt_P_100N_100_2,dat	2469661	117857,3	20,95466	96,32589	0	87	21,88665	-4,26%
3	r103,txt_P_100N_100_3,dat	2475846	116788,6	21,19938	95,97262	0	79	22,25708	-4,75%
4	r104,txt_P_100N_100_4,dat	2471977	118988,4	20,77495	96,2001	0	76	21,80734	-4,73%
5	r105,txt_P_100N_100_5,dat	2471921	118785,3	20,80999	96,282	0	80	21,73327	-4,25%
6	r106,txt_P_100N_100_6,dat	2471371	118676,3	20,82447	96,09111	0	75	21,78309	-4,40%
7	r107,txt_P_100N_100_7,dat	2473521	116595,6	21,21453	96,4347	0	86	22,16976	-4,31%
8	r108,txt_P_100N_100_8,dat	2475957	116656,4	21,22436	96,28024	0	124	22,27179	-4,70%
9	r109,txt_P_100N_100_9,dat	2474025	116851,8	21,17232	96,32028	0	105	22,15365	-4,43%
10	r110,txt_P_100N_100_10,dat	2470359	117812,6	20,96854	96,44328	0	137	21,91172	-4,30%
11	r111,txt_P_100N_100_11,dat	2471978	118396,5	20,87882	96,20149	0	112	21,85582	-4,47%
12	r112,txt_P_100N_100_12,dat	2472629	117638,7	21,01885	96,41503	0	100	22,01837	-4,54%
1	C201,txt_P_100N_100_1,dat	2467593	141183,1	17,47796	96,41095	0	117	18,21371	-4,04%
2	C202,txt_P_100N_100_2,dat	2469661	140198	17,61552	96,41163	0	117	18,39026	-4,21%
3	C203,txt_P_100N_100_3,dat	2475846	138497,4	17,87648	96,49374	0	96	18,67435	-4,27%
4	C204,txt_P_100N_100_4,dat	2471977	140999,2	17,53185	96,37067	0	113	18,34285	-4,42%
5	C205,txt_P_100N_100_5,dat	2471921	141514,5	17,46762	96,15459	0	120	18,25386	-4,31%
6	C206,txt_P_100N_100_6,dat	2471371	141072,6	17,51844	96,2186	0	102	18,30001	-4,27%
7	C207,txt_P_100N_100_7,dat	2473521	138423,9	17,86917	96,39118	0	88	18,65018	-4,19%
8	C208,txt_P_100N_100_8,dat	2475957	138631,4	17,86	96,19338	0	81	18,70192	-4,50%
1	RC201,txt_P_100N_100_1,dat	2467593	157281,3	15,68904	96,24084	0	92	16,33863	-3,98%
2	RC202,txt_P_100N_100_2,dat	2469661	156082,5	15,8228	96,28308	0	97	16,50642	-4,14%
3	RC203,txt_P_100N_100_3,dat	2475846	153899,1	16,08746	96,36293	0	93	16,76512	-4,04%
4	RC204,txt_P_100N_100_4,dat	2471977	157247,7	15,72027	96,24268	0	93	16,44322	-4,40%
5	RC205,txt_P_100N_100_5,dat	2471921	157361,8	15,70853	96,282	0	98	16,40267	-4,23%
6	RC206,txt_P_100N_100_6,dat	2471371	156936,1	15,74763	96,34644	0	87	16,41774	-4,08%
7	RC207,txt_P_100N_100_7,dat	2473521	154154,7	16,04571	96,30426	0	73	16,72562	-4,07%
8	RC208,txt_P_100N_100_8,dat	2475957	154202,9	16,05649	96,19338	0	91	16,77979	-4,31%
1	R201,txt_P_100N_100_1,dat	2467593	119221,8	20,69749	96,28331	0	119	21,6905	-4,58%
2	R202,txt_P_100N_100_2,dat	2469661	118216,5	20,891	96,41163	0	96	21,86409	-4,45%
3	R203,txt_P_100N_100_3,dat	2475846	116171,6	21,31197	96,18907	0	124	22,20456	-4,02%
4	R204,txt_P_100N_100_4,dat	2471977	118278,6	20,89962	96,32797	0	121	21,81722	-4,21%
5	R205,txt_P_100N_100_5,dat	2471921	118820,3	20,80386	96,32455	0	108	21,76672	-4,42%
6	R206,txt_P_100N_100_6,dat	2471371	118693,9	20,82139	96,13357	0	110	21,7776	-4,39%
7	R207,txt_P_100N_100_7,dat	2473521	116604,6	21,21289	96,39118	0	117	22,18127	-4,37%
8	R208,txt_P_100N_100_8,dat	2475957	116702,4	21,21599	96,41082	0	100	22,22524	-4,54%
9	R209,txt_P_100N_100_9,dat	2474025	116685,6	21,20249	96,1903	0	86	22,17714	-4,39%
10	R210,txt_P_100N_100_10,dat	2470359	117930,2	20,94764	96,18576	0	75	21,92947	-4,48%
11	R211,txt_P_100N_100_11,dat	2471978	118564	20,84932	96,11628	0	82	21,85083	-4,58%

Tabella 4.7: Confronto tra l'approccio euristico per il MIRP ed il Buond, per le istanze con 100 clienti

Come si può vedere dalla Tabella 4.7, confrontando il valore Q/V ottenuto risolvendo il il MIRP con l'approccio proposto, con il bound euristico ottenuto, la distanza percentuale è intorno al 4% ciò ci garantisce che la soluzione ottenuto è piuttosto buona.

		"Heuristic" Bound						TS Split Bound				
	Name istance	Q	V	Q/V	% vehicle	compare %	time	Q	V	%Gap (V)	% vehicle	time
1	RC1_2_1,txt_P_100N_200_1,dat	3649231	348994,3	10,45642	96,93393	3,05%	86,299	3649231	340371,2	2,53%	99,97849	2
2	RC1_2_2,txt_P_100N_200_2,dat	3636024	354776,7	10,24877	96,9864	2,99%	44,56	3636024	347635	2,05%	99,98037	2
3	RC1_2_3,txt_P_100N_200_3,dat	3644736	353063,7	10,32317	97,00722	2,97%	77,841	3644234	346408,2	1,92%	99,97174	1
4	RC1_2_4,txt_P_100N_200_4,dat	3643994	357325,6	10,19797	97,01175	2,97%	72,055	3643994	349107,2	2,35%	99,98425	5
5	RC1_2_5,txt_P_100N_200_5,dat	3648247	354591,6	10,28859	97,00261	2,97%	47,5	3648247	346785,6	2,25%	99,96961	1
6	RC1_2_6,txt_P_100N_200_6,dat	3647085	352199,4	10,35517	96,97082	3,02%	44,557	3647085	342957,9	2,69%	99,98972	4
7	RC1_2_7,txt_P_100N_200_7,dat	3643138	351371,4	10,36834	97,0138	2,96%	110,329	3643138	343240,9	2,37%	99,97582	3
8	RC1_2_8,txt_P_100N_200_8,dat	3641600	358003,2	10,17198	97,03279	2,94%	52,129	3641600	351350,3	1,89%	99,97227	1
9	RC1_2_9,txt_P_100N_200_9,dat	3643404	352292,2	10,34199	96,983	2,99%	38,628	3643404	343243,3	2,64%	99,96897	4
10	RC1_2_10,txt_P_100N_200_10,dat	3632066	354006,2	10,25989	96,97426	3,00%	32,603	3632066	345600	2,43%	99,97627	2

Tabella 4.8: Confronto tra il bound e l'algoritmo di split delivery, per le istanze con 200 clienti

Come si può vedere dalla Tabella 4.8, l'algoritmo con lo split delivery realizza una distanza inferiore che è sempre minore del 3% rispetto alla soluzione calcolata con il nostro algoritmo, tale valore ci garantisce che il nostro algoritmo sta producendo una buona soluzione. Per la colonna Q/V che è calcolata prendendo in considerazione il volume trasportato e la distanza percorsa utilizzando l'algoritmo di routing proposto rappresenta un "bound" per il MIRP.

		IlS+alg Routing						"Heuristic" Bound	
	Name istance	Q	V	Q/V	% vehicle	time inve	time routing	Q/V	Compare Q/V with bound
1	RC1_2_1,txt_P_100N_200_1,dat	4949628	493787,6	10,0238	96,7034	0	222	10,45642	-4,14%
2	RC1_2_2,txt_P_100N_200_2,dat	4939122	503553,7	9,80853	96,76709	0	160	10,24877	-4,30%
3	RC1_2_3,txt_P_100N_200_3,dat	4940033	499894,7	9,882148	96,77345	0	157	10,32317	-4,27%
4	RC1_2_4,txt_P_100N_200_4,dat	4947848	504852,3	9,800585	96,74438	0	151	10,19797	-3,90%
5	RC1_2_5,txt_P_100N_200_5,dat	4953138	501642,9	9,873832	96,76619	0	183	10,28859	-4,03%
6	RC1_2_6,txt_P_100N_200_6,dat	4941058	497828,3	9,925226	96,78297	0	192	10,35517	-4,15%
7	RC1_2_7,txt_P_100N_200_7,dat	4940387	497220,5	9,936008	96,68294	0	185	10,36834	-4,17%
8	RC1_2_8,txt_P_100N_200_8,dat	4939724	506418,3	9,754236	96,71442	0	191	10,17198	-4,11%
9	RC1_2_9,txt_P_100N_200_9,dat	4942317	498856,6	9,90729	96,80274	0	216	10,34199	-4,20%
10	RC1_2_10,txt_P_100N_200_10,dat	4952145	504679,2	9,812462	96,61531	0	184	10,25989	-4,36%

Tabella 4.9: Confronto tra l'algoritmo euristico per il MIRP e il bound, per le istanze con 200 clienti

Come si può vedere dalla Tabella 4.9, confrontando il valore Q/V ottenuto risolvendo il MIRP, con l'approccio proposto, con il bound euristico ottenuto, la distanza percentuale è intorno al 4% ciò ci garantisce che la soluzione ottenuto è piuttosto buona.

Gestione Combinata delle scorte e dell'instradamento dei veicoli in ambiente aleatorio

Lo studio degli approcci al problema combinato del magazzino e del trasporto merci viene completato in questo capitolo, affrontando gli aspetti della modellazione in ambito stocastico. Degli approcci di modellazione e di risoluzione presenti in letteratura per tale problema, noto come *Stochastic Vendor Managed Inventory* (SVMI), ne è stato scelto uno basato sui processi decisionali di Markov (MDP) non omogeneo, su un orizzonte finito. Il modello MDP per il problema SVMI diventa particolarmente impegnativo, da un punto di vista computazionale, con l'aumentare della dimensione del problema: la crescita delle dimensioni del problema deriva principalmente dall'aumento del numero di rivenditori, dalla capacità di stoccaggio dei rivenditori e dal tipo di veicolo utilizzato. Anche all'aumentare dell'orizzonte di pianificazione il processo MDP risulta di complessità computazionale crescente. Per gestire computazionalmente un crescente numero di rivenditori, appare ragionevole l'idea di raggruppare i rivenditori in gruppi disgiunti ed assegnare un veicolo a ciascun gruppo. In tal modo si ottengono tanti problemi SVMI quanti sono i gruppi di rivenditori. Se da una parte questo approccio potrebbe portare ad una soluzione non ottimale del problema, dall'altro permette al decisore di prendere in considerazione altri fattori (vincoli aggiuntivi), quali le distanze di viaggio e le ore di guida. Un approccio alternativo al raggruppamento menzionato prima è fondato sull'esistenza di relazioni monotone tra le azioni da dover attuare (sulle scorte) e i livelli di inventario presenti presso i singoli rivenditori; tali relazioni consentono di sviluppare euristiche di soluzione per il problema globale. Una tale euristica verrà presentata in questa sede e sperimentata su istanze campione del problema in esame. Quanto alle possibili applicazioni del problema SVMI, una abbastanza tipica nel mondo reale è quella di organizzare le consegne di carburanti liquidi o gassosi, preliminarmente stoccati in serbatoi dispersi in varie località perché pericolosi da stoccare in quantità elevate in un unico luogo. Ciò è particolarmente vero per una zona residenziale o campus universitario, dove serbatoi di stoccaggio dispersi sono di norma utilizzati. In tal caso, il venditore può utilizzare misuratori sofisticati per monitorare da remoto i livelli di riempimento di questi serbatoi ed intervenire prontamente a ripristinare il livello di sicurezza. Un'altra possibile applicazione del problema SVMI consiste nella distribuzione di denaro agli sportelli automatici, dove per problemi di sicurezza, deve essere tenuta una quantità limitata di denaro (che, d'altra parte, viene monitorato facilmente da remoto). Il requisito di puntualità nelle consegne di denaro agli sportelli bancomat è particolarmente stringente perché i clienti devono essere in grado di accedere ai propri conti liberamente.

5.1 Literature Review

White and White (1989) realizzarono un'utile *review* sui processi decisionali di Markov. Il paper di Yang et al. (2000) è un recente studio sul problema del *vehicle routing* stocastico che ha alcune caratteristiche simili al problema del SVMI. Nel problema di distribuzione indagato dagli autori, che si riferisce allo *stochastic vehicle routing problem (SVRP) with restocking*, includono un deposito e più rivenditori. L'obiettivo è quello di determinare un percorso ottimale per il veicolo al fine di visitare i rivenditori ed il tempo in cui il veicolo ritorni al deposito per il rifornimento prima di visitare il rivenditore successivo. I costi in esame sono i costi di trasporto e il costo di rifornimento. Le differenze tra l'SVRP *with restocking* ed il problema dell'SVMI consistono sulla disponibilità delle informazioni dello stato e sulla selezione delle vie in cui re-instradare i veicoli. Nel SVRP vengono incluse nel tour anche le visite al deposito per effettuare il rifornimento, poiché ciò viene fissato prima che il veicolo lasci il deposito. D'altra parte nel problema SVMI, il veicolo è autorizzato a recarsi in qualsiasi posto (rivenditore o deposito). Inoltre, i livelli d'inventario correnti di tutti i rivenditori sono disponibili ad ogni momento decisionale, invece nel SVRP *with restocking* si suppone di conoscere solo il livello d'inventario del cliente che si sta visitando. L'SVRP è un caso particolare del ben studiato problema di vehicle routing (VRP). A differenza dei problemi SVMI, il VRP si concentra in generale sulla progettazione ottimale dei percorsi di consegna di raccolta, dove i veicoli provengono da un deposito unico e visitano più località disperse in una certa area geografica. Il problema delle scorte è di solito ignorato o assai semplificato. In un recente documento, Kleywegt et al. (2002), studiano una variazione del VRP, che viene indicato come inventory routing problem (IRP), è formulato come un processo decisionale di Markov. Sono proposti dei metodi di approssimazione per risolvere l'IRP stocastico. Per il

caso in cui un solo cliente è visitato per ogni percorso dal veicolo, vale a dire l'IRP stocastico con consegne dirette, i risultati computazionali sono presentati. Gli studi precedenti sono focalizzati sull'integrazione tra i problemi d'inventario e quelli di trasporto considerando il problema solo dal punto di vista deterministico, Bertazzi and Speranza (1999) forniscono una buona survey su tali problematiche. Nella rassegna gli autori classificarono questi problemi in modelli a tempo continuo ed a tempo discreto. Ulteriori classificazioni si basano sul numero di origini e destinazioni per i problemi logistici. L'articolo di Federgruen e Zipkin (1984) è stato tra i primi a studiare i modelli stocastici in cui i problemi di trasporto e quelli d'inventario risultano essere integrati. Nel paper viene proposto un algoritmo per risolvere il problema d'inventario e quello per il routing in modo separato per poi combinare entrambe le soluzioni. Il problema di allocazione delle scorte è formulato come constrained non-linear optimization problem. Chien et al. (1989) formularono il problema combinato dell'inventory allocation e del vehicle routing come un modello matematico intero misto. Questo problema consiste nel rifornire con una quantità limitata, proveniente da un unico deposito, le scorte dei diversi clienti usando una flotta di veicoli. Un rilassamento Lagrangiano è stato sviluppato per risolvere tale programma. Nel lavoro di Cetinkaya and Lee (2000), un renewal model viene utilizzato per studiare il rifornimento delle scorte e la pianificazione della spedizione per i sistemi di Vendor Managed Inventory. In tale impostazione, la domanda dei clienti è casuale. L'obiettivo è quello di determinare la quantità di rifornimento e la politica shipment-release. Il vehicle routing non è considerato. Pertanto, il problema della distribuzione studiato nel paper è più semplice del SVMI. Un'applicazione integrata per i problemi di scorte e di trasporto è presentata in Dror and Ball (1987). Nel documento, gli autori studiano il problema della distribuzione del gasolio tra i clienti utilizzando una flotta di veicoli. L'obiettivo del problema è quello di minimizzare la consegna annuale e i costi di stock-out. È stata presa in considerazione sia la domanda deterministica che stocastica. Un algoritmo euristico è stato sviluppato sulla base di una procedura di interscambio e un algoritmo d'assegnamento LP-based. Cachon and Fisher (2000) studiarono il modello d'inventario con un unico fornitore e più rivenditori. Ogni rivenditore è soggetto ad una domanda stocastica stazionaria. I loro risultati numerici suggeriscono che l'informazione ha molto più valore quando è usata per ridurre il lead time e le dimensioni dei lotti rispetto a quando è semplicemente condivisa con il fornitore.

5.2 Processo Decisionale di Markov (PDM) per modellare l'Inventory Routing Problem (IRP)

Un singolo prodotto viene distribuito da un unico soggetto distributore (vendor) ad *N* differenti clienti (customers) mediante una flotta di *M* veicoli omogenei, di capacità limitata e pari a C_v, mentre ogni cliente ha una capacità di stoccaggio pari a C_n. Il problema è modellato ad istanti temporali discreti (*t=0,1,….*), che corrispondono ai diversi giorni dell'orizzonte temporale. La domanda dei diversi clienti, nei differenti giorni dell'arco temporale, è rappresentata da vettori aleatori indipendenti con una distribuzione di probabilità congiunta *F* che non cambia nel tempo. Il vendor può misurare il livello d'inventario X_{nt} di ogni customer *n* ad ogni istante di tempo *t*. Il *Vendor* deve individuare quali *Customer* rifornire, di quanto rifornire ogni *Customer* e come combinare i singoli *Customer* all'interno della rotta con cui si rifornirannno i diversi *Customer*, ed infine quale rotta assegnare al singolo veicolo degli *M* veicoli a disposizione. Inoltre è possibile tener conto, durante la fase di routing, di ulteriori vincoli operazionali (come il carico di lavoro giornaliero per ogni guidatore, le finestre temporali di ogni *Customer* , la capacità di stoccaggio, ecc). E' possibile che un veicolo compia più rotte nello stesso giorno. Il costo di ogni singola rotta può essere calcolato tenendo conto non solo del costo di percorrenza dei singoli archi (c_{ij} per l'arco da *i* a *j*) ma considerando anche un costo che è proporzionale alle unità di prodotto trasportate lungo l'arco. Se la quantità d_n è consegnata al *Customer n* il *Vendor* guadagna $r_n(d_n)$. La stocasticità della domanda presuppone una probabilità non nulla che il singolo *Customer* vada in *stock out*, tale eventualità non può essere eliminata. Lo *Shortage* è penalizzato in funzione obiettivo con il termine $p_n(s_n)$ se la domanda non soddisfatta è s_n per il *Customer n*. La domanda che non si riesce a soddisfare viene considerata persa e non posta in arretrato. Se il livello d'inventario del Customer *n* è x_n all'inizio del giorno, e la quantità d_n è consegnata al cliente *n*, allora il costo d'inventario è uguale a $h_n(x_n+d_n)$. Il costo d'inventario può essere ulteriormente modellato come una funzione "media" del livello d'inventario per ogni cliente nell'orizzonte temporale. L'obiettivo è scegliere una politica di distribuzione che massimizza il valore atteso del valore di sconto (rewards minus costs) su un orizzonte infinito di tempo.

5.3 Formulazione del Problema

Indicando con X_{nt} il livello d'inventario, ovvero lo stato, del cliente *n* al tempo *t*, risulta $X_{nt} \in [0,C_n]$, dove C_n è il massimo livello di inventario ammissibile per il cliente n. Da qui, $X_t=(X_{1t}, X_{2t},\dots, X_{Nt},) \in X$ è il vettore di stato al tempo *t* e X è lo spazio degli stati, ovvero $X= [0,C_1]\times [0,C_2]\times\dots [0,C_N]$. Lo spazio delle azioni *A(x)*, per ogni stato *x*, è l'insieme di tutti gli itinerari che soddisfano tutti i vincoli operazionali. Sia $A_t \in A(X_t)$ indica l'itinerario scelto al tempo *t*. Per ogni itinerario *a* e per ogni arco (*i,j*), sia $k_{ij}(a)$ il numero di volte che l'arco (*i,j*) sia attraversato da un veicolo mentre esegue l'itinerario *a*. Inoltre, per ogni cliente *n*, $d_n(a)$ indica la quantità di prodotto che è consegnata al cliente n mentre esegue l'itinerario *a*. il vincolo sulla capacità di stoccaggio presso il cliente non può essere violato, ciò viene espresso come $X_{nt}+d_n(A_t)\leq C_n$ per tutti i clienti *n* ed ad ogni istante temporale *t*. Si assume che nessuna unità di prodotto venga usata tra il tempo in cui viene misurato il livello d'inventario e l'istante in cui viene consegnata la merce. Se delle unità di prodotto venissero consumate in tale intervallo temporale allora si potrebbe consegnare una quantità maggiore di prodotto. Sia U_{nt} la domanda del *Customer n* al tempo *t*. Allora, la quantità di prodotto usata dal cliente *n* al tempo *t* è data da $min\{X_{nt}+d_n(A_t), U_{nt}\}$. Quindi lo *shortage* del cliente *n* al tempo *t* è dato da $S_{nt}=max\{U_{nt}-(X_{nt}+d_n(A_t)),0\}$ ed il prossimo livello d'inventario del cliente *n* al tempo *t+1* è dato da $X_{n,t+1}=max\{X_{nt}+d_n(A_t)-U_{nt},0\}$. La funzione di distribuzione congiunta *F* della domanda del cliente fornisce la funzione di transizione markoviana denominata *Q*. Per ogni stato $x \in X$, per ogni itinerario $a \in A(x)$ e ogni sotto insieme (misurabile) B⊆X, sia $U(x,a,B) \equiv \{U \in \Re_+^N : max\{x_n+d_n(a)-U_n,0\} \in B\}$. Allora $Q[B|x,a] \equiv F[U(x,a,B)]$. In altre parole per ogni stato $x \in X$ e ogni itinerario $a \in A(x)$,

$$P[X_{t+1} \in B | X_t = x, A_{t=a}] = Q[B|x,a] \equiv F[U(x,a,B)]$$

Sia $g(x,a)$ il guadagno atteso al singolo stadio, se il processo è nello stadio *x* al tempo *t* e l'itinerario $a \in A(x)$ è implementato. Allora nei termini della notazione introdotta in precedenza:

$$g(x,a) \equiv \sum_n r_n\big(d_n(a)\big) - \sum_{(i,j)} c_{ij}k_{ij}(a) - \sum_n h_n\big(x_n+d_n(a)\big) - \sum_n E^{F_n}\big[p_n\big(max\{U_{nt}-(X_{nt}+d_n(A_t)),0\}\big)\big]$$

Dove E^{F_n} rappresenta il valore atteso rispettando la distribuzione di probabilità marginale F_n di U_n. L'obiettivo è massimizzare lo sconto totale su un orizzonte infinito. Sia $\alpha \in [0,1)$ il fattore di sconto. Sia $V^*(x)$ il valore ottimo atteso per lo stato iniziale *x*:

$$V^*(x) \equiv \sup_{\{A_t\}_{t=0}^{\infty}} E\left[\sum_{t=0}^{\infty} \alpha^t g(X_t,A_t) | X_0 = x\right] \qquad (5.1)$$

Le azioni A_t sono tali che $A_t \in A(X_t)$ per ogni *t*, e A_t devono dipendere solo dalla "storia" $(X_0, A_0, X_1, \dots, X_t)$ del processo fino al tempo *t*, nell'ipotesi comune che quando il decisore decide un itinerario al tempo *t* non ha visione di cosa accadrà oltre *t*. Una politica stazionaria deterministica π prescrive un azione $a \in A(x)$ basandosi solamente sulle informazioni contenute nello stato corrente x del processo. Per ogni politica stazionaria deterministica π ed ogni stato $x \in X$, il valore atteso $V^\pi(x)$ è dato:

$$V^\pi(x) \equiv E^\pi\left[\sum_{t=0}^{\infty} \alpha^t g\big(X_t,\pi(X_t)\big) \middle| X_0 = x\right] = g\big(x,\pi(x)\big) + \alpha \int_X V^\pi(y) Q[dy\,|x,\pi(x)]$$

Da un risultato di Bertsekas and Shreve (1978), segue che sotto ipotesi non molto restrittive (e.s. *g* limitata e α<1) per determinare il valore ottimo atteso in (1) è sufficiente restringere l'attenzione alla classe Π delle politiche stazionarie deterministiche. Segue che per ogni stato $x \in X$,

$$V^*(x) \equiv \sup_{\pi\in\Pi} V^\pi(x) = \sup_{a\in A(x)} \left\{g(x,a) + \alpha \int_X V^*(y) Q[dy\,|x,a]\right\} \qquad (5.2)$$

Una politica π^* viene detta ottima se $V^{\pi^*} = V^*$.

5.4 Risoluzione PDM

Il problema presentato in precedenza è estremamente difficile da risolvere. Tutti gli approcci presenti in letteratura semplificano il problema in un modo o nell'altro. Al fine di determinare l'ottimo valore V^* e la politica ottima π^*, si deve risolvere all'ottimo l'equazione (5.2). Ciò richiede l'esecuzione di alcune operazioni, computazionalmente difficili. Quanto alla stima del valore ottimo di V^*, poiché V^* appare sia nel lato sinistro che in quello destro dell'equazione (5.2), di solito gli algoritmi proposti in letteratura calcolano tale valore sfruttando delle stime (approssimazioni) progressive di $\mathrm{V}^*(\mathrm{x})$ per ogni $x \in X$. Tale approccio, però, è praticabile solo se lo spazio di stato *X*, misurato nella numerosità dei clienti considerati, risulta relativamente piccolo. Nel caso dell'IRP, la dimensione dello spazio degli stati è esponenziale rispetto al numero di *clienti*. Quindi, anche se i livelli d'inventario fossero resi discreti, lo spazio degli stati *X* risulterebbe comunque troppo grande per calcolare $\mathrm{V}^*(\mathrm{x})$ per ogni $x \in X$, (da 4 clienti in su, nella pratica). La stima del valore atteso dell'integrale nella (5.2), come si può intuire, è non banale. Per molte applicazioni, questo è un integrale con molte dimensioni, ciò richiede un grosso sforzo computazione per essere calcolato accuratamente. Nel caso dell'IRP, il numero delle dimensioni è pari al numero di *Customer*, il quale può essere di alcune centinaia e, purtroppo, i metodi numerici d'integrazione convenzionali non sono adoperabili per la risoluzioni d'integrali con tali dimensioni. Il problema di massimizzazione nell'equazione (5.2) deve essere risolto per determinare l'azione ottima, da applicare per ogni stato. Nel caso dell'IRP, questo problema è molto difficile da risolvere, poiché il solo problema di *vehicle routing*, al quale si può ricondurre il problema, è *NP-hard*. Ci sono alcuni algoritmi convenzionali per risolvere PDM, fra cui quelli in Bertsekas (1995) e in Puterman (1994). Tali algoritmi sono utilizzabili solo se il problema da risolvere è semplice. Come già evidenziato questa condizione non è valida per l'IRP, poiché lo spazio degli stati *X* è estremamente grande, il valore atteso è difficile da calcolare ed infine il problema di ottimizzazione nella (5.2) è difficilmente risolvibile. L'approccio sviluppato da Kleywegt et al. 2002 sviluppa un'efficiente programmazione dinamica basata su metodi di approssimazione. La motivazione per cui usare dei metodi di approssimazione è la complessità computazionale dell'IRP. La motivazione per cui usare una specifica programmazione dinamica basata su metodi di approssimazione è la seguente: si supponga che V^* possa essere approssimato da $\hat{V}$ tale che $\left\|\mathrm{V}^* - \hat{V}\right\|_\infty \leq \varepsilon$ tale che $\left|\mathrm{V}^*(x) - \hat{V}(x)\right|_\infty \leq \varepsilon$ per tutti gli stati $x \in X$. Si scelga una politica $\hat{\pi}$ tale che:

$$g(x, \hat{\pi}(x)) + \alpha \int_X \hat{V}(y) Q[dy\,|x, \hat{\pi}(x)] \geq \sup_{a \in \mathrm{A}(\mathrm{x})} \left\{ g(x,a) + \alpha \int_X \hat{V}(y) Q[dy\,|x,a] \right\} - \delta$$

per tutti gli stati $x \in X$, ciò coincide con il valore $\mathrm{V}^{\hat{\pi}}$ della politica $\hat{\pi}$ più vicina al valore ottimo V^*.

5.5 IRP con Direct Delivery

Da qui in avanti ci si concentrerà sul caso di IRP con *direct delivery* (Kleywegt et al. 2002), in cui un singolo veicolo visita uno ed un solo cliente (variante dell'IRP denominata IRPDD). La formulazione dell'IRPDD è la stessa della formulazione dell'IRP eccetto che per le seguente caratteristiche:

- Lo spazio delle azioni *A(x)* per ogni stato *x* è l'insieme di tutti gli itinerari che coincidono con le rotte che visitano solo un cliente e soddisfano il *work load*, le *time windows* ed i vincoli di capacità.
- Ogni itinerario *a* è costituito da un insieme di itinerari individuali a_n, *n=1,….,N*. L'itinerario a_n indica il numero di visite al cliente *n* effettuato da ogni veicolo e la quantità di prodotto da dover consegnare al cliente *n* da parte di ogni veicolo. Con t_n si indica il tempo necessario ad un veicolo per compiere il viaggio di andata e ritorno dal *Vendor* al *Customer n*.

I costi di trasporto possono essere associati ad ogni cliente invece che agli archi della rete. Sia c_n il costo di trasporto per effettuare una consegna al cliente *n* e sia $k_n(a_n)$ il numero di volte che il cliente *n* sia visitato da un veicolo quando si esegue l'azione a_n. Allora

$$g(x,a) \equiv \sum_{n=1}^{N}\{r_n(d_n(a_n)) - c_n k_n(a_n) - h_n(x_n + d_n(a_n)) - E^{F_n}[p_n(max\{U_n - (x_n + d_n(a_n)), 0\})]\} \quad (5.3)$$

Anche se adottare la politica del *Direct Delivery* semplifica il *routing*, il problema dell'IRPDD rimane sempre molto difficile da risolvere quando il numero di clienti è maggiore di quattro ed il numero di veicoli è limitato, poiché lo spazio degli stati cresce esponenzialmente all'aumentare del numero di clienti. Per esempio, se *Z* indica il numero dei livelli d'inventario per cliente, la dimensione dello spazio degli stati $|X| = Z^N$.

5.6 Heuristic SVMI

La risoluzione del problema SVMI modellato con un MDP diventa computazionalmente più difficile da risolvere con l'aumentare della dimensione del problema. In questo caso (problema *"single product"*), la crescita delle dimensioni del problema deriva principalmente dall'aumento del numero di rivenditori e dalla capacità di stoccaggio dei rivenditori e dal tipo di veicolo utilizzato. Anche all'aumentare dell'orizzonte di pianificazione il processo MDP risulta più difficile da risolvere dal punto di vista computazionale. È praticamente improbabile applicare questo approccio ad un problema "*multi product*" visto che già nel caso precedente la risoluzione dello stesso è non banale. In questa sede, l'obiettivo è individuare dei buoni metodi euristici che diano una buona soluzione dal punto di vista della qualità della stessa, una volta implementate in algoritmi che siano in grado di risolvere problemi realistici. In tal senso, l'attenzione sarà dunque rivolta soprattutto al problema della gestione delle scorte poiché la stocasticità del problema deriva dalla natura aleatoria della domanda, mentre la fase di routing non è altro che una diretta conseguenza della pianificazione delle quantità da consegnare. Per ogni $i = 1,2,\dots,N$, h_i è l'inventory holding cost per unità di prodotto e per unità di tempo, b_i^1 è il guadagno per unità di prodotto vendute, b_i^2 è il costo di penalità per ogni ordine non soddisfatto, b_i^3 è il costo di approvvigionamento per unità di prodotto. Si assuma che tutti questi parametri siano maggiori di zero. Ancora, $D_{t,i}^{l,k}$ è una variabile casuale con distribuzione nota, che rappresenta il numero di unità richieste dal retailer i tra il tempo t e $t + d_{ik}$, mentre d_{lk} è il tempo necessario al veicolo per andare dal cliente i al cliente k. I risultati matematici per il controllo del livello d'inventario, che saranno enunciati, forniscono sufficienti condizioni per individuare un livello soglia affidabile di rifornimento del generico cliente. Nel seguito, $\bar{F}_{t,i}^{l,k} = 1 - F_{t,i}^{l,k}$, dove $F_{t,i}^{l,k}$ è la funzione cumulativa della distribuzione $D_{t,i}^{l,k}$ e si noti che si può scrivere il vettore x come $x = (x_i, x_i^c)$, per $i = 1,2,\dots,N$.

Teorema 5.1 (Inventory Control) per $t = 1,2,\dots,T$ e $l \in \{1,2,\dots,N\}$ e $i \in K\backslash\{o,l\}$, e per tutti $k \in K$, si assuma che:

$$(b_i^1 + b_i^2)\bar{F}_{t,i}^{l,k} \geq h_i d_{lk}$$

Allora esiste $a^*(k)$ per lo stato $s_t((x_i, x_i^c), x_v, l)$, per il quale è non decrescente in x_i.

Considerando il precedente risultato si evince che l'azione ottima per l'*inventory* a^* è non decrescente per i livelli d'inventario del *Cutomer* i, indicato con x_i. Sulla base di questo risultato, si può sviluppare un'euristica in cui le azioni d'inventario ottimali siano determinate per gli stati che corrispondono ad un certo valore di x_i. Le azioni sull'inventario sono calcolate assumendo una relazione lineare a tratti tra a^* e x_i. Per il problema SVMI, il Teorema 5.1 fornisce una relazione monotona tra l'azione delle scorte ottimali e livelli di inventario dei rivenditori che è possibile utilizzare nella procedura per la determinazione della soluzione euristica. Per questa euristica, è ovvio che maggiori saranno i numeri delle sezioni lineari secondo le quali è approssimata la funzione $F_{t,i}^{l,k}$ e migliore sarà la soluzione ottenuta. D'altra parte, l'aumento del numero di sezioni lineari nella soluzione euristica richiede più tempo di calcolo. Nel problema SVMI, non è chiaro quando il veicolo dovrà ritornare dal cliente corrente, mentre la successiva destinazione del veicolo è determinata ad ogni istante decisionale. Nell'istante corrente *t*, si suppone di conoscere il tempo in cui avverrà la visita successiva al cliente corrente. Ulteriormente, si suppone che l'azione per il livello d'inventario sia di tipo *base-stock* e che esiste un livello d'inventario di riferimento S_t^l tale che l'azione per il livello d'inventario ottimo sia:

$$a^* = \min\{x_v, max\{0, S_t^l - x_l\}\}.$$

In queste condizioni, il problema del controllo del livello d'inventario è una variante del classico problema del *Newsvendor*. Di conseguenza, il livello d'inventario di riferimento S_t^l può essere determinato basandosi sulla distribuzione della domanda del cliente corrente. Il livello d'inventario di riferimento può essere identificato nel modo seguente. Per $l \in \{1,2,\dots,N\}$, si assuma che il tempo fino alla prossima visita a tale cliente sia noto e indicato con δ. Allora, il guadagno atteso per il cliente l tra il tempo t ed il tempo $t+\delta$ è:

$$g_t(\tilde{x}_l, x_l) = -h_l \delta \tilde{x}_l + b_l^1 E[min\{Q_t^l, \tilde{x}_l\}] - b_l^2 E[max\{0, Q_t^l - \tilde{x}_l\}] - b_l^3(\tilde{x}_l - x_l)$$

dove $\tilde{x}_l = x_l + a$ e Q_t^l è la domanda del cliente l tra il tempo t ed il tempo $t+\delta$. Inoltre, in seguito, si vedrà che $g_t(\tilde{x}_l, x_l)$ è concava in $\tilde{x}_l$. Soltanto per lo scopo di determinare il livello d'inventario di riferimento, è ragionevole stimare il tempo della visita successiva al cliente corrente. Sia $\bar{H}_t^l = 1 - H_t^l$, dove H_t^l è la distribuzione cumulativa della domanda del cliente corrente tra t e il tempo della visita successiva al cliente corrente. Il seguente teorema fornisce la formula per calcolare il livello d'inventario di riferimento, S_t^l.

Teorema 5.2 si assuma che H_t^l e $g_t(\tilde{x}_l, x_l)$ siano note, allora la base-stock ottima per il livello d'inventario per tale cliente, S_t^l tale che:

$$\bar{H}_t^l(S_t^l) = \frac{h_l \delta + b_l^3}{b_l^1 + b_l^2}$$

Si noti che Q_t^l è una variabile aleatoria discreta ed esiste al più una S_t^l che soddisfa l'equazione del teorema. Quindi, scegliendo il valore di S_t^l tale che $\bar{H}_t^l(S_t^l)$ è vicino a $(h_l\delta + b_l^3)/(b_l^1 + b_l^2)$, si può mostrare che il risultato del teorema precedente è equivalente al problema del *Newsvendor* con le seguenti assunzioni: $c_u = b_l^1 + b_l^2 - h_l\delta - b_l^3$ e $c_o = h_l\delta + b_l^3$.

L'algoritmo seguente schematizza come applicare il Teorema 5.2 per determinare l'azione di tipo *base-stock* per il problema SVMI.

Algorithm 5.1.*: Base-Stock Inventory Algorithm

1. At current time t, estimate the time until the next visit to current retailer l and call it δ.
2. Compute the cumulative distribution of Q_t^l, namely H_t^l.
3. Determine S_t^l such that
$$\bar{H}_t^l(S_t^l) \approx (h_l\delta + b_l^3)/(b_l^1 + b_l^2)$$
4. The base-stock inventory action is $a(S_t^l) = min\{x_v, \max\{0, S_t^l - x_l\}\}$.

5.7 Test sul modello Newsvendor confrontato con politica (s,S)

La distribuzione di probabilità che si è adottata per descrivere la domanda giornaliera consumata da un singolo prodotto è la distribuzione Poissoniana. Tale scelta è legata alla duttilità di tale distribuzione poiché, allo stesso tempo, è in grado di descrivere sia fenomeni rari, come per esempio quelli descritti con una binomiale, sia fenomeni molto più frequenti (usando, in tal caso, la sua capacità di approssimare anche una normale). Si è realizzato un modello di magazzino in cui vi è una domanda aleatoria di cui sono noti forma e parametri tipici. Si è supposto che tale domanda sia una normale di cui conosciamo la media (300) e la varianza (1). Inoltre si è supposto che il β del modello del *Newsvendor* sia pari al 90%, inoltre l'intervallo tra due rifornimenti (δ) vale 3 epoche. È stato effettuato il confronto con un analogo sistema che viene controllato con la politica (s,S), in cui la soglia (s) è uguale al valore medio consumato in ogni epoca (100) mente il valore S coincide con il valore ottimale di rifornimento ottenuto con il modello del *Newsvendor* (301). Tali esperimenti sono stati effettuati con un orizzonte temporale di 30 epoche (un orizzonte temporale pari ad un mese) ed ogni esperimento è stato ripetuto 30 volte per poter stimare l'intervallo di confidenza dell'indice di performance. L'indice di performance che si è usato per discriminare le prestazioni di ogni sistema è il valore cumulato di domanda giornaliera persa per via del valore del livello d'inventario non idoneo a soddisfare la domanda giornaliera. Prima si è supposto che in entrambi i modelli la distribuzione della domanda

coincidesse con quella supposta, per calcolare il valore ottimale del *news vendor problem*, successivamente entrambi i sistemi sono stati sottoposti ad una domanda giornaliera il cui valore medio è sempre identico al caso precedente ma la distribuzione della domanda non è più una normale ma bensì una Poissoniana. Nel primo caso come è facile pensare il valore medio cumulato (sull'intero orizzonte di pianificazione) di domanda persa per entrambi i modelli è praticamente nullo, il che implica che entrambi i modelli funzionano correttamente, quando sono sottoposti alla stessa domanda che abbiamo supposto per calcolare il valore ottimale per rifornire il magazzino. Quando si è supposto, per entrambi i modelli, che la domanda giornaliera fosse una distribuzione di Poisson si è verificato che il modello controllato attraverso la quantità ottimale calcolata attraverso il modello del Newsvendor realizza un indice di performance con un ordine di grandezza maggiore rispetto al sistema controllato con una politica (s, S). Ciò implica che la politica (s, S) è più robusta rispetto alla politica news vendor problem.

			Lost Demand	
	Type	Demand Distribution	mean	wide
1	NewsVendor	Normal	0,03	0,05
2	**(s,S)**	Normal	0,00	0,00
3	NewsVendor	Poisson	27,43	7,33
4	**(s,S)**	Poisson	2,87	2,00

Tabella 5.1: Robustezza dei due modelli con caso ideale e con domanda Poissoniana

Inoltre si è effettuato un ulteriore test in cui la varianza della domanda giornaliera si suppone essere una normale con media pari a 100, identica al caso precedente, ma in questo caso la varianza non sarà più pari ad 1 ma pari a 10 (un ordine di grandezza superiore). Anche in tal caso il sistema (s, S) realizza un valore medio di domanda persa inferiore rispetto all'altro modello.

			Lost Demand	
	Type	Demand Distribution	mean	wide
1	NewsVendor	Normal	21,83	7,05
2	**(s,S)**	Normal	2,50	1,55

Tabella 5.2: Robustezza dei due modelli in cui la varianza è pari a 10

È stato effettuato un ultimo test test in cui abbiamo fatto variare dell'1% la domanda giornaliera rispetto a quella ipotizzata in precedenza ed abbiamo sottoposto i due modelli a due tipi domanda giornaliera la prima una normale mentre la seconda è una Poissoniana come si può vedere nella tabella seguente la politica (s, S) funziona molto meglio rispetto all'altra politica.

			Lost Demand	
	Type	Demand Distribution	mean	wide
1	NewsVendor	Normal	5,67	1,27
2	**(s,S)**	Normal	0,13	0,10
3	NewsVendor	Poisson	59,20	11,96
4	**(s,S)**	Poisson	2,77	1,39

Tabella 5.3: Robustezza dei due modelli in cui la media è aumentata dell'1%

Questi test effettuati dimostrano che il modello del *news vendor* non è un modello robusto ed in condizioni d'elevata incertezza le performance che si ottengono sono poco soddisfacenti. Invece un politica di controllo (s, S) che in un certo senso controlla in modo dinamico il valore del livello d'inventario è più affidabile e presenta performance migliori rispetto ad un modello che calcola tutto off-line.

5.8 Roll-out

Per illustrare le potenzialità e le performance dell'approccio risolutivo illustrato nel capitolo precedente, in questo paragrafo è stato immerso quell'approccio in un framework in cui la domanda di consumo dei singoli prodotti non è più costante ma stocastica. La domanda dei singoli prodotti è stata modellata con un processo di Poisson il cui valore medio coincide con μ, il tasso di consumo giornaliero, che veniva ipotizzato costante nel paragrafo precedente. L'orizzonte di pianificazione che viene utilizzato in questo particolare esperimento è H=7, mentre 2 il numero di giorni contenuti nella soluzione che vengono utilizzati prima di riaggiornare i livelli d'inventario correnti dei singoli prodotti e rieseguire l'algoritmo, sono stati utilizzati tali valori poiché da alcune sperimentazioni preliminari abbiamo visto che questa scelta coincideva con la migliore aggregazione possibile. L'orizzonte temporale preso in considerazione negli esperimenti è di 30 giorni. Per ogni istanza abbiamo effettuato 30 run poiché essendo in regime di domanda stocastica i dati ottenuti hanno valore solo se si fornisce un intervallo di confidenza. La nuova misura che è stata introdotta riguarda la quantità (cumulata su tutto il periodo di pianificazione e per tutti i prodotti) di domanda persa per via del livello d'inventario dei singoli prodotti insufficiente a soddisfare la quantità di prodotto richiesta, tale indice è stato denominato "Demand Lost".

Le istanze utilizzate in tale sperimentazione coincidono con alcune di quelle utilizzate nel capitolo precedente. Sono stati selezionati 3 gruppi di istanze: quelle con 50 clienti, quelle con 100 di tipo RC1 (con clienti dislocati in modo casuale ma raggruppati) di Solomon e, infine, quelle con 200 clienti (di Homberger) sempre di tipo RC1. Il primo set di istanze prese in considerazione è quelle con 50 clienti. Si può osservare che la domanda persa è molto inferiore (di qualche ordine di grandezza) rispetto al volume di merce trasportato; infatti, tale percentuale è inferiore all'1%, precisamente vale intorno allo 0,15%. Ciò significa che l'algoritmo proposto precedentemente, immerso in un ambiente dinamico, ha delle buone performance poiché, come si può vedere, anche l'indice di prestazione V/D è stimato con un intervallo di confidenza molto stresso (per ogni istanza) e questo implica che vengono garantite delle buone prestazioni per ogni replica.

		V			Demand Lost			Demand Lost su V	D			V/D			
	Istance	min	mean	max	min	mean	max	%	min	mean	max	min	mean	max	%Vehicle
1	P_100N_50_1,dat_abs1n50,dat	3059755,17	3062724,50	3065693,83	4648,45	4713,13	4777,81	0,15%	2138638,64	2140921,47	2143204,29	1,43	1,43	1,43	95,85
2	P_100N_50_2,dat_abs2n50,dat	3072382,56	3074801,97	3077221,37	4485,60	4533,20	4580,80	0,15%	2169241,61	2171426,83	2173612,06	1,41	1,42	1,42	95,79
3	P_100N_50_3,dat_abs3n50,dat	3052824,23	3056140,07	3059455,90	4506,64	4569,73	4632,82	0,15%	1709528,62	1711533,33	1713538,05	1,78	1,79	1,79	95,57
4	P_100N_50_4,dat_abs4n50,dat	3070391,02	3073719,87	3077048,72	4513,30	4591,43	4669,56	0,15%	2005024,88	2007187,67	2009350,46	1,53	1,53	1,53	95,62
5	P_100N_50_5,dat_abs5n50,dat	3062078,51	3065349,83	3068621,16	4325,09	4386,63	4448,18	0,14%	1877151,67	1879451,53	1881751,40	1,63	1,63	1,63	95,37

Tabella 5.4: Istanze con 50 clienti

Il secondo set di istanze prese in considerazione è quello con 100 clienti di tipo RC1. Anche in questo caso la domanda persa è molto inferiore al volume di merce trasportato. Infatti, tale percentuale è inferiore all'1%, precisamente vale intorno allo 0,15% e ciò implica che l'algoritmo proposto precedentemente mantiene ottime prestazioni anche all'aumentare del numero di clienti serviti. Ancora, si può vedere che il rapporto V/D è stimato con un intervallo di confidenza molto stretto: per ogni istanza, al più differisce per la seconda cifra decimale e ciò implica che vengono garantite prestazioni quasi identiche per ogni replica.

		V			Demand Lost			%Demand Lost su V	D			V/D			
	Istance	min	mean	max	min	mean	max	%	min	mean	max	min	mean	max	%Vehicle
1	P_100N_100_1,dat_rc101,txt	6126458,21	6131464,93	6136471,65	9179,63	9257,87	9336,10	0,15%	411816,86	412133,21	412449,55	14,87	14,88	14,88	95,59
2	P_100N_100_1,dat_rc101,txt	6118694,26	6122087,77	6125481,27	9191,22	9302,90	9414,58	0,15%	408919,19	409223,56	409527,92	14,95	14,96	14,97	95,60
3	P_100N_100_3,dat_rc103,txt	6115636,79	6120755,17	6125873,55	9125,97	9237,07	9348,16	0,15%	402952,34	403271,93	403591,53	15,17	15,18	15,18	95,67
4	P_100N_100_4,dat_rc104,txt	6115550,43	6119538,97	6123527,51	9084,08	9156,43	9228,79	0,15%	410820,13	411089,86	411359,59	14,88	14,89	14,89	95,65
5	P_100N_100_5,dat_rc105,txt	6119968,51	6123391,37	6126814,23	9336,36	9427,07	9517,77	0,15%	412058,74	412348,33	412637,91	14,84	14,85	14,86	95,47
6	P_100N_100_6,dat_rc106,txt	6124389,93	6130153,30	6135916,67	8909,26	9007,23	9105,21	0,15%	412031,52	412440,36	412849,19	14,86	14,86	14,87	95,69
7	P_100N_100_7,dat_rc107,txt	6124076,96	6127917,80	6131758,64	9217,05	9301,87	9386,68	0,15%	404822,78	405137,46	405452,13	15,12	15,12	15,13	95,80
8	P_100N_100_8,dat_rc108,txt	6127259,63	6131351,03	6135442,43	9235,32	9327,90	9420,48	0,15%	403969,01	404270,74	404572,47	15,16	15,17	15,17	95,80

Tabella 5.5: Istanze con 100 clienti

Il terzo ed ultimo set di istanze prese in considerazione è quelle con 200 clienti sempre di tipo RC1. Come si può notare, anche in questo caso la domanda persa è molto inferiore rispetto al volume di merce trasportato. Infatti, anche in questo caso, tale percentuale è inferiore all'1%: precisamente vale intorno allo 0,15% e ciò implica che

l'algoritmo proposto nel precedente capitolo immerso in un ambiente con domanda stocastica mantiene ottime prestazioni anche all'aumentare del numero di clienti serviti. Tant'è che il rapporto V/D è espresso da un intervallo di confidenza molto stretto (per ogni istanza, al più differisce per la seconda cifra decimale) e ciò implica che vengono garantite prestazioni quasi identiche per ogni replica. Infine, si osservi che in questo ultimo set si registra rispetto ai precedenti un aumento del volume di merce trasportata.

		V			Demand Lost			%Demand Lost su V	D			V/D			
	Istance	min	mean	max	min	mean	max	%	min	mean	max	min	mean	max	%Vehicle
1	P_100N_200_1,dat_RC1_2_1,txt	12267809,99	12273120,10	12278430,21	18178,29	18298,93	18419,58	0,15%	1287854,23	1288490,72	1289127,22	9,52	9,52	9,53	96,25
2	P_100N_200_2,dat_RC1_2_2,txt	12236801,37	12241952,67	12247103,97	18565,22	18681,93	18798,65	0,15%	1311311,85	1312023,71	1312735,56	9,33	9,33	9,33	96,21
3	P_100N_200_3,dat_RC1_2_3,txt	12251310,05	12256549,43	12261788,82	18501,34	18629,57	18757,79	0,15%	1306806,03	1307448,32	1308090,60	9,37	9,37	9,38	96,24
4	P_100N_200_4,dat_RC1_2_4,txt	12243305,63	12250020,07	12256734,50	18275,18	18402,97	18530,75	0,15%	1321262,33	1322180,61	1323098,89	9,26	9,26	9,27	96,22
5	P_100N_200_5,dat_RC1_2_5,txt	12264585,07	12270420,97	12276256,86	18513,44	18636,47	18759,49	0,15%	1312278,74	1312921,86	1313564,99	9,34	9,35	9,35	96,26
6	P_100N_200_6,dat_RC1_2_6,txt	12244366,76	12250629,50	12256892,24	18351,42	18502,47	18653,52	0,15%	1300663,32	1301357,18	1302051,05	9,41	9,41	9,42	96,26
7	P_100N_200_7,dat_RC1_2_7,txt	12248288,80	12255504,83	12262720,87	17648,56	17762,27	17875,97	0,14%	1303228,87	1303979,16	1304729,45	9,39	9,40	9,40	96,30
8	P_100N_200_8,dat_RC1_2_8,txt	12240109,69	12246363,97	12252618,24	18455,15	18585,97	18716,78	0,15%	1325732,78	1326528,56	1327324,34	9,23	9,23	9,23	96,27
9	P_100N_200_9,dat_RC1_2_9,txt	12240178,63	12247731,13	12255283,64	18143,01	18282,10	18421,19	0,15%	1304797,92	1305633,73	1306469,55	9,38	9,38	9,38	96,30
10	P_100N_200_10,dat_RC1_2_10,txt	12216139,36	12222250,20	12228361,04	18613,13	18733,07	18853,00	0,15%	1313676,56	1314404,68	1315132,81	9,30	9,30	9,30	96,29

Tabella 5.5: Istanze con 200 clienti

In conclusione, l'algoritmo proposto nel paragrafo precedente, integrato in uno schema di *roll-out*, produce buone prestazioni. Anche se il tempo computazionale, inevitabilmente, aumenta con l'aumentare della dimensione del problema le prestazioni ottenute rimangono sempre ottimali.

Conclusioni

L'obiettivo guida del lavoro di ricerca è stato quello di consentire un'efficiente integrazione di tecniche metaeuristiche di esplorazione della regione ammissibile in ambienti di simulazione Monte Carlo volti a valutare la qualità delle soluzioni trovate. Sono stati sviluppati ed implementati diversi algoritmi euristici di ottimizzazione per problemi reali di logistica del trasporto merci, concentrandosi sulla necessità ed opportunità di approfondire e specializzare le caratteristiche dei diversi meccanismi di intensificazione e diversificazione nell'ambito delle metodologie proposte (i.e. local branching, beam search, nested partitions, iterated local search). Sul piano applicativo, i temi specifici hanno riguardato i processi di accumulo e rinvio delle merci (containerizzate e sfuse) che si realizzano nelle aree portuali e retroportuali dei terminali marittimi e gli algoritmi proposti per alcuni problemi reali sono stati sviluppati con l'obbiettivo di individuare buone soluzioni in tempi accettabili e sono stati convalidati nell'ambito di progetti di ricerca industriale.

In particolare è stato proposto un algoritmo migliorativo della tecnica basilare del local branching. Tale algoritmo, denominato Two Stage Local Branching, è stato sperimentato positivamente per i problemi di retangular packing che possono essere posti alla base di alcuni importanti modelli della logistica portuale e retroportuale. Per quanto riguarda la logistica portuale, è stato proposto e sviluppato un ambiente integrato di simulazione ottimizzazione che mira a cogliere gli aspetti aleatori del Berth Allocation Problem che si manifestano al momento della programmazione operativa delle operazioni di banchina e che si riflettono sui tempi di permanenza delle navi in banchina. L'ambiente innovativo è stato costruito attorno ad un'euristica costruttiva di Beam Search e ad una seconda euristica migliorativa di Simulated Annealing. Per quanto riguarda la logistica retro portuale, è stato sviluppato un algoritmo per la gestione combinata delle scorte di più prodotti da distribuire sul territorio e del conseguente instradamento dei veicoli adibiti al trasporto. Grazie ad un upper bound calcolato sulla quantità di merce trasportata per unità di distanza, è stato possibile stabilire sperimentalmente la buona qualità delle soluzioni proposte dall'algoritmo di gestione combinata. Infine, lo stesso algoritmo è stato immerso in uno schema di roll-out, per tener conto degli aspetti aleatori presenti nel problema, legati all'aleatorietà della domanda di ogni prodotto. Il risultato che si è avuto è decisamente molto incoraggiante poiché la domanda di merce persa (non soddisfatta) è di gran lunga inferiore alla quantità di domanda soddisfatta.

Bibliografia

Aarts, E. H. L. Korst, J. H. M. and Van Laarhoven P. J. M. (1997), Simulated annealing. In Emile H. L. Aarts and Jan Karel Lenstra, editors, Local Search in Combinatorial Optimization, 91-120. Wiley-Interscience, Chichester, England.

Adelman, D. (2004), A Price-directed Approach to Stochastic Inventory/Routing, Operations Research 52, 499-514.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G. and Lokketangen, A. (2010), `Industrial Aspects and Literature Survey:Combined Inventory Management and Routing'', Computers and Operations Research 37,9, 1515-1536.

Andradóttir, S. (2007) Simulation Optimization in Handbook of simulation: principles, methodology, advances, applications, and practice, Banks, J. (ed). John Wiley & Sons, Inc., Hoboken, NJ, USA.

Anily, S., Federgruen, A. (1990), One Warehouse Multiple Retailer Systems with Vehicle Routing Costs, Management Science 36, 92-114.

Anily, S., Federgruen, A. (1991), Rejoinder to 'One Warehouse Multiple Retailer Systems with Vehicle Routing Costs', Management Science 37, 1497-1499.

Archetti, C., Bertazzi, L., Laporte, G. and Speranza, M. G. (2007), A branch-and-cut algorithm for a vendor-managed inventory-routing problem. Transportation Science, 41(3):382-391.

Archetti, C., Bertazzi, Hertz, A. and Speranza, M. G. (2009), A hybrid heuristic for an inventory-routing problem. Technical report, Dipartimento di Metodi Quantitativi, Università di Brescia, Brescia (Italy).

Assad, A., Dahl, R. and Golden, B. (1984), Analysis of a large scale vehicle routing problem with an inventory component. Large Scale System, 7(2-3), 181-190.

Balas, E. Ceria, S. Dawande, M. Margot, F. and Pataki, G. (2001), OCTANE: A New Heuristic For Pure 0-1 Programs. Operations Research 49, 2, 207–225.

Balakrishnan, A. Magnanti, T.L. and Mirchandani, P. (1997), Network Design, in: M. DellAmico, F. Maffioli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, New York, 311–334.

Bard, J., L. Huang, M. Dror, and P. Jaillet. A branch and cut algorithm for the vrp with satellite facilities. IIE Trans.Oper.Engrg., 30:821-834, 1998a.

Bard, J., Huang, L., Jaillet, P., Dror, M. (1998b), A Decomposition Approach to the Inventory Routing Problem with Satellite Facilities, Transportation Science 32, 189-203

Bechhofer, R.E. (1954) A single-sample multiple decision procedure for ranking means of normal populations with known variances. Annals of Mathematical Statistics 25(1): 16-39.

Bell, W., Dalberto, L., Fisher, M., Greenfield, A., Jaikumar, R., Kedia, P., Mack, R., Prutzman, P. (1983), Improving the Distribution of Industrial Gases with an On-line Computerized Routing and Scheduling Optimizer, Interfaces 13, 4-23.

Beltrami, L. and Bodin, F. (1974), Networks and vehicle routing for municipal waste collection. Net- 561 works, 4(1):65-94.

Benoist, T. Gardi, F. Jeanjean, A. (2011), Randomized Local Search for Real-Life Inventory Routing, Transportation Science 45, 381–398.

Berkey, J.O. Wang, P.Y. (1987), Two-dimensional finite bin packing algorithms, Journal of the Operational Research Society, 38, 423–429.

Bertazzi, L., Paletta, G., Speranza, M.G. (2002), Deterministic Order-up-to Level Policies in an Inventory Routing Problem, Transportation Science 36, 119-132.

Bertazzi, L., Paletta, G., Speranza, M.G. (2005), Minimizing the Total Cost in an Integrated Vendor–Managed Inventory System, Journal of Heuristics 11, 393-419.

Bertazzi, L. and Speranza, M. G. (1999), “Models and Algorithms for the Minimization of Inventory and Transportation Costs: A Survey,” in New Trends in Distribution Logistics (STAHLY, P., ed.), pp. 137–157, New York, NY: Springer.

Bertazzi, L., Speranza, M.G. (2002), Continuous and Discrete Shipping Strategies for the Single Link Problem, Transportation Science 36, 314-325.

Bertazzi, L., Speranza, M.G. (2005), Worst-case Analysis of the Full Load Policy in the Single Link Shipping Problem, International Journal of Production Economics 93-94C, 217-224.

Bertazzi, L., Speranza, M.G., Ukovich, W. (1997), Minimization of Logistic Costs with Given Frequencies, Transportation Research B 31, 327-340.

Bertazzi, L., Speranza, M.G., Ukovich, W. (2000), Exact and Heuristic Solutions for a Shipment Problem with Given Frequencies, Management Science 46, 973-988.

Bertsekas, D. P. (1995), Dynamic Programming and Optimal Control. Athena Scientific, Belmont, MA.

Bertsekas, D. P. and Shreve, S. E. (1978), Stochastic Optimal Control: The Discrete Time Case. Academic Press, New York, NY.

Bielli, M., Boulmakoul, A. and Rida, M. (2006) Object oriented model for container terminal distributed simulation. European Journal of Operational Research 175(3), 1731-1751,

Bierwirth, C. and Meisel, F. (2010) A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research 202(3), 615-627.

Bramel, J. and Simchi-Levi, D. (1995), A location based heuristic for general routing problems. Operations Research, 43(4):649-660.

Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J., and Lusby, R. (2009). Models for the discrete berth allocation problem: a computational comparison. DTU Technical Report 14-2009 Delft, Denmark.

Cachon, G. P. and Fisher, M. (2000), Supply Chain Inventory Management and the Value of Shared Information, Management Science, 46, 1032–1048.

Campbell, A.M., Clarke, L., Kleywegt, A., Savelsbergh, M.W.P. (1998), The Inventory Routing Problem, in: Fleet Management and Logistics, T.G. Crainic and G. Laporte (eds), 95-113, Kluwer, Boston.

Campbell, A.M., Hardin, J. (2005), Vehicle Minimization for Periodic Deliveries, European Journal of. Operational Research 165, 23, 668-684.

Campbell, A.M., Savelsbergh, M.W.P. (2004a), A Decomposition Approach for the Inventory–Routing Problem, Transportation Science 38, 488-502.

Campbell, A.M., Savelsbergh, M.W.P. (2004b), Delivery Volume Optimization, Transportation Science 38, 210–223.

Campbell, A.M., Savelsbergh, M.W.P. (2004c), Efficiently Handling Practical Complexities in Insertion Heuristics, Transportation Science 38, 369-378.

Canonaco, P., Legato, P. and Mazza, R.M. (2006) CaLeMa - A simulation tool for the vessel process (release 1.0). DEIS - Università della Calabria, Rende (CS) Italy.

Canonaco, P., Legato, P. and Mazza, R.M. (2007) An integrated simulation model for channel contention and berth management at a maritime container terminal in Proceedings 21st European Conference on Modelling and Simulation, Zelinka, I. et al (eds), 353 – 362.

Cetinkaya, S. and Lee, C. Y. (2000), Stock Replenishment and Shipment Scheduling for Vendor-Managed Inventory Systems, Management Science, vol. 46, pp. 217–232.

Chan, L. M., Federgruen, A., and Simchi-Levi, D. (1998), Probabilistic analyses and practical algorithms for inventory-routing models. Operations Research, 46, 1, 96-106.

Chardaire, P. Lutton, J. L. and Sutter, A. (1995), Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms. European Journal of Operational Research, 579, 86-565.

Chen, C. H. (1996). A Lower Bound for the Correct Subset-Selection Probability and Its Application to Discrete Event System Simulations. IEEE Transactions on Automatic Control, 41:1227-1231.

Chen, C. H. Wu, S. D. and Dai, L. (1999), Ordinal Comparison of Heuristic Algorithms Using Stochastic Optimization. IEEE Transactions on Robotics and Automation, 15: 44-56.

Chen, H. C. Chen, C. H. and Yücesan, E. (1999). Computing Efforts Allocation for Ordinal Optimization and Discrete Event Simulation. To appear in IEEE Transactions on Automatic Control.

Chien, T., Balakrisnan, A., and Wong, R. (1989), An Integrated Inventory Allocation and Vehicle Routing Problem, Transportation Science, 23, 67–76.

Cordeau, J.-F., Laporte, G., Legato, P. and Moccia, L. (2005) Models and tabu search heuristics for the berth-allocation problem. Transportation Science 39(4), 526–538.

Cordeau, J.-F., Laporte, G., Savelsbergh, M.W.P., Vigo, D. (2007), Short-Haul Routing, in Handbooks in Operations Research and Management Science: Transportation G. Laporte and C. Barnhart (eds.). Forthcoming.

Crainic T. G. Perboli, G. Rei W. Tadei, W. (2010). Efficient Heuristics for the Variable. Size Bin Packing Problem with. Fixed Costs. Technical Report. Cirrelt (Montreal, Canada).

Danna, E. Rothberg, E. Le Pape, C. (2005), Exploring relaxation induced neighborhoods to improve MIP solutions, Mathematical Programming,. 102, 1, 71-90.

De Haan, L. (1981), Estimation of the minimum of a function using order statistics. J. Armer. Statist. Assoc. 76 467-469.

Dror, M., Ball, M. (1987), Inventory/Routing: Reduction from an Annual to a Short-Period Problem, Naval Research Logistics Quarterly 34, 891-905.

Dror, M., Ball, M., Golden, B. (1985), A Computational Comparison of Algorithms for the Inventory Routing Problem, Annals of Operations Research 4, 3-23.

Eiben, A. E. and Schippers, C. A. (1998). On evolutionary exploration and exploitation. Fundamenta Informaticae, 35, 35–50.

Expertfit version 6.00. Copyright 1995-2004 by Averill M. Law.

Federgruen, A., Simchi–Levi, D. (1995), Analysis of Vehicle Routing and Inventory–Routing Problems, in: Handbooks in Operations Research and Management Science (Ball, M.O., Magnanti, T.L., Monma, C.L. and Nemhauser, G.L., eds.), Vol. 8, 297-373, North–Holland.

Federgruen, A., Zipkin, P. (1984), A Combined Vehicle Routing and Inventory Allocation Problem, Operations Research 32, 1019-1032.

Feller, W.(1968), An Introduction to Probability Theory and its Applications. John Whiley.

Fischetti, M. and Lodi, A. (2003), Local Branching. Mathematical Programming Ser. B, 98, 23-47.

Fischetti, M. Glover, F. and Lodi, A. (2005), The feasibility pump, Mathematical Programming, 104, 91-104.

Fischetti, M. Polo, C. Scantamburlo, M. (2004), A Local Branching Heuristic for Mixed-Integer Programs with 2-Level Variables, Networks 44, 2, 61-72.

Fischetti, M., Romanin Jacur, G. and Salazar Gonzàles, J.J. (2003), Optimization of the Interconnecting Network of a UMTS Radio Mobile Telephone System. European Journal of Operational Research, 144, 56-67.

Fu, M.C. (2001) Simulation optimization in Proceedings of the 2001 Winter Simulation Conference, Peters, B.A., Smith, J.S., Medeiros, D.J., and Rohrer, M.W. (eds), pp. 53-61.

Gallego, G., Simchi–Levi, D. (1990), On the Effectiveness of Direct Shipping Strategy for the One-Warehouse Multi-Retailer R-Systems, Management Science 36, 240-243.

Gallego, G., Simchi–Levi, D. (1994), Rejoinder to ”A Note on Bounds for Direct Shipping Costs”, Management Science 40, 1393.

Garey, M. R. and Johnson, D. S. (1979), Computers and intractability; a guide to the theory of NPcompleteness. W.H. Freeman.

Gaur, V., Fisher, M. (2004), A Periodic Inventory Routing Problem at a Supermarket Chain, Operations Research 52, 813-822.

Gilmore, P.C. Gomory, R.E. (1961), A linear programming approach to the cutting-stock problem (part I), Operations Research, 9, 849–859.

Gilmore, P.C. Gomory, R.E. (1965), Multistage cutting-stock problems of two and more dimensions, Operations Research, 13, 90–120.

Glover, F. (1986), Future paths for integer programming and links to artificial intelligence. Comp. Oper. Res., 13, 533-549.

Glover, F. and Laguna, M. (1997) Tabu Search. Kluwer Academic Publisher, Boston, Dordrecht, London.

Glover, F. and Laguna, M. (1997a), General Purpose Heuristics For Integer Programming: Part I. Journal of Heuristics 2, 343–358.

Glover, F. and Laguna, M. (1997b) General Purpose Heuristics For Integer Programming: Part II. Journal of Heuristics 3, 161–179.

Goldsman, D.M., Kim, S.-H., Marshall, W.S., and Nelson, B.L. (2002) Ranking and selection for steady-state simulation: procedures and perspectives. INFORMS Journal on Computing 14(1), 2-19.

Guan, Y. and Cheung, R.K. (2004). The berth allocation problem: models and solution methods. OR Spectrum 26, 75–92.

Hansen, P., Oguz, C. and Mladenovic, N. (2008). Variable neighborhood search for minimum cost berth allocation. European Journal of Operational Research 191, 636–649.

Hemmelmayr, V. and Doerner, K. F. and Hartl, R. F. and Savelsbergh, M. W. P. (2010), Vendor managed inventory for environments with stochastic product usage. European Journal of Operational Research 202, 686-695.

Hendriks, M., Laumanns, M., Lefeber, E., and Udding, J.T. (2010). Robust cyclic berth planning of container vessels. OR Spectrum 32, 501–517.

Hopper, E. (2000), Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods, Ph.D. thesis, University of Wales, Cardiff, UK.

Imai, A., Nishimura, E. and Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port. Transportation Research Part B 35, 401–417.

Imai, A., Sun, X., Nishimura, E., and Papadimitriou, S. (2005). Berth allocation in a container port: using a continuous location space approach. Transportation Research Part B 39 (3), 199–221.

Ingber, L. (1996), Adaptive simulated annealing (ASA): Lessons learned. Control and Cybernetics . Special Issue on Simulated Annealing Applied to Combinatorial Optimization, 25,1,33-54.

Kim, K.H. and Moon, K.C. (2003). Berth scheduling by simulated annealing. Transportation Research Part B 37, 541–560.

Kim, S.H. and Nelson, B.L. (2006). Selecting the best system. Chapter 18 in Handbooks in Operations Research and Management Science: Simulation, Henderson, S.G. and Nelson, B.L. (eds), Elsevier.

Kirkpatrick, S. Gelatt, C. D. and Vecchi, M. P. (1983), Optimization by simulated annealing. Science, 13 May 1983, 220(4598), 671-680.

Kleywegt, A. J., Nori, V., and Savelsbergh, M. W. P. (2002), The Stochastic Vehicle Routing Problem with Direct Deliveries, Transportation Science, 36, 94–118.

Jaillet, P., Bard, J.F., Huang, L. and Dror, M. (2002). Delivery Cost Approximations for Inventory Routing Problems in a Rolling Horizon Framework. Transportation Science 36, 3, 292-300.

Labbè M. and Louveaux F.L., (1997), Location Problems, in: M. DellAmico, F. Maffioli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, New York, 261-281.

Lee, Y. and Chen, C.Y. (2009). An optimization heuristic for the berth scheduling problem. European Journal of Operational Research 196 (2), 500–508.

Lee, L.H., Chew, E.P. and Manikam, P. (2006). A general framework on the simulation-based optimization under fixed computing budget. European Journal of Operational Research 174 (3), 1828–1841.

Legato, P. and Mazza, R.M. (2001) Berth planning and resources optimisation at a container terminal via discrete event simulation. European Journal of Operational Research 133(3), 537-547.

Legato, P., Mazza, R.M. and Trunfio, R. (2008) Simulation-based optimization for the quay crane scheduling problem, in Proceedings of the 2008 Winter Simulation Conference, Mason, S.J., Hill, R., Moench, L., and Rose, O. (eds), pp. 2717–2725.

Lim, A. (1998). The berth planning problem. Operations Research Letters 22, 105-110.

Lodi, A. Martello, S. Vigo, D. (1999), Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS Journal on Computing, 11, 345–357.

Lodi, A. Martello, S. Vigo, D. (2002), Recent advances on two-dimensional bin packing problems, Discrete Applied Mathematics, 123, 373–390.

Lodi, A. Martello, S. Vigo, D. (2004), Models and bounds for two-dimensional level packing problems, Journal of Combinatorial Optimization, 8, 3, 363–379.

Løkketangen, A. (2002), Heuristics for 0-1Mixed-Integer Programming. In P.M. Pardalos and M.G.C. Resende (ed.s) Handbook of Applied Optimization, Oxford University Press, 474–477.

Lourenço, H. R. Martin, O. and Stűtzle, T. (2002), Iterated local search. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, volume 57 of International Series in Operations Research & Management Science, pages 321.353. Kluwer Academic Publishers, Norwell, MA.

Lundy, M. and Mees, A. (1986), Convergence of an annealing algorithm. Mathematical Programming, 34, 1, 111-124.

Martello, S. Vigo, D. (1998), Exact solutions of the two-dimensional finite bin packing problem, Management Science, 44, 388–399

Meisel, F. and Bierwith, C. (2006). Integration of berth allocation and crane assignment to improve the resource utilization at a seaport container terminal. Operations Research Proceedings 2005, Haasis H.-D., Kopfer H., Schonberger J. (Eds.) Springer, Berlin, pp. 105-110.

Mladenovìc, N. and Hansen, P. (1997), Variable Neighborhood Search. Computers and Operations Research 24, 1097–1100.

Moin, N. H. and Salhi, S. (2007) Inventory routing problems: A logistical overview. Journal of Operational Research Society, 58, 1185-1194.

Monaci, M. Toth, P. (2006), A set-covering based heuristic approach for bin-packing problems, INFORMS Journal on Computing, 18, 1, 71–85.

Moorthy, R. and Teo, C.P. (2006). Berth management in container terminal: the template design problem. OR Spectrum 28, 495-518.

Nediak, M. and Eckstein, J. (2001), Pivot, Cut, and Dive: A Heuristic for 0-1 Mixed Integer Programming. Research Report RRR 53-2001, RUTCOR, Rutgers University, October.

Nemhauser, G. L. and Wolsey, A. L. (1988), Integer and Combinatorial Optimization. John Wiley & Sons, New York.

Oliveira, J.F. Ferreira, J.S. (1994) A faster variant of the Gilmore and Gomory technique for cutting stock problems, JORBEL—Belgium Journal of Operations Research, Statistics and Computer Science, 34,1, 23–38.

Osman, I. H. (1993), Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Annals of Operations Research, 41, 421-451.

Osman, I. H. and Laporte, G. (1996), Metaheuristics: A bibliography. Annals of Operations Research, 63, 513-623.

Papadimitriou, C. H. and Steiglitz, K. (1982), Combinatorial Optimization - Algorithms and Complexity. Dover Publications, Inc., New York.

Pisinger, D. Sigurd, M. M. (2007), Using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem, INFORMS Journal on Computing, 19, 1, 36-51.

Puchinger, J. Raidl, G. R. (2007), Models and algorithms for three-stage two-dimensional bin packing, European Journal of Operational Research, 183, 1304–1327

Puchinger, J. Raidl, G. R. Koller, G. (2004), Solving a real-world glass cutting problem, in: Evolutionary Computation in Combinatorial Optimization—EvoCOP 2004, in: J. Gottlieb, G.R. Raidl (Eds.), LNCS, vol. 3004, Springer, 162–173.

Puterman, M. L. (1994), Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY.

Qu, W. Bookbinder, J. Iyogun, P. (1999). An integrated inventory-transportation system with modified periodic policy for multiple products, European Journal of Operational Research 115, 254-269.

Rinott, Y. (1978) On two-stage selection procedures and related probability-inequalities. Communications in Statistics - Theory and Methods A7(8), 799-811.

Reeves, C. R. (1993), editor. Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publishing, Oxford, England.

Ross, S. (1994). A First Course in Probability. Prentice Hall Inc.

Savelsbergh, M. W. P., Song, J.-H. (2007), Inventory Routing with Continuous Moves, Computers and Operations Research, 34, 1744-1763.

Savelsbergh, M. W. P., Song, J.-H. (2008), An optimization algorithm for the inventory routing problem with continuous moves, Computers and Operations Research, 35, 2266-2282.

Shi, L. and Olafsson, S. (2000), Nested partitions method for global optimization, Operations Research. 48,3, 390-407.

Song, J.-H., Savelsbergh, M. W. P. (2007), Performance Measurements for Inventory Routing, Transportation Science, Transportation Science, 41, 44-54.

Speranza, M.G., Ukovich, W. (1994), Minimizing Transportation and Inventory Costs for Several Products on a Single Link, Operations Research 42, 879-894.

Stűtzle, T. (1999), Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications. DISKI - Dissertationen zur Kunstliken Intelligenz. infix, Sankt Augustin, Germany.

Vanderbeck, F. (1998), A nested decomposition approach to a 3-stage 2-dimensional cutting stock problem, Management Science 47, 2, 864–879.

Vis, I. F. A. and De Koster, R. (2003) Transvesselment of containers at a container terminal: an overview. European Journal of Operational Research 147,1, 1-16.

Viswanathan, S., Mathur, K. (1997), Integrating Routing and Inventory Decisions in One Warehouse Multiretailer Multiproduct Distribution System, Management Science 43, 294-312.

Wang, F. and Lim, A. (2007) A stochastic beam search for the berth allocation problem. Decision Support Systems 42(4), 2186–2196.

White, C. C. and White, D. J. (1989), Markov Decision Processes, European Journal of Operational Research, 39, 1–16.

Yaman, H. (2002). Concentrator location in telecommunication network. Master Dissertation, University of Bruxelles.

Yang, W. H., Mather, K., and Ballou, R., (2000), Stochastic Vehicle Routing Problem with Restocking, Transportation Science, 34, 99–112.

Yun, W.Y. and Choi, Y.S. (1999) A simulation model for container-terminal operation analysis using an object-oriented approach. International Journal of Production Economics 59, 221-230.

Zhiglijavsky, A. A. (1991). Theory of Global Random Search. Kluver Academic Publishers, Norwell, MA.

Zhou, P.F. and Kang, H.G. (2008). Study on Berth and Quay-crane Allocation under Stochastic Environments in Container Terminal. Systems Engineering — Theory & Practice 28 (1) 161-169.