UNIVERSITÀ DELLA CALABRIA

Facoltà di Ingegneria

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)

DOTTORATO DI RICERCA IN RICERCA OPERATIVA, XXI CICLO (2006-08)

Il Problema del VMI: un approccio risolutivo basato sul metodo di decomposizione

Coordinatore: Prof. Lucio Grandinetti

Dottorando: Francesco Rende

Relatori:
Prof. Roberto Musmanno

Ing. Demetrio Laganà

Settore Scientifico-Disciplinare (SSD) : **MAT/09**

Il Problema del VMI: un approccio risolutivo basato sul metodo di decomposizione

Francesco Rende

Tesi di dottorato in Ricerca Operativa

Indice

Prefazione

L'organizzazione logistica della distribuzione delle merci ai clienti è un aspetto che interviene fortemente nelle politiche di pianificazione dei processi produttivi e distributivi. Nei casi in cui la rete logistica include attori che sono geograficamente molto distanti tra di loro (anche centinania o migliaia di chilometri) si rende necessario immagazzinare delle scorte generando un sistema distributivo *multistadio*. In questi casi, quando esistono più stadi di immagazzinamento che separano la produzione dal consumo, le fluttuazioni della domanda sono amplificate dalla rete di distribuzione. Questo effetto, noto in letteratura come effetto *Bullwhip* e descritto da Forrester ([11]), è causato dalle diverse politiche di gestione degli ordini di rifornimento attuate ad ogni stadio della distribuzione. Un sistema distributivo può essere classificato di tipo *pull* o di tipo *push*. In un sistema distributivo di tipo *pull* ogni centro di distribuzione decide cosa è richiesto per ogni periodo di pianificazione ed ordina direttamente dai propri fornitori. Viceversa, in un sistema distributivo di tipo *push*, il centro di distribuzione principale (*vendor*) determina le esigenze di ogni altro centro distributivo cliente (*customer*), e provvede agli ordini di ripristino verso i *customer*. I vantaggi derivanti dall'adozione di un sistema centralizzato di tipo *push* sono molteplici e si realizzano in consistenti

risparmi sui costi di gestione dell'intera catena distributiva. A differenza del sistema di tipo *pull*, nel sistema di tipo *push* il flusso informativo è bidirezionale lungo la *Supply Chain*, dal momento che il *vendor* decide gli ordini di distribuzione verso i centri periferici, sulla base delle quantità di scorta monitorate e sulla base delle previsioni di domanda per gli stessi centri periferici. Questo tipo di approccio è noto nella letteratura scientifica come VMI (*Vendor Managed Inventory*) ed implica la soluzione di due problemi tradizionalmente affrontati in modo separato: il problema della gestione ottimale delle scorte (cosa rifornire, in che quantità e quando) ed il problema di *routing* distributivo (schedulare in modo ottimale le consegne giornaliere ai clienti).

Nel presente lavoro di tesi il paradigma del VMI è stato affrontato per studiare i modelli matematici che descrivono il problema nel caso in cui il processo distributivo riguarda non un singolo prodotto ma un numero elevato di prodotti. L'analisi della complessità computazionale del problema così definito ha suggerito l'adozione di un approccio risolutivo di tipo euristico basato sul metodo di decomposizione del problema in due distinti sotto problemi: un problema di pianificazione ed un problema di *vehicle routing*. Infine, la realizzazione di un dimostratore *software* ha consentito di condurre una sperimentazione computazionale sulla base di *data set* rappresentativi di scenari applicativi reali.

Capitolo 1

Introduzione

1.1 Una forma di collaborazione nella *Supply Chain*: Il VMI

La realizzazione di progetti di collaborazione tra aziende è oggi considerata un'importante opportunità per raggiungere e mantenere posizioni di vantaggio competitivo. Il ricorso a relazioni di *partnership* tra cliente e fornitore è, infatti, un fenomeno diffuso in molti settori industriali, che continua a ricevere l'attenzione dei *manager*, consulenti e ricercatori. In molti casi, lo scambio d'informazioni tra clienti e fornitori si limita alla comunicazione dei soli dati relativi alla disponibilità dei prodotti, ai prezzi, alle date di consegna. Tradizionalmente, il fornitore riceve gli ordini dal cliente, controlla la disponibilità dei prodotti richiesti e ne pianifica la consegna al cliente. Non essendo in grado di conoscere né quanto gli verrà ordinato, né quando riceverà l'ordine il fornitore è quindi costretto ad elaborare delle previsioni di vendita, che per la loro natura

aleatoria, inducono a mantenere scorte di sicurezza spesso non necessarie o che talvolta possono rivelarsi non adeguate a soddisfare la domanda reale. In tale scenario, anche il cliente, non conoscendo la disponibilità delle scorte presso il fornitore, né lo stato di avanzamento degli ordini, è portato a mantenere livelli di giacenza tali da salvaguardarsi dal rischio di possibile mancate forniture. Questo fenomeno, causato da carenze negli scambi informativi tra cliente e fornitore è stato efficacemente descritto da Houlihan ([13]) che ha fornito una rappresentazione grafica della spirale negativa creata da una scorretta gestione della domanda in una *supply chain*. Sostanzialmente, gli attori della *supply chain* si trovano ad operare alternativamente in condizioni di impossibilità di soddisfare gli ordini dei clienti e di eccessiva presenza di scorte con la conseguenza che i clienti sono insoddisfatti a causa di inevasi, solleciti, allungamento dei tempi di consegna, mancate vendite, scorte elevate e rischio di obsolescenza dei prodotti. Il *Vendor Managed Inventory* è una tecnica utile per affrontare e risolvere tali problematiche. Si tratta di una forma di collaborazione tra cliente e distributore che prevede che quest'ultimo si assuma la responsabilità di decidere il livello di inventario più appropriato da mantenere presso il magazzino del cliente, per l'insieme dei prodotti forniti, al fine di garantire un livello di servizio concordato tra le parti. Ciò significa che il fornitore si assume l'onere di monitorare il livello delle scorte del cliente e, in base al consumo previsto, organizzare i rifornimenti periodici, decidendo quantità, tempi e spedizioni. Tutto ciò è reso possibile da una serie di informazioni, riguardanti le vendite dei prodotti e lo stato corrente delle scorte, che il cliente mette a disposizione del fornitore. Un importante requisito per un'efficace implementazione del VMI risulta essere la presenza di un adeguato scambio informativo tra cliente e fornitore che permetta la

condivisione delle informazioni necessarie. La riduzione del tempo di trasferimento dei dati e degli errori di *data entry* può essere facilitato attraverso l'utilizzo della tecnologia EDI (*Electronic Data Interchange*) o di Internet che consentono la comunicazione tempestiva delle informazioni dai nodi periferici (*customer*) della rete di distribuzione al nodo centrale (*vendor*).

I benefici del VMI sono risultati essere evidenti sin dai primi casi d'implementazione tra la fine degli anni ottanta e gli anni novanta in alcune famose aziende tra le quali Wal-Mart, Procter & Gamble, Whitbread Beer, Barilla, etc.. Alcuni dei vantaggi del VMI più comunemente riportati in letteratura sono:

- fidelizzazione del cliente;
- riduzione del livello di scorte;
- riduzione del numero e della frequenza degli *stock out*;
- riduzione del costo e degli errori di *data entry*;
- aumento della flessibilità nella pianificazione della produzione e della distribuzione;
- miglioramento del livello di servizio del cliente.

Di contro è opportuno sottolineare che l'implementazione del VMI non è priva di difficoltà e complicazioni. Tra i principali fattori d'ostacolo vi sono la presenza di obiettivi contrastanti tra i diversi attori della *supply chain*, la riluttanza da parte delle aziende nel condividere informazioni riservate, nonché la necessita di trovare incentivi per evitare comportamenti opportunistici.

1.2 Il problema del VMI nella letteratura scientifica

Dal punto di vista della definizione del modello matematico il VMI trova implementazioni modellistiche attraverso i problemi di IRP (*Inventory Routing Problem*). Tali problemi si pongono l'obiettivo di risolvere in maniera integrata due problematiche che sono tradizionalmente risolte separatamente: la gestione ottimale delle scorte e la distribuzione ottimale delle merci. La letteratura scientifica sull' IRP è abbastanza ricca per quanto riguarda lo studio dei casi in cui ci si riconduce ad una sola classe o tipologia di prodotti (*single product*) nei diversi centri periferici, ed in cui la flotta di veicoli sia omogenea per capacità di carico. Ricorrenti esempi di modelli IRP, nel caso di *single product*, sono quelli relativi alla distribuzione del gas o carburante (in generale materie prime caratterizzate da un elevato costo di produzione) da uno o più centri di produzione (*plants*) verso centri di commercializzazione o consumo della materia prima. Tali esempi sono particolarmenti diffusi nelle aree geografiche molto vaste dove la localizzazione dei centri di consumo può risultare molto distante dai centri di produzione (ad esempio USA o CANADA) D'altra parte, si registra un numero limitato di lavori scientifici per quanto riguarda i casi in cui il numero di prodotti, oggetto di inventario e distribuzione, risulta essere elevato (*multi product*) come ad esempio nel caso delle aziende che operano nel settore della distribuzione alimentare o in generale della GDO (Grande Distribuzione Organizzata). Laddove si trovano modelli più flessibili sotto questi aspetti, spesso ci si riconduce a formulazioni *standard* o a formulazioni troppo complesse e, di conseguenza, difficilmente applicabili nei casi reali. La modellistica scientifica sul IRP consiste essenzialmente in:

- modelli di pianificazione giornaliera con domanda deterministica o stocastica;

- modelli di pianificazione di medio termine (modelli *multiday*) con domanda deterministica o stocastica;
- modelli di *routing* permanente per gli scopi di pianificazione a lungo termine.

Per quanto riguarda i modelli di pianificazione giornaliera, l'obiettivo è minimizzare le rotte per le consegne ai clienti su un orizzonte di pianificazione giornaliero. Questo approccio risulta molto riduttivo nella misura in cui pospone tutte le consegne dei clienti che non rientrano nel giorno prescelto, lasciando molte consegne per i giorni futuri e generando in alcuni casi soluzioni inammissibili. Tuttavia tali modelli presentano una buona trattabilità computazionale. I maggiori contributi presenti in letteratura sono dovuti a Beltrami e Bodin ([6]), Federgruen e Zipkin ([10]) e Assad et al. ([1]). I modelli di pianificazione di medio termine sono più consistenti dei modelli giornalieri. Quest'ultimi, benchè implichino approcci risolutivi complessi, producono soluzioni ammissibili robuste.

Una caratteristica fondamentale dei modelli *multiday* è la rappresentazione degli effetti di lungo termine sulle decisioni di breve termine; un altro aspetto peculiare di questi modelli è la selezione dei clienti per la pianificazione nel breve periodo. Un approccio metodologico innovativo ai problemi di *inventory routing* su un orizzonte temporale di alcuni giorni è stato proposto da Dror, Ball e Golden ([2], [4]). Tale approccio è stato esteso su un orizzonte temporale di lungo termine da Jaillet, Bard, Huang, e Dror ([4], [5], [9], [3]) utilizzando una variabile casuale normalmente distribuita con parametri noti per modellare la domanda di prodotto presso i clienti. A partire dal modello di domanda, si determina il giorno ottimale di rifornimento di ogni singolo cliente, indipendentemente dagli altri clienti. Il giorno in cui è previsto il reintegro delle

scorte, il cliente avrà un livello di inventario tale da garantirgli una buona probabilità di non andare in *stockout*, fino alla consegna del prodotto. Nell'approccio di Jaillet et al. ([3]) sono inclusi nella pianificazione solo quei clienti con giorni ottimali di rifornimento entro due settimane successive al periodo di pianificazione (*planning day*). Inizialmente si risolve un problema di assegnamento dei clienti ai giorni, seguito da una fase di *post*-ottimizzazione per evitare situazioni di eccessivo carico di consegne nelle due settimane di pianificazione. Infine, si risolvono tanti problemi di *vehicle routing* in cui la quantità consegnata ad ogni cliente è molto prossima alla quantità di reintegro delle scorte presso il cliente medesimo. Questo approccio è utilizzato in un ambiente di pianificazione del tipo *rolling horizon*, in cui si attuano i risultati forniti dalla soluzione del modello per la prima settimana, e si usano i dati aggiornati alla fine della prima settimana per pianificare la seconda settimana.

I modelli di pianificazione a lungo termine, denominati anche modelli di *routing* permanente o *routing* periodico, comportano la determinazione di un giorno di schedulazione delle consegne che può essere ripetuto sempre. Questo approccio, descritto da Christofides e Beasley ([14]) e da Gaudioso e Paletta ([12]), risulta particolarmente efficiente per la determinazione della dimensione della flotta di veicoli piuttosto che per la pianificazione delle consegne. L'intrinseca natura stocastica del problema rende questo approccio poco efficiente. Negli ultimi anni sono stati prodotti molti studi incentrati sulla natura stocastica della domanda. Kleywegt et al. ([16], [15]]) propongono un modello decisionale Markoviano per l'*Inventory Routing Problem Stocastico* (SIRP) ed un approccio risolutivo con i metodi della programmazione dinamica. Nonostante i risultati computazionali siano interessanti e promettenti, tale approccio è poco perfor-

mante per i casi reali. Inoltre, nelle situazioni reali è molto difficile riuscire a reperire le informazioni necessarie per costruire una distribuzione di probabilità che rappresenti, in modo statisticamente corretto, le variabili che definiscono il problema.

1.3 Contributi

Obiettivo del lavoro di tesi è stato lo studio dei modelli e dei metodi risolutivi dei problemi di IRP nel costesto dei centri di distribuzione che operano in alcuni settori all'interno dei quali il processo di distribuzione coinvolge un elevato numero di referenze o prodotti. Questo è il caso delle realtà aziendali che operano nel settore della distribuzione di prodotti alimentari e diversi (*grocery*) e che è stato oggetto di analisi nel lavoro di tesi. Sulla base delle esigenze rilevate in tale settore, è stato formulato un modello matematico rappresentativo del problema e successivamente sono state definite delle euristiche risolutive del modello matematico. In particolare, il problema è stato studiato sia rispetto alle problematiche del centro di distribuzione (nodo *vendor* della rete di distribuzione) sia rispetto alle problematiche dei punti vendita riforniti dal centro di distribuzione (punti vendita periferici, in seguito definiti, nodi *customer*) effettuando le seguente attività:

- definizione di un modello matematico, definito *Vendor Model*, che si pone l'obiettivo di minimizzare il costo complessivo della distribuzione nel caso in cui il problema è affrontato dal punto di vista del centro di distribuzione;
- definizione di un modello matematico, definito *Customer Model*, che si pone l'obiettivo di minimizzare il numero di consegne effettuate presso il *customer* salva-

guardando al tempo stesso il livello di servizio; il problema è analizzato dal punto di vista delle esigenze di rifornimento dei clienti;

- definizione di una metodologia risolutiva del modello matematico mediante un approccio a due fasi (conosciuto in letteratura come metodo di decomposizione) che suddivide il problema in due sottoproblemi: un primo problema di pianificazione delle rotte ed un secondo problema di *vehicle routing* per l'instradamento dei veicoli lungo una rotta di consegna;

- la definizione di tre diverse euristiche risolutive basate sulla ricerca locale e sulla tecnica di *Nested Partitioning*;

- sperimentazione computazionale sulla base di opportuni *data set* rappresentativi di scenari aziendali reali e realizzazione di un dimostratore *software*.

Capitolo 2

Un approccio risolutivo basato sul metodo di decomposizione

2.1 Introduzione

Lo studio del modello di distribuizione secondo il paradigma del VMI è stato affrontato con riferimento ad una catena logistica di secondo grado (*two echelon*) caratterizzata da un unico fornitore (*one origin*) che dispone di P prodotti (*multi product*), riforniti a M *customer* (*multiple destinations*), attraverso l'utilizzo di una flotta di veicoli con capacità omogenea su un orizzonte di pianificazione finito. Il numero di *echelon* indica il numero di livelli presenti tra il primo attore della catena, che nel caso di studio coincide con il CeDi, e l'ultimo soggetto della filiera corripondente ai punti vendita periferici.

L'obiettivo del modello matematico è la definizione di un'efficiente strategia di rifornimento degli ordini ai *customer* in grado di determinare le quantità di prodotto da

inviare, i periodi di rifornimento e le rotte di consegna da attivare da parte del fornitore, con l'obiettivo di minimizzare il costo totale di trasporto delle consegne sull'intero periodo di pianificazione.

Dal punto di vista della definizione del modello matematico l'attenzione è posta al caso in cui il numero di prodotti coinvolti nel processo distributivo è caratterizzato da un evelato numero di prodotti e/o categorie di prodotti ordinati dai diversi *customer* (*Multi Product* IRP, nel seguito faremo riferimento a tale tipologia di problemi con l'acronimo MIRP). Saranno presentati due diversi modelli matematici per il MIRP: il primo modello, denominato (*Vendor Model*) si pone l'obiettivo di risolvere il problema dal punto di vista del Centro di Distribuzione (nodo *Vendor* della rete di distribuzione) mediante la definizione di una funzione obiettivo che minimizza i costi complessivi legati alla distribuzione dei prodotti; il secondo modello, denominato *Customer Model*, si pone l'obiettivo di ottimizzare il processo di distribuzione dal punto di vista del *Customer* in modo tale da garantire un determinato livello di servizio ed evitare situazioni di *stock-out*.

La risoluzione del problema affrontato presenta una elevata complessità computazionale a causa:

- della presenza di un elevato numero di variabili decisionali e di vincoli operativi legati al rispetto delle consegne;
- dell'elevato numero di rotte ammissibili di consegna;
- dell'estensione del periodo di pianificazione.

Una buona trattabilità computazionale di questo modello si può ottenere riducendo opportunamente il numero di rotte ed aggregando i periodi finali dell'orizzonte temporale. A tale scopo si propone un approccio risolutivo basato sul metodo di decomposizione che aggiunge, rispetto a quanto già proposto in letteratura da Campbell e Savelsbergh ([7]), la possibilità di rifornire uno stesso *customer* più volte nella stessa giornata, consegnando una diversa quantità di prodotto.

2.2 La fase di *planning*

Il modello di pianificazione adottato ha una struttura temporale di tipo *rolling horizon*. Tale struttura temporale consente di ridurre il numero di variabili decisionali aggregando le variabili relative alle rotte da attivare nei giorni posti alla fine del periodo di pianificazione T . In particolare, l'approccio *rolling horizon* prevede che:

1. in ogni giorno dell'orizzonte temporale si risolva il modello di *planning* con i dati aggiornati dall'inizio del periodo di pianificazione fino al giorno immediatamente precedente ai giorni da pianificare;

2. si utilizzino i dati di *output* del modello di *planning* per inizializzare l'euristica di *routing* relativamente ai primi t giorni dell'orizzonte temporale;

3. si implementino i risultati del *routing* e si ponga l'orologio temporale in avanti di t giorni.

Sulla base di tale modello è possibile ottenere una significativa riduzione del numero di rotte assumendo che i prodotti, o le categorie di prodotti (insieme di prodotti con

tasso medio di consumo giornaliero dello stesso ordine), possano essere consegnati con una stessa rotta soltanto se appartengono allo stesso *cluster*. Un *cluster* è un insieme di coppie (p, i), rappresentative della consegna del prodotto p al *customer* i, associate, compatibilmente con i vincoli di capacità, ad un determinato veicolo per un lungo periodo (generalmente coincidente con il periodo di pianificazione T). Due *cluster* si definiscono disgiunti se non hanno prodotti in comune. Per definire una buona partizione dell'insieme dei prodotti in *cluster*, occorre:

- generare un numero sufficientemente elevato di *cluster*;
- definire il costo di ciascun *cluster*;
- assegnare ogni prodotto ad un solo *cluster* risolvendo un problema di *set partitioning*.

2.2.1 Definizioni preliminari

Il MIRP è definito su una rete logistica rappresentata da un grafo $G = (V, E)$ non orientato, dove $V = \{0, \ldots, n\}$ è l'insieme dei vertici e $E = \{(i, j) : i \in V, j \in V, i < j\}$ rappresenta l'insieme degli archi del grafo. Sia $c_{ij} = c_{ji} \geq 0$ il costo, non negativo, associato all'arco $(i, j) \in E$ e $T = \{0, \ldots, H\}$ un insieme finito e discreto di istanti temporali (giorni), a partire da un istante iniziale 0 sino ad un istante temporale finale denominato H. Nel periodo di pianificazione, ogni veicolo v, di una flotta di veicoli con capacità omogenea Q, effettua la distribuzione di un insieme di prodotti p al *customer* $i \in M = V \backslash \{0\}$. Si assume che il nodo $\{0\}$ sia rappresentativo del *vendor*.

Lo stato iniziale di ogni *customer* è definito dai seguenti parametri:

- P_i è l'insieme dei prodotti richiesti dal *customer* $i \in M$ all'interno del periodo di pianificazione;
- I_{pi}^0 livello iniziale dell'inventario del prodotto $p \in P_i$ presso il *customer* $i \in M$ all'inizio del periodo di pianificazione;
- U_{pi} indica il massimo livello di inventario del prodotto $p \in P_i$ presso il *customer* $i \in M$;
- μ_{pi} indica il tasso di consumo medio del prodotto $p \in P_i$ presso il *customer* $i \in M$; tale valore è assunto essere costante nel periodo di pianificazione.

Sulla base di tali parametri, la quantità minima di prodotto $p \in P_i$ necessaria al *customer* $i \in M$ per evitare una situazione di *stock-out* nel giorno $t \in T$ può essere calcolata mediante la seguente espressione:

$$LL_{pi}^t = max\{0, t\mu_{pi} - I_{pi}^0\}$$

viceversa, la quantità massima di prodotto $p \in P_i$ che può essere consegnata al *customer* $i \in M$ nel giorno $t \in T$ è:

$$UL_{pi}^t = U_{pi} + t\mu_{pi} - I_{pi}^0$$

Dal punto di vista delle politiche di gestione delle scorte, e senza perdita di generalità, è possibile assumere che ogni *customer* $i \in M$ adotti la politica EOQ (*Economic Order Quantity*) per ogni prodotto $p \in P_i$. I valori forniti in *output* dalla politica EOQ di gestione delle scorte sono:

- $q_{pi} = \sqrt{\frac{2f_p\mu_{pi}}{h_{pi}}}$ la quantità di prodotto $p \in P_i$ da consegnare al *customer* $i \in M$; dove f_p rappresenta il costo fisso di riordino del prodotto $p \in P_i$ per il *customer* $i \in M$;

- K_{pi} il numero di consegne del prodotto $p \in P_i$ al *customer* $i \in M$ nel periodo di pianificazione T;

- t_{pik} il tempo di riordino della k-*esima* consegna, $k = 1, \ldots, K_{pi}$, di prodotti $p \in P_i$ al *customer* $i \in M$.

La formulazione della funzione obiettivo tiene conto della penalità associata ai casi in cui una consegna è anticipata o posticipata rispetto al giorno indicato dalla politica di gestione delle scorte. In particolare, la quantità π^e_{pik} (*early penalty cost*) rappresenta il costo di penalità che interviene nel caso in cui la k-*esima* consegna del prodotto $p \in P_i$ è effettuata prima del giorno t_{pik}. Viceversa, la quantità π^l_{pik} (*delay penalty cost*) rappresenta il costo di penalità che interviene nel caso in cui la k-*esima* consegna del prodotto $p \in P_i$ è effettuata successivamente al giorno t_{pik}

Obiettivo del MIRP è la determinazione della politica di distribuzione capace di minimizzare, nel caso del *Vendor Model*, i costi totali di trasporto e di penalità e, nel caso del *Customer Model*, il numero di consegne complessive effettuate ad ogni *customer* $i \in M$.

2.2.2 *Vendor Model planning*

Il modello matematico è definito dalle seguenti variabili decisionali:

$$x_r^t = \begin{cases} 1 & \text{se la rotta } r \text{ è schedulata nel giorno } t, \\ 0 & \text{altrimenti,} \end{cases}$$

per ogni $r \in R$, and $t \in T$.

$$z_{pik}^t = \begin{cases} 1 & \text{se la consegna } k\text{-}esima \text{ del prodotto } p \text{ al } customer\ i \text{ avviene il giorno } t, \\ 0 & \text{altrimenti,} \end{cases}$$

per ogni $p \in P_i$, $i \in N$, $k = 1, \ldots, K_{pi}$, and $t \in T$.

$$y_{pik}^t = \begin{cases} 1 & \text{se la consegna } k\text{-}esima \text{ del prodotto } p \text{ al } customer\ i \text{ è stata già effettuata il giorno } t, \\ 0 & \text{altrimenti,} \end{cases}$$

per ogni $p \in P_i$, $i \in N$, $k = 1, \ldots, K_{pi}$, e $t \in T$.

Le variabili y_{pik}^t sono utilizzate per tenere traccia delle consegne effettuate in tutto il periodo di pianificazione. Infine, la variabile d_{pir}^t è utilizzata per indicare la quantità di prodotto p consegnata al *customer* i con la rotta r il giorno t, per ogni $(p, i) \in r$, $r \in R$, e $t \in T$.

In questo caso la funzione obiettivo assume la seguente forma:

$$\text{Minimize } \sum_{t\in T}\sum_{\forall r\in R} c_r x_r^t + \sum_{\forall r\in R}\sum_{\forall (p,i)\in r}\sum_{k=1}^{K_{pi}}\Big(\sum_{t=1}^{t^*_{pik}-1}\pi^e_{pik}(t^*_{pik}-t)z^t_{pik} + \sum_{t=t^*_{pik}+1}^{H}\pi^l_{pik}(t-t^*_{pik})z^t_{pik}\Big) \tag{2.1a}$$

ed è soggetta ai seguenti vincoli:

$$LL^t_{pi} \le \sum_{0\le s\le t}\sum_{\forall r\in R:(p,i)\in r} d^s_{pir} \le UL^t_{pi},\ \forall p\in P_i, i\in N, t\in T \tag{2.1b}$$

$$\sum_{\forall (p,i)\in r} d^t_{pir} \le Q x^t_r,\ \forall r\in R, t\in T \tag{2.1c}$$

$$x^t_r = \sum_{k=1}^{K_{pi}} z^t_{pik},\ \forall r\in R, (p,i)\in r, t\in T \tag{2.1d}$$

$$y^t_{pik} = z^{t-1}_{pik} + y^{t-1}_{pik},\ \forall (p,i)\in r, k=1,\dots,K_{pi}, t\in T\backslash\{0\} \tag{2.1e}$$

$$y^t_{pik} \le y^t_{pik-1},\ \forall (p,i)\in r, k=1,\dots,K_{pi}, t\in T\backslash\{0\} \tag{2.1f}$$

$$y^0_{pik} = 0,\ \forall (p,i)\in r, k=1,\dots,K_{pi} \tag{2.1g}$$

$$x^t_r \in \{0,1\},\ \forall r\in R, t\in T \tag{2.1h}$$

$$z^t_{pik} \in \{0,1\},\ \forall p\in P_i, i\in N, k=1,\dots,K_{pi}, t\in T \tag{2.1i}$$

$$y^t_{pik} \in \{0,1\},\ \forall p\in P_i, i\in N, k=1,\dots,K_{pi}, t\in T \tag{2.1j}$$

$$d^t_{pir} \ge 0,\ \forall (p,i)\in r, r\in R, t\in T. \tag{2.1k}$$

La funzione obiettivo (2.1a) si prefigge di minimizzare il costo totale associato alla distribuzione dei prodotti presso i *customer* della rete di distribuzione. La particolarità del modello è rappresentata sia dalla possibilità di servire lo stesso *customer* attraverso

differente rotte sia dalla presenza dei costi di penalità nel caso in cui una consegna avvenga in anticipo o in ritardo rispetto ai tempi previsti dalla politica di gestione delle scorte. Tale fattore contribuisce a minimizzare in maniera significativa il numero di consegne che si discostano dalle indicazioni (ottimali) fornite dalla politica EOQ.

Per quanto riguarda i vincoli, possiamo osservare che:

- i vincoli (2.1b) assicurano l'assenza di *stock-out* di ogni prodotto p al *customer* i;
- i vincoli (2.1c) assicurano che le quantità di prodotto p consegnate sulla rotta r non eccedano la capacità di carico dei veicoli utilizzati per la consegna;
- i vincoli (2.1d) assicurano che se una rotta r è schedulata il giorno t allora almeno una consegna del prodotto p al *customer* i deve avvenire il giorno t;
- i vincolo (2.1e) esprimono la condizione che se un prodotto p è stato già consegnato al *customer* i allora la consegna è avvenuta o nel giorno precedenete o nel periodo precedente;
- i vincoli (2.1f) affermano che la $(k-1)$-esima consegna del prodotto p al cliente i deve avvenire prima della k-esima consegna;
- i vincoli (2.1g) esprimomo la condizione che le consegne dei prodotti non possono essere effettuate prima dell'inizio del periodo di pianificazione;
- i rimanente vincoli esprimono le condizioni di non negatività delle variabili decisionali del modello matematico.

La difficoltà principale, per la risoluzione del modello matematico, risiede nel processo di determinazione delle rotte di consegna R. La ricerca della soluzione ottima richiede la conoscenza di ogni rotta di consegna $r \in R$ in modo tale che si possa definire l'insieme delle variabili x_r^t, per ogni $t \in T$.

L'idea alla base della riduzione della complessità del modello presentato è basato sull'utilizzo di apposite procedure di clusterizzazione dell'insieme delle rotte. Attraverso tale operazioni, è possibile ottenere una significativa riduzione delle rotte partendo dalla constatazione che prodotti o categorie di prodotti possono essere consegnati con una stessa rotta solo se appartengono allo stesso *cluster*. L'aggregazione dei *customer* in *cluster* avviene sulla base dei parametri caratteristici della politica di gestione delle scorte quali, ad esempio, il primo giorno di consegna di un prodotto e l'intervallo di tempo tra due consegne consecutivi di uno stesso prodotto.

Generazione dei *cluster* La prima fase del processo di clusterizzazione richiede la generazione del maggior numero di *cluster* contenenti prodotti tale che:

1. i giorni della prima consegna t_{pi1} ed il numero medio di giorni fra due consegne successive ΔT_{pi} dei prodotti p al *customer* i siano omogenei;

2. i *customer* a cui consegnare tali prodotti siano geograficamente vicini fra di loro e con il *vendor*.

Questa fase è quella che incide maggiormente sulla trattabilità computazionale del modello di *planning*. Sono state studiate ed implementate quattro diverse strategie di clusterizzazione. Ciascuna di esse è stata validata risolvendo il modello matematico

(2.1), in precedenza esposto, a partire dall'insieme R di rotte generato per ciascun *cluster*. Siano:

- t_{pi1} il giorno della prima consegna, nel periodo di pianificazione T, del prodotto p al *customer* i , per ogni $p \in P_i$, $i \in M$;
- ΔT_{pi} l'intervallo di tempo medio fra due successivi riordini del prodotto p al *customer* i (determinato sulla base della politica di gestione delle scorte EOQ):

$$\Delta T_{pi} = t_{pik+1} - t_{pik}$$

- T_w la durata del giorno lavorativo (espressa in ore);
- $(0; t_{pi}; \Delta T_{pi})$ il sistema di riferimento cartesiano con origine nel nodo *vendor* $\{0\}$;
- $(t_{pi1}, \Delta T_{pi})$ le coordinate cartesiane della coppia (p, i), per ogni $p \in P_i, i \in M$;
- $An((p, i))$ l'angolo associato alla coppia (p, i), per ogni $p \in P, i \in M$:

$$An((p, i)) = arctan\left[\frac{\Delta T_{pi}}{t_{pi1}}\right]$$

- $R((p, i))$ la distanza radiale della coppia (p, i), $p \in P, i \in M$, rispetto al nodo *vendor* $\{0\}$:

$$R((p, i)) = \sqrt{(t_{pi1})^2 + (\Delta T_{pi})^2}$$

Siano $An((p, i))$ e $R((p, i))$ le coordinate polari di ciascuna coppia (p, i). Ad ogni coppia (p, i) è assegnata una etichetta di clusterizzazione; tale etichetta è inizializzata

a zero ed è aggiornata con l'indice del *cluster* in cui la coppia (p, i) è inserita. Per ogni veicolo $v = 1, \ldots, F$ si definisce un insieme P_v di prodotti non etichettati. Inizialmente si ha $P_v = \bigcup_{j \in M} P_{jv}$, essendo $P_{iv} = \{(p, i) \in P_i \times i | q_{pi} \leq Q_v\}$.

Sia $AVR_m = \sum_{(p,i) \in P_v} \frac{R((p,i))}{|P_v|}$ il raggio polare medio calcolato sull'insieme di prodotti contenuti in P_v.

Ad ogni iterazione dell'algoritmo di clusterizzazione viene definita una famiglia di *cluster* C_v per la categoria di veicoli $v, v = 1, \ldots, F$.

Le quattro diverse procedure di clusterizzazione sono di seguito descritte.

Procedura CL1 Le coppie (p, i) sono ordinate per valori di t_{pi1} crescenti ed il generico *cluster* C_v^u è costruito selezionando le coppie (p, i) con lo stesso valore di t_{pi1}, sino a che il vincolo di capacità è soddisfatto. Per garantire l'ammissibilità temporale del *cluster* occorre verificare che la durata del TSP (*Travelling Salesman Problem*) dal nodo *vendor* a tutti i nodi rappresentativi dei *customer* serviti in C_v^u , sia minore di T_w. La Figura 2.1 mostra un esempio di clusterizzazione per veicoli di capacità Q_1 , su un insieme $I = \{1, 2, 3\}$ di tre clienti tali che $P_1 = \{A, D, F\}$; $P_2 = \{A, B, C\}$, $P3 = \{L, N\}$. Le coppie (p, i) sono definite da:

$$P_1 \times \{1\} = \{A_1, D_1, F_1\}, P_2 \times \{2\} = \{A_2, B_2, C_2\}, P_3 \times \{3\} = \{L_3, N_3\}.$$

I *cluster* generati con questa tecnica di clusterizzazione sono:

$$C_1^1 = \{A_1, A_2\}; C_1^2 = \{L_3\}; C_1^3 = \{D_1\}; C_1^4 = \{F_1\}; C_1^5 = \{B_2\}; C_1^6 = \{C_2, N_3\}.$$

Per ogni categoria di veicolo l'algoritmo $CL1$ genera una partizione delle coppie (p, i); in questo senso si parla di clusterizzazione senza sovrapposizioni (*not overlap clustering*).

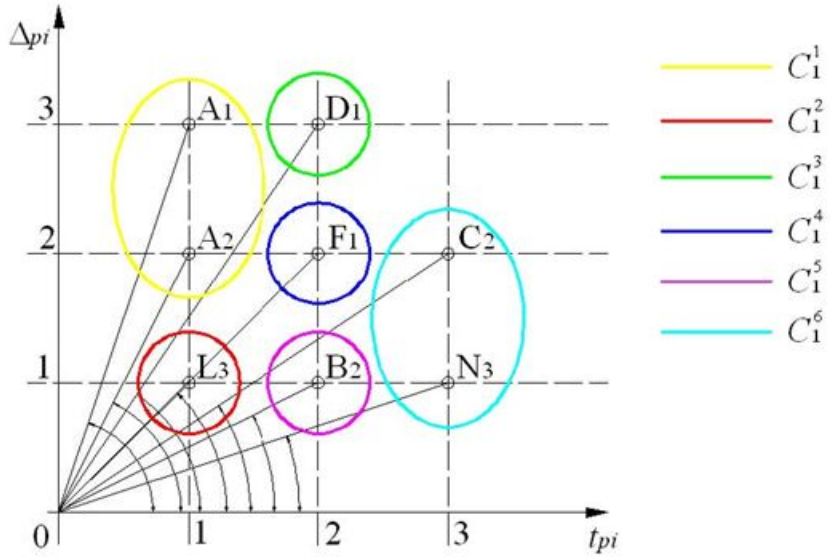

Figura 2.1: La procedura di clusterizzazione CL1

Siano:

- $TTSP(I)$ il tempo di viaggio del TSP sul sottografo $G(I)$, definito dal nodo *vendor* e dall'insieme I dei nodi *customer* da visitare;
- $t_{p[j]j}$ il tempo di viaggio calcolato sullo spigolo $(p[j], j)$ del *tour* del TSP che collega il predecessore del vertice j con j medesimo;
- $t_{js[j]}$ il tempo di viaggio calcolato sullo spigolo $(j, s[j])$ del *tour* del TSP che collega il vertice j con il suo successore;
- $t_{p[j]s[j]}$ il tempo di viaggio sul cammino minimo che collega il predecessore di j con il suo successore nel *tour* del TSP;

- t_{0j} il tempo di viaggio sul cammino minimo che collega il nodo *vendor* $\{0\}$ con j;
- α_1 e α_2 due parametri scalari non negativi. Per semplicità sono stati settati $\alpha_1 = \alpha_2 = \frac{1}{2}$.

L'*outline* della procedura è di seguito descritta.

Algorithm 1 CL1

Require: Set $P = \{(p,i), \forall p \in P_i, i \in M\}$
Ensure: A set partitioning **CL1** $= \{CL1_1, \ldots, CL1_l\}$ of P

for $v = 1$ to V **do**
  $P_v = \cup_{i \in M} P_{iv}$, dove $P_{iv} = \{(p,i) \in P_i \times \{i\} \,|q_{pi} \leq Q_v\}$
  Inizializzazione dell'indice dei *cluster*: $u = 0$
  while $P_v \neq \emptyset$ **do**
    definizione di un nuovo *cluster*: $u = u + 1$
    Inizializzazione del *cluster* u con il veicolo v : $C_v^u = \emptyset$
    while $Q_v - \sum_{(p,i) \in C_v^u} q_{pi} \geq \min_{p_i \in P_i}\{q_{pi}\}$ **do**
      $C_v^u = \{p_i \in P_i | (p,i) = arg\max_{p_i \in P_v}\{t_{pi1}\}\}$
      $P_v = P_v - \{(p,i) \in C_v^u\}$
      $I_v^u = \{i \in V | (p,i) \in C_v^u \cap P_{iv}, \forall i = 1, \ldots, n\} \cup \{0\}$
      Calcola $T_{TSP}(I_v^u)$
      while $T_{TSP}(I_v^u) > T_w$ **do**
        Seleziona $\bar{i} = arg \max_{j \in TSP_v^u - \{0\}} \{\alpha_1(t_{p[j]j} + t_{js[j]} - t_{p[j]s[j]}) + \alpha_2(t_{0j})\}$
        Aggiorna l'insieme delle coppie (p,i) non etichettate: $P_v = P_v \cup \{(p,i) \in C_v^u \cap P_{iv}\}$
        Aggiorna il *cluster*: $C_v^u = C_v^u - \{p_i \in C_v^u \cap P_{iv}\}$
        Aggiorna l'insieme degli indici dei clienti: $I_v^u = I_v^u - \{\bar{i}\}$
        Calcola $T_{TSP}(I_v^u)$
      end while
    end while
  end while
end for

Procedura CL2 Il secondo metodo si basa sulla costruzione di famiglie di *cluster* C_v, $v = 1, \ldots, F$, per ogni categoria di veicoli, a partire dalle coppie (p,i) con il massimo

valore di $An((p,i))$, aggiungendo di volta in volta i prodotti con valori decrescenti di $An((p,i))$, fino a garantire il soddisfacimento del vincolo sulla capacità:

$$Q_v - \sum_{(pi)\in c_v^u} q_{pi} < \min_{p_i \in P_v} \{q_{pi}\}.$$

Sia I_v^n l'insieme dei *customer* visitati nel *cluster* C_v^n incluso il nodo *vendor*, e sia $G(I_v^u)$ il sottografo definito sull'insieme I; per tenere in considerazione la distanza geografica fra i *customer* ed il nodo *vendor* occorre verificare che la durata $T_{TSP}(I_v^u)$ del *tour* del TSP, calcolato su tutti i vertici di $G(I_v^u)$, non risulti maggiore di T_w . Superata questa seconda verifica, tutti i prodotti inseriti nel *cluster* C_v^u sono etichettati e l'insieme P_v viene aggiornato. Gli altri *cluster* di C_v sono generati con la stessa tecnica. L'iterazione v-esima dell'algoritmo ha termine quando tutti i prodotti presenti in P_v risultano etichettati. L'iterazione successiva determina la famiglia di *cluster* C_{v+1} per la categoria di veicoli $v+1$. La Figura 2.2 mostra il risultato del metodo CL2 applicato all'esempio introdotto in precedenza. Anche in questo caso la procedura fornisce una partizione dell'insieme delle coppie $(p,i) \in P = \bigcup_{i\in M} P_i$, per ogni $v = 1, \ldots, F$.
L'*outline* della procedura è di seguito descritta.

Algorithm 2 CL2

for $v = 1$ to F **do**
  $P_v == \cup_{i \in M} P_{iv}$, dove $P_{iv} = \{(p,i) \in P_i \times \{i\} | q_{pi} \leq Q_v\}$
  Inizializzazione dell'indice dei *cluster*: $u = 0$
  while $P_v \neq \emptyset$ **do**
    Definizione di un nuovo *cluster*: $u = u + 1$
    Inizializzazione del *cluster* u con il veicolo v: $C_v^u = \emptyset$
    while $Q_v - \sum_{p_i \in C_v^u} q_{pi} \geq \min_{p_i \in P_i}\{q_{pi}\}$ **do**
      $C_v^u = \{p_i \in P_i | p_i = \underset{p_i \in P_v}{argmax}\{An(p_i)\}\}$
      $P_v = P_v - \{p_i \in C_v^u\}$
      $I_v^u = \{i \in V | p_i \in C_v^u \cap P_{iv}, \forall i = 1, \ldots, n\} \cup \{0\}$
      Calcola $T_{TSP}(I_v^u)$
      while $T_{TSP}(I_v^u) > T_w$ **do**
        Seleziona $\bar{i} = arg \underset{j \in TSP_v^u - \{0\}}{max}\{\alpha_1(t_{p[j]j} + t_{js[j]} - t_{p[j]s[j]}) + \alpha_2(t_{0j})\}$
        Aggiorna l'insieme delle coppie (p,i) non etichettate: $P_v = P_v \cup \{p_i \in C_v^u \cap P_{iv}\}$
        Aggiorna il *cluster*: $C_v^u = C_v^u - \{p_i \in C_v^u \cap P_{iv}\}$
        Aggiorna l'insieme degli indici dei clienti: $I_v^u = I_v^u - \{\bar{i}\}$
        Calcola $T_{TSP}(I_v^u)$
      end while
    end while
  end while
end for

Procedura CL3 Il terzo metodo si basa sulla procedura CL2 per la costruzione di famiglie di *cluster* C_v tali che:

$$\left| C_v^u \bigcap C_v^{u+1} \right| = 1, u = 1, \ldots, U, v = 1, \ldots, F.$$

Nello specifico la procedura genera una famiglia di *cluster* tali che l'ultima coppia (p,i) inserita nel *cluster* C_v^u è la prima coppia a far parte del *cluster* C_v^{u+1}. Per esempio,

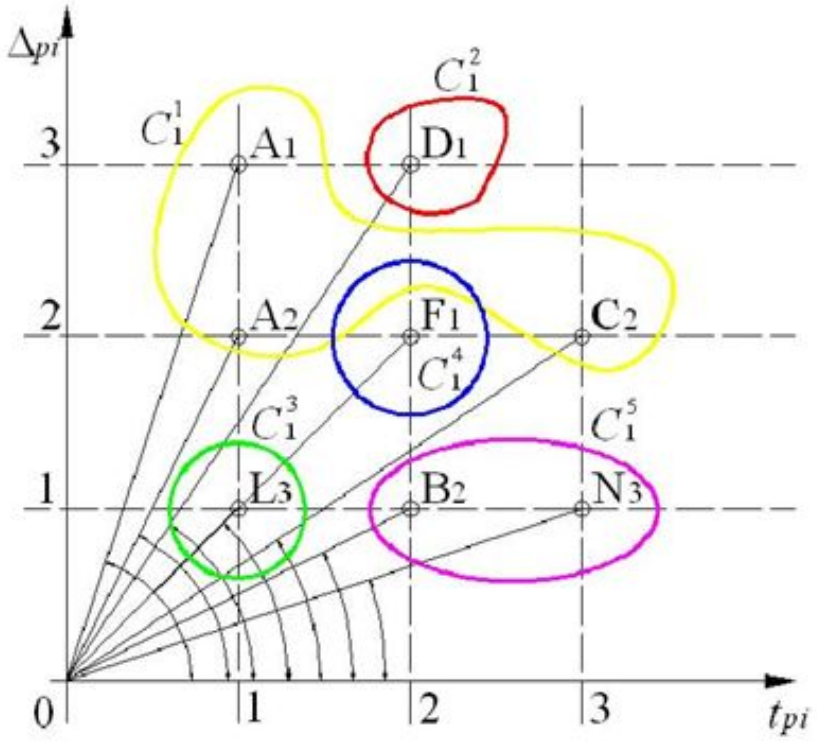

Figura 2.2: La procedura di clusterizzazione CL2

se $C_1^1 = \{A_1, A_2, C_2\}$ è il primo *cluster* generato con veicoli di capacità Q_1, il secondo *cluster* è inizializzato con la coppia C_2 , cosicché $C_1^2 = \{C_2, D_1\}$. Il livello di sovrapposizione fra due successivi sottoinsiemi di C_v, $v = 1, \ldots, F$, è rappresentato da una sola coppia (prodotto, *customer*). Questo metodo determina un ricoprimento dell'insieme di coppie (p, i), per ogni categoria di veicoli. L'*outline* della procedura è di seguito descritta.

Algorithm 3 CL3

1: **for** $v = 1$ to F **do**
2: $P_v == \cup_{i \in M} P_{iv}$, dove $P_{iv} = \{(p,i) \in P_i \times \{i\} | q_{pi} \leq Q_v\}$
3: Inizializzazione dell'indice dei *cluster*: $u = 0$
4: **while** $P_v \neq \emptyset$ **do**
5: Definizione di un nuovo *cluster*: $u = u + 1$
6: Inizializzazione del *cluster* u con il veicolo v: $C_v^u = \emptyset$
7: **while** $Q_v - \sum_{p_i \in C_v^u} q_{pi} \geq \min_{p_i \in P_i} \{q_{pi}\}$ **do**
8: $C_v^u = \{p_i \in P_i | p_i = \underset{p_i \in P_v}{argmax} \{An((p,i))\}\}$
9: $P_v = P_v - \{(p,i) \in C_v^u\}$
10: $I_v^u = \{i \in V | p_i \in C_v^u \cap P_{iv}, \forall i = 1, \ldots, n\} \cup \{0\}$
11: Calcola $T_{TSP}(I_v^u)$
12: **while** $T_{TSP}(I_v^u) > T_w$ **do**
13: Seleziona $\bar{i} = arg \underset{j \in TSP_v^u - \{0\}}{max} \{\alpha_1(t_{p[j]j} + t_{js[j]} - t_{p[j]s[j]}) + \alpha_2(t_{0j})\}$
14: Aggiorna l'insieme delle coppie (p,i) non etichettate: $P_v = P_v \cup \{(p,i) \in C_v^u \cap P_{iv}\}$
15: Aggiorna il *cluster*: $C_v^u = C_v^u - \{(p,i) \in C_v^u \cap P_{iv}\}$
16: Aggiorna l'insieme degli indici dei clienti: $I_v^u = I_v^u - \{\bar{i}\}$
17: Calcola $T_{TSP}(I_v^u)$
18: **end while**
19: **end while**
20: **if** $|C_v^u| > 1$ **then**
21: Seleziona l'ultima coppia $\overline{p_i}$ inserita in C_v^u
22: Costruisci un nuovo *cluster*: $u = u + 1$
23: $C_v^u = \{\overline{p_i}\}$
24: **else**
25: Costruisci un nuovo *cluster*: $u = u + 1$
26: $C_v^u = \emptyset$
27: **end if**
28: **end while**
29: **end for**

Procedura CL4 Anche il quarto metodo si ispira alla metodologia CL2 per la costruzione di famiglie di *cluster* C tali che:

$$\left| C_v^u \bigcap C_v^{u+1} \right| = 2, u = 1, \ldots, U, v = 1, \ldots, F.$$

Come per $CL3$ il risultato finale è un ricoprimento dell'insieme di coppie (p, i) per ogni tipo di veicolo . Il livello di sovrapposizione fra due *cluster* della stessa famiglia è costituito da due coppie (prodotto, *customer*): le ultime due coppie p_i inserite nel *cluster* C_v^u sono utilizzate per inizializzare il *cluster* C_v^{u+1} , a condizione che $|C_v^u| \geq 2$

L'*outline* della procedura è rappresentata nello schema sottostante.

Algorithm 4 CL4

1: **for** $v = 1$ to F **do**
2: $P_v == \cup_{i \in M} P_{iv}$, dove $P_{iv} = \{(p,i) \in P_i \times \{i\} | q_{pi} \leq Q_v\}$
3: Inizializzazione dell'indice dei cluster: $u = 0$
4: **while** $P_v \neq \emptyset$ **do**
5: Definizione di un nuovo cluster: $u = u + 1$
6: Inizializzazione del cluster u con il veicolo v: $C_v^u = \emptyset$
7: **while** $Q_v - \sum_{(p,i) \in C_v^u} q_{pi} \geq \min_{p_i \in P_i} \{q_{pi}\}$ **do**
8: $C_v^u = \{p_i \in P_i | p_i = \underset{p_i \in P_v}{argmax} \{An((p,i))\}\}$
9: $P_v = P_v - \{(p,i) \in C_v^u\}$
10: $I_v^u = \{i \in V | (p,i) \in C_v^u \cap P_{iv}, \forall i = 1, \ldots, n\} \cup \{0\}$
11: Calcola $T_{TSP}(I_v^u)$
12: **while** $T_{TSP}(I_v^u) > T_w$ **do**
13: Seleziona $\bar{i} = arg \underset{j \in TSP_v^u - \{0\}}{max} \{\alpha_1(t_{p[j]j} + t_{js[j]} - t_{p[j]s[j]}) + \alpha_2(t_{0j})\}$
14: Aggiorna l'insieme delle coppie (p,i) non etichettate: $P_v = P_v \cup \{p_i \in C_v^u \cap P_{iv}\}$
15: Aggiorna il *cluster*: $C_v^u = C_v^u - \{p_i \in C_v^u \cap P_{iv}\}$
16: Aggiorna l'insieme degli indici dei clienti: $I_v^u = I_v^u - \{\bar{i}\}$
17: Calcola $T_{TSP}(I_v^u)$
18: **end while**
19: **end while**
20: **if** $|C_v^u| > 2$ **then**
21: Seleziona le ultime due coppie $\overline{(p,i)}$ e $\widetilde{(p,i)}$ inserite in C_v^u
22: Costruisci un nuovo *cluster*: $u = u + 1$
23: $C_v^u = \{\overline{(p,i)}, \widetilde{(p,i)}\}$
24: **else**
25: Costruisci un nuovo *cluster*: $u = u + 1$
26: $C_v^u = \emptyset$
27: **end if**
28: **end while**
29: **end for**

La fase successiva consiste nella determinazione del costo dei *cluster* cosi come descritto nel paragrafo successivo.

Calcolo dei costi dei *cluster* Alcuni *cluster* possono avere dei prodotti in comune, pertanto ad una stessa rotta r_v possono essere associati prodotti appartenenti a *cluster* diversi. Sia R_v^u l'insieme delle rotte relative all'u-esimo *cluster* C_v^u alle quali è assegnato il veicolo v di capacità Q_v, e sia I_v^u l'insieme degli indici dei clienti presenti nel *cluster*; occorre definire il costo minimo da associare a ciascuno dei *cluster* generati in precedenza. I costi di servizio di un *cluster* sono i costi per la consegna delle quantità dei prodotti richiesti dai clienti. Il periodo temporale di pianificazione deve avere una durata sufficiente affinché tutti i prodotti siano consegnati almeno una volta, e deve essere tale da potere stimare i costi di consegna di tutti i prodotti che formano il *cluster*. Pertanto, il costo di servizio di un *cluster* dipende dall'ubicazione geografica rispetto al *vendor* dei *customer* che richiedono i prodotti, dalla capacità di stoccaggio e dai tassi di consumo dei prodotti. I coefficienti e le variabili che servono per definire il modello matematico per la stima dei costi dei *cluster* sono i seguenti:

- c_{rv} indica il costo(durata) della rotta $r_v \in R_v$ per la consegna di un sottoinsieme di prodotti presenti nel cluster C_v^u;
- z_{rv} è la variabile decisionale che indica il numero di volte che la rotta r_v viene eseguita nel periodo di pianificazione T;
- y_{pirv} è la variabile decisionale che rappresenta la quantità totale del prodotto p consegnata al *customer* i con la rotta r_v nel periodo di pianificazione T.

La formulazione del modello matematico è la seguente:

$$\text{Minimize } z_v^u = \sum_{r_v : r_v \in R_v^u} C_{rv} Z_{rv} \tag{2.2a}$$

ed è soggetta ai seguenti vincoli:

$$\sum_{i:i\in I_{rv}} \sum_{p:p\in P_i} y_{pirv} \leq min(Q_v, \sum_{i:i\in I_{rv}} \sum_{p:p\in P_i} C_{pi}) z_{rv}, \forall r_v in R_v^u. \tag{2.2b}$$

$$y_{pirv} \leq min(Q_v, C_{pi}) z_{rv}, \forall i \in I_{rv}, \forall p \in P_i, \forall r_v \in R_v^u. \tag{2.2c}$$

$$\sum_{r_v : i\in I_{rv}} y_{pirv} = T\mu_{pi}, \forall i \in I_v^u, \forall p \in P_i. \tag{2.2d}$$

$$z_{rv} \text{ intere, } y_{pirv} \geq 0 \text{ ed intere.} \tag{2.2e}$$

Per quanto riguarda i vincoli, possiamo osservare che:

- i vincoli (2.2b) assicurano che il volume complessivo consegnato dalla rotta $r_v \in R_v$ non eccede il minimo fra la capacità del veicolo Q_v e la capacità di stoccaggio totale dei *customer* serviti $\sum_{i:i\in I_{rv}} \sum_{p:p\in P_i} C_{pi}$, moltiplicato per il numero di volte che la rotta r_v viene eseguita nel periodo T;

- i vincoli (2.2c) esprimono la condizione tale per cui la quantità totale del prodotto p consegnata al cliente i con la rotta r_v non deve risultare maggiore del minimo fra la capacità del veicolo Q_v e la capacità di stoccaggio di tale prodotto C_{pi}, moltiplicato per il numero di volte che tale rotta è eseguita nel periodo T;

- il vincolo (2.2d) impongono che la quantità totale del prodotto p consegnata al

customer i con tutte le rotte $r_v \in R_v$ deve essere uguale alla quantità di tale prodotto consumata dal *customer* i nel periodo di pianificazione $T : T\mu_{pi}$.

La soluzione di questo modello fornisce il costo z_v^u del cluster C_v^u e le quantità y_{pirv} di ciascun prodotto p da consegnare al *customer* i con le rotte $r_v \in R_v^u$.

Assegnamento dei prodotti ai *cluster* L'ultimo passo di questa complessa procedura consiste nella risoluzione del problema di assegnamento di ogni prodotto ad uno ed un solo *cluster*, per ogni famiglia di *cluster* C_v, $v = 1, \ldots, F$.
Si tratta di risolvere il seguente problema di *set partitioning*:

$$\text{Minimize} \sum_{C_v \in C} \sum_{U : c_v^u \in C_v} z_v^u x_v^u \tag{2.3a}$$

con i seguenti vincoli:

$$\sum_{C_v \in C} \sum_{u : u \in C_v^u \in C_v} = a_{iv}^u x_v^u, \forall i \in M. \tag{2.3b}$$

$$x_v^u \in \{0, 1\}, \forall C_v \in C, \forall u \in C_v. \tag{2.3c}$$

dove:

- $x_v^u \in \{0, 1\}$ è una variabile decisionale che vale 1 se il *cluster* C_v^u è selezionato, o altrimenti;
- $a_{iv}^u \in \{0, 1\}$ è un coifficiente booleano che vale 1 se il prodotto i si trova nel *cluster* C_v ,0 altrimenti;

- z_v^u è il costo del *cluster* C_v^u , determinato con il modello precedente (2.2).

La soluzione del problema (2.3) determina l'allocazione ottimale (di costo minimo) di ciascuna coppia (p, i) in uno ed un solo *cluster*. In conclusione, la risoluzione dei problemi (2.2) e (2.3) è preliminare alla risoluzione del problema di *planning* (2.1). La soluzione ottima del problema (2.2) determina il costo minimo di ciascun cluster definito con uno dei quattro metodi CL1, CL2, CL3 o CL4 descritti in precedenza. Infine, la soluzione ottima del problema (2.3) definisce in quale ed unico *cluster* ogni coppia (prodotto, *customer*) deve essere allocata. La ricerca della soluzione ottima o di una buona soluzione ammissibile per i due predetti problemi avviene con l'ausilio della libreria CPLEX.

2.2.3 *Customer Model planning*

Nel caso del modello di pianificazione definito *Customer Model*, il modello matematico è definito mediante le seguenti variabili decisionali:

$$y^t = \begin{cases} 1 & \text{se il } \textit{customer} \text{ è visitato il giorno } t \in T, \\ 0 & \text{altrimenti.} \end{cases}$$

$$x_{pk}^t = \begin{cases} 1 & \text{se la consegna } k = 1, \ldots, K_p \text{ del prodotto } p \in P_i \text{ presso il } \textit{customer} \text{ avviene il giorno } t \in T, \\ 0 & \text{altrimenti.} \end{cases}$$

$d_p^t =$ quantità di prodotto $p \in P_i$ consegnata nel giorno $t \in T$ al *customer*.

$\delta_{pk}^+ =$ numero di giorni di ritardo per la k-*esima* consegna del prodotto $p \in P_i$ presso il *customer*.

$\delta_{pk}^- =$ numero di giorni di anticipo per k-*esima* consegna del prodotto $p \in P_i$ presso il *customer*.

In questo caso, l'indice i associato al *customer* è stato omesso in quanto tale modello deve essere risolto per ogni *customer* $i \in M$.

La funzione obiettivo del modello matematico assume la seguente forma:

$$\text{Minimize} \sum_{t \in T} c^t y^t + \sum_{p \in P_i} \sum_{k=1}^{K_p} \overline{\pi}^e_{pk} \delta^-_{pk} + \sum_{p \in P_i} \sum_{k=1}^{K_p} \overline{\pi}^l_{pk} \delta^+_{pk} \tag{2.4a}$$

con i seguenti vincoli:

$$\sum_{t \in T} x_{pk}^t = 1, \ \forall\, p \in P_i, \forall\, k = 1, \ldots, K_p \tag{2.4b}$$

$$\sum_{p \in P_i} \sum_{k=1}^{K_p} x_{pk}^t \leq |P_i| y^t, \ \forall\, t \in T \tag{2.4c}$$

$$\delta_{pk}^+ - \delta_{pk}^- = \sum_{t \in T} t x_{pk}^t - t_{pk}, \ \forall\, p \in P_i, \forall\, k = 1, \ldots, K_p \tag{2.4d}$$

$$\delta_{pk}^- \leq LD, \ \forall\, p \in P_i, \forall\, k = 1, \ldots, K_{pi} \tag{2.4e}$$

$$\delta_{pk}^+ \leq LD, \ \forall\, p \in P_i, \forall\, k = 1, \ldots, K_{pi} \tag{2.4f}$$

$$\sum_{p \in P_i} \sum_{k=1}^{K_p} \bar{\pi}_{pk}^e \delta_{pk}^- \leq \Pi, \tag{2.4g}$$

$$\sum_{p \in P_i} \sum_{k=1}^{K_p} \bar{\pi}_{pk}^l \delta_{pk}^+ \leq \Pi, \tag{2.4h}$$

$$LL_p^t \leq \sum_{s=1}^{t} d_p^s \leq UL_p^t \ \forall\, p \in P_i, \forall\, t \in T \tag{2.4i}$$

$$(q_p - LD\mu_p) \sum_{k=1}^{K_p} x_{pk}^t \leq d_p^t \leq (q_p + LD\mu_p) \sum_{k=1}^{K_p} x_{pk}^t, \ \forall\, p \in P_i, \forall\, t \in T, \tag{2.4j}$$

$$x_{pk}^t \in \{0, 1\}, \ \forall\, p \in P_i, \forall\, k = 1, \ldots, K_p, \forall\, t \in T \tag{2.4k}$$

$$y^t \in \{0, 1\}, \ \forall\, t \in T \tag{2.4l}$$

$$\delta_{pk}^+, \ \delta_{pk}^- \in \mathcal{Z}_+ \ \forall\, p \in P_i, \forall\, k = 1, \ldots, K_p \tag{2.4m}$$

$$d_p^t \in \Re_+ \ \forall\, p \in P_i, \forall\, t \in T \tag{2.4n}$$

La funzione obiettivo (2.4a) si prefigge di minimizzare:

1. il numero di volte in cui un determinato *customer* è visitato all'interno del periodo

di pianificazione T;

2. il costo totale di penalità per le consegne effettuate in anticipo o in ritardo rispetto al tempo di riordino indicato dalla politica di gestione delle scorte.

L'utilizzo dei fattori moltiplicativi all'interno della funzione obiettivo ha consentito di rendere omogenee le diverse componenti in termini di costi. In particolare, la funzione (2.4a) è stata resa omogenea introducendo i seguenti fattori moltiplicativi (costi) c^t, $\bar{\pi}^e_{pk}$ e $\bar{\pi}^l_{pk}$ dove:

- $c^t = h_p q_p + 2c_{0i}$ rappresenta il costo per servire un determinato *customer* ed è calcolato sommando due diverse componenti: il costo di instradamento (*routing cost*) $2c_{0i}$ ed il costo di inventario $h_p q_p$;
- i coifficienti $\bar{\pi}^e_{pk}$ e $\bar{\pi}^l_{pk}$ sono i costi di penalità per unità di tempo nel caso di consegne effettuate in anticipo o in ritardi rispetto alla data indicata dalla politica EOQ: tali valori sono determinati cosi come descritto in ([17]).

Per quanto riguardi i vincoli del modello matematico:

- i vincoli (2.4b) assicurano che la consegna k-*esima* del prodotto $p \in P_i$ avviene in un giorno del periodo di pianificazione;
- i vincoli (2.4c) assicurano che il numero di consegne pianificate nel giorno t non superi il numero di prodotti che possono essere consegnati al *customer* $i \in M$;
- i vincoli (2.4d) esprimono la distanza, in giorni, della k-*esima* consegna del prodotto p ripetto al tempo di riordino t_{pik};

- i vincoli (2.4e) e (2.4f) sono stati introdotti per limitare il numero di giorni per cui la *k-esima* consegna può essere ritardata o anticipata. Il parametro LD rappresenta il massimo numero di giorni per cui la consegna può essere anticipata o ritardata (*upper bound*);

- i vincoli (2.4g) e (2.4h) consentono di limitare il valore del costo di penalità all'interno di una determinata soglia Π;

- i vincoli (2.4i) esprimono l'assenza di *stock-out*;

- i vincoli (2.4j) impongono che la quantità di prodotto p consegnata al *customer* i deve essere compresa tra la quantità di prodotto consegnata LD giorni prima e LD giorni dopo la consegna;

- i vincoli (2.4k)–(2.4n) impongono le condizioni di non negatività delle variabili decisionali del modello matematico.

La ricerca della soluzione ottima del modello matematico descritto in (2.4) presenta una complessità computazionale NP-Hard a causa dell'elevato numero di prodotti che devono essere consegnati nell'intero periodo di pianificazione. La maggiore complessità del modello (2.4) risiede nella cardinalità dell'insieme P_i e nel valore di K_p in quanto ciò implica, per un determinato t, un incremento non lineare del numero di variabili decisionali x_{pk}^t. Inoltre, nel caso in cui il periodo di pianificazione T è molto ampio, il numero di variabili che dipendono dal tempo aumentano drasticamente rendendo il processo di ricerca della soluzione ottima ancora più complesso. Obiettivo del *Customer Model* è quello di determinare le quantità d_p^t di $p \in P_i$ consegnate ad ogni *customer*

all'interno del periodo di pianificazione T. Successivamente, sarà effettuata la schedulazione delle consegne giornaliere in modo tale che ogni *customer* sia servito nel rispetto di determinate finestre temporali.

2.3 La fase di *vehicle routing*

Il problema dell'instradamento dei veicoli sulla rete di distribuzione attiene in primo luogo alle scelte di breve periodo, poiché tramite esso sono stabilite le rotte ed i carichi per ogni singolo vettore. Tuttavia, poiché tali decisioni sono collegate alla tipologia di infrastrutture presenti e dalla flotta di veicoli a disposizione dell'organizzazione, il *Vehicle Routing Problem* apparre intimamente connesso anche ai piani strategico e tattico delle aziende. Sulla base degli elementi distintivi della rete distributiva esistono diverse tipologie del problema. Le principali caratteristiche che determinano la tipologia di problema sono:

1. numero di attività; una distribuzione può ritenersi *single-stage* se prevede la sola consegna delle merci, altrimenti implica ulteriori processi per il recupero prodotti, imballi, resi, ecc., dando vita alla così detta *reverse logistics*;

2. produzione, domanda deterministica e/o stocastica: gli attori a monte e a valle della rete logistica possono operare con comportamento mediamente noto oppure con aleatorietà tali da dover ricorrere a studi statistici;

3. dimensione della flotta; è possibile differenziare tra le distribuzioni attuate con un solo vettore e quelle, riguardanti la maggior parte di realtà industriali, che

ricorrono ad una flotta di mezzi di trasporto, al crescere della quale aumenta la difficoltà del problema logistico;

4. capacità dei veicoli; considerando di non poter utilizzare vettori a capacità infinita è necessario correlare le caratteristiche della rete con i vincoli dati dai mezzi di trasporto scelti e dalle loro capacità di carico, in termini di peso o di volume;

5. orizzonte di pianificazione; la valutazione dei collegamenti da istaurare per la consegna delle merci può afferire ad un orizzonte temporale limitato o più ampio, relativo a domande e livelli di produzione costanti o dipendenti dal tempo. In un caso o nell'altro si presentano problematiche e modelli differenti;

6. orizzonte di consegna; a seconda della tipologia di merci distribuite o del mercato in cui si trovano ad operare, le organizzazioni si possono trovare a fronteggiare la necessità di recapitare entro un certo intervallo temporale dalla data di emissione dell'ordine corrispettivo;

7. obiettivi di minimizzazione dei costi totali della rete di distribuzione;

2.3.1 L'algoritmo di *vehicle routing* per il VMI

La soluzione del modello di pianificazione non specifica la struttura delle rotte da attivare. Infatti, le informazioni relative ai tempi di partenza dei veicoli ed alle sequenze di consegna dei prodotti presso i *customer* inclusi nelle rotte non sono determinati alla fine della fase di pianificazione. Restano, pertanto, da determinare gli itinerari ed i programmi di consegna delle rotte. A tale scopo, si osserva che le consegne dei prodotti presso i *customer*, nelle quantità stabilite dal modello di pianificazione, devono essere

effettuate nel rispetto dei vincoli rappresentati dalle finestre temporali associate ai vari *customer*. La realizzazione del piano delle consegne, associate ai primi giorni del periodo di pianificazione T, avviene risolvendo una serie di problemi di *Vehicle Routing* capacitivo con *Time Windows* (VRPTW). Il VRPTW è una estensione del *Capacitated Vehicle Routing Problem* (CVRP) in cui ad ogni *customer* i è associato un intervallo di tempo $[a_i, b_i]$ detto, appunto, *time window*. I dati del problema sono rappresentati dai seguenti valori:

- tempo di inizio $t_i \in [a_i, b_i]$ del servizio presso *customer*;
- tempo di servizio di ciascun *customer*, espresso dall'intervallo di tempo durante il quale il veicolo, che effettua il servizio, staziona presso il cliente;
- matrice dei tempi di viaggio, la cui generica *entry* t_{ij} rappresenta il tempo di percorrenza dell'arco (i, j);
- t_0 che rappresenta l'istante di partenza dei veicoli dal deposito.

Il VRPTW è, di norma, un problema asimmetrico, in quanto i valori dei *time window* inducono implicitamente un orientamento delle rotte.
Riassumendo la soluzione di un'istanza di VRPTW consiste nella determinazione delle rotte tali che:

1. ogni rotta parte e termina nel deposito;
2. ogni cliente è visitato da una sola rotta;
3. la somma delle domande dei clienti visitati da una rotta non supera la capacità Q del veicolo che li serve;

4. per ogni cliente i, il servizio ha inizio in un istante compreso nell'intervallo temporale $[a_i, b_i]$ e il veicolo rimane occupato per un tempo pari a s_i.

I punti (1) e (3) sono già soddisfatti nella fase di costruzione dei *cluster* durante la risoluzione del modello di pianificazione. Infatti, le rotte di consegna sono insiemi di coppie (p, i), tali che ogni coppia appartiene ad uno ed un solo *cluster* e la somma delle quantità dei prodotti consegnate sono minori delle capacità di carico del veicolo. I dati di *input* del modello di VRPTW sono ricavati dai dati di *output* del modello di pianificazione. In particolare, sono presi in considerazione i seguenti elementi:

- lista dei clienti $i \in I_{rv}$ presenti in ciascuna rotta di consegna $r_v \in R$ attivata nei primi t giorni del periodo di pianificazione;
- massima lunghezza temporale delle rotte $r_v \in R$ in $[0, t] \subseteq T$;
- domanda d^t_{pirv} di ogni *customer* $i \in I_{rv}$ da inserire nelle rotte $r_v \in R$ in $[0, t] \subseteq T$;
- capacità Q_v di ogni categoria di veicoli;
- finestra temporale $[a_i, b_i]$ relativa ad ogni *customer* $i \in I_{rv}$. L'*output* del modulo di VRPTW è l'ordinamento temporale delle consegne delle rotte attivate nel periodo $[0, t] \subseteq T$.

L'algoritmo implementato per il VRPTW è basato su un inserimento euristico dei clienti: l'algoritmo, ad ogni iterazione, decide quale nuovo cliente $i \in I_{rv}$ deve essere inserito nella rotta corrente $r_v \in R$ e tra quali *customer* adiacenti $pred(i)$ e $succ(i)$ esso deve essere inserito.

Nella scelta di i l'algoritmo cerca di massimizzare il beneficio ottenuto servendo un *customer* nella rotta corrente anziché in una rotta individuale. Il migliore inserimento fattibile per allocare un *customer* è determinato dalla minimizzazione di una misura della distanza e del tempo aggiuntivi richiesti per visitarlo. Le quantità da consegnare a ciascun *customer* $i \in I$ vengono salvate in una lista, ordinandole per *deadline*. Successivamente, si inizia a scorrere la lista contenente le richieste da allocare e, compatibilmente con i vincoli di finestre temporali associate ai clienti, si cerca di inizializzare la rotta (ponendo $r_1 = \{0, i, 0\}$ dove i è il *customer* con *deadline* più bassa) descritta dalla categoria di veicoli con capacità Q_1 . Se non si riesce ad inizializzare la rotta per il veicolo corrente, si passa ad un altro veicolo. Altrimenti, una volta che la rotta è stata inizializzata, si procede all'inserimento del successivo cliente. Per determinare quale *customer* inserire, per ognuno di quelli non allocati, si verifica che la somma delle domande dei *customer* presenti nella rotta corrente (incluso il nuovo *customer* da inserire) sia minore o uguale della capacità del veicolo corrente. Se il vincolo di capacità del veicolo è soddisfatto, per ogni coppia di *customer* appartenenti alla rotta corrente si verifica che il nuovo inserimento sia ammissibile; ovvero, una volta inserito il nuovo *customer*, si verifica che la traslazione delle finestre temporali che ne consegue, sommata alla lunghezza corrente della rotta, risulti minore o uguale alla massima lunghezza prevista per la stessa. Se anche il vincolo di massima lunghezza temporale della rotta è soddisfatto, si calcola il valore della funzione f_1, definita come:

$$f_1(j, i, k) = \alpha (t_{ji} + t_{ik} - \mu t_{jk}) + (1 - \alpha) \left(t_k^{'} - t_k \right)$$

in cui valgono le relazioni:

- $0 \leq \alpha \leq 1$, $\mu \geq 0$;
- t_{ji} è il tempo di viaggio del cammino minimo da j ad i;
- t_{ik} è il tempo di viaggio del cammino minimo da i a k;
- t_{jk} è il tempo di viaggio dello spigolo (j, k);
- t_l^i è il tempo di inizio in cui viene servito il *customer* k supposto che il *customer* i sia inserito tra j e k.

Una volta esaurite le coppie di possibili inserimenti, tra tutti i valori di f_1 calcolati si determina il minimo, che è utilizzato per calcolare il valore della funzione f_2 associata al singolo *customer* da allocare. Terminato lo scorrimento della lista dei clienti da allocare, si determina il massimo tra i valori della funzione $f2$, definita come:

$$f_2\left(pred(i), i, succ(i)\right) = \lambda t_{0i} - f_1(pred(i), i, succ(i)) \text{ con } \lambda \geq 0.$$

Se tale massimo esiste, allora è stato determinato il migliore *customer* da inserire. In questo ultimo caso, il migliore *customer* viene inserito tra i vertici della coppia per la quale corrisponde il massimo valore di f_2, ovvero il minimo valore di f_1. Una volta effettuato l'inserimento si ripete la procedura di inserimento (con le verifiche connesse) sullo stesso segmento di rotta. Nel caso in cui, invece, il migliore *customer* da inserire non sia stato trovato, si passa ad un nuovo segmento di rotta. Il tutto si ripete , per ogni giorno dell'intervallo temporale $[0, t] \subseteq T$, sino a quando la lista dei veicoli non è vuota oppure risulta vuota la lista dei *customer* da servire.

Una *outline* della procedura euristica implementata è di seguito riportata.

Algorithm 5 Routing

Require: L'insieme M^t è l'insieme dei *customer* serviti nel giorno $t \in T$, l'insieme $P^t = \bigcup_{i \in M^t} P_i^t$ è l'insieme dei prodotti consegnati nel giorni t ad ogni *customer* i, l'insieme S^t è l'insieme delle quantità d_{ip} consegnate, m è la flotta di veicoli di capacità omogenea Q

Ensure: l'insieme R^t è l'insiee delle rotte da attivare nel giorno $t \in T$

1: **for all** $t \in T$ **do**
2: **while** $M^t \neq \emptyset$ **do**
3: $\rho^* = -\infty$
4: **for all** $i \in M^t$ **do**
5: **for all** $p_i \in P_i^t$ **do**
6: **for** $r_h \in R^t$ **do**
7: **for all** $\{(j-1, d_{j-1p_{j-1}}), (j, d_{jp_j})\} \subseteq r_h$ **do**
8: **if** $Feasible((j, d_{jp_j}), (i, d_{ip_i}))$ and $Profit((j, d_{jp_j}), (i, d_{ip_i})) > \rho^*$ **then**
9: $r^* = r_h$
10: $(j^*, d_{j^*p_{j^*}}) = (j, d_{jp_j})$
11: $(i^*, d_{i^*p_{i^*}}) = (i, d_{ip_i})$
12: $\rho^* = Profit((j, d_{jp_j}), (i, d_{ip_i}))$
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **end for**
18: $Insert((j^*, d_{j^*p_{j^*}}), (i^*, d_{i^*p_{i^*}}))$
19: $P_i^t = P_i^t \backslash \{p_{i^*}\}$
20: **if** $P_i^t = \emptyset$ **then**
21: $M^t = M^t \backslash \{i^*\}$
22: **end if**
23: $Update(r^*)$
24: **end while**
25: **end for**

E' possibile osservare che la funzione $Feasible((customer, product), (customer, product))$ verifica la possibilità di inserire la coppia $(customer, product)$ nel rispetto dei vincoli di capacità e delle finestre temporali. Inoltre, La funzione $Profit((customer, product),$

$(customer, product)$) valuta il profitto derivante dall'inserimento della coppia $(customer, product)$. Sulla base dei valori di profitto determinati è individuata la coppia $(customer, product)$ inserita nella rotta attraverso la funzione $Insert((customer, product), (customer, product))$. Compito della funzione *update* è la determinazione sia della capacità residua da trasportare sia del costo totale della rotta.

2.4 Un *bound* per il MIRP

In questa sezione è presentata una procedura per determinare un *bound* per il problema del MIRP. Assumiamo che la capacità di inventario per ogni prodotto presso ogni *customer* sia illimitata così come il numero di veicoli della flotta utilizzata per la distribuzione. L'obiettivo della procedura è di ottenere una stima accurata della minima distanza percorsa dalla flotta per soddisfare gli ordini (domanda) emessi dai *customer*. Per minimizzare la distanza totale percorsa sull'interno periodo di pianificazione la politica per la gestione delle consegne deve essere tale da effettuare le consegne solo nel corso del primo giorno del periodo di pianificazione T. Pertanto, è possibile effettuare la stima della quantità minima di prodotto che deve essere consegnata presso ogni *customer* durante il perido di pianificazione, e quindi consegnare tale quantità nel primo giorno del periodo di pianificazione minimizzando la distanza complessiva percorsa dalla flotta di veicoli nel periodo di pianificazione. Si evidenzia che la quantità totale di prodotto è la quantità utile a soddisfare la domanda del prodotto presso i *customer* nel periodo di pianificazione T.

Il seguente modelo matematico consente di determinare la quantità minima di ogni

prodotto da consegnare ad ogni *customer* durante l'interno periodo di pianificazione T in modo tale da garantire assenza di situazioni di *stock-out* e nel rispetto del numero di consegne programmate sulla base della politica di gestione delle scorte.

MIRP Bound

$$\text{Minimize} \sum_{t \in T} \sum_{p \in P_i} d_p^t \tag{2.5a}$$

con i seguenti vincoli

$$LL_p^t \leq \sum_{s=1}^{t} d_p^s \leq UL_p^t \qquad \forall\, p \in P_i, \forall\, t \in T \tag{2.5b}$$

$$\sum_{t \in T} z_p^t = K_p \qquad \forall\, p \in P_i \tag{2.5c}$$

$$d_p^t \leq U_p z_p^t \qquad \forall\, p \in P_i, \forall\, t \in T \tag{2.5d}$$

$$z_p^t \in \mathcal{Z}_+ \qquad \forall\, p \in P_i, \forall\, t \in T \tag{2.5e}$$

$$d_p^t \in \Re_+ \qquad \forall\, p \in P_i, \forall\, t \in T \tag{2.5f}$$

- i vincoli (2.5b) sono i classici vincoli relativi alla gestione delle scorte;
- i vincoli (2.5c) sono i vincoli (2.4b) del modello (2.4) espressi in forma differente sulla base delle seguente operazioni: somma dei vincoli sull'indice k per ogni $p \in P_i$ e aggregazione delle variabili x_{pk}^t mediante l'introduzione delle variabili surrogate $z_p^t = \sum_{k=1}^{K_p} x_{pk}^t$;
- i vincoli (2.5d) rappresentano un rilassamento del vincolo (2.4j) ottenuto rimuovento la parte sinistra del vincolo originale, impostando $LD = \frac{U_p - q_p}{\mu_p}$ ed utilizzando le variabili surrogate z_p^t;
- i vincoli (2.5d) fungono da vincoli di priorità nel senso che il prodotto p non può essere consegnato il giorno t se in esso non sono schedulate consegne;
- le variabili (2.5e) rappresentano il numero di consegne del prodotto p nel giorno t;
- le variabili (2.5f) rappresentano la quantità di prodotto p consegnata nel giorno t; la quantità minima complessiva di ogni prodotto che deve essere consegnata ad ogni *customer* può essere definita da $\underline{d}_{ip} = \sum_{t \in T} \underline{d}_p^t$, per ogni $p \in P_i$, dove $\underline{d}_p^t$ sono i valori restituiti dal modello (2.5);
- un *lower bound* $\underline{D}$ sulla distanza minima percorsa, necessaria a soddisfare la domanda di prodotto di tutti i *customer*, è ottenuto applicando l'algoritmo di *routing* rilassando i vincoli sulle finestre temporali dei *customer* e consegnando, per ogni *customer* $i \in M$, la quantità $\sum_{p \in P_i} \underline{d}_{ip}$. Il risultato è un *upper bound* del

rapporto volume consegnato per chilometro percorsi:

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

Capitolo 3

Algoritmi Euristici per il VMI

3.1 Introduzione

Spesso la determinazione della soluzione ottima di un problema NP-difficile può risultare troppo onerosa in termini di tempo di calcolo. Ad esempio, il problema del commesso viaggiatore su un grafo di n nodi presenta un numero di *tour* ammissibili dell'ordine di $O(n!)$. La risoluzione esatta del problema implica tempi di calcolo così elevati da rendere necessario l'individuazione di metodi risolutivi alternativi in grado di produrre soluzioni ammissibili di buona qualità in tempi di calcolo ragionevoli. Si pone in evidenza che, tipicamente, la determinazione di buone soluzioni ammissibili è sufficiente nelle applicazioni reali (soprattutto se di grandi dimensioni). Questo è essenzialmente dovuto ad una serie di fattori:

- molti dei parametri in gioco nelle applicazioni reali sono delle stime che possono essere soggette ad errore, per cui è necessaria una analisi di sensibilità del modello

la cui implementazione inciderebbe notevolmente sui tempi di calcolo per la ricerca di una soluzione ottima robusta;

- spesso è utile disporre di una buona soluzione ammissibile, per il problema in esame, al fine di valutare velocemente degli scenari di lavoro (fase operativa);
- spesso si lavora in tempo reale, per cui si vuole avere una *buona* soluzione ammissibile in tempi molto ridotti (minuti o secondi di tempo di calcolo);
- a volte le applicazioni reali presentano molti vincoli di natura difficile, cioè difficilmente rappresentabili con modelli di programmazione lineare intera la cui risoluzione, attraverso algoritmi esatti, richiede tempi di calcolo eccessivi.

In generale, è possibile definire un algoritmo euristico come un qualunque metodo che fornisce una soluzione per un determinato problema. Generalmente, si richiede che un algoritmo euristico determini una soluzione ammissibile del problema in un tempo di calcolo polinomiale rispetto alla dimensione dell'istanza da risolvere. Un buon algoritmo euristico deve essere in grado di determinare una buona soluzione ammissibile, ovvero una soluzione il cui valore è abbastanza vicino al valore della soluzione ottima, in tempi di calcolo ridotti. Nel mondo reale, esistono dei problemi per i quali la determinazione della soluzione ottima è computazionalmente NP-difficile. In questi casi, anche la ricerca di una soluzione ammissibile attraverso una procedura euristica potrebbe richiedere elevati tempi di calcolo. Considerando una istanza I di un problema di minimo, un algoritmo euristico A fornisce una soluzione di valore $z^A(I)$ tale che

$$z^A(I) \geq z^*(I)$$

dove $z^*(I)$ è il valore ottimo dell'istanza. Si dice che l'algoritmo fornisce un *upper bound* sul valore della soluzione ottima. Ovviamente, nel caso di problema di massimo la relazione sarà del tipo $z^A(I) \leq z^*(I)$, ossia l'algoritmo A fornisce un *lower bound* sul valore della soluzione ottima. Un algoritmo euristico può essere anche in grado di determinare una soluzione ottima di un problema, ma non può mai essere in grado di dimostrare l'ottimalità di tale soluzione (a meno che non sia combinato con la soluzione di qualche rilassamento del problema). Gli algoritmi euristici possono essere classificati nel seguente modo:

- Algoritmi Costruttivi
  - partono da una soluzione vuota;
  - determinano in modo iterativo i nuovi elementi da aggiungere in soluzione fino ad arrivare ad una soluzione completa.
- *Greedy*, basati su tecniche di ottimizzazione ad enumerazione implicita.
- Algoritmi di Ricerca Locale
  - partono da una soluzione iniziale (generalmente ammissibile);
  - cercano iterativamente di migliorarla, effettuando delle (semplici) modifiche alla soluzione corrente;
  - terminano quando non è più possibile migliorare la soluzione corrente con le modifiche del tipo prescelto (ottimo locale).
- Algoritmi Meta-euristici

– sono delle evoluzioni degli algoritmi di ricerca locale;

– utilizzano speciali tecniche per evitare di fermarsi in un ottimo locale;

– devono evitare il verificarsi di *cicli* nell'evoluzione dell'algoritmo.

Il tempo di calcolo, e la qualità delle soluzioni prodotte, cresce man mano che si passa dagli algoritmi costruttivi agli algoritmi di ricerca locale e ai meta-euristici. A seconda dell'applicazione che si vuole risolvere e della qualità delle soluzioni che si vogliono ottenere, converrà scegliere un tipo di algoritmo piuttosto che un altro. Un importante aspetto delle procedure euristiche è che il tempo di calcolo richiesto è sensibilmente minore di quello che sarebbe stato richiesto da un algoritmo esatto per risolvere lo stesso problema.

Le procedure euristiche definite in questo capitolo fanno riferimento al modello matematico del VMI denominato *Customer Model*, così come definito in (2.4).

3.2 Procedure di base

In questa sezione sono riportare delle funzioni di base che verranno utilizzate dalle procedure euristiche definite nelle sezioni successive. Le procedure richiamate dalle procedure euristiche sono:

- $d_{ip} \leftarrow Get\_Order(i, p, \Pi_p)$. E' la procedura utilizzata per calcolare la quantità da riordinare d_{ip} in grado di soddisfare il vincolo (2.4i); tale valore è calcolato sulla base della politica di gestione delle scorte Π_p;

- l'insieme $S_i^t = \{d_{ip} : t_{ipk} = t, \forall\, p \in P_i, \forall\, k = 1, \ldots, K_{ip}\}$ è l'insieme delle quantità d_{ip} che devono essere consegnate dal *vendor* al *customer* i nel giorno 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 $ND_i = \{ND_i^0, \ldots, ND_i^H\}$ è l'insieme contenente tutte le consegne al *customer* i nell'orizzonte di pianificazione. E' possibile notare che ND_i^t è il numero di consegne al *customer* i il giorno $t \in T$;

- $ND_i \leftarrow Get\_OrderNumber(i, P_i)$. E' la procedura utilizzata per calcolare il numero di consegne presso il *customer* i sull'intero periodo di pianificazione;

- $\{0, 1\} \leftarrow Check\_Earlier(i, p, k, t_{ipk})$. E' una procedura che ritorna 1 se la k-esima consegna del prodotto p al *customer* i può essere schedulata in anticipo (il giorno $t - 1$) rispetto al giorno programmato t, 0 altrimenti. Il controllo consente di assicurare che, nel giorno $t_{ipk} - 1$, la massima quantità di inventario $UL_{ip}^{t_{ipk}-1} = U_{ip} + ((t_{ipk} - 1) - t_{ipk-1})\mu_{ip} - I_{ip}^{t_{ipk-1}}$, dove $I_{ip}^{t_{ipk-1}}$ è la quantità di prodotto p nel giorno t_{ipk-1}, soddisfa la seguente relazione:

$$I_{ip}^{t_{ipk}-1} + q_{ip} \leq UL_{ip}^{t_{ipk}-1};$$

- $\{0, 1\} \leftarrow Check\_Later(i, p, k, t_{ipk})$. E' una procedura che ritorna 1 se la k-esima consegna del prodotto $p \in P_i$ al *customer* $i \in M$ può essere schedulata in ritardo (nel giorno $t + 1$) rispetto al giorno programmato t, 0 altrimenti. Inoltre, con riferimento alla prima consegna ($k = 1$), la procedura controlla che la differenza tra il livello di inventario nel giorno t_{ip1} e la quantità di prodotto consegnata nello

stesso giorno è maggiore o uguale della quantità di prodotto consumata il giorno successivo, ovvero che:

$$I_{ip}^{t_{ip1}} - q_{ip} \geq ((t_{ip1} + 1) - t_{ip1})\mu_{ip}$$

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

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

è minore o uguale al livello massimo di inventario nel giorno t_{ipk}, ovvero:

$$q_{ip} + I_{ip}^{t_{ipk-1}} + ((t_{ipk} + 1) - t_{ipk})\mu_{ip} \leq UL_{ip}^{t_{ipk-1}}$$

E' possibile osservare che la quantità $q_{ip} + I_{ip}^{t_{ipk-1}} + ((t_{ipk} + 1) - t_{ipk})\mu_{ip}$, che rappresenta il livello di inventario nel giorno t_{ipk-1}, è sufficiente a soddisfare una maggiore domanda di prodotto, nel giorno $t_{ipk}+1$, pari a $q_{ip}+((t_{ipk}+1)-t_{ipk})\mu_{ip}$

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

$$S_i = \{\ldots, \bar{S}_i^{t_{ipk}-1}, \ldots, \forall\, t \in T\},$$

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

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

$$S_i = \{\ldots, \bar{S}_i^{t_{ipk}+1}, \ldots, \forall\, t \in T\},$$

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

3.3 Algoritmo di *Local Search*

Consideriamo un problema di minimizzazione, e una sua soluzione ammissibile x, con associato il valore della funzione obiettivo $f(x)$. La ricerca locale consiste nel definire un intorno di x (detto, nella terminologia della ricerca locale, vicinato), e nell'esplorarlo in cerca di soluzioni migliori, se ve ne sono. Se, in questo vicinato di x, si determina una soluzione y per cui $f(y) < f(x)$, allora ci si sposta da x a y e si riparte da y con l'esplorazione del suo intorno. Se, invece, nel vicinato di x non si determina alcuna soluzione migliore, allora vuol dire che x è un minimo locale. Nella ricerca locale classica, arrivati in un minimo locale, l'algoritmo si ferma e restituisce tale minimo come valore di *output*. Ovviamente, non si ha nessuna garanzia che tale valore costituisca una soluzione ottima del problema; anzi, tipicamente, esso può essere molto distante dall'ottimo globale. Le metaeuristiche, in effetti, sono state concepite proprio per cercare di ovviare al problema di rimanere intrappolati in un minimo locale. In base a quanto detto sopra, possiamo riassumere schematicamente l'algoritmo generale di ricerca locale come segue, indicando con x una generica soluzione ammissibile del problema, $N(x)$ il

suo vicinato e $f(x)$ la funzione obiettivo.

Algoritmo di ricerca locale

1. Scegli una soluzione iniziale x;
2. Genera le soluzioni nel vicinato $N(x)$;
3. Se in $N(x)$ esite una soluzione y tale che $f(y) < f(x)$, allora poni $x := y$ e vai al passo 2, altrimenti STOP.

Lo spostamento da x a y al passo 3 viene spesso chiamato mossa. Nella ricerca locale, come si vede, si ha un miglioramento della funzione obiettivo in corrispondenza di una mossa. Per applicare l'approccio della ricerca locale a un particolare problema, bisogna fare alcune scelte di fondo. In particolare, è necessario avere a disposizione una soluzione iniziale ammissibile. Questa può essere generata da un'euristica ad *hoc* per il particolare problema, o in alternativa può essere generata casualmente. E' possibile anche eseguire l'algoritmo a partire da diverse soluzioni iniziali, ottenendo così diverse soluzioni euristiche, e scegliere poi la migliore. Inoltre, bisogna definire in modo preciso e opportuno il vicinato di una soluzione ed avere a disposizione un algoritmo efficiente in grado di assicurare la sua completa esplorazione. Infatti, risulterebbe poco utile definire un vicinato che sia teoricamente ampio, ma non essere in grado successivamente di esplorarlo in tempi ragionevoli. L'efficacia e l'efficienza degli algoritmi di ricerca locale dipendono da tali scelte.

Nel caso del modello di pianificazione definito per il VMI, il metodo di ricerca locale, defininito HC (*Hill Climbing*)-MIRP, implementa una procedura di ricerca locale di tipo *random*. I giorni t di T sono definiti *moveable* e inseriti in T_m se il numero delle

consegne, in essi schedulate, è minore o uguale al valore $\gamma \cdot |P_i|$, dove γ è un valore costante di soglia. Inizialmente, l'algoritmo HC-MIRP controlla se tutte le consegne che occorrono in un giorno t, *moveable*, possono essere anticipare o posticipate rispetto a t, ed etichetta come *tabu* i giorni che non soddisfano tale condizione rimuovendoli dall'insieme T_m. Successivamente, effettua una selezione casuale di un giorno *moveable* t a partire dall'insiemee T_m ed effettua una operazione di *shift* per anticipare o posticipare tutte le consegne che prendono luogo in quel particolare giorno.

I passi procedurali della tecnica utilizzata sono di seguito riportati:

1. se $S = \emptyset$, allora gli insiemi S_i^t, S^t, S_i e S sono inizializzati cosi come descritto nella sezione 3.2.

2. per ogni *customer* $i \in M$

   (a) calcolare il numero di consegne per ogni $t \in T$;

   (b) definire l'insieme $T_m = \{t \in T : ND_i^t \leq \gamma \cdot |P_i|,\ \gamma > 0\}$, dove γ è un valore costante di soglia;

   (c) se $T_m = \emptyset$ allora arresta la procedura, altrimenti controlla se tutte le consegne che occorrono ad ogni $t \in T_m$ possono essere anticipate o posticipate. Se nessuna delle consegne può essere spostata, allora etichettare t come *tabu* e eliminarla da T_m;

   (d) selezionare in maniera *random* il giorno t dall'insieme T_m ed effettuare lo *shift* di tutte le consegne, quindi ritornare al passo (b)

3. applicare l'algoritmo di *routing* definito nella sezione 2.3 per costruire le rotte di consegna per ogni $t \in T$.

Per semplicità di esposizione si definiscono le seguenti funzioni, utilizzate di seguito nell'*outline* della procedura euristica:

Algorithm 6 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**

Algorithm 7 $CHECKSHIFT$

1: Select $t_p = argmax_{t \in \{t^*-1, t^*+1\}} \{I_{ip}^{t^*-1} Check\_Earlier(i, p, k, t^*), I_{ip}^{t^*+1} Check\_Later(i, p, k, t^*)\}$
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**

Algorithm 8 ROUTING

1: **for all** $t \in T$ **do**
2: **if** $S^t \neq \emptyset$ **then**
3: Inizializza M^t e 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**

L'*outline* dell'algoritmo è di seguito riportato:

Algorithm 9 HC-MIRP

Require: L'insieme M, S, il grafo G, la flotta di veicoli m con capacità Q, il periodo di pianificazione T

Ensure: L'insieme R delle rotte di consegna sull'intero periodo di pianificazine T, l'insieme S aggiornato. Inizialmente pone l'insieme $R = \emptyset$

1: *Richiamare la procedura* $INITIALIZE()$
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** $p \in P_i$, tale che $t_{ipk} = t_m$ **do**
13: **if** $Check\_Earlier(i, p, k, t_{ipk}) = Check\_Later(i, p, k, t_{ipk}) =$ FALSE **then**
14: MOVE = FALSE
15: $B_i = B_i \cup \{t_m\}$ e $T_m = T_m \backslash \{t_m\}$
16: break
17: **end if**
18: **end for**
19: **end for**
20: **if** $T_m \neq \emptyset$ **then**
21: Sia t_m un giorno, definito *moveable*, ottenuto da una selezione casuale dall'insieme T_m
22: **for all** $p \in P_i$, tale che $t_{ipk} = t$ **do**
23: *Richiamare la procedura* $CHECKSHIFT()$
24: **end for**
25: **end if**
26: **end while**
27: **end for**
28: *Richiamare la procedura* $ROUTING()$

E' possibile osservare che l'algoritmo di ricerca locale proposto effettuta la selezione del giorno t_m, definito *moveable*, in maniera casuale dall'insieme T_m. Il raggiungimento di più alti livelli di *performance*, sia in termini di efficienza computazionale sia rispetto al valore del rapporto tra volume di prodotto consegnato e chilometri percorsi, hanno suggerito di utilizzare la procedura locale all'interno di una meta euristica così come descritto nella sezione successiva.

3.4 Algoritmo di ILS (*Iterated Local Search*)

La tecnica della ricerca locale iterativa è un metodo di ricerca locale che genera una sequenza di soluzioni mediante l'utilizzo di una euristica di supporto ed è capace 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* di ricerca locale iterativa è stato introdotto da Lourenço et al. ([18]). In maniera più formale, il *framework* ILS può essere così definito:

- Θ un insieme di soluzioni ammissibili;
- θ una soluzione di Θ;
- $J : \Theta \rightarrow \Re$ una funzione obiettivo, $J(\theta)$ 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 del metodo ILS è il concetto di ricerca diagonale in Θ^*. In pratica, partendo da una soluzione di ottimo locale θ^*, una nuova soluzione $\theta^{'}$ è ottenuta effettuando una pertubazione di Θ^*. Di conseguenza, una nuova soluzione ottima locale è ottenuta come $LS(\theta^{'}) = \theta^{'*}$. Infine, la migliore soluzione ottima locale, selezionata tra Θ^* e $\Theta^{'*}$ è memorizzata come migliore soluzione corrente: Θ^*_{new}.
Nel costesto del MIRP la procedura di ILS è di seguito illustrata.
Per ogni *customer*:

1. definire una soluzione iniziale costruendo un piano di consegne per l'intero periodo di pianificazione. Tale soluzione può essere ottenuta applicando, ad esempio, le indicazioni fornite dalle politiche *standard* utilizzate per la gestione delle scorte;

2. invocare una procedura di *merge* sulla base dei piani di consegna ottenuti al passo precedente con l'obiettivo di ridurre, sull'intero periodo di pianificazione, il numero di giorni che prevedono consegne;

3. applicare la procedura HC-MIRP per migliorare il piano delle consegne così ottenuto;

4. invocare una procedura, definita di *shift*, in grado di perturbare la soluzione ottenuta al passo precedente;

5. se la fase di *shift* genera una nuova soluzione (perturbata), allora ripetere i passi 3 e 4, altrimenti terminare l'elaborazione.

L'*outline* della procedura è di seguito riportata:

Algorithm 10 ILS-MIRP

Require: Gli insiemi M, S, il grafo G, la flotta m di veicoli con capacità Q, il periodo di pianificazione T

Ensure: L'insieme R delle rotte di consegna sull'intero periodo di pianificazione T, l'insieme S aggiornato. Inizialmente pone l'insieme $R = \emptyset$

1: **for all** $i \in M$ **do**
2: Sia S_i l'insieme delle quantità di prodotto da consegnare, in accordo alla politica di gestione delle scorte, al *customer* i nel corso del periodo di pianificazione T; utilizzare la procedura $d_{ip} \leftarrow Get\_Order(i, p, \Pi_p)$ per ogni $p \in P_i$ e per ogni $t \in T$ per la fase di inizializzazione;
3: Sia R_i l'insieme delle rotte di consegna per il *customer* i. Inizializzare $R_i = \emptyset$;
4: Sia S_i^p l'insieme delle quantità di prodotto da consegnare al *customer* i a valle del processo di perturbazione. Inizializzare $S_i^p = \emptyset$;
5: $S_i = Merge(i, S_i, T)$
6: **while** TRUE **do**
7: $S_i =$ ripetere i passi 3 - 26 dell'algoritmo HC-MIRP
8: $S_i^p = \text{Shift}(i, S_i, T)$
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: $S = \bigcup_{i \in M} S_i$
17: **for all** $t \in T$ **do**
18: **if** $S^t \neq \emptyset$ **then**
19: Inizializzare M^t e P^t
20: $R = R \cup R^t \leftarrow \text{Routing}(M^t, P^t, S^t, G, m, Q)$
21: **end if**
22: **end for**

3.5 Algoritmo di *Nested Partitioning*

Preliminarmente alla definizione della procedura di tipo NP (*Nested Partitioning*) per il problema del MIRP si evidenziano le caratteristiche principali della tecnica di NP cosi come presentata da Shi and Ólafsson ([19], [20]). Consideriamo il seguente problema di ottimizzazione combinatoriale:

$$\min_{\theta \in \Theta} J(\theta),$$

dove Θ è una regione ammissibile finita e $J(\theta) : \Theta \rightarrow \Re$ è una funzione che esprime una misura di *performance*. Nell'ambito del problema MIRP, con riferimento al *Customer Model*, Θ corrisponde all'insieme dei valori ammissibili per le variabili y^t, x^t_{pk}, d^t_p, δ^+_{pt}, δ^-_{pt} e che ricadono nella regione ammissibile definita dai vincoli (2.4b)- (2.4n), mentre la funzione $J(\theta) : \Theta \rightarrow \Re$ non è altro che la funzione obiettivo definita in (2.4a). In generale, le procedure di *Nested Partitioning* prevedono di suddividere la funzione Θ in sotto regioni e successivamente di valutare il potenziale di ogni regione in modo tale da concentrare gli sforzi computazionali nelle regioni che presentano un alto *Valore Potenziale*. La regione inizialmente analizzata è rappresentata dalla regione Θ. Tale procedimento è utilizzato in maniera iterativa, in modo tale che, ad ogni ciclo del processo di ricerca, la regione potenzialmente più promettenete è annidata all'interno della regione più promettente dell'ultimo ciclo effettuato. Nell'ambito della *outline* della procedura di *Nested Partitioning* saranno utilizzate le seguenti definizioni:

- *Regione Valida* è una regione individuata sulla base di un ben determinato schema di partizione. L'insieme di tutte le *Regioni Valide* è indicato con Σ e Σ_0 indica la *Regione* contenete esattamente un'unica soluzione ammissibile (in seguito faremo

riferimento a tali regioni come regioni *singleton*);

- *Regola di Partizione* è la regola utilizzata per suddividere una determinata regione σ in M_σ sotto regioni. M_σ può dipendere dal sottoinsieme di partenza, ma non dal numero di iterazioni che sono state effettuate per la sua generazione;

- se la Regione $\sigma \in \Sigma$ è costituita da un insieme di regioni η, allora σ è detta una *sub-region* di η, e η è detta una *super-region* di σ;

- $I : \Sigma \to \Re$ è una funzione, denominata *promising index*, per la stima del potenziale di una determinata regione in accordo con la funzione di *performance*. Un esempio di tale funzione è il seguente:

$$I(\sigma) = min_{\theta_i \in \{\theta_1, \ldots, \theta_n\}} J(\theta_i).$$

Lo schema generale di una procedura di *Nested Partitioning* è il seguente:

1. **Partitioning**: nel corso della iterazione u effettuare il partizionamento della regione $\sigma(u)$ nelle sotto regioni $M_{\sigma(u)}$: $\sigma(u) = \{\sigma_1(u), \ldots, \sigma_{M_{\sigma(u)}}(u)\}$; inoltre, aggregare, in un'unica regione, l'intera regione circostante:

$$\Theta \backslash \{\sigma_1(u), \ldots, \sigma_{M_{\sigma(u)}}(u)\}$$

   Inoltre, generare le $M_{\sigma(u)} + 1$ regioni valide:

$$\Sigma = \{\sigma_1(u), \ldots, \sigma_{M_{\sigma(u)}}(u), \sigma_{M_{\sigma(u)+1}}(u)\}$$

2. **Random sampling**: per ogni regione $\sigma_i(u)$, $i = 1, \ldots, M_{\sigma(u)} + 1$, è definito un insieme di soluzioni per mezzo di una funzione *random*: $\{\theta_i^1, \ldots, \theta_i^{N_j}\}$ di N_j; i valori $J(\theta_i^1)$, ..., $J(\theta_i^{N_j})$ rappresentano i corrispondenti indici di *performance* delle soluzioni così determinate.

3. **Stima degli indici di *performance* più promettenti**: l'indice $I(\sigma_i(u))$ della regione $\sigma_i(u)$ è stimato come

$$\hat{I}(\sigma_i(u)) = min_{j=1,\ldots,N_j} J(\theta_i^j),\ \forall i = 1, \ldots, M_{\sigma(u)} + 1,$$

   dove $\hat{I}(\sigma_i(u))$ è il valore dell'indice di *performance* corrispondente alla regione $\sigma_i(u)$ e valutato sulla base delle soluzioni ammissibili selezionate in modo *random*.

4. **Backtracking**: selezionare la regione più promettente sulla base della seguente funzione:

$$\sigma_b = argmin_{i=1,\ldots,M_{\sigma(u)}+1} \hat{I}(\sigma_i(u)).$$

   Nel caso in cui due regioni hanno lo stesso indice di *performance*, allora la scelta avverrà in maniera casuale. Se $\sigma_b \subset \sigma(u)$, allora $u = u + 1$ e $\sigma(u) = \sigma_b$, altrimenti, se la regione circostante è la regione più promettente, allora si effettua un'operazione di *backtracking* che coinvolge una *super-region* più vasta.

5. **Stopping criterion**: il criterio di arresto della procedura è il criterio denominato *Cardinal Stopping Rule* e si basa sulle regole introdotte da De Haan ([8]) relative al teorema unilaterale degli intervalli di confidenza. Il processo di ricerca si arresta

nelle seguenti condizioni:

$$\frac{(J_{[2]} - J_{[1]})}{(1 - prob)^{-\chi} - 1} \leq \epsilon, \tag{3.1}$$

dove $J_{[2]}$ e $J_{[1]}$ sono rispettivamente la seconda e la prima miglior soluzione trovata, *prob* è la probabilità di trovare una soluzione con un indice di *performance* minore o uguale alla distanza ϵ dal valore ottimo. Il parametro χ è uguale a:

$$\chi = \frac{1}{\log n_{\Sigma_0(u)}} (\frac{J_{[\Sigma_0(u)]} - J_{[3]}}{J_{[3]} - J_{[2]}}),$$

dove:

- $n_{\Sigma_0(u)}$ è il numero delle regioni *singleton* raggiunte sino all'iterazione u;
- $J_{[\Sigma_0(u)]}$ è l'indice di *performance* della regione *singleton* peggiore individuata sino all'iterazione u;
- $J_{[3]}$ è l'indice di *performance* della terza miglior soluzione trovata sino all'iterazione u.

Il criterio di arresto è verificato ogni volta che la regione più promettente è una regione *singleton*. Nel caso in cui la condizione di arresto non sia soddisfatta allora si effettua un processo di *backtracking* all'intera regione ammissibile.

Nel caso del *Customer Model*, l'algoritmo di *Nested Partitioning* è utilizzato per generare la pianificazione della distribuzione per ogni *customer*. Una partizione è definita costruendo due diverse liste: una prima lista, definita *shifted*, in cui sono inseriti tutti i giorni che risultano essere anticipati o posticipati nel piano di consegne corrente

(rappresentante la partizione più promettente) e una seconda lista, definita di *giorni candidati*, contenente tutti i giorni che potrebbero essere anticipati o posticipati cosi come descritto nei passi 6-19 dell'algoritmo HC-MIRP. In particolare, l'algoritmo di *Nested Partitioning* assume la seguente forma:

1. **MIRP-Partitioning**. Inizializzare due distinte liste in rappresentanza di una partizione della regione ammissibile del problema; una lista, denominata *shifted list*, contenente l'insieme dei giorni *spostati* ed una lista, denominata *candidate list*, contenente l'insieme dei giorni che devono ancora essere *spostati*. La costruzione di tali liste avviene cosi come indicato nei passi 6-19 dell'algoritmo HC-MIRP. Se la lista dei giorni candidati è vuota allora la regione corrente è una regione *singleton*. Ad ogni passo dell'algoritmo, è definita la lista dei giorni candidati che può essere *spostata* in giorni adiacenti. Una differente pianificazione è ottenuta sulla base della scelta di anticipare o posticipare una determinata consegna. Pertanto, il numero di partizioni è definito dalla lunghezza della lista dei giorni candidati, e ogni sotto regione differesce dalle altre per il primo giorno selezionato della lista dei giorni *spostati*. La regione circostante è definita rimuovendo dalla lista dei candidati il primo giorno incluso nella *shifted list* della regione più promettente.

2. **MIRP-*Random sampling***. Il campionamento è effettuato sulla base di una scelta casuale dei valori contenuti nella partizione corrente; in tal modo si genera una soluzione *random*. La nuova soluzione è generata effettuando un'operazione di *shift* delle consegne previste per tutti i giorni definiti nella sotto regione corrente ed è completato anticipando o ritardando le consegne previste nei gorni candidati sino

a che la lista non diventa vuota. Ogni giorno è scelto, dalla lista di appartenenza, in modo casuale.

3. **MIRP-*Estimation of the most promising performance index***. L'indice di *performance* individuato per il MIRP è rappresentato dal numero di giorni di consegna $|ND(DP)|$ nell'arco dell'interno periodo di pianificazione, essendo $ND(DP) = \{ND_1, \ldots, ND_n\}$ l'insieme delle consegne schedulate nel piano di consegna DP. Osserviamo che tale indice consente di differenziare i piani di consegna e che l'indice di valore minimo è selezionato come il miglior valore:

$$DP^b = argmin_{\{DP \in \Omega(DP)\}}\{|ND(DP)|\},$$

dove $\Omega(DP)$ è l'insieme dei piani di consegna.

4. **MIRP-*Backtracking***. La procedura di *backtracking* è attivata nei seguenti due casi:

   - il primo caso si verifica quando nel corso della fase di ricerca la regione più promettente non è all'interno della partizione ma nella regione circostante; in questo caso la procedura invoca la fase di *backtracking* sull'intera regione ammissibile;
   - il secondo caso si verifica quando la procedura di ricerca individua una regione *singleton* (ovvero una regione che non è possibile partizionare in quanto la lista dei giorni candidati è vuota). In tal caso, il valore dell'indice di *per-*

formance è memorizzato, quindi si verifica il criterio di arresto prima di ripartire per partizionare l'intera regione ammissibile.

5. **MIRP-*Stopping criterion***. Il criterio di arresto della procedura è basato sulla regola di arresto, di carattere generale, denominata *Cardinal Stopping Rule* (3.1) che nel caso in oggetto assume la seguente forma:

$$\frac{(ND_{[2]} - ND_{[1]})}{(1 - prob)^{-\chi} - 1} \leq \epsilon, \tag{3.2}$$

dove $ND_{[1]} = ND(DP^b)$ e $\chi = \frac{1}{\log n_{\Sigma(DP)_0(u)}}(\frac{ND_{[\Sigma(DP)_0(u)]} - ND_{[3]}}{ND_{[3]} - ND_{[2]}})$. Si osservi che $n_{\Sigma(DP)_0(u)}$ è il numero di piani di consegna *Singleton* determinati sino all'iterazione u.

Capitolo 4

Esperimenti computazionali

4.1 Introduzione

Gli esperimenti computazionali sono stati effettuati utilizzando un insieme di *data set* rappresentativi di istanze reali del problema. La sperimentazione è stata condotta utilizzando il seguente ambiente di esecuzione:

- Pentium M con processore Intel Centrino con frequenza di *clock* di 2 Ghz e 2 Gbyte di memoria RAM;
- JRE (*Java Runtime Enviroment*)1.5;
- DBMS (*DataBase Management System*) Microsoft SQL Server 2005 per la memorizzazione dei *data set*.

La codifica degli algoritmi è stata effettuata utilizzando il linguaggio java (SDK 1.5). La risoluzione del modello (2.5), per la determinazione dei valori di *bound*, è stata

effettuata mediante l'utilizzo della libreria ILOG CPLEX (versione 10.1). In tal caso, il valore del parametro intero di controllo del livello di *probing* è stato fissato a 3 (cplex.setParam(IloCplex.IntParam.Probe,3)) e il tempo massimo di elaborazione è stato limitato al valore di 2 ore.

4.2 Definizione dei *Data Set* utilizzati per la sperimentazione

I *data set* utilizzati per la sperimentazione computazionale delle procedure euristiche sono stati costruiti sulla base dei seguenti parametri:

- periodo di pianificazione H: un valore selezionato tra le due possibili configurazione $H = 14$ o $H = 28$ giorni;
- numero di *customer* $|M| = 9$;
- numero di prodotti per ogni *customer* $|P_i| = 100w$, con $w = 1, 3, 5$;
- livello iniziale di inventario (giacenza) I_{ip}^0 del prodotto p presso il *customer* i generato in maniera *random* all'interno del *range* di valori $[150, 200]$;
- il livello massimo di inventario U_{ip} è stato fissato uguale al valore 200 per ogni $p \in P_i$, e per ogni $i \in M$;
- il tasso medio di consumo μ_{ip} del prodotto p presso il *customer* i è stato generato con una procedura *random* nel seguente intervallo di valori $[\frac{6000}{365}, \frac{9000}{365}]$, dove 6000

e 9000 sono rispettivamente il tasso medio minimo di consumo annuo ed il tasso medio massimo di consumo annuo del prodotto p presso il *customer* i;

- il costo di stoccaggio h_p per unità di prodotto p è stato generato in maniera *random* nell'intervallo di valori ammissibili $[0.13, 0.21]$;
- il costo fisso di riordino f_p del prodotto p è stato generato in maniera *random* nell'intervallo di valori $[50, 100]$;
- si è assunto di avere a disposizione una flotta di veicoli di capacità omogenea pari a $Q = 600$.

I dati caratteristici della rete logistica utilizzata per la sperimentazione sono riportati nelle tabelle 4.1 e 4.2 che rappresentano, rispettivamente, la matrice delle distanze (in chilometri) tra i nodi della rete e la matrice dei tempi di viaggio (espressi in ore).

Le finestre temporali utilizzate per il servizio dei *customer* sono le seguenti:

- $TW_1 = [7am - 11am]$ per i *customer* 1 e 3;
- $TW_2 = [9am - 13pm]$ per i *customer* 4 e 5;
- $TW_3 = [11am - 16pm]$ per i *customer* 6, 7, 8 e 9.

4.3 Analisi dei risultati computazionali

Il valore di soglia γ, utilizzato all'interno della procedura HC-MIRP, è stato fissato, in modo empirico, al valore 0.40. I valori associati alle regole di *Cardinal Stopping*

Tabella 4.1: Matrice delle distanze (chilometri)

	V	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
V	0.00	42.84	1.50	69.80	1.70	68.60	47.60	69.50	69.80	70.70
C1	71.90	0.00	73.40	1.40	73.60	1.00	68.90	1.70	1.80	1.70
C2	1.40	70.50	0.00	71.20	0.40	70.00	49.00	70.09	71.20	72.10
C3	72.30	1.10	73.80	0.00	74.00	0.80	69.20	2.10	0.80	0.70
C4	2.30	71.40	6.00	72.10	0.00	71.00	50.40	71.80	72.20	73.00
C5	71.70	1.20	73.20	1.10	73.40	0.00	68.70	1.90	1.70	1.90
C6	48.00	70.90	49.50	71.60	49.80	70.40	0.00	71.20	71.60	72.40
C7	72.90	1.90	74.40	2.60	74.40	1.40	69.90	0.00	3.10	3.40
C8	72.90	1.70	74.40	1.40	74.70	1.10	69.90	2.40	0.00	1.70
C9	73.80	1.00	75.30	1.60	75.60	1.70	70.80	3.00	2.40	0.00

Tabella 4.2: Matrice dei tempi di percorrenza (ore)

	V	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
V	0.00	1h5m	3m	1h7m	3m	1h4m	48m	1h6m	1h7m	1h8m
C1	1h4m	0.00	1h8m	4m	1h8m	5m	1h16m	6m	5m	4m
C2	4m	1h9m	0.00	1h12m	3m	1h9m	53m	1h10m	1h12m	1h13m
C3	1h4m	3m	1h8m	0.00	1h7m	3m	1h16m	5m	2m	1m
C4	7m	1h12m	2m	1h14m	0.00	1h11m	54m	1h13m	1h14m	1h15m
C5	1h2m	4m	1h6m	6m	1h5m	0.00	1h13m	5m	6m	7m
C6	47m	1h17m	50m	1h20m	50m	1h17m	0.00	1h19m	1h20m	1h21m
C7	1h3m	4m	1h7m	6m	1h6m	3m	1h15m	0.00	7m	7m
C8	1h4m	5m	1h8m	4m	1h8m	4m	1h16m	5m	0.00	5m
C9	1h5m	3m	1h9m	5m	1h9m	6m	1h17m	6m	7m	0.00

dell'algoritmo NP-MIRP sono stati impostati a $\epsilon = 0.1$ e $prob = 0.90$. Le tabelle dei risultati riportano le seguenti informazioni:

- la colonna *Instance* riporta il numero identificativo dell'istanza;
- la colonna $|M|$ riporta il numero di *customer* presenti nell'istanza;
- la colonna $|P_i|$ riporta il numero di prodotti per ogni *customer* i;
- la colonna $\bar{V}$ riporta il volume medio consegnato ai *customer*, nel periodo di pianificazione T; tale valore è determinato considerando i valori di *output* delle euristiche $\{\text{HC}, \text{NP}, \text{ILS}\}$ su tutte le istanze prese in considerazione;
- la colonna $\bar{D}$ riporta la distanza media percorsa, nel periodo di pianificazione T; tale valore è determinato considerando i valori di *output* delle euristiche $\{\text{HC}, \text{NP}, \text{ILS}\}$ su tutte le istanze prese in considerazione;
- la colonna $\frac{V_{HC}}{D_{HC}}$ riporta il volume per chilometro determinato attraverso la procedura euristica HC-MIRP;
- la colonna $\frac{V_{NP}}{D_{NP}}$ riporta il volume per chilometro determinato attraverso la procedura euristica NP-MIRP;
- la colonna $\frac{V_{ILS}}{D_{ILS}}$ riporta il volume per chilometro determinato attraverso la procedura euristica ILS-MIRP;
- la colonna $\frac{\bar{V}}{\bar{D}}$ riporta il volume medio per chilometro calcolato considerando i valori di *output* delle euristiche $\{\text{HC}, \text{NP}, \text{ILS}\}$ su tutte le istanze prese in considerazione;

- la colonna $UB_{\frac{V}{D}}$ riporta il valore dell'*upper bounds* del volume per chilometro calcolato così come descritto nella sezione 2.4;

- la colonna GAP riporta la differenza (*gap*) tra i valori ottenuti dalla procedura euristica rispetto al valore dell'*upper bound*, valutato come $\frac{(UB_{V/D}-V_H/D_H)\cdot 100}{UB_{V/D}}$, dove V_H/D_H rappresenta il volume per chilometro calcolato mediante la procedura euristica $H \in \{\text{HC}, \text{NP}, \text{ILS}\}$;

- la colonna $\frac{\bar{V}}{\bar{D}}$ riporta la media del volume per chilometro calcolata sulla base dei valori ottenuti su tutti i *data set*;

- la riga **Best** riporta il miglior valore del rapporto $\frac{V_b}{D_b}$;

- la riga **Average** riporta il valore del medio del rapporto;

- la riga **Wrost** riporta il peggior valore del rapporto $\frac{V_w}{D_w}$;

- la riga **Time** riporta il tempo medio di computazione dei risultati;

- la riga **BestGAP** riporta il miglior valore del *gap*;

- la riga **AverageGAP** riporta il valore medio del *gap*;

- la riga **WorstGAP** riporta il peggior valore del *gap*.

Si evidenzia che gli algoritmi NP-MIRP e ILS-MIRP hanno valori di *performance* molto vicini tra di loro sia rispetto ai valori medi sia rispetto ai valori ottimi (colonna *best*). Il miglior valore del *gap*, calcolato come $\frac{(V_b/D_b-V_w/D_w)\cdot 100}{V_w/D_w}$, è ottenuto mediante l'utilizzo dell'algorimto HC-MIRP ed è uguale a 3.05.

Tabella 4.3: Performance - Dataset $|P_i| = 100$ e $H = 14$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$\frac{V_{HC}}{D_{HC}}$	$\frac{V_{NP}}{D_{NP}}$	$\frac{V_{ILS}}{D_{ILS}}$	$UB_{\frac{V}{D}}$
1	9	100	8.92	9.07	9.16	9.46
2	9	100	9.01	9.16	9.17	9.58
3	9	100	9.05	9.21	9.12	9.51
4	9	100	9.11	9.23	9.20	9.54
5	9	100	8.84	9.05	8.96	9.55
6	9	100	8.97	9.14	9.13	9.45
7	9	100	9.09	9.25	9.13	9.64
8	9	100	9.08	9.17	9.16	9.48
9	9	100	9.05	9.19	9.15	9.42
10	9	100	9.05	9.03	9.10	9.50
Best			**9.11**	**9.25**	**9.20**	**9.64**
Average			**9.02**	**9.15**	**9.13**	**9.51**
Worst			**8.84**	**9.03**	**8.96**	**9.42**
Time (milli sec.)			**0.00**	**32541**	**4.60**	**15799.9**

La tabella 4.4 mostra i *gap*, rispetto al valore dell'*upper bound*, delle diverse euristiche utilizzate.

Il miglior valore medio si registra in corrispondenza dell'algoritmo NP-MIRP ed è uguale al valore 3.81. L'algoritmo di ricerca locale HC-MIRP fa registrare ottime prestazioni computazionali sulla base del fatto che implementa semplici criteri di ricerca locale. Leggermente più onerosa risulta l'elaborazione della procedura NP-MIRP a causa della struttura iterativa del processo di *backtracking* interno all'algoritmo.

La tabella 4.5 riporta i risultati per un *data set* caratterizzato da istanze che presentano un elevato numero di prodotti per ogni *customer*. La tabella 4.6 riporta, invece, i valori di *gap* rispetto al valore dell'*upper bound*.

I risultati riportati nelle tabelle 4.5 e 4.6 mostrano che il valore medio, relativamente ai tempi di calcolo ed al valore del volume per chilometro, sono ottenuti dalle

Tabella 4.4: Gap - Dataset $|P_i| = 100$ e $H = 14$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$GAP_{\frac{V_{HC}}{D_{HC}}}$	$GAP_{\frac{V_{NP}}{D_{NP}}}$	$GAP_{\frac{V_{ILS}}{D_{ILS}}}$
1	9	100	5.68	4.10	3.15
2	9	100	5.94	4.37	4.27
3	9	100	4.89	3.20	4.15
4	9	100	4.51	3.25	3.56
5	9	100	7.44	5.24	6.19
6	9	100	5.09	3.29	3.40
7	9	100	5.72	4.06	5.30
8	9	100	4.17	3.22	3.33
9	9	100	3.94	2.46	2.88
10	9	100	4.69	4.90	4.17
BestGAP			**3.94**	**2.46**	**2.88**
AverageGAP			**5.21**	**3.81**	**4.04**
WorstGAP			**7.44**	**5.24**	**6.19**

Tabella 4.5: Performance - Dataset $|P_i| = 300$ e $H = 14$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$\frac{V_{HC}}{D_{HC}}$	$\frac{V_{NP}}{D_{NP}}$	$\frac{V_{ILS}}{D_{ILS}}$	$UB_{\frac{V}{D}}$
1	9	300	9.47	9.57	9.53	9.76
2	9	300	9.43	9.42	9.46	9.77
3	9	300	9.46	9.55	9.50	9.73
4	9	300	9.46	9.48	9.49	9.76
5	9	300	9.42	9.46	9.45	9.76
Best			**9.47**	**9.57**	**9.53**	**9.77**
Average			**9.45**	**9.50**	**9.49**	**9.76**
Worst			**9.42**	**9.42**	**9.45**	**9.73**
Time (milli sec.)			**6.40**	**85556**	**34.40**	**90453.2**

Tabella 4.6: Gap - Dataset $|P_i| = 300$ e $H = 14$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$GAP_{\frac{V_{HC}}{D_{HC}}}$	$GAP_{\frac{V_{NP}}{D_{NP}}}$	$GAP_{\frac{V_{ILS}}{D_{ILS}}}$
1	9	300	2.95	1.92	2.33
2	9	300	3.53	3.63	3.22
3	9	300	2.82	1.90	2.41
4	9	300	3.05	2.85	2.74
5	9	300	3.48	3.07	3.17
BestGAP			**2.82**	**1.90**	**2.33**
AverageGAP			**3.17**	**2.67**	**2.78**
WorstGAP			**3.53**	**3.63**	**3.22**

procedure NP-MIRP e ILS-MIRP. In particolare, l'algoritmo NP-MIRP fa registrare il miglior risultato rispetto alla distanza dal valore dell'*upper bound* (miglior *gap*) facendo registrare il valore 1.59. E' importante evidenziare la maggiore complessità delle istanze esaminate, derivante dall'elevato numero di prodotti da consegnare. Ciò nonostante, le procedure euristiche hanno fatto registrare un leggero incremento dei tempi di calcolo della soluzione ottima locale.

La tabella 4.7 riporta i risultati per istanze con un numero molto elevato di prodotti per ogni *customer*. La tabella 4.8 riporta le differenze registrate rispetto ai valori dell'*upper bound*.

Si sono registrati tempi di calcolo molto ridotti, rispetto alle dimensioni delle istanze utilizzate, per ognuno degli algoritmi realizzati. La motivazione risiede nel fatto che la consegna di un grande numero di prodotti non può essere facilmente anticipata o ritardata nel periodo di pianificazione in quanto deve essere garantita l'assenza di situazioni di *stock out*. La soluzione finale è, infatti, ottenuta operando un numero limitato di iterazioni a partire dalla soluzione ammissibile iniziale.

Tabella 4.7: Performance - Dataset $|P_i| = 500$ e $H = 14$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$\frac{V_{HC}}{D_{HC}}$	$\frac{V_{NP}}{D_{NP}}$	$\frac{V_{ILS}}{D_{ILS}}$	$UB_{\frac{V}{D}}$
1	9	500	9.59	9.59	9.54	9.93
2	9	500	9.59	9.64	9.58	9.93
3	9	500	9.52	9.52	9.53	9.93
4	9	500	9.54	9.54	9.54	9.97
5	9	500	9.58	9.60	9.53	9.96
Best			**9.59**	**9.64**	**9.58**	**9.97**
Average			**9.57**	**9.58**	**9.54**	**9.95**
Worst			**9.52**	**9.52**	**9.53**	**9.93**
Time (milli sec.)			**0.00**	**118484**	**84.40**	**619853.2**

Tabella 4.8: Gap - Dataset $|P_i| = 500$ and $H = 14$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$GAP_{\frac{V_{HC}}{D_{HC}}}$	$GAP_{\frac{V_{NP}}{D_{NP}}}$	$GAP_{\frac{V_{ILS}}{D_{ILS}}}$
1	9	500	3.47	3.47	3.97
2	9	500	3.47	2.97	3.57
3	9	500	4.12	4.12	4.02
4	9	500	4.27	4.27	4.27
5	9	500	3.84	3.64	4.34
BestGAP			**3.47**	**2.97**	**3.57**
AverageGAP			**3.83**	**3.69**	**4.03**
WorstGAP			**4.27**	**4.27**	**4.34**

Le tabelle 4.9, 4.10, 4.11 e 4.12 riportano i risultati ottenuti nel caso in cui il periodo di pianificazione è stato esteso a 28 giorni. In questo caso, è possibile osservare che nel caso di istanze con più di 100 prodotti per *customer* il risolutore del modello per la determinazione del *bound* non restituisce, nel tempo limite di elaborazione prefissato, una soluzione ammissibile.

Tabella 4.9: Performance - Dataset $|P_i| = 100$ e $H = 28$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$\frac{V_{HC}}{D_{HC}}$	$\frac{V_{NP}}{D_{NP}}$	$\frac{V_{ILS}}{D_{ILS}}$
1	9	100	9.12	9.32	9.18
2	9	100	9.29	9.34	9.25
3	9	100	9.22	9.33	9.28
4	9	100	9.19	9.23	9.15
5	9	100	9.02	9.14	9.07
6	9	100	9.19	9.20	9.28
7	9	100	9.18	9.25	9.20
8	9	100	9.14	9.31	9.20
9	9	100	9.06	9.13	9.11
10	9	100	9.13	9.20	9.13
Best			**9.29**	**9.34**	**9.28**
Average			**9.15**	**9.25**	**9.19**
Worst			**9.02**	**9.13**	**9.07**
Time (milli sec.)			**3.20**	**343245**	**12.40**

Le prestazioni dell'algoritmo NP-MIRP sono superiori rispetto, alle alte procedure, in relazione all'indicatore rappresentato dal rapporto volume di prodotto per chilometri percorsi mentre sono inferiori rispetto ai tempi di elaborazione. Il miglior risultato, su quest'ultimo *data set*, si ottiene, infatti, attraverso l'utilizzo della procedura NP-MIRP e fa registrare un *best gap* uguale a 0.62.

Tabella 4.10: Gap - Dataset $|P_i| = 100$ e $H = 28$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$GAP_{\frac{V_{HC}}{D_{HC}}}$	$GAP_{\frac{V_{NP}}{D_{NP}}}$	$GAP_{\frac{V_{ILS}}{D_{ILS}}}$
1	9	100	7.46	5.43	6.85
2	9	100	6.30	5.79	6.70
3	9	100	6.49	5.38	5.88
4	9	100	6.28	5.87	6.68
5	9	100	8.26	7.04	7.75
6	9	100	6.90	6.80	5.99
7	9	100	7.09	6.38	6.89
8	9	100	8.17	6.47	7.57
9	9	100	8.17	7.46	7.66
10	9	100	6.69	5.97	6.69
BestGAP			**6.28**	**5.38**	**5.88**
AverageGAP			**7.18**	**6.26**	**6.87**
WorstGAP			**8.26**	**7.46**	**7.75**

Tabella 4.11: Performance - Dataset $|P_i| = 300$ e $H = 28$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$\frac{V_{HC}}{D_{HC}}$	$\frac{V_{NP}}{D_{NP}}$	$\frac{V_{ILS}}{D_{ILS}}$
1	9	300	9.57	9.60	9.55
2	9	300	9.53	9.58	9.53
3	9	300	9.53	9.60	9.58
4	9	300	9.59	9.63	9.59
5	9	300	9.50	9.55	9.53
Best			**9.59**	**9.60**	**9.59**
Average			**9.54**	**9.59**	**9.55**
Worst			**9.50**	**9.55**	**9.53**
Time (milli sec.)			**12.20**	**718953**	**97.00**

Tabella 4.12: Performance - Dataset $|P_i| = 500$ e $H = 28$ giorni.

Instance	$\|M\|$	$\|P_i\|$	$\frac{V_{HC}}{D_{HC}}$	$\frac{V_{NP}}{D_{NP}}$	$\frac{V_{ILS}}{D_{ILS}}$
1	9	500	9.64	9.68	9.64
2	9	500	9.67	9.71	9.68
3	9	500	9.65	9.66	9.67
4	9	500	9.61	9.65	9.64
5	9	500	9.63	9.67	9.67
Best			**9.67**	**9.71**	**9.68**
Average			**9.64**	**9.67**	**9.66**
Worst			**9.61**	**9.65**	**9.64**
Time (milli sec.)			**15.40**	**915837**	**187.40**

Capitolo 5

Realizzazione di un dimostratore *software*

5.1 Introduzione

La fase di sperimentazione computazionale è stata effettuata sulla base di un dimostratore software in grado di interagire con la base di dati contenente i *data set* appositamente predisposti. Il paradigma del VMI prevede che per ogni giorno del periodo di pianificazione siano effettuate le seguenti operazioni, così come descritto in Figura 5.1:

- rilevazione dei livelli di inventario, o simulazione del processo di vendita, presso i nodi *customer*;
- determinazione dell'insieme di rotte da attivare sull'intero orizzonte temporale; la pianificazione è determinata per tutti i giorni dell'intervallo di pianificazione T;

- determinazione dei percorsi da effettuare per l'esecuzione dei piani di rifornimento; in questa fase è applicato l'algoritmo di *vehicle routing* che consente di ottimizzare le distanze percorse dai veicoli utilizzati per la distribuzione;
- emissione dei piani di rifornimento e conseguente aggiornamento dei livelli di inventario dei *customer*.

La libreria di ottimizzazione sviluppata per il modello di pianificazione integrata del VMI è predisposta per essere integrata con i sistemi informativi aziendali in modo tale che rendere intuitivo ed efficiente l'intero processo di gestione del VMI. Offre, inoltre, una semplice interfaccia grafica per la visualizzazione dei dati di *output* delle procedure di ottimizzazione così come illustrato nella sezione successiva.

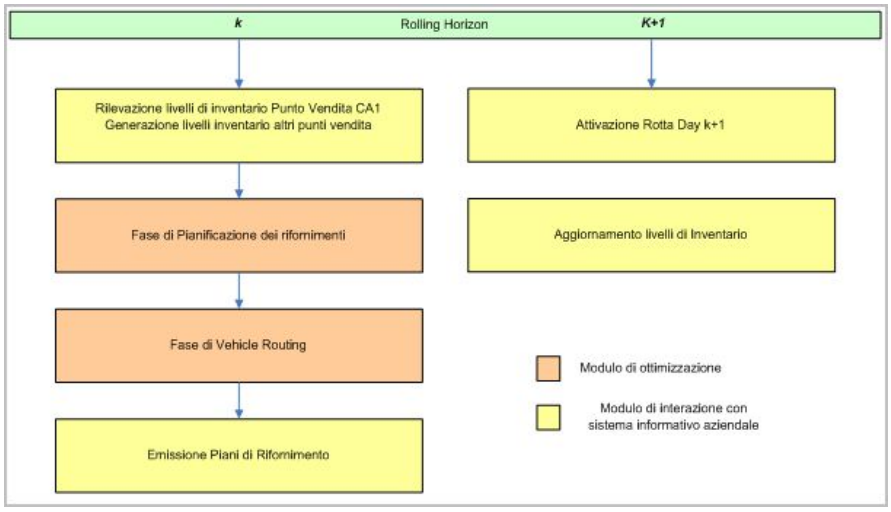

Figura 5.1: Processo di pianificazione integrata secondo il modello VMI

5.2 Architettura *software* del dimostratore

La Figura 5.3 rappresenta l'architettura logica del dimostratore *software* realizzato per la sperimentazione computazionale delle procedure euristiche risolutive del modello del VMI. La progettazione del dimostratore è stata effettuata sulla base di criteri di modularità in modo tale da favorire successive attività di manutenzione, integrazione della libreria di ottimizzazione in sistemi terze parti e lo sviluppo di nuovi moduli funzionali. Il dimostratore è stato interamente sviluppato mediante l'utilizzo del linguaggio java (*Java Develepment Kit* 1.5).

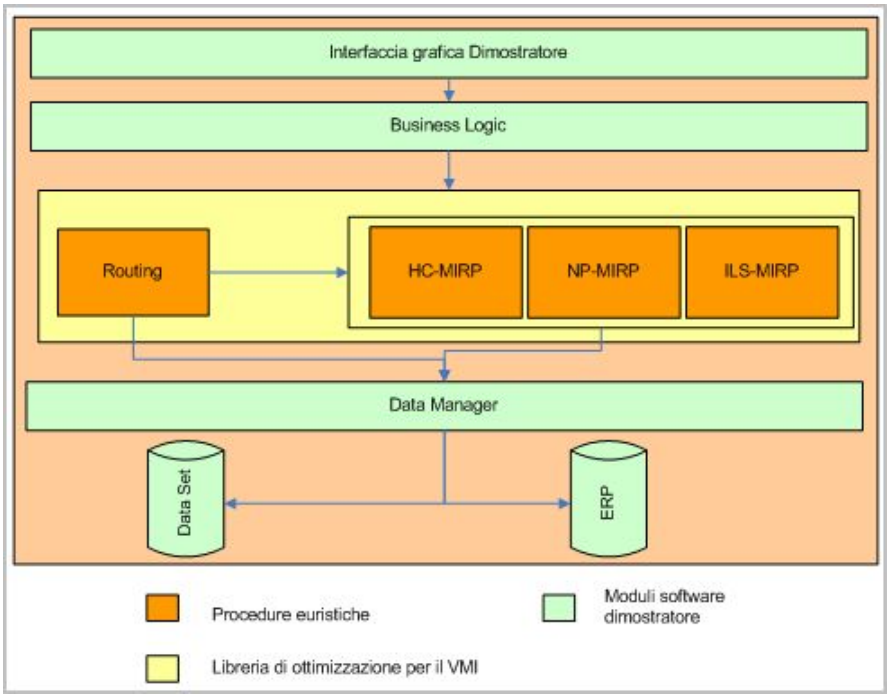

Figura 5.2: Architettura logica del dimostratore *software*

La Figura 5.3 evidenzia la presenza dei seguenti moduli funzionali:

- *GUI*: è il modulo che implementa la componente *view* delle *web-application*; è stata sviluppata utilizzando la tecnologia java e le JSP (*Java Server Page*).

- *Business Logic*: sono i moduli che realizzano la logica applicativa attraverso un insieme di *java bean* che utilizzano i metodi dalla libreria del VMI fornendo, all'applicazione, un maggiore livello di astrazione;

- *VMI Library*: è l'insieme dei moduli che implementano le diverse procedure euristiche definite per la risoluzione del modello del VMI;

- *Data Manager*: è il modulo che si occupa di interagire con il DBMS (*Data Base Management System*) per la lettura dei dati in ingresso (lettura dei *data set*) e la memorizzazione dei dati di *output* (risultato del processo di pianificazione integrata).

Il modulo relativo all'interfaccia grafica è una *web application* che consente di visualizzare l'insieme dei dati forniti in *output* dalle procedure risolutive e che sono rappresentativi della soluzione del problema del VMI. In particolare, le funzionalità fruibili attraverso tale interfaccia grafica consentono di:

- eseguire una nuova pianificazione per un determinato periodo di pianificazione;

- effettuare la ricerca, all'interno dell'archivio storico, della pianificazione effettuata per un determinato periodo;

- visualizzare l'ultimo periodo di pianificazione e attivare le rotte del primo giorno di pianificazione.

Figura 5.3: Voci menù applicazione VMI

La Figura 5.3 riporta il *menù* utente a disposizione dell'operatore.

La Figura 5.4 riporta l'elenco delle ultime pianificazioni effettuate. I dati sono presentati all'operatore sia in forma di elenco, sia sotto forma di raggruppamento. Il raggruppamento è effettuato sulla base del periodo di pianificazione.

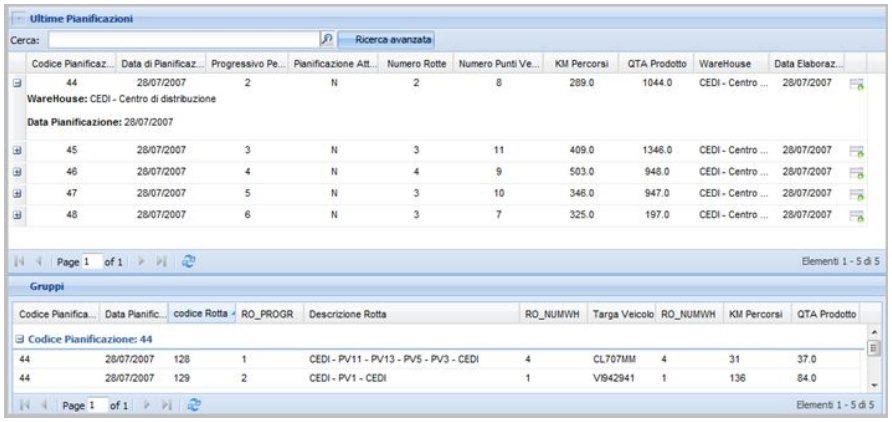

Figura 5.4: Lista ultime pianificazioni

Utilizzando l'icona che appare a lato di una determinata pianificazione (ultima colonna visibile), è possibile ottenere il dettaglio delle rotte da attivare a copertura delle esigenze di distribuzione, così come mostrato in Figura 5.5.

Attraverso l'utilizzo delle icone associate alle rotte è possibile effettuare le seguenti operazioni:

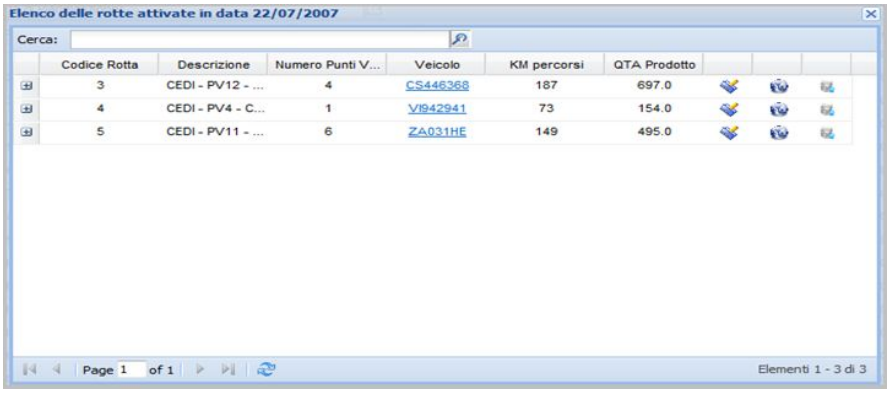

Figura 5.5: Elenco delle rotte da attivare

- accedere al dettaglio delle quantità di prodotto da distribuire ai punti vendita appartenenti alla rotta stessa (ordine per priorità di visita dei punti vendita);
- accedere alla mappa (se disponibile sulla rete internet) che consente di visualizzare il percorso da seguire, ottenendo informazioni circa le distanze ed i tempi di percorrenza;
- accedere alla funzionalità di attivazione di una rotta (consente di trasferire i dati all'interno del sistema ERP per la generazione dei documenti fiscali necessari al trasporto della merce).

Naturalmente, è possibile apportare delle modifiche sulle quantità di prodotto da consegnare ad un determinato punto vendita. Tale funzionalità è stata inserita al solo scopo di conferire massima flessibilità nell'utilizzo della metodologia di distribuzione. Infatti, nella gestione dei casi reali, potrebbe sempre capitare l'esigenza di gestire una particolare richiesta (non prevedibile) di fornitura di un particolare prodotto a causa

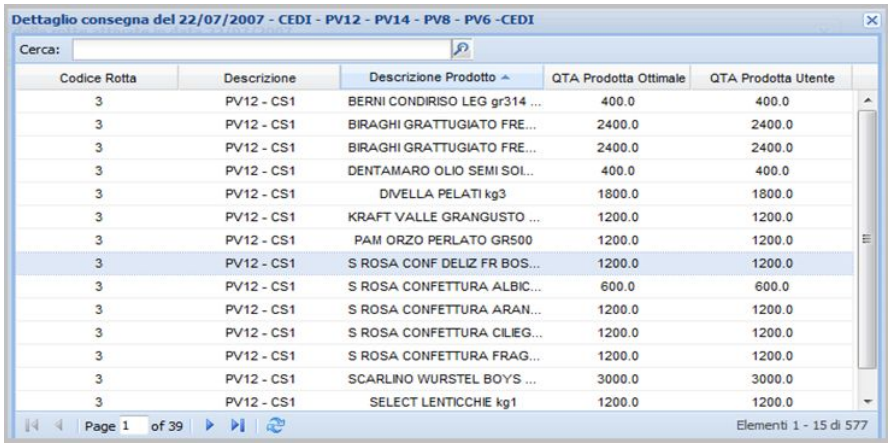

Figura 5.6: Dettaglio dei prodotti da consegnare su una rotta

di una maggiore vendita effettuata nell'ultimo periodo o a causa di una particolare esigenza da parte di un cliente.

Capitolo 6

Conclusioni

Il metodo proposto rappresenta un valido compromesso fra gli approcci scientifici attuali e le indicazioni ottenute dall'analisi di situazioni reali. Per mantenere i tempi di calcolo entro limiti accettabili si è deciso di utilizzare un approccio deterministico. Pur riconoscendo le caratteristiche stocastiche dei dati di *input* e l'aleatorietà derivante dall'uso di un orizzonte temporale di lungo periodo, il metodo proposto ha una struttura temporale di tipo *rolling horizon*. Un altro aspetto significativo dell'approccio proposto è che si desiderano valutare gli effetti di lunga durata delle decisioni prese nel breve periodo; ne consegue che l'orizzonte temporale di pianificazione deve essere sufficiente lungo, affinché i dati disponibili permettano di prevedere i tassi medi di consumo dei prodotti nel periodo programmato. Nella struttura *rolling horizon* il modello di pianificazione è risolto su un orizzonte temporale di k giorni, e viene utilizzato per la distribuzione nei primi j giorni. Il valore di j è generalmente piccolo al fine di sfruttare le nuove informazioni relative ai livelli di inventario ed ai tassi di consumo dei prodotti

presso i centri periferici. Tipicamente j è compreso fra uno e due giorni, in modo da fornire preliminarmente ai conducenti dei veicoli il piano di lavoro. Seguendo l'approccio proposto da Jaillet et al. ([3]) il processo decisionale è scomposto in due fasi. Nella prima fase si decide sulla pianificazione di lungo periodo, vale a dire su k giorni; mentre, nella seconda fase si prendono decisioni di breve termine, quindi su j giorni (la parte della pianificazione che sarà realizzata). La prima fase produce in ciascuno dei k giorni un assegnamento delle quantità dei prodotti da consegnare ai *customer*. La seconda fase prevede la costruzione degli itinerari per le consegne programmate. Nell'approccio descritto da Jaillet et al. ([3]) l'assegnazione delle consegne ai *customer* avviene sulla base delle informazioni relative al singolo prodotto che ogni giorno ciascun *customer* esporta al *vendor*. In questo modo si massimizza la quantità consegnata al *customer*; tuttavia non è detto che questo approccio determini nel lungo periodo una distribuzione ottimale, poichè non si tiene conto delle sinergie che possono esistere fra i singoli *customer*. Infatti, può accadere che due o più *customer* si trovino geograficamente vicini ed abbiano livelli di inventario e tassi medi di consumo abbastanza simili, cosicché può risultare conveniente servire questi *customer* con un'unica rotta, compatibilmente con le capacità di carico del veicolo. In questo modo, la pianificazione risulta più efficiente nel lungo periodo e determina una riduzione sensibile dei costi.

La novità dell'approccio proposto risiede principalmente nel fatto che la strategia di distribuzione e di rifornimento dei singoli *customer* presenta un *focus* orientato non solo alla minimizzazione dei costi di trasporto del *vendor*, ma anche al soddisfacimento delle aspettative dei centri periferici dal punto di vista della tempistica dei rifornimenti. In questo quadro, la scelta delle consegne ai clienti avviene in relazione alla prossimità

geografica ed all'omogeneità di consumo dei clienti stessi.

Le procedure euristiche definite e sperimentate in questo lavoro di tesi hanno evidenziato i benefici del modello del VMI nel caso particolare di un sistema logistico che opera sulla distribuzione di un elevato numero di prodotti. I risultati ottenuti hanno permesso di validare la bontà dell'approccio risolutivo utilizzato. In questo ambito, si ritiene che successive attività di ricerca possano migliorare i risultati ottenuti mediante la definizione di modelli matematici computazionalmente più trattabili, e la progettazione di algoritmi euristici o metaeuristici capaci di fornire *bound* stringenti in tempi di calcolo accettabili.

Bibliografia

[1] A. Assad, R. Dahl, and B. Golden. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems*, 7(2-3):181Ű–190, 1984.

[2] M. Ball. *Vehicle Routing: Methods and Studies*, chapter Allocation/routing: Models and algorithms, pages 199–221. Elsevier Science, Amsterdam, The Netherlands, 1988.

[3] J. Bard, M. Dror, L. Huang, and P. Jaillet. Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science*, 36(3):292Ű–300, 2002.

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

[5] J. Bard, L. Huang, M. Dror, and P. Jaillet. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science*, 33(2):189Ű–203, 1998b.

[6] L. Beltrami, F.and Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65Ű–94, 1974.

[7] A. Campbell and M. Savelsbergh. A decomposition approach for the inventory-routing problem. *Transportation Science*, 38(4):488Ű–502, 2004c.

[8] L. De Haan. Estimation of the minimum of a function using order statistics. *Methodology and Computing in Applied Probability*, 76:467–469, 1981.

[9] M. Dror, M. Ball, and B. Golden. Computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research*, 4(1):1Ű–23, 1985.

[10] A. Federgruen and P. Zipkin. A combined vehicle routing and inventory allocation problem. *Operations Research*, 32(5):1019Ű–1037, 1984.

[11] J. Forrester. Industrial dynamics. *MIT Press*, 1961.

[12] Paletta G. Gaudioso, M. A heuristic for the periodic vehicle routing problem. *Transportation Sci*, 26:86–92, 1992.

[13] J.B. Houlihan. A new approach. management decision. *International Supply Chain Management*, 26:13–19, 1988.

[14] N. Houlihan and J. Beasley. The period routing problem. *Networks*, 14:237–256, 1984.

[15] A. Kleywegt, V. Nori, and M. Savelsbergh. The stochastic inventory routing problem with direct deliveries. *Transportation Science*, 36:94Ű–118, 2002.

[16] A. Kleywegt, V. Nori, and M. Savelsbergh. Dynamic programming approximations for a stochastic inventory routing problem. *Transportation Science*, 38:42Ű–70, 2004.

[17] D. Laganà, P. Legato, and M. Savelsbergh. A decomposition approach for the multi-product inventory-routing problem. *Technical report, Department of Electronics, Computer and System Sciences, University of Calabria, Rende, Italy and The Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta (USA)*, 2009.

[18] H. R. Lourenço, O. C. Martin, and M. Stützle. *Handbooks of Metaheuristics*, volume 14, chapter Iterated Local Search, pages 321Ű–353. F. Glover and G. A. Kochenberger editors, 2001.

[19] L. Shi and S. Olafsson. Nested partitions method for global optimization. *Operations Research*, 48:390–407, 1991.

[20] L. Shi and S. Olafsson. Stopping rules for the stochastic nested partitions method. *Methodology and Computing in Applied Probability*, 2(1):37–58, 2000.