ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Seconda Facoltà di Ingegneria - Sede di Cesena
Corso di Laurea Magistrale in Ingegneria Informatica

PIANIFICAZIONE ED OTTIMIZZAZIONE DEI PERCORSI NELLA LOGISTICA DEI MAGAZZINI INDUSTRIALI

Elaborata nel corso di: Metodi e Modelli per il Supporto alle Decisioni LM

Tesi di Laurea di:
MANUEL FABBRI

Relatore:
Prof. DANIELE VIGO

Correlatore:
Ing. CLAUDIO GAMBETTI

ANNO ACCADEMICO 2011–2012
SESSIONE III

PAROLE CHIAVE

Logistica
Industria
Magazzino
Pianificazione
Percorso

Indice

Introduzione

Nell'ambito della logistica industriale, un problema di base riguarda la necessità di gestire i flussi di merci in entrata e in uscita dagli impianti; una componente fondamentale di questo problema concerne la pianificazione e l'ottimizzazione dei percorsi all'interno di un magazzino industriale.

Si è voluto affrontare parte di questo vasto argomento mediante un'esperienza di tirocinio presso **Onit Group s.r.l.**, una software house di Cesena che dal 1996 svolge la propria attività nel settore delle tecnologie informatiche e della consulenza mirata al management e alla organizzazione aziendale.

Il tirocinio, dal titolo *Algoritmi di pianificazione percorsi consentiti indoor e calcolo distanze*, ha affrontato la seguente necessità aziendale: la tendenza a modificare sempre più frequentemente, in base alle mutevoli esigenze del mercato, le modalità di utilizzo degli spazi negli impianti di stoccaggio rende determinante pianificare in maniera ottima e automatica i percorsi di movimentazione interna. La pianificazione dei percorsi deve essere eseguita per determinare il viaggio di un lavoratore o di una macchina all'interno di un'area, evitando gli ostacoli. Attraverso l'attività di tirocinio è stato possibile esaminare direttamente questo contesto di realtà aziendale, visitando gli stabilimenti della **società cooperativa agricola Orogel** di Cesena e osservando sul campo i processi di carico e scarico merci, lavorazione, produzione, confezionamento e stoccaggio per la linea dei prodotti surgelati.

Una volta analizzato tale dominio applicativo, nel quale si intende agire, l'obiettivo della tesi è di studiare algoritmi di pianificazione e ottimizzazione dei percorsi, all'interno di ambienti complessi, e implementarne una soluzione software valutandone le prestazio-

ni all'interno di uno scenario reale, per fornire uno strumento che possa essere di supporto in questo ambito.

Il progetto appena descritto è stato sviluppato con il supporto della **business unit industria** di *Onit*, che opera negli ambiti della progettazione e ottimizzazione dei processi produttivi, qualitativi e di logistica.

La tesi è organizzata come segue: nel capitolo 1 si presenta una panoramica generale del contesto lavorativo, introducendo i vari tipi di magazzini industriali e il dominio in cui si collocano. Questa conoscenza è necessaria come premessa delle fasi seguenti.

Nel secondo capitolo ci si focalizza sul prodotto dell'attività di ricerca bibliografica riguardante la modellazione dei magazzini e gli algoritmi di ottimizzazione dei percorsi. Si riportano quelli ritenuti più rilevanti, sui quali è stata prodotta una sintesi.

Il capitolo successivo tratta della soluzione software sviluppata, a fronte dei requisiti del problema, dell'analisi e del progetto. Inoltre, si pone l'attenzione anche sulla tecnologia utilizzata, mediante uno sguardo generale sul `framework .NET 4.0`.

Nel capitolo 4 si descrivono i risultati ottenuti, mediante un test del software applicato ad un caso reale, commentando analisi e simulazioni applicate a scenari produttivi inerenti all'azienda *Orogel*.

Si termina, infine, con una sezione conclusiva sull'attività svolta.

Capitolo 1

Il sistema logistico e i magazzini industriali

In questo capitolo si introduce il concetto di sistema logistico, per inquadrare l'ambito nel quale si colloca il magazzino industriale. Si inizia con una panoramica sulla logistica, per poi focalizzarsi sulle principali caratteristiche di un sistema logistico, infine di presentano le categorie principali dei magazzini industriali.

1.1 Introduzione alla logistica

Con il termine **logistica** si intende la pianificazione e il controllo dei flussi di materiale e informativi nelle organizzazioni. L'*obiettivo* della logistica è portare beni o servizi nel giusto luogo, con tempi certi e nelle condizioni volute, a costo minimo e a profitto maggiore. È una delle chiavi dell'economia moderna.

L'espansione dei mercati industriali a livello globale ha generato una forte competizione e, allo stesso tempo, la disponibilità di prodotti alternativi ha creato un tipo di clientela molto esigente, che richiede continuamente la disponibilità istantanea di nuovi modelli. Ai fornitori delle attività logistiche, quindi, è richiesto di svolgere più transazioni, in meno tempo, a costi minori e con grande accuratezza; per di più, la tendenza a una personalizzazione di massa dei prodotti porterà a una intensificazione di questo tipo di domanda.

I ritmi accelerati e l'ambito maggiore su cui le operazioni logistiche deve agire hanno portato e portano tuttora ad un cambiamento nel modello di pianificazione logistica aziendale.

Data la quantità di denaro coinvolto e l'incremento dei requisiti operativi, la pianificazione e il controllo dei sistemi logistici ha attirato maggior attenzione sia in ambito lavorativo che accademico. Per massimizzare il valore in un sistema logistico, vi sono un gran numero di decisioni di pianificazione da prendere, che vanno dalla scelta del sistema di stoccaggio, a quale elemento prelevare in successione per riempire l'ordine di un cliente fino alle decisioni aziendali di alto livello di investire sulla costruzione di un nuovo impianto.

Esiste una grande quantità di letteratura e di strumenti di supporto software che si focalizzano su ogni singolo ambito dei sistemi logistici. Da questa premessa, si può comprendere l'importanza della catena logistica nel mondo attuale e come le aziende siano interessate a possibili miglioramenti in uno qualunque dei suoi tanti aspetti.

1.2 Il sistema logistico

Il *sistema logistico* è l'insieme delle infrastrutture, delle attrezzature, delle risorse e delle politiche operative che permettono il movimento del flusso delle merci e delle relative informazioni, dall'acquisizione delle materie prime e dei materiali ausiliari attraverso la produzione, fino alla distrubuzione dei prodotti finiti ai clienti [3].

Le tre attività principali di cui si occupa ogni sistema logistico sono:

- **processamento degli ordini**;
- **gestione dello stoccaggio**;
- **trasporto della merce**.

Le prime due hanno come obiettivo di garantire la fruibilità del prodotto o bene, in modo che sia disponibile quando richiesto (**valore temporale**) e la terza ne deve garantire l'accessibilità (**valore spaziale**).

Si noti che ben il 10% del costo del prodotto è influenzato dall'aspetto logistico; al suo interno, il costo del trasporto incide solamente per 1/3! Gli altri costi più rilevanti sono relativi allo stoccaggio, all'inventario e all'amministrazione [2].

Come detto, la logistica si occupa di **integrare** tutte quelle attività di fornitura, gestione magazzino, gestione trasporti, eccetera, che contenute in un'azienda, con lo scopo di portare maggior valore temporale e spaziale. Per fare ciò, essa si serve di una **supply chain managament**.

1.2.1 Supply chain management

La **supply chain managament** (*SCM*) è un complesso sistema logistico che definisce *l'integrazione di tutte le attività associate con il flusso e la trasformazione di beni, dalle materie prime all'utente finale, allo stesso modo dei flussi informativi, mediante il miglioramento delle inter-relazioni presenti, per far raggiungere un vantaggio sostenibile e competitivo all'azienda* [1]. Rappresenta un complesso sistema logistico nel quale le materie prime sono trasformate in prodotti finiti da distribuire agli utenti finali. Un possibile percorso di questo tipo è rappresentanto in figura 1.1: i materiali grezzi sono portati dai fornitori (*supplier*) agli impianti di produzione (*manufacturing plant*), che creano componenti o parti semi-lavorate, mentre i prodotti finiti vengono assemblati in uno stabilimento differente (*assembly plant*). Anche la distribuzione è suddivisa in due passi: due centri di distribuzione centrali (*CDC*) che riforniscono diversi centri di distribuzione regionali (*RDC*). Ovviamente, al seconda del tipo di prodotto e di domanda, verrà realizzata la catena più adatta, eliminando o aggiungendo impianti e collegamenti fra le varie fasi. Questi ultimi, inoltre, possono essere più o meno complessi e andare da una semplice linea di trasporto con camion a una rete più articolata che coinvolge diversi sistemi e compagnie di trasporto.

Gli ambiti aziendali coinvolti dalla *supply chain* sono vari: *marketing*, *vendite*, *ricerca e sviluppo*, *previsioni*, *produzione*, *acquisti*, *logistica*, *sistemi informativi*, *finanza*, *servizio cliente*.

I flussi principali che la *supply chain* deve gestire sono riguardanti: *prodotti* (in entrata e in uscita), *servizi* (in entrata e in uscita),

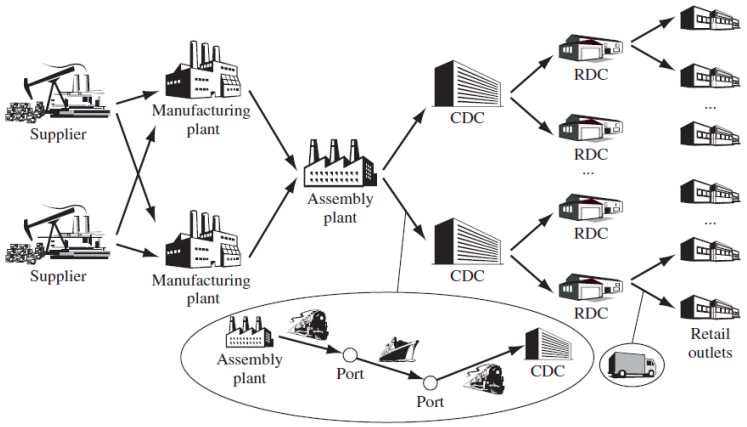

Figura 1.1: Una esempio di supply chain (fonte: Ghiani, Laporte, Musmanno)

informazioni (in entrata e in uscita), *risorse finanziarie* (in entrata e in uscita), *domanda* (in entrata), *previsioni* (in uscita).

Tutto ciò per raggiungere gli obiettivi desiderati a proposito di:

- *soddisfazione del cliente*;
- *valore*;
- *redditività*;
- *vantaggio competitivo*.

La *SCM* non è sempre stata quella odierna, ma si è evoluta nel tempo. Negli anni '60, nell'ambito dell'industria, c'erano uffici frammentati per le diverse attività da svolgere; nei seguenti 40 anni, si è andati verso un'integrazione della logica, per aree di competenza (stoccaggio, produzione, trasporto), dove ogni area è progettata a seconda delle proprie peculiarità. Questo ha portato a un miglioramento nell'efficienza del processo produttivo e a un contenimento dei costi, in quanto un singolo ambito tiene conto di più aspetti nel processo di decisione, interagendo con gli altri. Si crea la logistica *inbound* (per flussi in entrata) e *outbound* (in uscita) dell'azienda.

1.2.2 Caratteristiche di un sistema logistico

Ci sono varie dimensioni che caratterizzano un sistema logistico; di seguito vengono analizzate le più comuni:

1. Sistemi **pull** o **push**:

   - **Push** (*make-to-stock*): il sistema è orientato alla produzione. Si ha la struttura distributiva tradizionale a diffusione capillare sul territorio: dal sistema produttivo ai depositi centrali, passando per quelli periferici, fino ai punti vendita. È prioritaria la capacità di stoccaggio rispetto alla distribuzione. La produzione è basata sulle previsioni e la domanda è anticipata; si hanno alti costi di magazzino e più bassi costi di trasporto, in quanto si possono aggregare maggiormente i flussi delle forniture, rispetto ai sistemi *pull*.
   - **Pull** (*make-to-order*): i prodotti sono realizzati solo su richiesta dei clienti; il magazzino del produttore è basso o nullo e si hanno alti costi di trasporto. Questa logica di sistema è stata ideata parallelamente all'evoluzione della SCM; per cercare un vantaggio competitivo, mediante l'eccellenza del servizio con bassi livelli di scorte, i magazzini centrali si trasformano in *centri di distribuzione* (*CEDI*), e diminuiti di numero; i magazzini periferici si trasformano in *punti di transito*, che operano solo smistamento e distribuzione. Si ha una struttura distributiva snella. Le informazioni risalgono a ritroso, dai punti vendita ai punti di transito, fino ai *CEDI*, gli ordini vengono consolidati e poi spediti. Viene data la priorità alla capacità di smistamento e distribuzione.
   - **Mixed** (*make-to-assembly*): è un sistema misto fra i due; si accumulano parti di approvvigionamento (*push*), per poi essere efficienti nella distribuzione ai clienti (*pull*) a fronte di un ordine.

2. Supply chain a **integrazione verticale** o con **logistica di terze parti**:

- **Integrazione verticale**: in questo caso tutti i componenti (sorgenti di materie prime, stabilimenti, trasporti) appartengono alla stessa azienda. Più frequentemente, diverse e indipendenti compagnie operano in una supply chain. Vantaggioso in un mercato stabile.
- **Logistica di terze parti**: parte della logistica viene affidata a terzi (*i.e.*, operatori specializzati). In questo caso, l'azienda ha un basso o nullo costo di investimento su infrastrutture e si riducono gli effetti negativi dei periodi con bassa domanda, ma rischia la perdita di contatto con il cliente e i costi logistici unitari sono in genere maggiori.

3. Magazzino a **gestione del dettagliante** o a **gestione del venditore**:

   - **A gestione del dettagliante**: egli controlla il suo magazzino e decide quando e quanto rifornirsi; in questo caso si ha una logica di tipo *pull* per la distribuzione.
   - **A gestione del venditore**: egli monitora i magazzini dei clienti rivenditori e decide quando e quanto rifornirli; la distribuzione a valle è decisa dal punto a monte. Questa modalità si è sviluppata solo di recente ed è realizzabile in accordo fra le due parti o in caso di forte leadership del mercato del venditore. Quest'ultimo ha maggiori risparmi dovuti ad una miglior coordinazione della logistica, mentre i clienti non sono tenuti ad investire nel controllo del magazzino.

Come si osserva nella figura 1.2, i prodotti seguono il flusso della *supply chain* (eccetto per quelli obsoleti, danneggiati o non funzionanti che devono ritornare alla sorgente), al contrario le informazioni seguono il percorso inverso, dai consumatori ai fornitori. In un sistema *pull*, ad esempio, gli ordini fatti dai clienti finali vengono presi in carico dai venditori e trasmessi al produttore, che li trasforma in ordini per i fornitori. Analogamente, nei sistemi *pull*, le vendite storicizzate sono usate per fare previsioni future sulla domanda di prodotto e la relativa richiesta di materiali.

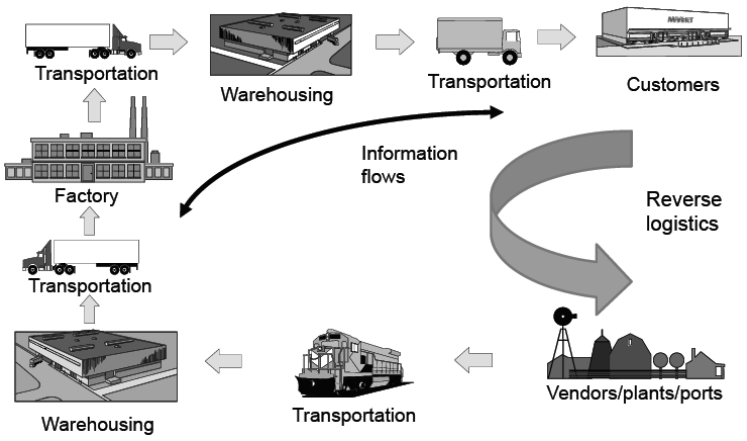

Figura 1.2: Flusso dei prodotti e delle informazioni (fonte: Ballou, 2004)

Facendo riferimento al caso reale in *Orogel*, un'azienda manufatturiera che lavora sul prodotto fresco, opera in modalità *push*. Lo stoccaggio dei prodotti ha un forte impatto sui costi, necessitando la presenza di diverse celle di immagazzinamento, presenti sia a gestione manuale, sia automatica. Ognuno di questi impianti può arrivare a contenere migliaia di posti pallet, soprattutto grazie alla grande capienza fornita da quelli a movimentazione interna automatica, da ciò se ne deduce che, nel caso della linea di prodotti surgelati, il consumo di energia risulta particolarmente oneroso, dovendo mantenere temperature di gran lunga sotto lo 0. Il ritmo della produzione, ovvero la trasformazione del prodotto fresco in surgelato, è dettato dal calendario dei raccolti di campagna, mentre le operazioni di confezionamento sono basate sulle previsioni di vendita.

1.2.3 Funzionamento di un sistema logistico

Riprendendo le tre attività principali facenti parte di un sistema logistico accennate prima, si descrive, in breve, il suo funzionamento.

Fase 1 - gestione degli ordini: questo compito è a grande dispendio di tempo (fino al 70% dell'intero tempo di ciclo dell'ordine [5]); qui si collega il flusso delle informazioni con il flusso dei prodotti. A fronte di una richiesta, se ne verifica la disponibilità, si produce o estrae dal magazzino il prodotto e infine si spedisce, mantenendo le informazioni sullo stato dell'ordine a disposizione del cliente. Ad ogni modo, il ricorso a supporti elettronici e informatici hanno accelerato di molto questa fase, nel corso degli ultimi anni.

Fase 2 - gestione dello stoccaggio: riguarda l'immagazzinamento sia delle materie prime da manifatturare, che dei prodotti semilavorati (o componenti da assemblare) e dei prodotti finiti da vendere. I magazzini sono utili per gestire la richiesta di tipo stagionale o tamponare quella casuale di un prodotto, per migliorare il livello di servizio (riducendo i tempi di consegna, se il magazzino è più vicino al bacino della clientela), per sfruttare le economie di scala nel trasporto. In una logica di tipo *push* si può reagire alla variabilità della domanda e tamponare eventuali inefficienze a monte. D'altro canto, il mantenimento di un magazzino può risultare molto costoso per varie ragioni: la proprietà incorre in un costo di opportunità, rappresentato dal ritorno di investimento che avrebbe realizzato se il denaro fosse stato meglio investito. Secondo, deve essere sostenuto il costo della struttura, privata, pubblica o in affitto che sia. L'obiettivo della gestione di magazzino è determinare il livello di stoccaggio per minimizzare i costi operativi totali, soddisfacendo i requisiti di livello di servizio dei clienti.

Fase 2/3 - strategie di stoccaggio/trasporto: le politiche di stoccaggio e di trasporto sono fortemente collegate. Come viene rappresentato in figura 1.3, ce ne possono essere tre tipologie principali:

- **Spedizione diretta** dal produttore all'utente finale. Il *lead-time* (*i.e.* l'intervallo di tempo fra il momento dell'ordine e di arrivo del prodotto) è ridotto e si risparmia sui costi dell'utilizzo di un magazzino. Non è conveniente se le spedizioni sono piccole e i clienti sparsi, perché si avrebbe una flotta di veicoli

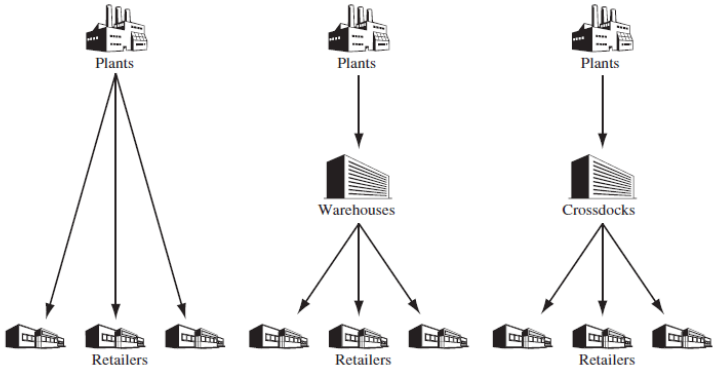

Figura 1.3: Strategie di stoccaggio/trasporto (fonte: Ghiani, Laporte, Musmanno)

semivuoti. È comune in caso di richieste di camion pieni o per beni degradabili che non consentono aggregazione.

- **Immagazzinamento**: strategia intermedia. Il magazzino riceve i beni in ingresso, li stocca, e a fronte di ordini li spedisce. Se ne può avere uno singolo centralizzato oppure vari regionali decentralizzati. Con questa strategia si riduce il *lead-time*, ma aumentano i costi di gestione.

- **Crossdocking**: avviene la cosiddetta distribuzione *appena in tempo*, dalla fabbrica al magazzino e da esso al cliente. Il magazzino diventa una piattaforma logistica nella quale i prodotti rimangono per tempi brevi; le merci vengono trasferite da trasporti sulla lunga distanza a trasporti sulla corta distanza. Si ha una minor gestione dello stoccaggio, ma si introduce la complessità di sincronizzare le fasi di ingresso e uscita. Questa strategia richiede volumi di beni elevati, a bassa variabilità, e facilmente maneggiabili, con la coordinazione dei flussi che avviene mediante un supporto informativo.

Fase 3 - trasporto della merce: il trasporto delle merci gioca un ruolo chiave nell'economia odierna. Consente, infatti, che il consumo avvenga lontano dalla produzione, espandendo il raggio d'azione

dei mercati, stimolando la competizione diretta fra aziende produttrici di paesi diversi e incoraggiando le compagnie a sfruttare le economie di scala. Si ha, così, la disponibilità globale di certi prodotti e si migliora la fruibilità dei beni degradabili. Il trasporto può avvenire con mezzi privati, con terzisti che lavorano in esclusiva per l'azienda oppure con corrieri. Di seguito, se ne analizzano le principali caratteristiche:

- **Canale di distribuzione**: identifica il percorso seguito dal prodotto, con la presenza di possibili intermediari tra il produttore e il cliente (broker, grossista e dettagliante). In genere, i *beni industriali* sono distribuiti al massimo mediante i primi due intermediari, mentre i *beni di consumo* possono utilizzare anche il terzo.

- **Aggregazione della merce**: l'aggregazione può avvenire mediante un'apposita *infrastruttura* (consolidando spedizioni singole in certi nodi della rete di trasporto), in modalità *multi-stop* (le spedizioni singole verso la destinazione finale passano per un percorso che può servire diversi clienti) oppure con consolidamento *temporale* (le spedizioni vengono ritardate o anticipate per riuscire a spedire grandi quantità).

- **Modalità di trasporto**: può avvenire con aereo, camion, treno, nave, condutture o in maniera inter-modale. La scelta dipende dal tipo di merce (se consolidata in *pallet* o *container*), dallo stato del materiale (solido, liquido) e dalla deperibilità. Bisogna inoltre considerare i tempi di consegna che si desiderano rispettare, le distanze da percorrere e i costi del tipo di trasporto.

1.2.4 Gestire un sistema logistico

Ci sono vari obiettivi da tenere presente nella definizione di una strategia logistica, di cui i principali sono: la *riduzione degli investimenti* da compiere (scegliendo, ad esempio, fra logistica interna o esternalizzata), la *riduzione dei costi operativi* e il *miglioramento del livello di servizio*. Quest'ultimo dipende dal *lead-time* e può essere:

- *molto breve*, se è stato richiesto un prodotto disponibile al dettaglio,
- *breve*, se il prodotto deve essere ordinato dai centri di disribuzione regionali o dal magazzino centrale, e
- *lungo*, se il prodotto è terminato e deve ancora essere creato.

Il *lead-time*, o **order cycle time**, viene rappresentata come una variabile casuale con distribuzione di probabilità multinomiale, basata sui tre eventi di cui sopra.

È necessario scegliere un *compromesso fra costi e i livello di servizio*, tenendo presente che vi è anche una *relazione tra quest'ultimo e le vendite*. Il **livello di servizio ottimale** è quello che *garantisce il massimo delle entrate*, nel punto in cui è massima la differenza fra vendite e costi.

Ci sono tutta una serie di decisioni da intraprendere nel progetto di un sistema logistico e che riguardano: strutture, materiali, modalità, rifornimenti, trasporti, eccetera. Esse si possono raggruppare in 3 principali macro-aree:

- **Decisioni strategiche**: hanno un orizzonte di diversi anni, basate su dati imprecisi o incompleti, e sono prese dai vertici aziendali. Esse riguardano la dimensione, la configurazione e la locazione delle strutture.
- **Decisioni tattiche**: hanno un orizzonte temporale fino al massimo di un anno, basate su dati disaggregati e sono prese dai dirigenti di medio livello. Esse riguardano l'allocazione delle risorse e la pianificazione della produzione e della distribuzione.
- **Decisioni operative**: hanno un orizzonte di giorni, basate su dati precisi e prese dai dirigenti di basso livello. Esse riguardano, ad esempio, la presa in carico di ordini e lo smistamento dei veicoli.

In quest'ambito, si possono utilizzare dei supporti alle decisioni, come, ad esempio, **l'analisi quantitativa**. Essa è essenziale per prendere decisioni intelligenti e può essere svolta in diversi modi:

mediante il *benchmarking* (si valuta un sistema esistente comparato ad uno standard industriale), la *simulazione* (valutazione di alternative specifiche applicate ad un modello) e *l'ottimizzazione* (di una certa configurazione rispetto a un valore di prestazioni).

Come è immaginabile, data la vastità e l'importanza che l'argomento ricopre tuttora, esistono molti pacchetti software nell'area della logistica, ma la loro trattazione esula dallo scopo di questa tesi.

1.3 I magazzini industriali

Come descritto sopra, il sistema logistico comprende anche i sistemi di stoccaggio e di distribuzione, per congiungere produttori e consumatori. Il magazzino, quindi, eroga un *servizio logistico* (rendere disponibile il prodotto nel posto, al momento e al costo giusto), collocato nella *supply chain*. Il *sistema distributivo* è formato da due strutture distinte:

- il *canale logistico*: rappresenta la rete distributiva per il concentramento, la selezione, lo smistamento e il trasporto delle merci. Di queste funzioni, le prime tre sono svolte nei magazzini.

- il *canale commerciale*: costituito dalle strutture atte alla vendita del prodotto.

Il magazzino può essere visto come *contenitore* e *trasformatore* dei flussi di materiali che riceve, operando sia sulla quantità che sulla quantità dei beni. Esso è importante sia in *ottica temporale*, in quanto consente di realizzare economie legate ad approvvigionamenti e di ottimizzare la produzione, sia in *ottica qualitativa*, rendendo possibile il consolidamento dei carichi, con conseguente riduzione dei costi di trasporto.

I magazzini vengono principalmente suddivisi in base alla posizione in cui sono collocati nella supply chain. Ci sono, dunque, i magazzini **di produzione** e i magazzini **di distribuzione**. Nei primi si hanno grandi quantità gestite e sono suddivisi in magazzini di *materie prime* (provenienti dai fornitori), *interoperazionali* (posti fra

due fasi del processo produttivo, contenenti prodotti semi-lavorati o componenti) e di *prodotti finiti* (con prodotto in attesa di essere venduto). I magazzini di distribuzione, invece, hanno una grande varietà di prodotti e sono suddivisi in *centralizzati* o *periferici/regionali*, classificati a seconda della dimensione del proprio bacino di utenza:

- *Magazzini centralizzati*: consentono un immagazzinamento di grandi quantità di materiali, con l'obiettivo di massimizzare l'efficienza nella ricezione, stoccaggio, estrazione e spedizione dei beni. Si hanno flussi separati per ogni linea di prodotti.

- *Magazzini regionali*: questi centri sono più vicini alla distribuzione finale. Si hanno grandi quantità in ingresso, ricevute e immagazzinate, mentre le spedizioni sono più piccole e assemblate. Si formano gli ordini in maniera coordinata fra diverse attività, con prodotti provenienti da diversi flussi (con la necessaria sincronizzazione).

L'unità di movimentazione del bene (anche detta **unità di carico** o semplicemente *UdC*), che spesso è un pallet o un altro tipo di contenitore, può dipendere dal cliente o dal livello logistico in cui ci si trova. L'ordine, di solito, riguarda UdC di diversi prodotti.

Inoltre, bisogna tenere presente che nel **costo** del magazzino, la manipolazione della merce incide, all'incirca, per il 50%, mentre la ricezione, la spedizione, lo stoccaggio incidono per il restante 50%, in parti uguali. La spedizione e la ricezione sono fasi difficili da automatizzare, mentre la manipolazione dipende dalla tecnologia di immagazzinamneto e recupero usata.

1.3.1 Gestione del magazzino

Per ogni punto di stoccaggio, lungo la supply chain, è necessario dicedere la quantità e il periodo delle ordinazioni di prodotti, in modo da minimizzare il costo totale nel raggiungimento di un certo livello di servizio. I **costi più rilevanti** riguardano:

- *approvvigionamento*, che comprende i costi di processamento degli ordini, di acquisto, di lavoro e di trasporto dei prodotti;

- *mantenimento* dei magazzini;
- *perdite* a causa di vendite mancate o di ritorno degli ordini;
- *obsolescenza*.

La gestione del magazzino può essere classificata in base a diversi parametri, come, ad esempio, l'utilizzo di modelli *deterministici* oppure *stocastici* (nei quali la domanda non è sempre soddisfatta), la movimentazione *veloce* o *lenta* dei materiali, a domanda *statica* o *dinamica* nel tempo, con punti di stoccaggio *singoli* o *multipli* (più difficile) e a rifornimento *istantaneo* o *non istantaneo*.

A seconda delle assunzioni scelte che meglio si adattano alle esigenze aziendali, fra le alternative elencate sopra, si utilizzeranno modelli diversi, più o meno complessi, per gestire la logistica nel magazzino.

1.3.2 Struttura interna

La struttura interna di un magazzino riflette le operazioni che devono essere svolte al suo interno, mirando a minimizzare i costi di movimentazione; essa dipende da tre fattori primari:

- le caratteristiche fisiche dei prodotti da stoccare;
- il numero dei prodotti da immagazzinare;
- i volumi di prodotto manipolati in ingresso e in uscita.

Tipicamente, in ogni magazzino ci sono:

- una o più *zone di ricezione* dove la merce in ingresso è scaricata e controllata;
- una *zona di immagazzinamento* dove le unità di carico sono stoccate;
- una o più *zone di spedizione* dove sono assemlati gli ordini dei clienti e caricati i veicoli in uscita.

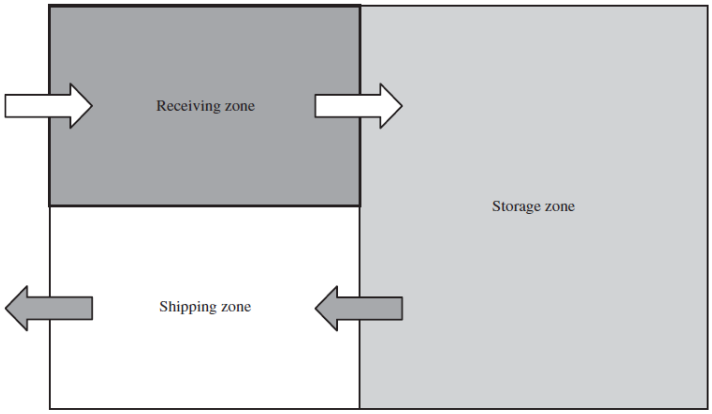

Figura 1.4: Magazzino con una zona di immagazzinamento (fonte: Ghiani, Laporte, Musmanno)

Nel magazzino di figura 1.4, la quantità di merce in entrata è pressoché simile alla quantità di merce in uscita. A volte, invece, in magazzini in cui la rotazione delle merci è più elevata, la zona di immagazzinamento è suddivisa in ingresso in una *zona di riserva*, che riceve grandi quantità di imballo, le quali vengono separate in unità di lavoro e in uscita in una *zona di inoltro*, dove si ha una suddivisione più fine.

Se ne può vedere un esempio in figura 1.5.

1.3.3 Mezzi di immagazzinamento

La scelta della tipologia del mezzo di immagazzinamento da adottare deriva fortemente dalle caratteristiche del bene da stoccare e dal numero medio di elementi di ogni prodotto che compongono l'ordine di un cliente. Per ogni tipologia si valutano tre parametri di riferimento, che sono:

- la **ricettività** della merce, ovvero la capacità totale di stoccaggio,
- la **selettività**, che è pari al numero delle UdC direttamente accessibili diviso la ricettività,

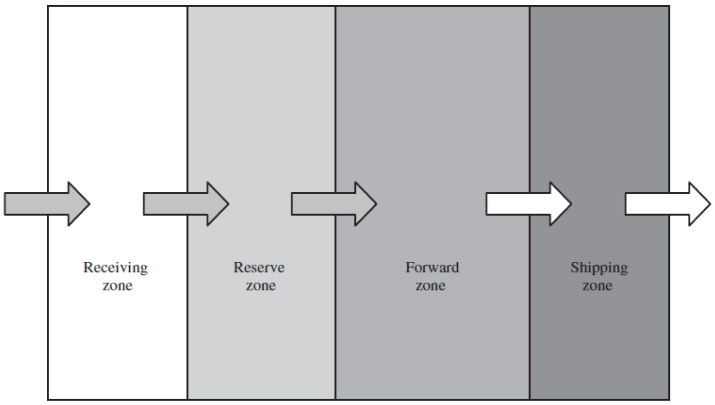

Figura 1.5: Magazzino con zona di riserva e zona di inoltro (fonte: Ghiani, Laporte, Musmanno)

- e la **potenzialità di movimentazione**, cioè il numero di UdC movimentabili in un'ora.

Considerando lo stoccaggio di beni di tipo solido, le alternative principali di immagazzinamento sono tre:

- **Pile**: gli elementi sono posti in cartoni o pallettizzati e organizzati in pile (o cataste), ovvero stoccando le UdC una sopra l'altra. Questo metodo necessita un investimento di capitali pressoché nullo ed è adatto per immagazzinare beni a bassa domanda. Si ha un alto coefficiente di sfruttamento volumentrico e superficiale dello spazio disponibile e il layout è altamente riconfigurabile, data l'assenza di strutture fisse. D'altra parte, l'organizzazione a cataste è caratterizzata da una bassa selettività (dato che solo le UdC poste in cima sono immediatamente accessibili) e da una ridotta potenzialità di movimentazione.
- **Scaffali**: in questo caso gli elementi da stoccare sono posti in scaffali metallici, mediante scatole o pallet. Essi hanno il vantaggio di poter stoccare anche unità di carico non sovrapponibili, di consentire un accesso più facile agli elementi contenuti e di poter allocare contenitori anche eterogenei nella forma. Un

esempio di questo sistema si può vedere nella figura 1.6. Inoltre, l'impiego degli scaffali dà la possibilità di automatizzare i processi di deposito e recupero, mediante apposite macchine descritte in seguito, ottimizzando lo spazio disponibile nella struttura. In tal modo, infatti, è possibile raggiungere un'altezza massima più alta rispetto alla modalità di stoccaggio a pila e restringere la larghezza dei corridoi, aumentando la capacità totale di stoccaggio. Per contro, essi richiedono un investimento di capitali superiore rispetto alle altre tipologie. I magazzini a scaffalatura possono essere di tipo **drive in** o **drive through**; la differenza fra i due è che nel primo caso si ha una gestione **LIFO** (*last in first out*) delle merci, in quanto l'immissione e l'estrazione delle unità di carico avvengono dallo stesso lato dello scaffale, mentre nel secondo si ha una gestione **FIFO** (*first in first out*) delle merci, quindi le unità di carico vengono immesse da un lato dello scaffale e prelevate dall'altro. La selettività di questa tipologia rimane comunque bassa; per aumentarla, si può fare ricorso a scaffalature di tipo *bifrontali*, che raddoppiano l'accessibilità agli elementi. Fino adesso si sono considerati solo magazzini **statici**, nei quali è solo l'operatore a muoversi e a movimentare le merci al suo interno; i magazzini **dinamici** estendono questa capacità anche alla struttura e quindi al materiale. Alcuni esempi sono rappresentati dagli *scaffali traslanti* (o *magazzino compattabile*), dal *live storage* e dai *canali in contropendenza*. I primi consentono un'elevata utilizzazione superficiale e volumetrica, ma è importante considerare il tempo necessario allo scorrimento delle scaffalature e della selettività ridotta. I magazzini live storage forniscono una gestione *LIFO* delle unità di carico, mediante scaffalature inclinate costituite, in genere, da un piano di scorrimento a rulli. Infine, i canali in contropendenza consentono una gestione *FIFO* delle unità. Questi ultimi due metodi di immagazzinamento sono convenienti se si considera la selettività a livello di gruppo e non di singola unità di carico, perché, in genere, ad ogni canale corrisponde una linea d'ordine; in caso contrario la selettività risulta piuttosto bassa.

Figura 1.6: Un sistema di immagazzinamento a scaffali (fonte: Ghiani, Laporte, Musmanno)

- **Cassettiere**: questa tipologia consiste nel disporre di piccoli scaffali o cassettiere, per stoccare elementi di piccole dimensioni con semplice accesso da parte dell'uomo. In genere vengono collocati nell'area di spedizione, nella quale si effettuano operazioni di *picking*. Anche in questo caso, si può optare per due ulteriori categorie di stoccaggio, con magazzini statici, anche detta **operatore verso materiali** o *picker-to-product*, che utilizzano i media di stoccaggio appena presentati, o con magazzini dinamici, anche detta **materiali verso operatore** o *order-to-picker*. Questi ultimi sono più convenienti quando si deve assicurare una potenzialità di movimentazione elevata e quando gli oggetti non hanno dimensioni particolarmente grandi. Con l'utilizzo di **caroselli** o **minitraslo** si eliminano i tempi persi nello spostamento dell'operatore, che costituiscono la maggior parte del tempo totale di prelievo nei magazzini statici.

1.3.4 Approcci per stoccaggio e recupero merce

I magazzini sono spesso classificati in base al metodo utilizzato per stoccare e recuperare la merce. La scelta del tipo di accesso da effettuare è molto importante, in quanto da essa dipende la definizione ottimale del layout su cui modellare il magazzino, basata sull'**obiettivo di minimizzare i tempi delle operazioni**. La gestione manuale, ad esempio, richiede spazi di manovra maggiori rispetto alla gestione automatizzata. Come accennato, si possono avere sistemi *picker-to-product*, dove gli ordini sono recuperati e assemblati da operatori umani, o *order-to-picker*, i quali automatizzano il processo. A partire da questi due, sono possibili ulteriori classificazioni. Un primo esempio è costituito dalla versione ibrida, vale a dire dai sistemi *picker-to-belt*, dove gli elementi sono recuperati da uomini e poi, mediante un nastro trasportatore, giungono alla fase di assemblaggio. Oppure, nei sistemi *picker-to-product*, gli operatori possono viaggiare alle locazioni di stoccaggio mediante dispositivi automatici, che in genere ricoprono un singolo corridoio (quindi, al massimo due lati, appartenenti a due scaffali diversi); essi prendono il nome di *person-aboard automated storage/retrieval systems* (*AS/RSs*). Nel caso in cui gli operatori si spostino a piedi o mediante carrelli, potendo visitare più corridoi, il sistema prende il nome di *walkride and pick systems* (*W/RPSs)*.

I sistemi *order-to-picker* più popolari sono gli *AS/RSs* (di cui si illustra una sua rappresentazione in figura 1.7; questi **magazzini automatici** consistono in una serie di corridoi di stoccaggio, ognuno dei quali è servito da una singola macchina di deposito e prelievo (e.g., **trasloelevatore**). Ogni corridoio, inoltre, è dotato alla fine di una *stazione di carico e trasporto*, accessibile in comune con il sistema di gestione esterno (**rulliera**, **nastro trasportatore** o **navetta**).

1.3.5 Magazzini automatici

I magazzini automatici rientrano nell'approccio per stoccaggio e recupero merce di tipo *order-to-picker*. In *Orogel* si sono potuti analizzare diversi impianti automatici con molteplici sistemi di trasporto. Primo fra tutti, si è potuto osservare un **trasloelevatore** all'opera:

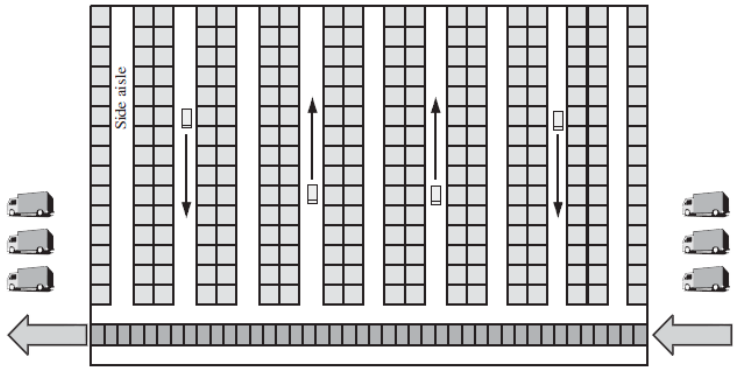

Figura 1.7: Rappresentazione di un AS/RS (fonte: Ghiani, Laporte, Musmanno)

posto in un corridoio, esso può agire lungo i due lati di due scaffali diversi, avendo la capacità di spostarsi lungo tre assi, in altezza, larghezza e profondità. La comunicazione fra i vari corridoi avviene mediante una o più **navette**, dotate di un sistema di traslazione che permette loro di spostarsi lungo una sola direzione; inoltre, lo scambio di materiale fra i trasloelevatori e le navette, avviene tramite apposite rulliere che fungono da buffer, per rendere *asincrona* l'interazione fra le due macchine. Data l'altezza di questi impianti, spesso sono suddivisi in piani, ognuno dei quali può interagire con l'esterno mediante appositi ingressi e uscite; per spostarsi in altezza, vengono utilizzati degli **elevatori**.

Un altro tipo di sistema di movimentazione visto in funzione, in una cella automatica più piccola, è formato da una coppia di **carroponti**: essi, con il meccanismo di moto fissato al soffitto, sono in grado di movimentare la merce dall'alto, spostandosi lungo tre assi. In tal modo, riescono ad avere un raggio d'azione sull'intera superficie della cella, con l'unico vincolo di non sovrapporsi.

Questi sistemi riducono il tempo di percorrenza (che costituisce circa il 70% del tempo della gestione manuale dei prelievi nel magazzino [2]), sono caratterizzati da bassi costi di esercizio, assicurando elevate prestazioni per quel che riguarda potenzialità di movimentazione, ricettività e **tracciabilità** (la possibilità di avere il

controllo su tutti i movimenti riguardanti una certa unità di carico). La gestione automatica tramite computer ottimizza le variabili di processo, come ad esempio l'ordine con il quale svolgere le operazioni di prelievo o deposito e i percorsi da compiere. Per costruire un tale sistema, però, è necessario sostenere un elevato costo riguardo la struttura del magazzino e la struttura di movimentazione e di controllo. Tali spese, perciò, sono da considerarsi accettabili solo a fronte di requisiti elevati su ricettività e potenzialità, valutando anche il fatto che una struttura di tale tipo è fortemente vincolata e non consente economiche riconfigurazioni.

In *Orogel* si è potuto osservare come funzioni quali la **giacenza puntuale**, la **tracciabilità del prodotto** e lo **stoccaggio automatico** siano svolte interamente via software, con la supervisione di diversi addetti e operatori, mediante un apposito *warehouse management system* (**WMS**), chiamato *On.Plant*. Oltre a questo, inoltre, ogni impianto automatico dispone di un software a sé stante che consente la supervisione dello stesso, controllando la correttezza delle operazioni di movimentazione degli elementi e che permette, in caso di problemi, un tempestivo intervento da parte degli operatori. Ad ogni pallet in ingresso viene associata una precisa locazione di deposito, calcolata con appositi algoritmi, che tengono conto, ad esempio, della conformazione, del tipo di articolo, eccetera. Una volta portato all'entrata dell'impianto di destinazione (o manualmente da un carrellista oppure mediante altri impianti automatici di trasporto), esso, comandato dal software, si attiva per condurre l'elemento nel punto di deposito assegnato. Analogamente, in caso di vendita, tutti gli articoli necessari sono portati all'uscita automaticamente, dopodiché un operatore, servendosi di un carrello elevatore, si occupa di caricare i pallet sul camion.

Si è notato come questo tipo di gestione automatizzata del magazzino permetta di **minimizzare i percorsi** da compiere al suo interno, **massimizzare gli spazi** a disposizione, poter gestire in maniera agevole la **scadenza** dei prodotti, il tutto rispondendo tempestivamente alle esigenze di ricezione e spedizione delle merci

Inoltre, in *Orogel* si è visto come gli impianti automatici non sono solo utilizzati per la movimentazione interna, ma anche per mettere in comunicazione i vari impianti di stoccaggio fra loro. È

presente, infatti, un sistema **automotore** che collega due stabilimenti (collocati su due lati opposti di una strada) e che trasporta pallet da una struttura ad un'altra. Esso è costituito da parti fisse (*guide*) e da parti mobili (*scambi*) che analogamente ad una ferrovia guidano e alimentano altri componenti mobili (*bilancelle*) lungo tutto il tragitto da compiere. Un software consente di controllare lo stato dell'intero impianto e di inviare comandi alle bilancelle.

In questo ambito è stato possibile aumentare personamente l'esperienza diretta, grazie ad attività in *Onit* che sono seguite al tirocinio. Il coinvolgimento in prima persona nell'implementazione del software di supervisione di nuovi impianti automatici per il transito della merce e di interfacciamento con l'automotore, ha permesso di conoscere e di comprendere più a fondo l'ampio dominio dell'automazione industriale.

1.3.6 Criteri di allocazione dei prodotti

L'allocazione dei prodotti nell'area di stoccaggio è una delle *decisioni operazionali* più importanti, in quanto influisce sulla *pianificazione dei depositi e dei prelievi*. Queste due fasi, infatti, sono da effettuare in *real-time*; in un ambiente dinamico quale il magazzino, il sistema non dispone a priori di tutte le informazioni, formate anche da una componente casuale, riguardanti gli ordinima le ha solo al momento.

Per ottimizzare le movimentazioni è necessario, dunque, progettare adeguatamente le aree di stoccaggio, seguendo precisi criteri per quel che riguarda l'allocazione dei prodotti nei media di immagazzinamento; a tale scopo, si valutano le scelte basandosi su vari *parametri di movimentazione*:

- *Indice di rotazione*: indica il grado di sostituzione delle scorte su un certo periodo di tempo. Dato un generico prodotto i, l'indice di rotazione è definito da:

$$IR_i = \frac{Fu_i}{\bar{G}_i} = \left[\frac{1}{T}\right]$$

  dove Fu_i è il flusso in uscita del prodotto i nel periodo T e $\bar{G}_i$ è la giacenza media del prodotto i. Ipotizzando due prodotti a e

b con $IR_a > IR_b$, si deduce che il prodotto a, dal punto di vista della movimentazione, è più importante b.

- *Indice di movimentazione*: valuta il numero di movimenti effettuati mediamente nel periodo T; per un generico prodotto i:

$$IM_i = \frac{M_i}{T} = \left[\frac{1}{T}\right]$$

dove IM_i è pari alle unità di carico del prodotto i movimentante nel periodo T. Analogamente a prima, un prodotto con IM maggiore di un altro risulta più importante per la movimentazione.

- *Indice di accesso*: calcola il numero di accessi nell'unità di tempo alla cella del magazzino dedicata al prodotto i-esimo. Dato un prodotto i, vale:

$$IA_i = \frac{IM_i}{n°celle_i} = \left[\frac{accessi}{T \cdot vano}\right]$$

dove $n°celle$ è pari al numero di celle dedicate al prodotto i, se esistono, oppure è la giacenza media del prodotto; in quest'ultimo caso vale $IR_i = IA_i$. Esso sintetizza la probabilità di accesso alla cella in questione sul periodo di tempo T.

Sulla base di questi parametri si sceglie, di conseguenza, la **politica di stoccaggio** più adatta, la quale può essere di vari tipi:

- *Dedicata*: si assegna ad ogni prodotto un numero definito di celle dedidate, fissandone la massima potenzialità ricettiva. Questo metodo facilita la tracciabilità e la ricerca, ma si rischia un sottoutilizzo della capacità complessiva. Si segue la disposizione dei prodotti per indice di accesso decrescente, se si vogliono ottimizzare i tempi di percorrenza, oppure per aree merceologiche se si vuole facilitarne la ricerca.

- *Basato su classi*: si suddivide il magazzino in zone dedicate a classi di prodotti e ogni articolo è stoccato (casualmente) nella zona della sua classe. Il tempo medio di accesso si riduce

all'aumentare del numero di classi individuate, ma con esse aumenta anche la complessità gestionale. In genere, viene valutato come 3 o 4 il numero di classi che consentono un buon compromesso [4].

- *Casuale*: questo tipo di politica ha l'approccio opposto a quella dedicata: l'allocazione di un prodotto è decisa dinamicamente, in base all'occupazione corrente e alla previsione della domanda. Questo metodo consente la migliore utilizzazione della capacità e quindi di superficie richiesta, ma il tempo medio di accesso è pari alla media dei tempi di accesso a tutti i vani.

Le ultime due poliche necessitano l'utilizzo di supporti informatici per tenere traccia della locazione e degli spostamenti dei singoli prodotti.

Un ulteriore accorgimento è relativo a come **disporre gli articoli** rispetto ai punti di ingresso e di uscita del magazzino; più vicina è la collocazione dei prodotti a questi punti, infatti, minore è la somma dei tempi di stoccaggio e di prelievo. Perciò risulta più vantaggioso disporre gli articoli a partire dai vani più velocemente accessibili, secondo un ordine decrescente degli indici di accesso.

1.3.7 Movimentazione interna

La movimentazione interna si suddivide nella fase di **deposito** e in quella di **prelievo**. Nella prima, una volta assegnata la posizione ai prodotti, avviene l'allocazione vera e propria; specularmente a questa, nella fase di prelievo, note le ubicazioni degli articoli, si attua l'operazione di **picking**. Generalmente, la merce in ingresso al magazzino presenta una maggior omogeneità, al contrario di quella in uscita: infatti, gli ordini dei clienti dettaglianti coinvolgono, spesso, un insieme di prodotti a maggior eterogeneità rispetto a quelli provenienti dai fornitori. La parte di *picking* risulta, di solito, più dispendiosa di quella di deposito, in termini di locazioni diverse da visitare e quindi di percorsi da compiere, perché richiede l'accesso a più classi diverse di prodotti. Le considerazioni presentate di seguito valgono anche per i depositi, ma si preferisce focalizzarsi

sulla fase di prelievo, ritenuta, per l'appunto, più significativa in termini di complessità.

Ogni ordine di vendita e spedizione, si traduce nella seguente serie di operazioni:

1. la formazione dei **batch**, ovvero l'insieme delle missioni di prelievo,
2. il calcolo dell'ordine di prelievo,
3. l'eventuale schedulazione della macchina,
4. e il caricamento dei veicoli.

Nel primo punto gli ordini sono formati in *batch* (lotti); essi possono essere combinati in un certo numero di modi, a seconda che il magazzino sia suddiviso in zone oppure no. Nel primo stadio si stima la dimensione ottima del batch ($d*$) che bilancia gli sforzi di picking e di ordinamneto. Nel secondo stadio i batch sono creati con la politica *first come first served*, aggregando $d*$ ordini consecutivi. Il dimensionamento dei batch mira a minimizzare il calcolo di lavoro totale, dato dalla somma dei tempi di picking e di ordinamento; il calcolo di $d*$ può avvenire utilizzando modelli più o meno complessi, a seconda dei requisiti.

Nel *W/RPS*, dove gli operatori si spostano a piedi o mediante carrelli motorizzati e visitano più corridoi, il secondo punto è uno dei più dispendiosi fra i quattro, richiedendo più del 70% del tempo [2]. Il routing sulla la scelta e l'ordinamento dei percorsi da prendere fa parte di una classe più grande di problemi di ottimizzazione nota come **VPR** (*vehicle routing problem*), o come **RTSP** (*road travelling salesman problem*) nel caso di operatore singolo. Si possono utilizzare semplici algoritmi euristici per ottenere il compromesso voluto fra tempi di calcolo e qualità della soluzione.

Infine, il carico dei veicoli avviene impacchettando i prodotti, spesso in pallet o container. I primi sono adatti per essere trasportati via camion, mentre i secondi via treno o nave. L'obiettivo, per minimizzare i costi, è rendere minimo il numero di contenitori necessari per racchiudere gli oggetti: ciò si traduce in ulteriori problemi (di tipo geometrico) con più o meno vincoli, ad esempio a seconda

che gli oggetti siano sovrapponibili, e a vari gradi di complessità, in base al numero di dimensioni da prendere in considerazione.

1.4 In sintesi

In questo capitolo si sono volute introdurre le principali tematiche del dominio a cui ci si è trovati di fronte all'inizio dell'attività di tirocinio. Partendo da una panoramica sulla logistica, per poi entrare nelle principali caratteristiche di un sistema logistico e infine terminare con un breve dettaglio sui magazzini industriali, si è notato come le problematiche da affrontare, nella costruzione e nella gestione di un sistema logistico, si estendano a tutti i livelli della catena e riguardino problemi della più diversa natura: si va dalle decisioni strategiche riguardo le strutture logistiche da inserire nella *supply chain*, passando per la modellazione dei layout dei magazzini, alla politica di stoccaggio e prelievo nel singolo impianto di deposito merce. Questa è la premessa su cui si basa l'attività di studio presentata nel capitolo seguente.

Capitolo 2

Pianificazione e ottimizzazione dei percorsi

In questo capitolo si riporta il lavoro di studio e ricerca bibliografica, con la formulazione di una sintesi adatta al problema della pianificazione e ottimizzazione dei percorsi, nell'ambito dei magazzini industriali presentati al capitolo precedente.

Dopo una introduzione all'argomento, si descrive la parte di modellazione del magazzino, enfatizzando l'importanza di una rappresentazione formale della sua struttura e dello spazio, che è la premessa necessaria alla successiva elaborazione algoritmica.

Nella sezione seguente si delinea come poter ottimizzare i percorsi a partire dal modello creato al passo precedente, presentando algoritmi che possono essere utilizzati allo scopo.

Infine si mostra uno schema che modella i possibili sistemi che realizzano queste funzioni, riportato dalla letteratura.

2.1 Introduzione

La pianificazione dei percorsi all'interno dei magazzini industriali riguarda tutta *l'attività di modellazione, progettazione ed esecuzione atta ad ottimizzare i movimenti degli operatori (umani o macchine) al loro interno*, generalmente durante la fase di spostamento della merce.

Come è stato introdotto nel capitolo precedente, la movimentazione degli elementi avviene principalmente per stoccare la merce in ingresso o per prelevare quella in uscita (ad esempio per la vendita o per l'asservimento a linee di produzione). Pianificare il più possibile gli spostamenti degli operatori risulta di fondamentale importanza per ottimizzare i tempi e i costi di lavoro in un magazzino, dimodoché si possano evadere gli ordini nella maniera più veloce e meno costosa possibile. Questi due parametri dipendono principalmente dalla **distanza** che è necessario percorrere durante lo svolgimento di una particolare operazione.

È necessario, dunque, recuperare tutte le informazioni a disposizione: a partire dalla struttura di un magazzino, ipotizzando la presenza di una serie di scaffali, e data la relativa metodologia per spostarsi al suo interno, è utile derivarne un modello che riassuma le informazioni necessarie alla pianificazione e ottimizzazione dei percorsi. Questo modello deve essenzialmente sintetizzare i punti accessibili dagli operatori durante gli spostamenti, ovvero tutte le varie locazioni degli scaffali, e gli spazi utilizzabili per raggiungerli, fermo restando i vincoli strutturali, come la presenza di corridoi percorribili solo in un senso di marcia o eventuali ostacoli.

Tutte queste informazioni sono definite **a priori** e possono essere sfruttate in fase di configurazione dei percorsi, la prima volta per tutte, fino ad un eventuale cambiamento di una delle condizioni sopraelencate. Le variabili su dove e come organizzare le merci nel magazzino non sono argomento di studio di questa tesi, per cui, non sapendo con certezza quali articoli e quali locazioni riguarderanno i prossimi ordini, non è possibile calcolare il percorso per recuperare i prodotti, una volta per tutte.

In base a quanto detto, questa fase deve necessariamente essere svolta **al momento**, solo quando si hanno tutte le informazioni sull'ordine. *Una buona pianificazione dei percorsi deve, quindi, saper gestire depositi e prelievi di un insieme di locazioni scelte casualmente e dare soluzioni accettabili, in termini di tempi e costi, in tutti questi possibili insiemi.* Essa è necessaria ad una movimentazione efficace della merce.

2.2 Modellazione del magazzino

Il primo passo nella pianificazione e ottimizzazione dei percorsi è di avere un **modello** formale del magazzino; com'è noto, questo artefatto consente di semplificare la realtà, evidenziando solamente le informazioni ritenute essenziali e interessanti, in base all'obiettivo che si vuol perseguire, e di poter essere elaborato da una macchina.

Di seguito si descrivono brevemente una serie di idee riferite a implementazioni di sistemi esistenti, ognuno costruito sulle proprie peculiarità; avere una visione d'insieme, documentadosi su soluzioni già presenti, è un passo utile per decidere le scelte migliori da adottare in seguito.

2.2.1 Panoramica sulla modellazione

Usualmente la pianificazione dei percorsi si svolge per determinare il viaggio che deve compiere un lavoratore o una macchina (ad esempio, un robot) attraverso un'area, la quale è anche riferita come *spazio gestito*. Un percorso pianificato può essere calcolato basandosi su dati che descrivono lo spazio che deve essere gestito (un magazzino, in questo caso). I metodi tradizionali della pianificazione determinano dove sono posti gli ostacoli all'interno dello spazio (muri, macchinari, scaffalature, eccetera) e cercano di identificare un percorso che li aggiri. Tali metodi sono computazionalmente molto costosi e spesso non forniscono il risultato voluto; ad esempio, si potrebbe voler determinare la distanza del percorso calcolato, che non è possibile fare sempre con questo tipo di sistemi.

Al contrario, esistono metodi che determinano i percorsi in uno spazio gestito basati sul modello del layout che definisce le tratte possibili: invece di definire lo spazio in termini di ostacoli (cioè dove non può esserci un percorso), si può rappresentare l'insieme delle aree discrete all'interno delle quali è possibile spostarsi [6]. Il **modello del layout** definisce le tratte percorribili come una **rete di nodi**, i quali indicano i punti dove iniziano e finiscono le tratte e le loro intersezione. Quando una certa operazione richiede di spostarsi lungo lo spazio, il sistema di pianificazione può generare un percorso dal punto di partenza alla locazione di destinazione ba-

sandosi sul modello del layout. Inoltre, questo tipo di definizione permette una maggior flessibilità nella specifica dei percorsi, aumentando l'efficacia nella gestione. Un miglioramento significativo riguarda l'abilità di calcolare la distanza del percorso, potendo così derivare anche una stima del tempo atteso per completare una certa operazione.

Le *locazioni* all'interno dello spazio possono essere connesse ai possibili percorsi basandosi sulle loro coordinate. Ogni elemento o area di lavoro può dunque essere associata, in base alle sue coordinate, a una particolare tratta; la determinazione dei percorsi può essere prima elaborata sui percorsi associati alle locazioni da raggiungere e poi relativa alla singola tratta, ordinando le specifiche destinazioni ad essa collegate.

Il concetto di percorso **migliore** spesso si traduce in quello **più breve**. Infatti, di frequente è richiesto che il viaggio da intraprendere fra le locazioni desiderate sia anche quello più corto; per fare ciò, di solito si impiegano algoritmi euristici, che includono la ricerca *depth first*, *breadth first* oppure una combinazione di esse [6]. Può anche essere necessario dover considerare dei vincoli aggiuntivi, come l'obbligo di dover utilizzare un particolare macchinario (ad esempio, un certo carrello elevatore) per svolgere il compito.

Il calcolo della distanza può essere eseguito sia per obiettivi di pianificazione, ad esempio per determinare quanto tempo può impiegare un compito, e/o di valutazione, individuando se un compito sia stato svolto nei tempi attesi. Molti processi all'interno di un magazzino otterrebbero dei benefici dalla conoscenza della distanza fra due locazioni e, di conseguenza, dalla distanza totale compiuta relativa a un ordine di picking.

Oltre a ciò, in altre implementazioni, si svolge un controllo di funzionalità per stabilire la **completezza** e la **consistenza** del modello del layout, cioè se è possibile viaggiare da un qualsiasi nodo a un altro della rete, potendo raggiungere tutte le locazioni mediante essi. Generalmente, questo controllo viene fatto prima di memorizzare la rete di nodi come modello, durante la generazione del layout model; in altro modo, può essere svolto contemporaneamente alla pianificazione di percorsi, determinando se un cammino è possibile o meno.

2.2.2 Generazione del modello

La generazione del modello del layout, per derivare lo spazio gestito, richiede i dati di anagrafica del magazzino, che includono l'indicazione delle coordinate (x, y) di ogni elemento strutturale presente al suo interno, sia esso un ostacolo, una stazione di lavoro, o altro. Una rappresentazione a griglia dello spazio gestito può indicare quali aree sono occupate e quali no, così, invece di rappresentare un modello di pianificazione dello spazio in termini di ostacoli, il sistema può determinare quali sono le aree libere e definire i possibili percorsi per attraversarle. Questi ultimi sono rappresentati come segmenti di linea delimitati da una coppia di coordinate della rappresentazione a griglia, con associata una misura della lunghezza. Il sistema poi identifica le intersezioni (sovrapposizioni fra le linee) e genera la rete di nodi; essa viene utilizzata come modello per pianificare i viaggi attraverso il magazzino. In tal modo, si può anche avere l'informazione riguardante la distanza complessiva sostenuta attraverso la percorrenza di un insieme di tratte.

In un approccio *a percorsi possibili* si può identificare l'inizio e la fine di uno scaffale di un corridoio, calcolando la lunghezza di quest'ultimo, e si può tracciare una freccia o una linea che raggiunge tutte le locazioni dello scaffale lungo il corridoio. Mediante una ulteriore freccia si può stabilire la connessione al corridoio successivo, e così via. La lunghezza di tali segmenti è determinabile basandosi sui dati di anagrafica del magazzino. Il risultato di questa serie di operazioni è un insieme di lati e frecce che definiscono la rete, dove ogni segmento indica un possibile percorso.

Si può anche tenere conto di informazioni aggiuntive da salvare assieme al modello. Ad esempio, ogni percorso può essere associato a un vincolo di percorrenza, come la larghezza della tratta e la dimensione della risorsa che può o meno attraversarla, la direzione obbligatoria di passaggio oppure la sua suddivisione in corsia destra e corsia sinistra. In sintesi, qualsiasi tipo di vincolo operativo è quindi associabile ai percorsi, sia di natura arbitraria che derivante da caratteristiche intrinseche della struttura.

2.3 Ottimizzazione dei percorsi

In questa sezione vengono esposte alcune comuni soluzioni nel calcolo dei percorsi per la fase di movimentazione interna di un magazzino. Anche in questo ambito la letteratura è molto vasta, quindi si riporta una sintesi sulla teoria che ha ispirato il capitolo successivo.

Spesso i riferimenti fatti dalla letteratura su questo genere di algoritmi sono raccontati mediante metafore su mappe stradali, ma possono valere allo stesso modo se riportati in un contesto di movimentazione interna nei magazzini.

2.3.1 Introduzione al calcolo dei percorsi

Come accennato precedentemente, si possono utilizzare alcune euristiche per calcolare il miglior tragitto attraverso lo spazio gestito del magazzino. Una combinazione di queste può impiegare una veloce ricerca *depth first* per trovare una prima soluzione ammissibile, per poi modificarla mediante una ricerca *breadth first* per ottenere miglioramenti alla soluzione iniziale.

In aggiunta, si può fare ricorso a *criteri di stop*, che indicano soglie numeriche superate le quali si intende fermare l'algortimo; ad esempio, si possono applicare per limitare la ricerca *breadth first* ad un numero di nodi da visitare pari a quello della soluzione iniziale. Questi criteri possono non portare ad alcun beneficio in termini di qualità, ma servono per limitare l'intensità computazionale del calcolo del percorso, come in caso di requisiti temporali da soddisfare.

In aggiunta a questi algoritmi, si possono utilizzare ulteriori metriche per la generazione dei percorsi, ad esempio decidere se impiegare il calcolo della distanza **Euclidea**, **Manhattan** o una loro combinazione. La prima considera il tratto fra due punti come una linea retta, vista come l'ipotenusa di un triangolo rettangolo tra le due coordinate di un punto di inizio e un punto di fine. La distanza Manhattan considera il tratto fra due punti come definito in due segmenti, il primo che si sposta solo lungo l'asse X e il secondo solo lungo l'asse Y, fra le coordinate dei punti di start e di stop. In questo modo, tutti i percorsi sono composti da soli angoli retti.

Nelle sottosezioni seguenti si riportano delle categorie di problemi e soluzioni note che modellano viaggi da ottimizzare in una rete di percorsi.

2.3.2 Vehicle routing problem

Il **vehicle routing problem**, o VRP, consiste nel determinare i percorsi per una flotta di veicoli che devono raggiungere un insieme di utenti. Il VRP può essere definito su di un grafo $G = (V, A, E)$, dove V è l'insieme dei vertici, A è l'insieme degli archi e E è l'insieme dei lati. Il vertice 0 rappresenta il punto di deposito, nel quale sono collocati gli m veicoli, mentre i sottoinsiemi dei *vertici richiesti* $U \subseteq V$ e degli *archi e lati richiesti* $R \subseteq A \cup E$ rappresentano gli utenti. L'obiettivo del VRP è determinare l'insieme degli m tour a costo minimo che attraversano il deposito, i vertici, gli archi e i lati richiesti.

In una rappresentazione a grafo, gli archi e i lati corrispondono a tratte percorribili e i vertici a intersezioni fra esse. Utenti singoli sono rappresentati da un vertice richiesto, mentre i sottoinsiemi di clienti distribuiti in maniera quasi continua in un insieme di utenti sono modellati come archi o lati richiesti.

All'interno di un magazzino, la rappresentazione pratica sarebbe quella di un insieme di risorse (i veicoli) che devono essere utilizzate al meglio per trasportare la merce e raggiungere i punti di deposito/prelievo (i clienti) previsti, attraverso il grafo dei percorsi.

Se $R = \emptyset$, il VRP è chiamato *node routing problem* (NRP), mentre se $U = \emptyset$ è chiamato *arc routing problem* (ARP). I problemi NRP sono stati studiati più approfonditamente degli ARP e sono usualmente riferiti direttamente come VRP. Se vale $m = 1$ e in assenza di vincoli particolari, il NRP diventa il classico *travelling salesman problem* (TSP), che consiste nel determinare un singolo circuito su tutti i vertici di G, mentre l'ARP diviene il *rural postman problem* (RPP), che mira a trovare un singolo circuito che include gli archi e i lati di R. Il RPP si riduce al *Chinese postman problem* (CPP) se ogni arco o lato deve essere attraversato ($R = A \cup E$).

I più comuni vincoli operazionali sono:

- il numero di veicoli m può essere fissato o essere una variabile decisionale, eventualmente soggetta a un vincolo di limite massimo;
- la merce totale trasportata da un veicolo, in ogni istante, non deve eccedere la sua capacità;
- la durata di ogni tratta non deve eccedere la durata del turno di lavoro;
- alcuni clienti devono essere serviti entro un intervallo di tempo prestabilito;
- il servizio di un cliente deve essere svolto da un singolo veicolo o da un insieme di essi;
- i clienti sono soggetti a relazioni di precedenza.

Quando è necessario prendere in considerazione esplicitamente vincoli di tempo, nella formazione delle tratte, il VRP è spesso riferito come *vehicle routing and scheduling problem* (VRSP).

Vincoli di precedenza appaiono nel caso in cui i beni debbano essere trasportati fra specifiche coppie di punti di carico e scarico merci. In questo caso, ogni coppia deve essere servita dallo stesso veicolo e ogni punto di carico deve essere visitato prima del punto di scarico a lui associato. Simili relazioni di precedenza vengono imposte quando i veicoli devono prima eseguire un certo numero di consegne e in seguito una serie di carichi.

Ipotizzando l'assegnazione di un costo di attraversamento per ogni arco e lato, l'**obiettivo** più comune è quello di minimizzare i costi totali nella percorrenza del grafo, sommati ai costi fissi relativi all'uso dei veicoli.

Nella risoluzione di questo tipo di problemi si fa spesso ricorso a euristiche, per due motivi principali: innanzitutto, gli algoritmi esatti per il VRP sono spesso funzioni ricorsive piuttosto complesse, che permettono di risolvere solo piccole istanze del problema; inoltre, soluzioni ammissibili devono spesso essere generate in un breve, ragionevole, lasso di tempo.

2.3.3 Travelling salesman problem

In assenza di vincoli operativi, esiste sempre una soluzione ottimale al NRP nel quale si usa un solo veicolo [5]. Quindi, il NRP si riduce a *travelling salesman problem*, che consiste nel trovare il tour a costo minimo che visita tutti i vertici e il deposito. In ogni soluzione ammissibile del TSP su un grafo G, ogni vertice $U \cup \{0\}$ compare almeno una volta e due vertici successivi sono collegati da un percorso a costo minimo. Come conseguenza, il TSP può essere riformulato su un **grafo ausiliario direzionato completo** $G' = (V', A')$, dove $V' = U \cup \{0\}$ è l'insieme dei vertici e A' è l'insieme degli archi. Ad ogni arco $(i, j) \in A'$ è associato un costo c_{ij} pari a quello del percorso di costo minimo da i a j in G. Tali costi soddisfano la *disuguaglianza triangolare*:

$$c_{ij} \leq c_{ik} + c_{kj}, \qquad \forall (i, j) \in A', \forall k \in V', k \neq i, j$$

Data questa proprietà, esiste una soluzione ottimale del TSP che è un tour Hamiltoniano in G', ovvero un circuito nel quale ogni vertice in V' appare esattamente una volta.

Se $c_{ij} = c_{ji}$ per ogni coppia di vertici distinti $i, j \in V'$, allora il TSP è detto *simmetrico* (STSP), altrimenti è detto *asimmetrico* (ATSP). Ovviamente, le soluzioni ottenute sull'ATSP sono ammissibili anche per il STSP, anche se è un approccio spesso poco efficiente [5].

Queste premesse sono utili per anticipare il modello del *road travelling salesman problem*.

2.3.4 Road travelling salesman problem

Come anticipato nel capitolo 1, il problema della scelta del percorso da compiere per raggiungere un insieme di locazioni del magazzino è noto come *order picker routing*. Il *road travelling salesman problem* (RTSP) è una variante del classico TSP, che consiste nel determinare il tour di costo minimo su un sottoinsieme dei vertici del grafo G, per un singolo veicolo ($m = 1$).

Il RTSP ha complessità NP-hard, ma nel caso dei magazzini è spesso risolvibile in tempo polinomiale a causa delle particolari caratteristiche della rete dei percorsi. Facendo riferimento alla figura

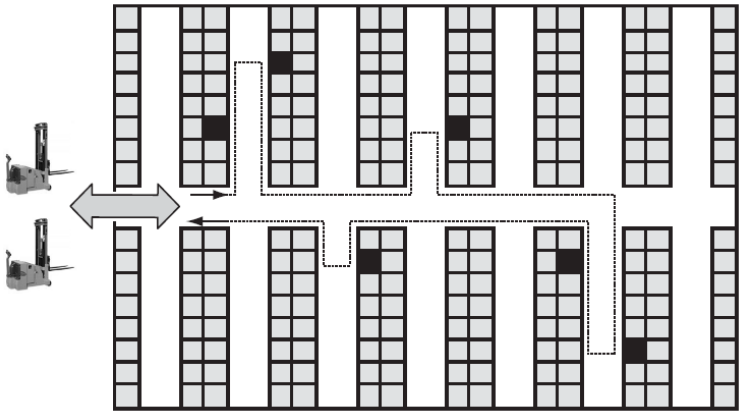

Figura 2.1: Esempio di order picker routing (fonte: Ghiani, Laporte, Musmanno)

2.1, nel caso in cui ogni corridoio abbbia un solo ingresso, la tratta a percorrenza più breve è data visitando prima le locazioni di stoccaggio raggiungibili dai corridoi della parte alta della figura e poi visitando quelli nella parte bassa.

Se invece i corridoi hanno più interruzioni (ad esempio per la presenza di più corridoi trasversali), il problema può essere risolto all'ottimo utilizzando l'algoritmo di programmazione dinamica di *Ratliff e Rosenthal*, il cui caso peggiore di complessità computazionale è dato da una funzione lineare sul numero dei lati dei corridoi. In ogni modo, alla presenza di diversi corridoi ortogonali, il numero di stati e transizioni aumenta rapidamente e l'uso delle procedure dinamiche di programmazione diviene non praticabile. Per questi motivi, vengono utilizzate due semplici euristiche:

- **Euristica S-shape**: ogni corridoio che contiene almeno un elemento che deve essere recuperato viene attraversato interamente. I corridoi che non includono elementi da prelevare sono saltati.

- **Euristica largest gap**: in questo contesto, il *gap* è la distanza fra due elementi adiacenti che devono essere recuperati in un

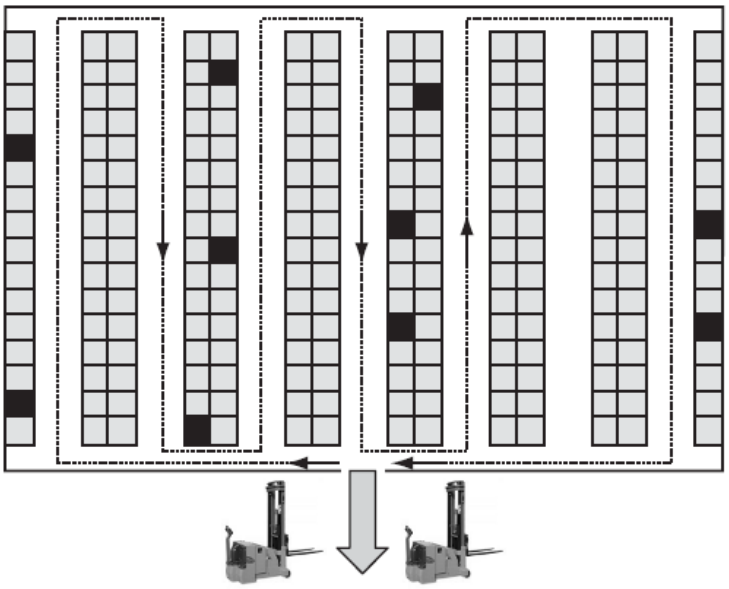

Figura 2.2: Euristica s-shape (fonte: Ghiani, Laporte, Musmanno)

corridoio, o la distanza fra l'ultimo elemento che deve essere prelevato in un corridoio e il corridoio trasversale più vicino. L'operatore si posizione dal lato del corridoio più vicino alla porta di ingresso/uscita, andando verso quello che include almeno un elemento da recuperare. Dopodiché egli attraversa il corridoio interamente, mentre entra ed esce nei rimanenti corridoi una o due volte, sempre dallo stesso lato, a seconda del *gap* maggiore del corridoio. I percorsi generati in questa maniera possono risultare problematici all'atto pratico della movimentazione; le inversioni a U nelle corsie sono potenzialmente pericolose e poco pratiche in un magazzino con corridoi a larghezza ridotta.

Basandosi su una mappa che prevede di prelevare degli elementi, nelle due figure riportate si applica in un caso l'euristica *s-shape* 2.2 e nell'altro l'euristica *largest gap* 2.3.

In alternativa, come per il `TSP`, anche l'`ATSP` può essere riformulato su un sottografo ausiliario direzionato completo $G'' = (V'', A'')$,

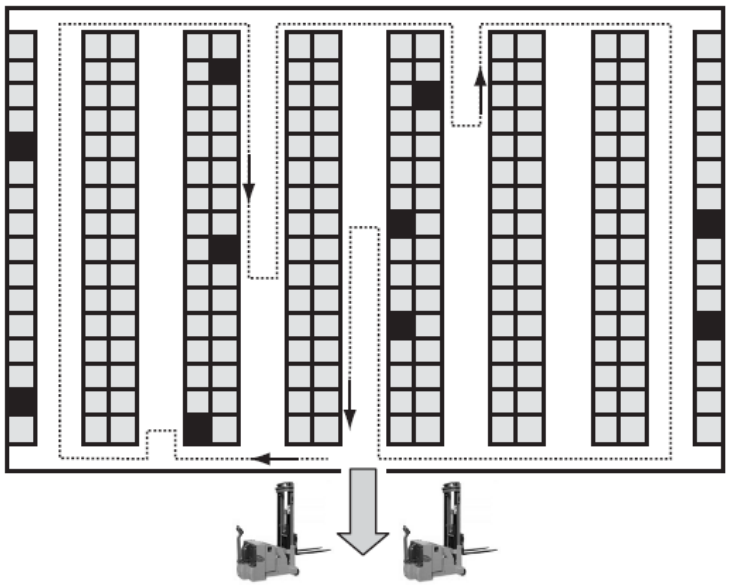

Figura 2.3: Euristica largest gap (fonte: Ghiani, Laporte, Musmanno)

dove $V'' = U' \cup \{0\}$ è l'insieme dei vertici, con $U' \subseteq U$ sottoinsieme dei vertici da visitare, e A' è l'insieme degli archi. Come nella definizione precedente, ad ogni arco $(i, j) \in A'$ è associato un costo c_{ij} pari a quello del percorso di costo minimo da i a j in G, che soddisfano la *disuguaglianza triangolare*.

Per questo esiste una soluzione ottimale dell'`ATSP` che è un tour Hamiltoniano in G'', ovvero un circuito nel quale ogni vertice in V'' appare esattamente una volta.

All'interno dei magazzini può essere utile adottare il modello dell'`ATSP` per rappresentare i percorsi all'interno, in quanto molto spesso le tratte sono direzionate per la presenza di corridoi o corsie a sensi unici di attraversamento.

2.3.5 Algoritmi euristici

Gli algoritmi euristici accennati in precedenza garantiscono di calcolare una soluzione che approssima l'ottimo. Il loro obiettivo è determinare una soluzione ammissibile di buona qualità in un tempo di calcolo accettabile, ottenendo un *bound* della soluzione ottima. Essi si dividono in due categorie:
Algoritmi euristici costruttivi: partono da una soluzione vuota e determinano iterativamente nuovi elementi fino a una soluzione completa. Essi seguono un *criterio di espansione* per scegliere quale elemento aggiungere in soluzione, in alcuni casi pre-ordinandoli. Per il `TSP` si possono applicare nell'espansione di un cammino o di un ciclo, fino alla soluzione completa. Si differenziano per la *scelta del ciclo o del vertice iniziale*, per il *criterio di selezione* del vertice da inserire e per il *criterio di inserzione* (ovvero, dove inserire il vertice). Alla generica iterazione h si ha un ciclo o cammino che coinvolge $S \subseteq V$ vertici; termina quando $S \equiv V$.

Fra questo tipo di algoritmi è importante ricordare il **nearest neighbor** e il **cheapest insertion**. Il primo sceglie il vertice iniziale in maniera arbitraria e poi procede con l'espansione di un cammino, scegliendo il vertice più vicino e aggiungendolo in fondo. Così facendo, però, spesso l'arco di chiusura risulta lungo e la soluzione dipende molto dal verdice iniziale.

Di seguito, viene riportato l'algoritmo *nearest neighbor* sotto forma di pseudo-codice:

```
it := 1;
s(1) := i;
while it < n do
    determina k : c_{s(it),k} :=min{c_{s(it),j} : j ∉ S};
    s(it + 1) := k;
    it := it + 1;
end while;
```

Descrivendo singolarmente le variabili:

- i è il vertice iniziale, scelto arbitrariamente;
- it è il contatore delle iterazioni;
- n è il numero di vertici totali;
- S è l'insieme dei vertici in soluzione;
- $s(i)$ indica il vertice in posizione i fra quelli inseriti nella soluzione s;
- $c_{i,j}$ indica il costo dell'arco che collega i vertici i e j.

La complessità risultante è $O(n^2)$, dipendente dal numero dei vertici.

L'algoritmo *cheapest insertion* procede per espansione di ciclo. La coppia iniziale è composta da un vertice arbitrario e dal quello a lui più distante, dopodiché si procede ad ogni passo selezionando il vertice per cui è minimo il costo di inserzione in S (insieme dei vertici presenti in soluzione). L'inserzione del vertice k tra i vertici in soluzione S_i e S_{i+1} comporta una variazione di costo data da:

$$\delta(k,i) = c_{S_i,k} + c_{k,S_{i+1}} - c_{S_i,S_{i+1}}, \qquad i \in S, k \notin S$$

Per ogni vertice che non appartiene ad S si può definire:

$$\mu(j) = argmin\{\delta(j,i) : i \in S\}$$

Questi valori si definiscono in tempo $O(n^2)$ e sono aggiornabili in tempo $O(n)$. Inoltre, il punto di inserzione in tal modo è definito univocamente, nella posizione successiva a $\mu(k)$.

Algoritmi euristici di ricerca locale: partono da una soluzione (anche non ammissibile) e cercano iterativamente di migliorarla effettuando semplici modifiche (*mosse*). Queste ultime possono essere, ad esempio, scambi di elementi fra quelli in soluzione e quelli che non lo sono. Tali algoritmi hanno termine quando non esistono più modifiche che migliorano la selezione corrente.

Tra i miglioramenti applicabili, si può scegliere se eseguire il primo scambio migliorante (*first improvement*) o il migliore tra quelli disponibili (*best improvement*). L'insieme delle soluzioni ottenibili dalla soluzione corrente mediante le mosse si chiama intorno o *neighborhood* (vicinato) e sono ottenute applicando una **neighborhood function** alla soluzione corrente.

Le ricerche locali hanno applicabilità generale e flessibilità, richiedono un valutatore di soluzioni, la verifica dell'ammissibilità, una funzione che genera il vicinato e una esplorazione del neighborhood efficiente. Il problema centrale risiede nella definizione del vicinato, che è fortemente dipendente dal problema. Un metodo generale è quello del neighborhood basato sugli scambi di sequenze o partizioni degli elementi presenti nella soluzione.

Applicando questo concetto al TSP, per generare un nuovo vicino si possono rimuovere k archi dalla soluzione, aggiungendone altrettanti non selezionati in precedenza. La cardinalità del vicinato dipende, quindi, dal valore di k scelto, in quanto gli archi rimossi possono essere ricombinati in modi sempre più diversi all'aumentare di k.

La qualità dell'ottimo (locale) raggiunto da questo tipo di algoritmi dipende molto dalla soluzione iniziale e dal vicinato, perciò si possono attuare alcuni miglioramenti, come la tecnica **multi-start**: essa genera diverse soluzioni iniziali (con perturbazioni casuali) e procede con una ricerca locale per ognuna.

Le prestazioni di un euristico sono valutabili sia *sperimentalmente*, osservando la qualità delle soluzioni su istanze di test, sia basandosi sul *caso peggiore*, in maniera analitica, misurando il massimo errore ottenibile rispetto alla soluzione ottima.

Una loro evoluzione è rappresentata dagli algoritmi **meta-euristici**: essi implementano algoritmi di ricerca locale che usano tecniche per uscire dai minimi locali, evitando il verificarsi di cicli, per cercare una soluzione migliore al di fuori di essi.

2.3.6 Algoritmi meta-euristici

Le tecniche *meta-euristiche* sono algoritmi di ricerca locale con metodi per uscire da minimi locali, che cercano di evitare i cicli durante la loro evoluzione. In generale sono molto efficaci e di facile implementazione, ma richiedono tempi di esecuzione molto più elevati rispetto agli euristici di prima [8].

Il primo che si presenta è il **tabu search**: esso rappresenta una generalizzazione della ricerca locale che consente l'accettazione di mosse peggiorative della soluzione. Viene utilizzata una memoria a breve termine, detta *tabu list*, per evitare di ritornare nelle ultime soluzioni visitate. La lunghezza di tale memoria è un parametro dell'algoritmo, d'ora in poi chiamato t.

Ipotizzando un problema di minimizzazione della funzione obiettivo, se ne riporta il funzionamento mediante pseudo-codice:

soluzione iniziale s di valore $z(s)$
$s^* := s$;
$k := 1$;
$T = \{s\}$;
`while not STOP CRITERION do`
    genera il neighborhood non tabu $N(s) \setminus T$
    trova la migliore soluzione $s' \in N(s) \setminus T$ rispetto a $z(\cdot)$
    `if` $z(s') < z(s)$ `then` $s^* := s$; $k_{best} := k$
    $s := s'$;
    $k := k + 1$;
    inserisci s' in T al posto della soluzione più vecchia
`end while`;

Descrivendo singolarmente le variabili:

- s è la soluzione attuale, aggiornata ad ogni iterazione;

- $z(s)$ è il valore della funzione obiettivo della soluzione s;
- k è il contatore delle iterazioni;
- $N(s)$ è l'insieme del vicinato della soluzione s;
- T è l'insieme delle soluzioni tabu;
- s' è la migliore soluzione trovata nell'insieme del vicinato, meno le soluzioni tabu, ad ogni iterazione;
- s^* è la migliore soluzione trovata attualmente, aggiornata ad s' solo è migliore di s;
- k_{best} è il numero dell'iterazione nella quale si è trovata l'ultima s^*;
- $s(i)$ indica il vertice in posizione i fra quelli inseriti nella soluzione s;
- $c_{i,j}$ indica il costo dell'arco che collega i vertici i e j;
- `STOP CRITERION`: sono i criteri di arresto dell'algoritmo e ce ne possono essere di diversi:
  - $N(s) \setminus T = \emptyset$: se l'insieme del vicinato meno le soluzioni tabu è uguale all'insieme vuoto;
  - se $k > k_{max}$, ovvero se si è raggiunti un certo limite di iterazioni k_max;
  - se si è raggiunti un tempo limite nell'elaborazione;
  - se $k - k_{best} > k_{non\ improving}$ ovvero se la soluzione non viene migliorata da un numero di iterazioni $k_{non\ improving}$;
  - se si è trovata s^* ottima, ad esempio uguale ad un lower bound.

L'aspetto critico riguarda la tabu list, infatti memorizzare T soluzioni può essere oneroso, in quanto ognuna è un vettore di n elementi; verificare se una soluzione è tabu ha complessità $O(n \cdot |T|)$. Per porre rimedio a questi calcoli dispendiosi si possono memorizzare solamente delle caratteristiche delle soluzioni, invece delle

soluzioni intere, oppure ricordare nella tabu list le mosse applicate, anche se in tal modo si proibisce un numero di soluzioni maggiore. Per fare ciò si ricorre all'utilizzo di una T^I che memorizza le *mosse inverse* delle ultime utilizzate, per evitare di ritornare nel punto di prima. Come detto, questa T^I è molto più restrittiva ti T, ma non garantisce comunque che non si abbiano cicli di periodo $\leq |T^I|$. Questa tabu list è chiamata **memoria di breve periodo**, siccome è possibile utilizzare anche dei *criteri di aspirazione*, ovvero: per sopperire alle condizioni troppo restrittive della tabu list, le mosse tabu possono essere comunque effettuate se conducono ad una buona soluzione. Per fare ciò si impiega una **memoria di lungo periodo** che consente sia *intensificazione*, preferendo mosse simili a quelle recenti, senza spostarsi troppo dalla regione attuale, che *diversificazione*, andando verso zone inesplorate. La lunghezza t della tabu list viene aggiornata dinamicamente in tal modo:

- se s^* è migliorata: $t = max\{t_{min}, t-1\}$ = **intensificazione**;
- se s^* è immutata: $t = min\{t_{max}, t+1\}$ = **diversificazione**.

Si possono ora immaginare diversi tipi di tabu list da applicare al *travelling salesman problem*, ad esempio:

- è tabu muovere il vertice i per t iterazioni:
  - si può salvare la lista dei vertici tabu,
  - o salvare l'iterazione in cui è stato mosso il vertice;
- basandosi su un neighborhood 2-opt, nel quale ogni vicino è ottenuto sostituendo una coppia di archi con una incrociata:
  - T può contenere il lato più corto rimosso (`Glover 86`);
  - T può contenere le coppie di archi più costose rimosse (`Knox 94`).

Infine, se il problema è fortemente vincolato la cardinalità del vicinato è molto piccolo; in questo caso si possono rilassare dei vincoli, riportarne le violazioni inserendo delle penalità nella funzione

obiettivo, regolandole in maniera dinamica; ad esempio, se da molte iterazioni le soluzioni risultano tutte inammissibili, si aumentano le penalità delle violazioni per riportarsi su zone ammissibili.

Un altro tipo di algoritmo meta-euristico molto noto è il **simulated annealing**, basato sull'utilizzo di una componente casuale nel movimento sul neighborhood. Esso funziona su questo principio: data s la solzione corrente, se esiste una soluzione migliorante s', appartenente al vicinato, la si esegue, altrimenti si applicano mosse peggioranti s'' con una probabilità che dipende da un parametro decrescente T, detto *temperatura*. Si è dimostrato che questo metodo è *asintoticamente esatto*, convergendo teoricamente all'ottimo globale [8].

Il simulated annealing si suddivide in una versione *non omogenea*, nella quale la temperatura diminuisce ad ogni mossa, ed in una *omogenea*, dove si mantiene T uguale fino ad un raggiungimento di uno stato di equilibrio.

L'algoritmo termina quando:

- non si migliora la soluzione da p iterazioni;
- non si accettano mosse da q iterazioni;

Una sua prima variante è formulata dal **reannealing**, che consiste nel memorizzare alla prima esecuzione il valore di T^0 col quale si è trovata la soluzione migliore, dopodiché avviare una seconda esecuzione di ricerca locale più accurata con $T = T^0$.

Una seconda variante è formulata dal **restricted neighborhood**, dove non si considerano le mosse che difficilmente producono buone soluzioni; ad esempio, nel `TSP` si userebbero mosse che collegano solo vertici vicini.

Sintetizzando, nel tabu search le mosse peggioranti sono consentite solo quando si è in un ottimo locale ed esso è poco basato sulla casualità, mentre il simulated annealing permette mosse peggioranti in qualunque momento ed è fortemente basato sulla casualità.

In conclusione, gli algoritmi meta-euristici forniscono schemi generali che necessitano di essere particolarizzate per i singoli problemi, per questo spesso non sono competitivi al confronto di algoritmi

sviluppati ad-hoc. I tempi di sviluppo sono però inferiori e i risultati sono normalmente migliori delle ricerche locali di base, anche se in tempi di calcolo superiori. Possono però diventare complessi e dipendenti da numerosi parametri.

2.3.7 Algoritmi per cammini minimi

Il problema dei cammini a costo minimo, o *shortest path problem* (SPP), rientra nella teoria dei grafi. Dato un grafo orientato $G = (V, A)$, pesato sugli archi con costi c_{ij}, e due vertici $s, t \in V$, si vuole determinare il cammino semplice di costo minimo dal vertice s al vertice t.

Se i costi degli archi sono qualsiasi, il SPP è NP-hard [9]. Esistono casi in cui il problema è a complessità polinomiale; ad esempio, se vale $c_{ij} \geq 0$ per ogni $(i, j) \in A$, si può utilizzare l'**algoritmo di Djikstra**, a complessità $O(n^2)$ - $O(n \cdot \log n)$.

Si dimostra che se i costi degli archi sono non negativi, il cammino di costo minimo è semplice ed elementare.

Inoltre, se il cammino minimo da v_i a v_j passa per v_k, allora esso è formato dal cammino minimo da v_i a v_k concatenato dal cammino minimo da v_k a v_j.

Per formulare matematicamente questo problema si può ricorrere ad un modello di programmazione lineare intera, con funzione obiettivo:

$$min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

e vincoli:

$$\sum_{(h,j)\in\Gamma^+(h)} x_{hj} - \sum_{(i,h)\in\Gamma^-(h)} x_{ih} = \begin{cases} 1 & \text{se } h = s \\ -1 & \text{se } h = t \\ 0 & \text{se } h \in V \setminus \{s, t\} \end{cases}$$

$$\sum_{(i,j):i,j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq V, S \neq \emptyset$$

$$0 \leq x_{ij} \leq 1 \qquad intero(i, j) \in A$$

Un arco che va dal vertice i al vertice j ha costo c_{ij} ed è considerato in soluzione se $x_{ij} = 1$, 0 altrimenti. La funzione obiettivo definisce una miminizzazione del costo totale degli archi in soluzione che formano il cammino. Il primo vincolo impone che la differenza fra il numero totale di archi entranti ($\Gamma^+(h)$) in un vertice h e il numero di archi uscenti ($\Gamma^-(h)$) da quel vertice sia pari a 1, se j è il vertice iniziale, -1 se h è il vertice finale, e pari a 0 se h è un vertice interno al cammino. Il secondo vincolo, invece, impone che per ogni possibile sottoinsieme di vertici, il numero di archi selezionati, presenti in esso, sia minore o uguale al numero dei vertici del sottoinsieme meno 1. In particolare, nel caso in cui i costi siano tutti non negativi, il secondo vincolo risulta ridondante e può dunque essere eliminato.

Questo problema è stato risolto efficientemente dall'**algoritmo di Dijkstra** (1959). Partendo dalle seguenti strutture dati:

- W è l'insieme dei vertici raggiunti in modo permanente dalla soluzione s,
- $L(v)$ è il costo del cammino minimo da s a v passante solo per $j \in W$,
- $pred(v)$ è il vertice che precede v nel cammino da s a v,

e dato il seguente

Teorema: *se* $L(v^*) = min_{v \in V \setminus W}\{L(v)\}$ *il cammino minimo da* s *a* v^* *è lungo* $L(v^*)$,
se ne descrive la definizione in pseudo-codice:

$W := \{s\}$;
$L(s) := 0$;
$pred(s) := 0$;
`for each` $v \in V \setminus \{s\}$ `do`
    $L(v) := c(s, v)$;
    $pred(v) := s$;
`while` $|W| < n$ `do`
    trova $v^* \in V \setminus W : L(v*) = \min_{v \in V \setminus W}\{L(v)\}$;
    $W := W \cup \{v^*\}$;
    `for each` $v \in V \setminus W$ `do`
        `if` $L(v*) + c(v*, v) < L(v)$ `then`

$L(v) := L(v^*) + c(v*, v)$;
$pred(v) := v^*$;
`end while`

Nel ciclo `while` si svolgono $n - 1$ iterazioni e per ogni iterazione si ha un numero di operazioni proporzionale a $|V \setminus W|$; in termini temporali, la complessità risulta $O(n^2)$.

Inizialmente solo il vertice iniziale è compreso in S; le etichette da assegnare ai vertici comprendono due informazioni: il predecessore e il costo totale per giungere a tale vertice. Quelli non raggiungibili da S hanno costo uguale a infinito, mentre quelli raggiungibili hanno costo pari all'arco di costo minimo che li connette ad S più il costo del vertice predecessore. Ad ogni iterazione si include in S il vertice che ha etichetta con costo minore e la sua diventa di assegnazione permanente. Dopodiché si aggiornano solo le etichette dei vertici che sono raggiungibili da S con un arco diretto, ma solo se il nuovo costo migliora quello precedente. Una volta trovato il percorso dal vertice iniziale a quello finale, le etichette ricostruiscono il cammino minimo. L'algoritmo è sviluppabile anche in forma tabellare: una colonna indica la componente connessa, $n - 1$ colonne rappresentanti i vertici (tranne quello iniziale) indicano il costo del cammino minimo per arrivare a tale vertice e altre $n - 1$ colonne indicano i vari predecessori. Ogni riga rappresenta un'iterazione e per ognuna si include in S il vertice con costo minore, aggiornando quella successiva eventualmente con i nuovi costi (se migliori) e i relativi predecessori, fino a connettere il vertice iniziale con quello finale.

Questo algoritmo può essere usato per calcolare i cammini minimi fra le varie coppie di vertici di una rete di percorsi, per costruire quel grafo ausiliario che consente di riformulare l'`ATSP`, applicando poi un algoritmo (meta-)euristico di ricerca locale.

2.4 Un possibile sistema

Dopo aver analizzato la letteratura, nei sistemi descritti emergono due tematiche: *la generazione del modello* e *la generazione dei percorsi*. Esse possono essere implementate anche in sistemi separati, non ne-

cessariamente compresi nello stesso apparato di gestione magazzino e in entità software e hardware diverse.

In figura 2.4 si può osservare un diagramma a blocchi di una implementazione hardware e software di un sistema di gestione magazzino. Esso comprende un generatore di modello della struttura che definisce una rete di nodi indicante le possibili tratte e un generatore di percorsi che rappresenta un tragitto basandosi sulla rete di nodi.

Prima di tutto, è necessario definire il layout del magazzino mediante le sue **caratteristiche strutturali**: esse includono le coordinate dei vari elementi presenti nello spazio e possono essere, ad esempio, relative a dati di anagrafica del magazzino. Possono inoltre indicare informazioni aggiuntive, come la direzione di viaggio in una particolare tratta, la larghezza di un corridoio, eccetera. Questi dati possono anche essere inseriti nel modulo di gestione magazzino da un utente, per modificare i percorsi identificati.

Ad esempio, in figura 2.5 è presentato un esempio grafico di un layout di un magazzino; esso descrive un modello di **spazio gestito**, citato in precedenza. Si può notare come il modello sia formato da diversi elementi: due serie di scaffali (una verticale e una orizzontale) separate da muri divisori visti come ostacoli, e un insieme di aree di lavoro. Questa è la parte che va a comporre i dati di anagrafica della struttura.

Ritornando alla figura 2.4, si ha il **layout analizer**, che permette al generatore di modello di ricevere in input i dati delle caratteristiche strutturali, processarli e generare un modello dello spazio gestito. Il suo compito è analizzare il layout, identificandone le strutture e i percorsi, per poi passare tali informazioni al modulo successivo.

Il **pathway definition module** si basa sulle informazioni che riceve dal blocco precedente per definire i possibili percorsi, mentre il **node identifier** esegue un'analoga operazione, ma atta all'identificazione e definizione dei nodi. I punti estremi di ogni tratta possono essere riconosciuti come nodi e la lunghezza del percorso può essere considerata come la distanza fra i due nodi. Inoltre, si possono identificare i nodi determinando i punti nei quali si intersecano i percorsi. Questi due moduli possono essere scambiati di ordine oppure eseguiti in concorrenza fra loro, a seconda delle implementazioni.

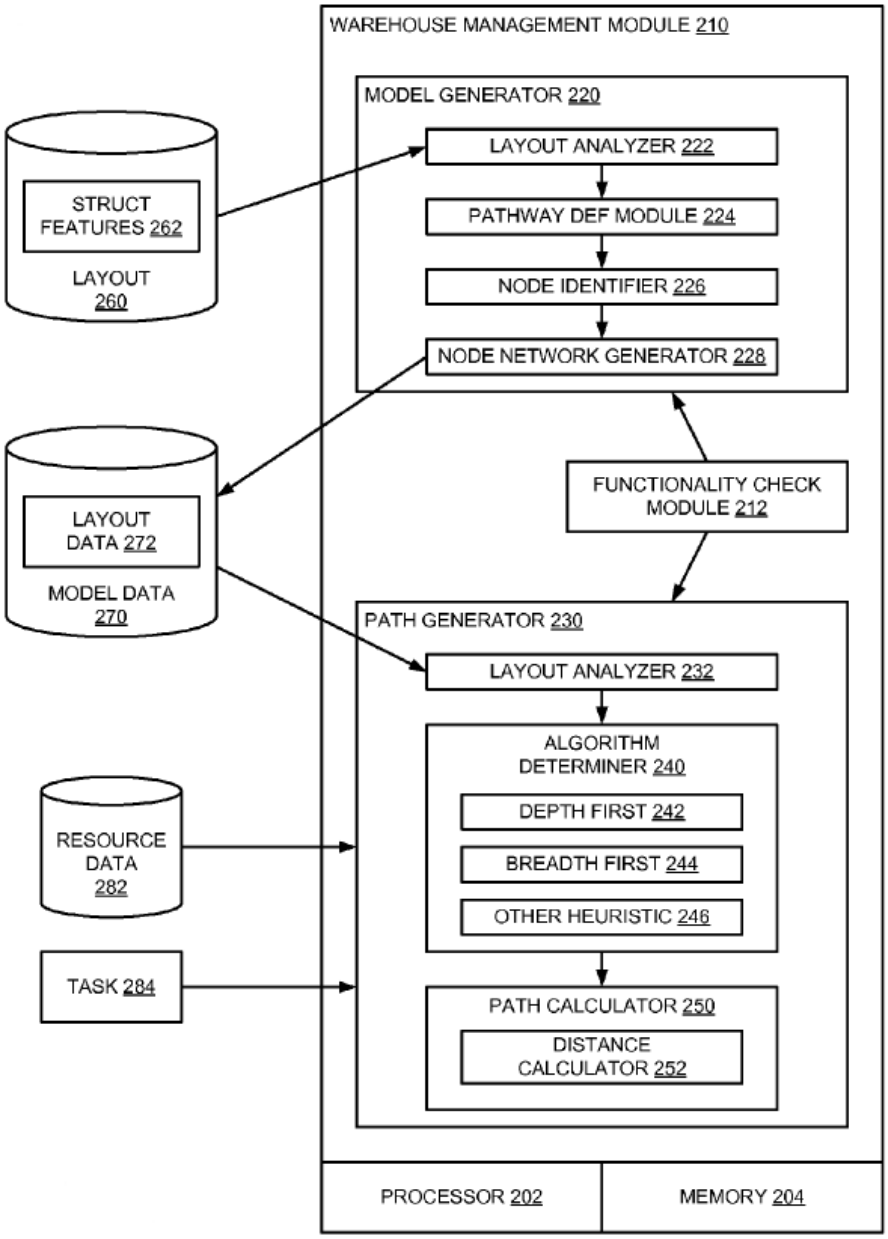

Figura 2.4: Un esempio di sistema di gestione magazzino (WMS) (fonte: Mandel et al., 2009)

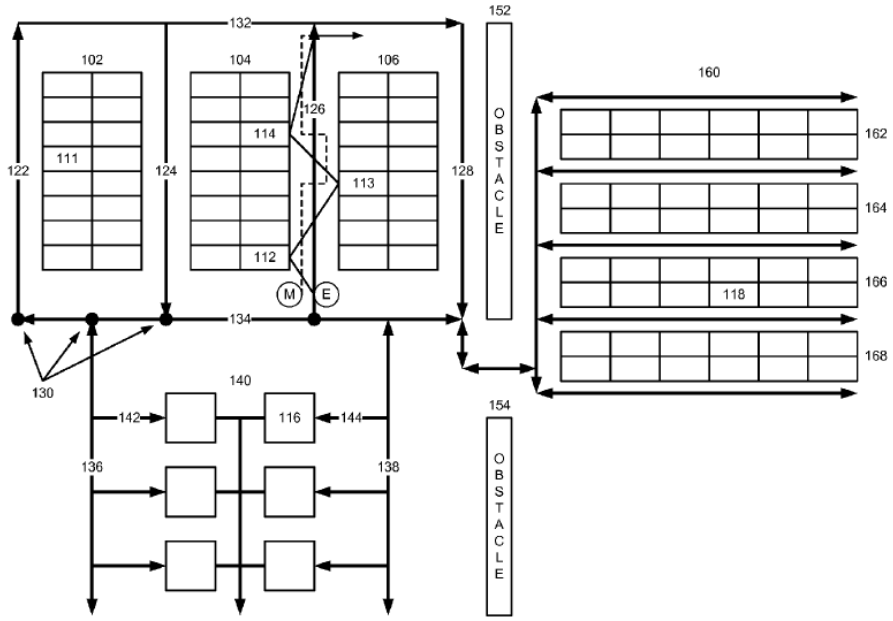

Figura 2.5: Esempio di layout di un magazzino (fonte: Mandel et al., 2009)

A seguito di queste fasi avviene la generazione della rete di nodi nel **node network generator**, che memorizza un modello con le informazioni sui nodi, sui percorsi e sui vincoli. I nodi sono organizzati in una rete che indica le relazioni di connessione fra di essi con le informazioni per determinare: dove sono posti i nodi all'interno di una griglia, rappresentante lo spazio gestito, dove sono presenti i possibili percorsi e come possono essere attraversati. Questo modello viene memorizzato come **model data**; esso contiene tutte le informazioni necessarie alla pianificazione dei percorsi.

Nella figura 2.5 sono inoltre riportati anche i nodi e i percorsi riconosciuti dal generatore di modello. Si nota come i nodi (numero `130`) siano il risultato di intersezioni fra i percorsi o considerati come estremi di una tratta. Si osserva che i percorsi, rappresentati da frecce, possono essere unidirezionali o bidirezionali e con `M` ed `E` sono indicate le due tipologie di metodi per il calcolo della distanza compiuta in un viaggio attraverso varie locazioni: *Manhattan* o *Euclidea*.

Il **path generator** è lanciato da un **task**, il quale è inteso come un'operazione che richiede la generazione di un percorso. I **resource data** sono dati che descrivono le varie risorse presenti che possono essere assegnate nello svolgimento di un compito. Il generatore di percorsi, basandosi su di essi, valuta le risorse da usare a seconda dei vincoli imposti dal compito.

Esso può essere composto di diversi moduli, fra i quali il primo è il **layout analizer**: questo riceve la rete di nodi calcolata in precedenza e si occupa di identificare le informazioni sui possibili percorsi da inviare al modulo successivo. La sua presenza garantisce la possibilità di disaccoppiare funzionalmente il **path generator** dal **model generator**.

L'**algorithm determiner** fornisce la capacità di utilizzare uno o più algoritmi euristici nella determinazione del percorso, come descritto in precedenza. Si utilizzano gli euristici in quanto il calcolo di una soluzione ottima ideale in questo ambito può essere impraticabile [6].

Il **path calculator** applica gli algoritmi selezionati, identificando un percorso per il task. Questa soluzione può poi essere migliorata, entro certi dati limiti di soglia.

2.5 In sintesi

In questo capitolo sono state riassunte diverse fonti di letteratura sullo stato dell'arte, riguardo la modellazione dei magazzini e l'ottimizzazione dei percorsi.

Nella prima parte si sono tratti spunti su come poter rappresentare in modo formale la struttura e lo spazio da gestire per arrivare ad avere una rete di percorsi che si colleghi con le locazioni presenti negli scaffali del magazzino.

Nella seconda parte si sono recuperati gli algoritmi che possono essere usati allo scopo di ottimizzare gli spostamenti nel raggiungimento di un certo insieme di locazioni.

Tali informazioni saranno alla base dell'analisi e del progetto software al capitolo seguente.

Capitolo 3

Un software per la pianificazione e ottimizzazione di percorsi

Questo capitolo illustra il processo di sviluppo che ha portato alla realizzazione del software riguardante la pianificazione e l'ottimizzazione dei percorsi.

Seguendo i principi dell'ingegneria del software [10], si descrivono i passi della creazione del progetto: dopo un'introduzione, si passa alla definizione dei requisiti e alla loro successiva analisi, per poi esaminare il problema derivante. Infine, si descrivono il progetto, i pattern e la tecnologia utilizzati e si mostra un esempio di esecuzione del sistema.

3.1 Introduzione

Dopo aver analizzato, nel capitolo 1, gli aspetti logistici che determinano l'importanza dei magazzini industriali nella gestione della *supply chain* e a seguito della ricerca bibliografica sulla modellazione e sugli algoritmi, riportata nel secondo capitolo, l'attività di tirocinio si è focalizzata sulla creazione di un progetto software, che si approcciasse al problema della pianificazione e ottimizzazione dei percorsi, portando una possibile soluzione.

Il progetto si è svolto mediante un processo di sviluppo a spirale: a partire da certi requisiti iniziali, definiti in comune accordo con il tutor aziendale Claudio Gambetti, essi si sono via via raffinati, aggiunti o modificati, in base ai risultati dei prototipi presentati di volta in volta e a seguito di confronti e scambi di idee sui possibili casi d'uso da comprendere. In maniera analoga, interazioni di questo tipo avvengono fra committenti e software house, nello sviluppo di prodotti applicativi.

Durante questo periodo, i componenti dela *business unit industria* di Onit sono stati disponibili a seguito di richieste di consigli o suggerimenti sia riguardo il modo in cui affrontare il problema, sia sull'utilizzo specifico delle tecnologie Microsoft che non rientravano nella mia diretta conoscenza.

3.1.1 Problema

Come anticipato, il problema che ci si pone riguarda l'ottimizzazione dei percorsi da compiere durante le operazioni di movimentazione interna in un magazzino, per raggiungere un insieme di locazioni coprendo la minore distanza possibile. Si vuole considerare il contesto in cui un singolo operatore in movimento è un uomo che si sposta a piedi o a bordo carrello e che deve effettuare una serie di depositi/prelievi dalle locazioni di stoccaggio degli scaffali.

3.1.2 Visione

La visione principale nello sviluppo di questo come di un qualsiasi altro sistema software è che *non c'è codice senza progetto, non c'è progetto senza analisi del problema, non c'è problema senza requisiti*. Partendo da questi presupposti, il primo passo da affrontare è la definizione dei requisiti principali da soddisfare in un prototipo, per poi raffinarli ripetendo l'operazione sino ad un risultato soddisfacente.

3.1.3 Obiettivi

L'obiettivo di questo progetto è di sintetizzare le conoscenze acquisite durante l'attività di tirocinio e di sfruttare le compentenze

pregresse fornite in ambito accademico per produrre una soluzione software che funga da prototipo aziendale per la pianificazione e l'ottimizzazione dei percorsi nei magazzini industriali.

3.2 Requisiti

Si vuole avere uno strumento mediamente interattivo che agevoli l'utente nella fase di configurazione dei percorsi e che provveda a fornire una successione di tratte da percorrere necessarie allo svolgimento di movimentazioni interne, che devono raggiungere un certo insieme di locazioni negli scaffali.

Dunque, come prima fase, si delineano i requisiti del sistema software, definiti in accordo con il tutor aziendale:

1. Avere la possibilità di modellare graficamente il layout di un magazzino, disponendo gli scaffali in uno spazio bidimensionale.

2. Avere la possibilità di definire, all'interno dello stesso spazio, una mappa principale dei corridoi percorribili dagli operatori nel magazzino, che si muovono a piedi o a bordo carrello.

3. Tale layout deve essere riconfigurabile in ogni momento, ciò comporta il prevedere

   (a) l'aggiunta, modifica o eliminazione degli scaffali, e

   (b) l'aggiunta, modifica o eliminazione delle tratte dei corridoi.

4. Gli scaffali devono essere modificabili nelle loro proprietà di codice identificativo, lunghezza, profondità, tipo (mono-fronte o bi-fronte), orientamento, posizione (coordinate x e y).

5. Le campate degli scaffali devono essere modificabili singolarmente nelle loro proprietà di codice identificativo, lunghezza e profondità.

6. Le tratte devono poter rappresentare il senso di percorrenza.

7. A partire dagli scaffali e dalla mappa principale dei corridoi deve essere possibile generare una mappa dettagliata che collega ogni campata di scaffale alla tratta del corridoio da cui è accessibile.

8. Questa mappa dettagliata deve essere modificabile nei percorsi.

9. A partire dalla mappa dettagliata, deve essere possibile selezionare un insieme di campate e un punto di deposito, e generare un percorso (circuito) che ne descriva l'ordine di percorrenza, in maniera quanto più ottimizzata, considerando un ragionevole tempo di calcolo.

10. Deve essere possibile evidenziare ogni tratta che collega due punti consecutivi, riportandone il costo in distanza percorsa, e visualizzare il percorso completo, anche in questo caso riportandone la distanza totale compiuta.

11. Tale operazione deve essere ripetibile più volte, selezionado o deselezionando diverse campate e deposito.

12. Si prevedono due tipi di utenti: il configuratore del layout, che si occupa di configurare il layout del magazzino e il pianificatore delle operazioni, che si occupa di gestire le operazioni di movimentazioni da svolgere all'interno.

3.3 Analisi dei requisiti

Questa fase è necessaria perché i requisiti sono espressi in linguaggio naturale, il quale è ambiguo; l'obiettivo dell'analisi è definire i termini in maniera più chiara e univoca possibile.

3.3.1 Glossario

Il glossario contiene un elenco delle definizioni dei termini presenti nei requisiti, espresse in linguaggio naturale.

- **Magazzino**: contenitore di scaffali, rappresentato da un layout.
- **Layout**: particolare disposizione degli elementi (scaffali, corridoi) che compongono il magazzino.
- **Scaffale**: mezzo di immagazzinamento degli oggetti presenti nel magazzino. Da un'ottica bidimensionale dall'alto, appare come un rettangolo. Può essere di tipo *mono-fronte* (una sola fila di campate, accessibile da entrambi i lati dello scaffale) o di tipo *bi-fronte* (due file di campate adiacenti, ognuna delle quali è accessibile da un solo lato dello scaffale).
- **Lunghezza** dello scaffale: misura del lato dello scaffale che consente l'accesso agli elementi che contiene.
- **Profondità** dello scaffale: misura del lato dello scaffale ortogonale alla lunghezza.
- **Posizione** dello scaffale: coordinate cartesiane rispetto a un punto di riferimento.
- **Orientamento**: angolo che forma il vettore normale di un elemento rispetto un asse di riferimento.
- **Campata**: sezione verticale di uno scaffale, delimitata da montanti, comprendente più piani lungo la sua altezza. Unità dello scaffale raggiungibile da un percorso.
- **Operatore**: risorsa che movimenta le merci nel magazzino.
- **Corridoio**: spazio disponibile compreso tra due scaffali consecutivi o tra uno scaffale e un generico ostacolo (ad esempio un muro) che può essere attraversato da un operatore.
- **Tratta**: è un segmento compreso fra due punti, su ognuno dei quali può terminare o congiungersi con altre tratte. La lunghezza del segmento rappresenta il costo di attraversamento della tratta.
- **Percorso**: è formato da una tratta o una successione di tratte.

- **Circuito**: è una successione di tratte che inizia e termina sullo stesso punto.

- **Mappa**, o rete, dei percorsi: rappresentazione grafica dei percorsi riconosciuti nel magazzino. Può essere principale (inserita dall'utente) o dettagliata (calcolata dal programma, a partire da quella principale e dal layout).

- **Senso di percorrenza**: può essere bidirezionale o unidirezionale.

- **Distanza**: la distanza di un percorso è calcolata come somma dei costi di attraversamento delle tratte che lo compongono.

- **Deposito**: punto di inizio e punto di fine del circuito.

3.3.2 Casi d'uso

Il modello dei casi d'uso è un catalogo delle funzionalità del sistema descritte usando `UML`. Ogni caso d'uso rappresenta una singola interazione ripetibile attuata da un utente (uomo o macchina) durante l'utilizzo del sistema, enfatizzando la prospettiva dell'utente, e ne definisce i requisiti funzionali.

Tipicamente, un caso d'uso include uno o più scenari che descrive le interazioni che avvengono tra un attore e il sistema, e documenta il risultato e le eccezioni che possono avvenire.

I casi d'uso possono includerne altri come parte di un più grande pattern di interazione e possono essere estesi da altri casi d'uso.

Descrizione dei requisiti presenti in figura 3.1:

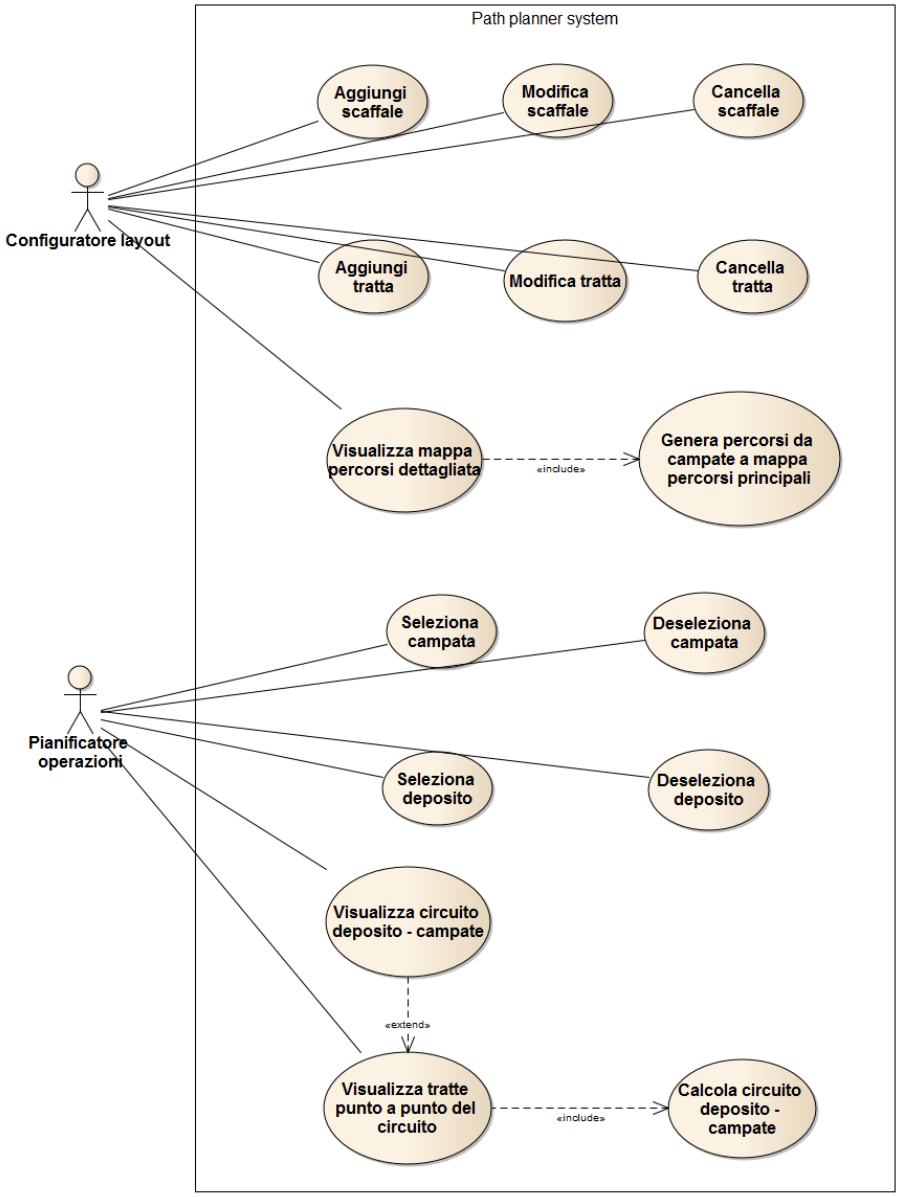

Figura 3.1: Casi d'uso

Campo	Descrizione
Nome	Aggiungi scaffale
Descrizione	Il configuratore del layout vuole aggiungere uno scaffale.
Attori	Configuratore layout, Path planner.
Precondizioni	-
Scenario principale	L'utente vuole aggiungere uno scaffale al layout del magazzino.
Scenari alternativi	-
Postcondizioni	Lo scaffale viene aggiunto al layout del magazzino.

Campo	Descrizione
Nome	Modifica scaffale
Descrizione	Il configuratore del layout vuole modificare uno scaffale.
Attori	Configuratore layout, Path planner.
Precondizioni	Lo scaffale deve essere già presente nel layout.
Scenario principale	L'utente vuole modificare uno scaffale al layout del magazzino.
Scenari alternativi	-
Postcondizioni	Lo scaffale viene modificato nelle sue proprietà.

Campo	Descrizione
Nome	Cancella scaffale
Descrizione	Il configuratore del layout vuole cancellare uno scaffale.
Attori	Configuratore layout, Path planner.
Precondizioni	Lo scaffale deve essere già presente nel layout.
Scenario principale	L'utente vuole cancellare uno scaffale dal layout del magazzino.
Scenari alternativi	-
Postcondizioni	Lo scaffale viene rimosso dal layout del magazzino.

Campo	**Descrizione**
Nome	Aggiungi tratta
Descrizione	Il configuratore del layout vuole aggiungere una tratta.
Attori	Configuratore layout, Path planner.
Precondizioni	-
Scenario principale	L'utente vuole aggiungere una tratta dal layout del magazzino.
Scenari alternativi	-
Postcondizioni	La tratta viene aggiunta al layout del magazzino.

Campo	**Descrizione**
Nome	Modifica tratta
Descrizione	Il configuratore del layout vuole modificare una tratta.
Attori	Configuratore layout, Path planner.
Precondizioni	La tratta deve essere già presente nel layout.
Scenario principale	L'utente vuole modificare una tratta dal layout del magazzino.
Scenari alternativi	-
Postcondizioni	La tratta viene modificata nelle sue proprietà.

Campo	**Descrizione**
Nome	Cancella tratta
Descrizione	Il configuratore del layout vuole cancellare una tratta.
Attori	Configuratore layout, Path planner.
Precondizioni	La tratta deve essere già presente nel layout.
Scenario principale	L'utente vuole cancellare una tratta dal layout del magazzino.
Scenari alternativi	-
Postcondizioni	La tratta viene cancellata dal layout del magazzino.

Campo	Descrizione
Nome	Visualizza mappa percorsi dettagliata
Descrizione	Il configuratore del layout vuole visualizzare graficamente la mappa dei percorsi dettagliata.
Attori	Configuratore layout, Path planner.
Precondizioni	La mappa dei percorsi dettagliata non è visualizzata. Deve essere possibile accedere al caso d'uso Genera percorsi da campate a mappa percorsi principali.
Scenario principale	L'utente vuole cancellare visualizzare la mappa dei percorsi dettagliata, che non è quella attualmente visualizzata dal sistema.
Scenari alternativi	-
Postcondizioni	La mappa dei percorsi dettagliata viene visualizzata.

Campo	Descrizione
Nome	Genera percorsi da campate a mappa percorsi principali
Descrizione	Genera i percorsi che collegano ogni campata di scaffale ai relativi corridoi presenti nella mappa dei percorsi principali.
Attori	Path planner.
Precondizioni	Nel layout deve essere presente almeno un elemento.
Scenario principale	Per visualizzare la mappa dei percorsi dettagliata, devono essere generati i percorsi.
Scenari alternativi	-
Postcondizioni	I percorsi che collegano le campate alla mappa principale vengono generati.

Campo	**Descrizione**
Nome	Seleziona campata
Descrizione	Il pianificatore delle operazioni vuole selezionare una campata.
Attori	Pianificatore operazioni, Path planner.
Precondizioni	La campata deve essere collegata alla alla mappa dei percorsi dettagliata. La campata non deve essere selezionata.
Scenario principale	Il pianificatore delle operazioni seleziona una campata da raggiungere in un'operazione di movimentazione interna.
Scenari alternativi	-
Postcondizioni	La campata viene selezionata.

Campo	**Descrizione**
Nome	Deseleziona campata
Descrizione	Il pianificatore delle operazioni vuole deselezionare una campata.
Attori	Pianificatore operazioni, Path planner.
Precondizioni	La campata deve essere selezionata.
Scenario principale	Il pianificatore delle operazioni deseleziona una campata da non raggiungere in un'operazione di movimentazione interna.
Scenari alternativi	-
Postcondizioni	La campata viene deselezionata.

Campo	Descrizione
Nome	Seleziona deposito
Descrizione	Il pianificatore delle operazioni vuole selezionare il punto deposito.
Attori	Pianificatore operazioni, Path planner.
Precondizioni	Deve essere presente almeno una tratta. Non deve essere selezionato alcun punto deposito.
Scenario principale	Il pianificatore delle operazioni seleziona il punto di deposito per un'operazione di movimentazione interna.
Scenari alternativi	-
Postcondizioni	Il punto deposito viene selezionato.

Campo	Descrizione
Nome	Deseleziona deposito
Descrizione	Il pianificatore delle operazioni vuole deselezionare il punto deposito.
Attori	Pianificatore operazioni, Path planner.
Precondizioni	Il punto deposito deve essere selezionato.
Scenario principale	Il pianificatore delle operazioni deseleziona il punto di deposito.
Scenari alternativi	-
Postcondizioni	Il punto deposito viene deselezionato.

Campo	**Descrizione**
Nome	Visualizza circuito deposito - campate
Descrizione	Il pianificatore delle operazioni vuole visualizzare il circuito deposito - campate calcolato dal sistema.
Attori	Pianificatore operazioni, Path planner.
Precondizioni	Deve essere stato calcolato il circuito deposito - campate.
Scenario principale	Il pianificatore delle operazioni vuole visualizzare il circuito da percorrere per un'operazione di movimentazione interna del magazzino.
Scenari alternativi	-
Postcondizioni	Il circuito deposito - campate viene visualizzato.

Campo	**Descrizione**
Nome	Visualizza tratte punto a punto del circuito
Descrizione	Il pianificatore delle operazioni vuole visualizzare le singole tratte fra deposito e campata o fra campata e campata del circuito calcolato dal sistema.
Attori	Pianificatore operazioni, Path planner.
Precondizioni	Deve essere stato calcolato il circuito deposito - campate.
Scenario principale	Il pianificatore delle operazioni vuole visualizzare il circuito da percorrere per un'operazione di movimentazione interna del magazzino.
Scenari alternativi	-
Postcondizioni	La singola tratta punto a punto fra deposito - campata o campata - campata viene visualizzata.

Campo	Descrizione
Nome	Calcola circuito deposito - campate
Descrizione	Il sistema deve calcolare il circuito deposito - campate.
Attori	Path planner.
Precondizioni	Deve essere selezionato un punto deposito e almeno una campata da raggiungere. La mappa dettagliata dei percorsi deve essere fatta in modo da rendere possibile la raggiungibilità di tutte le campate selezionate dal deposito.
Scenario principale	Il sistema richiede di calcolare il circuito deposito - campate.
Scenari alternativi	Se manca una delle precondizioni, il sistema notifica il tipo di errore
Postcondizioni	Il circuito deposito - campate viene calcolato.

3.3.3 Modello del dominio

Il modello del dominio è un modello concettuale di alto livello che cattura le informazioni essenziali riguardo le entità evocate nei requisiti; è una vista sugli oggetti che hanno a che fare con l'area di interesse e le loro relazioni. Esso viene utilizzato per rappresentare gli oggetti significativi all'interno di un dominio.

Il modello è rappresentato in figura 3.2.

3.4 Analisi del problema

L'analisi del problema è essenziale per valutare la complessità e i rischi, pianificare il lavoro e decidere i team di progettazione e sviluppo. Lo scopo primario è valutare la qualità del problema, se è gestibile facilmente o meno, per sapere come organizzare l'attività. È la fase più qualificante del processo di produzione e nella quale la dimensione dell'interazione assume notevole importanza fra quelle da analizzare, in quanto è quella che ha impatto maggiore sull'ar-

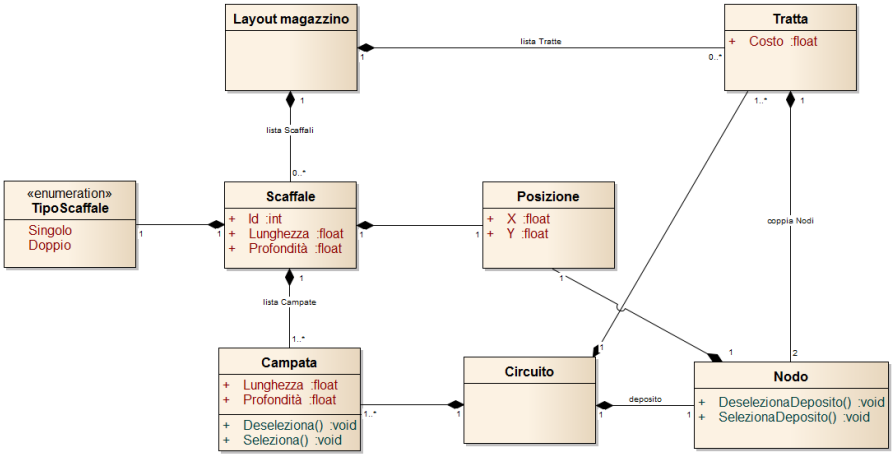

Figura 3.2: Modello del dominio

chitettura del sistema. Dunque come prima cosa bisogna pensare e simulare il sistema, senza focalizzarsi sul *come*, ma sul *cosa* deve fare.

Grazie all'esperienza formulata nel capitolo 1 e alla letteratura sintetizzata nel secondo capitolo, si ritiene di avere una sufficiente conoscenza per affrontare il problema, giudicato risolvibile mediante i concetti della programmazione orientata agli oggetti e facendo uso di una delle tante piattaforme esistenti che si basano su questo paradigma.

I punti rilevanti del problema riguardano la manipolazione delle entità del layout per arrivare, in una prima fase, ad ottenere una mappa completa dei percorsi fra tutti i punti considerati raggiungibili e, in una seconda fase, al sottoporre questa entità ai calcoli necessari per generare i percorsi richiesti dall'utente.

Una scelta importante che ricadrà in fase di progetto sarà quella di adottare degli algoritmi appositi per la fase di calcolo dei percorsi, facendo riferimento alla letteratura riportata nel capitolo 2.

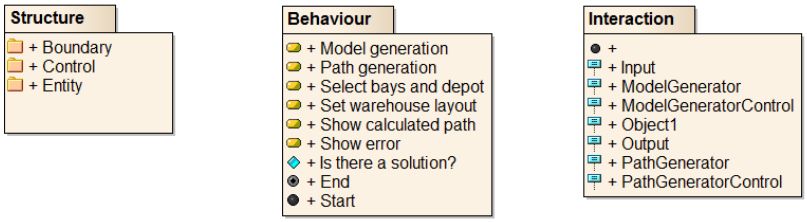

Figura 3.3: Architettura logica

3.4.1 Architettura logica

L'architettura logica è una *mappa del cosa*: indica come si collocano le varie parti all'interno del sistema complessivo, in modo da poter assemblare i vari componenti sviluppati indipendentemente; è ciò che rimane di un sistema quando non si può più togliere nulla, continuando a comprenderne la struttura e il funzionamento. È logica perché, a differenza di quella di progetto, osserva il problema in sé, non proietta la soluzione, visualizza la criticità dello stesso e dovrebbe essere abbastanza univoca rispetto al problema. Viene espressa in un linguaggio non ambiguo e che sia comprensibile a tutti.

Per rappresentare l'architettura logica è stato scelto il linguaggio `UML`. Il modello è suddiviso nelle tre dimensioni di *struttura*, *comportamento* e *interazione*, 3.3.

La struttura logica del sistema è riportata in figura 3.4. Essa è rappresentata mediante il pattern *boundary-control-entity*; questo approccio suddivide un sistema software nelle parti di visualizzazione, di implementazione e di dati associati.

La parte di *entity* descrive gli oggetti che rappresentano la semantica delle entità nel dominio applicativo, mettendo in luce quelle che sono state riconosciute in fase di analisi e le loro inter-relazioni, mentre la parte di *control* descrive gli oggetti che percepiscono gli eventi generati dall'utente e controllano l'esecuzione del processo di business e rappresenta un supporto per le operazioni e le attività. Infine la parte di *boundary* descrive gli oggetti che rappresentano l'interfaccia tra un attore ed il sistema; identifica la parte di confine fra il

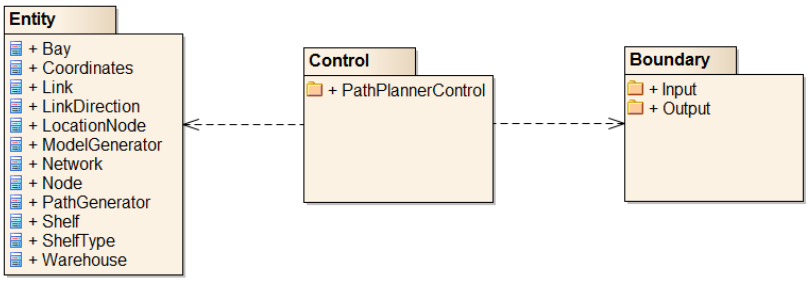

Figura 3.4: Struttura

sistema e l'utente.

Il diagramma `UML` delle entità è riportato in figura 3.5. Si sono riconosciute due entità principali che sono il `ModelGenerator` e il `PathGenerator`. La prima contiene il riferimento al magazzino (`Warehouse`), il quale ha una lista di scaffali (`Shelf`), a loro volta composti da un insieme di campate (`Bay`). Queste ultime due entità hanno anche il riferimento alla loro posizione (`Coordinates`) nello spazio. Il magazzino, inoltre, contiene anche due differenti tipi di reti di percorsi (`Network`); la prima è quella che viene definita dall'utente `MainPathsNetwork`), mentre la seconda (`DetailedPathsNetwork`) è quella generata dal sistema (e che l'utente può comunque modificare). La classe `Network` è definita da una lista di punti (`Node`), con la relative posizioni, che rappresentano gli estremi delle tratte (`Link`). Le campate si collegano concettualmente ai percorsi mediante un tipo di nodo più specifico, chiamato `LocationNode`, il quale ha il riferimento a una specifica campata.

La parte di *control*, rappresentata dal `PathPlannerControl`, è ulteriormente suddivisa in `ModelGeneratorControl` e `PathGeneratorControl`; ognuno che fornisce le rispettive funzionalità, come mostrato in figura 3.6.

Mediante un diagramma delle attività si è rappresentata la dimensione del *comportamento* del sistema 3.7. Come si può notare, l'esecuzione comincia con il settaggio del layout del magazzino (con l'input dell'utente) per poi far generare il modello completo dei percorsi al sistema. Nella fase di pianificazione successiva, l'utente

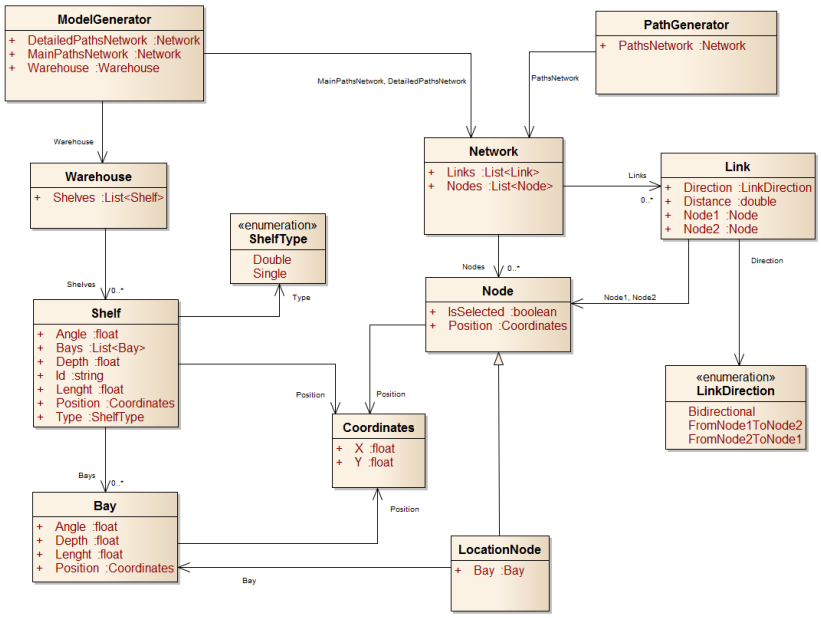

Figura 3.5: Entity

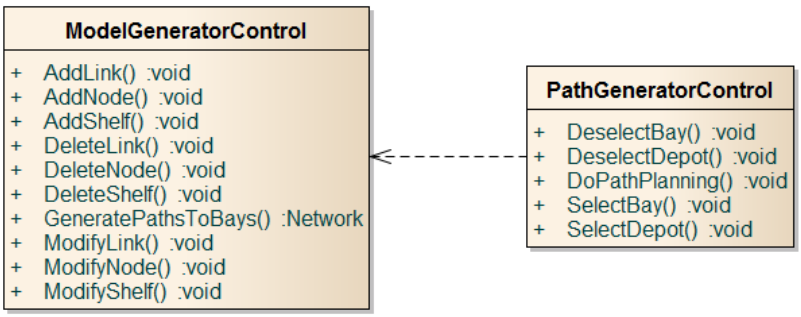

Figura 3.6: Dettaglio di PathPlannerControl

seleziona le campate e il deposito e il sistema esegue la generazione del percorso. A questo punto, se è stata trovata una soluzione, essa viene visualizzata, altrimenti si passa nello stato di visualizzazione dell'errore.

Infine, si riporta il diagramma delle interazioni 3.8. Si nota come sia il *control* a gestire il flusso di controllo, prendendo gli input dell'utente, svolgendo le operazioni richieste ed aggiornando la visualizzazione.

3.4.2 Piano di collaudo

Il piano di collaudo è necessario alla definizione della semantica delle operazioni fornite dalle entità scaturite dall'analisi. Per questo sono stati implementati un insieme di funzioni di test, allegati nella soluzione del progetto.

3.4.3 Piano di lavoro e analisi dei rischi

Durante l'analisi non sono stati evidenziati particolari rischi nella risoluzione del problema; si procede, dunque, a un piano di lavoro che prevede lo sviluppo di un primo prototipo quanto prima, che serva per poter rifinire i requisiti. Ciò innesca una serie di processi a spirale che terminano quando il risultato conseguito è giudicato soddisfacente da entrambe le parti.

3.5 Progetto

Sulla base della specifica dei requisiti prodotta in analisi, il progetto definisce come tali requisiti saranno soddisfatti, entrando nel merito della struttura che deve essere data al sistema software da realizzare. La progettazione rimane comunque una fase distinta dalla programmazione, che corrisponde alla traduzione in un particolare linguaggio di programmazione delle decisioni prese in sede di progettazione.

Una prima scelta progettuale riguarda la specifica piattaforma sulla quale implementare la soluzione al problema analizzato nella

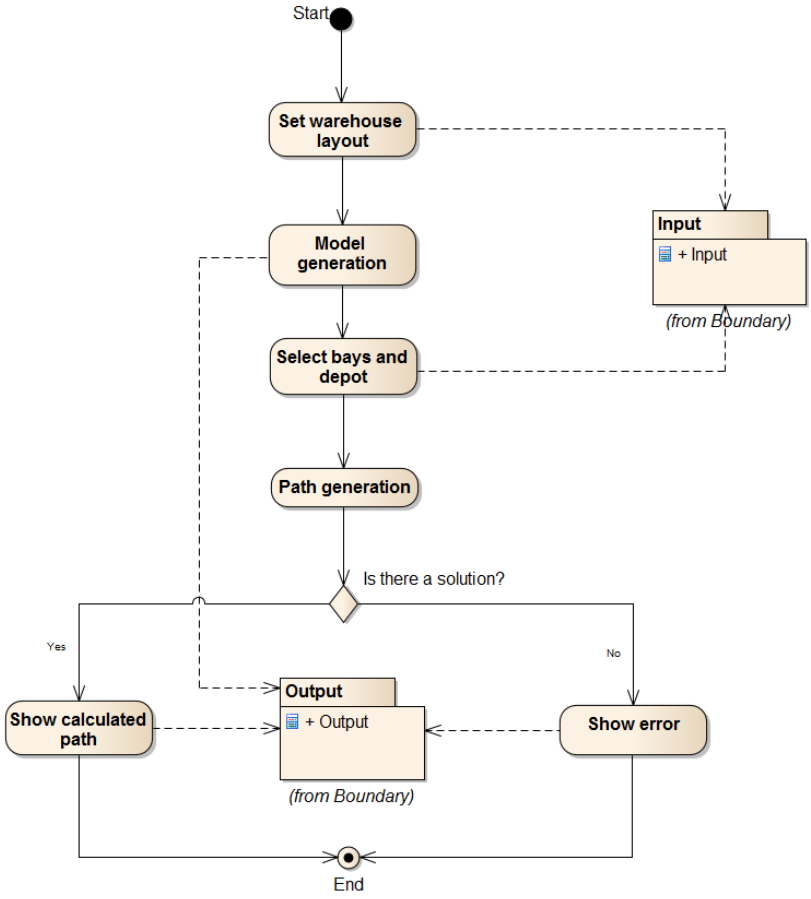

Figura 3.7: Behaviour

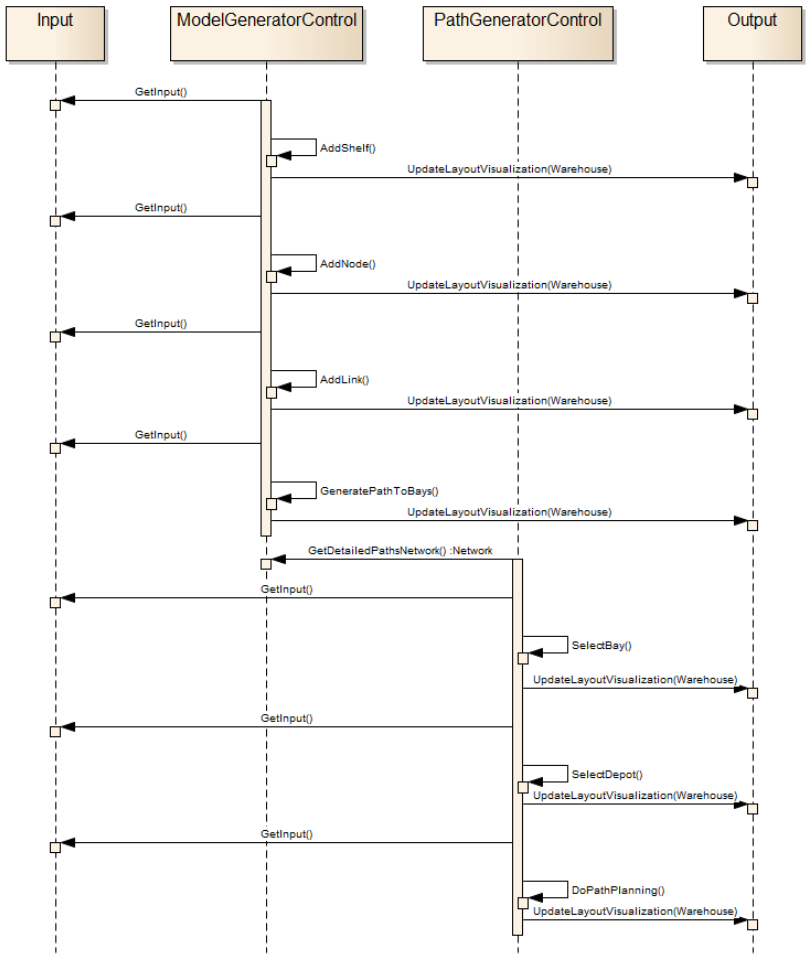

Figura 3.8: Interaction

sezione precedente: la decisione è ricaduta sul framework Microsoft `.NET` e l'utilizzo delle librerie `WPF`, implementandole mediante linguaggio `Visual C#`. Le tecnologie Microsoft sono fortemente sfruttate da Onit nello sviluppo dei suoi prodotti, dunque questa scelta è stata anche dettata dal fatto di poter imparare più a fondo, sia singolarmente che con il supporto della *BU industria*, questo tipo di strumenti.

Una seconda scelta progettuale, fortemente collegata alla prima, è sul pattern architetturale che si è deciso di utilizzare nel modellare il sistema.

Infine, un'altra scelta importante riguarda gli specifici algoritmi da utilizzare nella generazione del modello e nella generazione dei percorsi.

Si descrivono queste decisioni in dettaglio nelle seguenti sottosezioni.

3.5.1 Scelta tecnologica

Come anticipato, si sceglie di realizzare la soluzione utilizzando il framework `.NET 4.0`. Esso è un componente di Windows che supporta la compilazione e l'esecuzione di applicazioni o servizi Web; tale framework fornisce un ambiente di programmazione coerente orientato agli oggetti, un ambiente di esecuzione gestito, sviluppo e distribuzione semplificati e l'integrazione con vari linguaggi di programmazione.

È formato da due componenti principali: il *Common Language Runtime* (`CLR`) e la *libreria di classi* `.NET` framework. Il primo è un agente che gestisce il codice a *run time*, fornendo un ambiente con servizi di base quali gestione della memoria, thread, servizi remoti, sicurezza, ed efficienza, mentre il secondo è un insieme completo *object-oriented* di tipi riutilizzabili nello sviluppo, di cui fanno parte `ADO.NET`, `ASP.NET`, *Windows Form* e *Windows Presentation Foundation*.

All'interno di questo framework si sceglie di utilizzare *Windows Presentation Foundation* (`WPF`), ovvero un sistema di presentazione per applicazioni client Windows automonome o su browser, sottoinsieme dei tipi `.NET`. Contiene un set di funzionalità che includono *Extensible Application Markup Language* (`XAML`), controlli, binding ai

Figura 3.9: Associazione ai dati (fonte: MSDN)

dati, layout, grafica 2D e 3D, eccetera. Si ha la possibilità di sviluppare applicazioni usando sia un linguaggio di *markup* che *code-behind*. Il primo è realizzato da `XAML`, un linguaggio basato su `XML`, per implementare l'aspetto in modo dichiarativo, mediante una struttura di elementi ad albero. Il *code-behind*, al contrario, implementa le funzionalità in risposta a interazioni utente, gestione di eventi, chiamata alla logica di business e accesso ai dati. L'associazione ai dati può avvenire mediante la classe `Binding`, per relazionare la proprietà di dipendenza di un controllo di *destinazione* a una proprietà di un oggetto dati di *origine* dell'associazione. In tal modo, mediante un meccanismo definito dall'interfaccia `INotifyPropertyChanged`, il controllo associato alla sorgente viene notificato di aggiornarsi per presentare l'attuale valore della sorgente.

L'oggetto che realizza il *binding* ha una proprietà (*Mode*) per definire la modalità dell'associazione, ovvero *OneWay*, *TwoWay*, *OneWayToSource* e *OneTime*. Nella modalità *OneWay*, la proprietà di dipendenza viene aggiornata da quella sorgente, ma non può accadere il contrario; ciò succede, ad esempio, quando la destinazione dell'associazione è solo un output per il sistema. Per far aggiornare anche la sorgente, a fronte di cambiamenti della destinazione (che possono avvenire, ad esempio, a seguito di un input dell'utente) è necessario settare la proprietà *Mode* come *TwoWay*.

L'utilizzo di questo meccaniscmo è facilitato dall'utilizzo dal pattern di progettazione *Model-View-ViewModel*, che è stato scelto per rappresentare il sistema software e che viene presentato nella sotto-sezione successiva.

È importante dare, inoltre, un accenno al modello di *threading* di

WPF. Esso è progettato per evitare il ricorso alle difficoltà del threading, quando possibile. Le applicazioni WPF sono in genere avviate con due thread: uno per la gestione del rendering e l'altro per la gestione dell'interfaccia utente (*user interface*, UI). Il primo è eseguito in background, mentre il secondo riceve gli input, gestisce gli eventi, aggiorna la visualizzazione ed esegue il codice dell'applicazione. Il thread UI accoda gli elementi di lavoro in un oggetto denominato *Dispatcher*, che seleziona i task in base alle priorità e li esegue singolarmente fino al completamento. Ogni thread UI deve presentare almeno un oggetto *Dispatcher* e ogni oggetto *Dispatcher* può eseguire i task esattamente in un thread. Dunque, se è necessario svolgere azioni complesse, queste dovranno essere svolte in un thread separato che lasci quello UI riservato agli elementi in coda al *Dispatcher*; al termine, il risultato viene mandato al thread UI per la visualizzazione. Questo accade perché a un elemento grafico in Windows può accedere solo il thread che l'ha creato; un thread in background può chiedere a quello UI di eseguire un'operazione per suo conto, registrando un task sul *Dispatcher* del thread UI, con una certa priorità (metodi Invoke, BeginInvoke).

Si sceglie, infine, di realizzare la soluzione utilizzando il linguaggio di programmazione Visual C#, che è l'implementazione del linguaggio C# fatta da Microsoft.

Per ognuno di questi argomenti si potrebbe dedicare un capitolo, ma la loro approfondita trattazione esce dal contesto di questa tesi, sebbene il loro studio e apprendimento abbia fatto parte in maniera significativa dell'attività di tirocinio.

3.5.2 Il pattern Model-View-ViewModel

Il *Model-View-ViewModel* è un pattern architetturale che nasce come variazione al *Presentation-Model* di Martin Fowler (2004), operata nel 2005 da John Gossman (Microsoft).

L'approccio *Model-View-ViewModel* (MVVM) si presta naturalmente a piattaforme di applicazioni XAML, poiché influenza alcune specifiche capacità di WPF, come il data binding, i controlli e il comportamento. Questo pattern separa le responsabilità dell'aspetto del layout dell'interfaccia grafica dalla logica di presentazione, ripren-

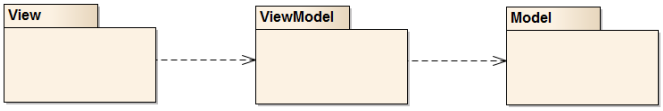

Figura 3.10: Model-View-ViewModel

dendo i concetti dal *Presentation Model* (che è nato per avere un'astrazione *platform-independent*), con l'obiettivo di avere un metodo standard per sfruttare le funzionalità chiave di `WPF`.

La **view** definisce la struttura, il layout e l'apparenza di ciò che vede l'utente; idealmente, viene definita interamente in `XAML`, con un limitato code-behind che non contiene business logic.

Il **model** è l'implementazione del modello di dominio dell'applicazione, che può includere dati del modello con logica di business e validazione. Spesso include un layer di accesso ai dati.

Il **view-model** è l'intermediario fra i due ed è responsabile di gestire la logica della *view*. Recupera i dati dal *model* e li rende disponibili alla *view*, in modo che siano facili da usare per essa, eventualmente trasformandoli. Fornisce anche le implementazioni dei comandi che l'utente avvia nella *view* ed è anche responsabile dei cambiamenti di stato locale che influenzano aspetti di visualizzazione.

È importante capire le interazioni fra questi tre componenti, come si nota dalla figura 3.10; la *view* è consapevole del *model* che è consapevole del *model*, al contrario il *model* non conosce il *view-model* che è inconsapevole della *view*.

Tipicamente, il *view-model* interagisce con il *model* mediante chiamate di metodi e quest'ultimo può anche riportare errori o cambiamenti ai dati sottostanti usando un set standard di eventi a cui il *view-model* si sottoscrive.

Uno dei benefici del `MVVM` coinvolge la fase di sviluppo, nella quale i designer e gli sviluppatori possono lavorare in maniera indipendente e concorrente sui componenti; ad esempio, i primi possono concentrarsi sulla *view*, mentre i secondi lavorano su componenti di *model* e *view-model*, creando unità di test senza usare l'interfaccia grafica. Inoltre, se la *view* è definita in `XAML` risulta semplice ridi-

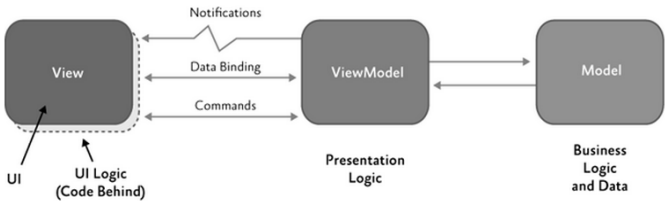

Figura 3.11: Interazione fra componenti del MVVM (fonte: MSDN)

segnarla senza agire sul resto del codice. I vantaggi riguardano maggiormente le applicazioni complesse e di lungo ciclo di vita, che divengono più facili da mantenere, testare, evolvere e riusare, mentre per applicazioni semplici o più brevi il lavoro addizionale per implementare il pattern può non valerne lo sforzo.

Facendo riferimento alla figura 3.11, la *view* può aggiornare automaticamente i valori visualizzati in risposta a cambiamenti del `VM` sottostante, se quest'ultimo implementa l'interfaccia `INotifyPropertyChanged`; inoltre, se la *view* vuole anche aggiornare il `VM`, il *binding* deve essere di tipo *two-way*. In genere, fra `V` e `VM` c'è una relazione uno a uno. In generale, si possono prevedere comportamenti in risposta a azioni dell'utente, da inserire nel *code-behind*, creando un *event-handler*; nel pattern `MVVM` la responsabilità di implementare l'azione è nel *view-model*, potendo connettere un controllo a un metodo del `VM` con un *command binding*.

In alcuni casi, la *view* può implementare comportamenti visuali nel *code-behind* che sarebbero difficili o inefficienti da esprimere in `XAML`. Essa riferisce il `VM` mediante la sua proprietà `DataContext`.

Come detto, il *view-model* non ha riferimenti alla *view* o alla sua specifica implementazione, ed implementa proprietà e comandi ai quali la `V` si può collegare, rendendo disponibili le funzionalità. Negli scenari più usuali, il `VM` converte e manipola i dati del *model* dimodoché siano più facili da consumare nella `V` e può anche definire proprietà addizionali che non farebbero parte del modello.

A volte non è semplice scegliere dove inserire certe funzionalità; la regola generale è che ogni cosa concernente la specifica apparenza della interfaccia grafica, che può essere ridisegnata in un secondo momento, deve rientrare nella *view*, ed il codice che manipola di-

rettamente elementi visuali deve risiedere nel *code-behind*. Analogamente, le parti di recupero e manipolazione dei dati da visualizzare dovranno andare nel `VM`.

Infine, il *model* contiene tutta la logica applicativa che si occupa del recupero e della gestione dei dati, facendo in modo che siano rispettate le regole di consistenza e validità; per massimizzare la riusabilità, non deve contenere casi o task specifici. Esso può implementare direttamente gli strumenti per collegare i dati alla `V`, ma nel caso in cui non lo faccia è il `VM` che sarà responsabile di fare il *wrapping* delle classi del modello e fornire le proprietà richieste all'interfaccia.

3.5.3 Algoritmo di generazione del modello

Per quanto riguarda la fase di generazione del modello, ovvero la parte in cui al sistema è richiesto di collegare le campate degli scaffali ai percorsi principali inseriti dall'utente, si decide di operare in questa maniera: dato l'angolo che lo scaffale forma con l'asse verticale del sistema di riferimento, ad ogni campata si associa un vettore normale, che ha direzione uguale e verso opposto a quella di accesso alla locazione. A partire dalla posizione centrale della campata, si percorre la direzione indicata dal versore, fino ad intersecare (se presente) il percorso inserito dall'utente. Se non è presente alcun percorso lungo quella direzione, o se viene intercettato prima un altro scaffale (che è visto come un ostacolo, in questo caso) la campata rimane non collegata. Si svolgono una serie di iterazioni che coinvolgono, per ogni campata, tutte le tratte tracciate dall'utente, andando ad ogni passo a trovare quella raggiungibile che, lungo la direzione di ricerca, risulta essere la più vicina.

Per prima cosa, è necessario stabilire un sistema di riferimento. Si considera il punto di origine (0,0) quello in alto a sinistra, con le ascisse cresenti verso destra e le ordinate crescenti verso il basso.

Inoltre, come si mostra in figura 3.12, si decide di rappresentare la posizione e l'orientamento di uno scaffale mediante un versore applicato nell'estremo evidenziato dello scaffale, con l'angolo relativo a quello formato con l'asse delle y.

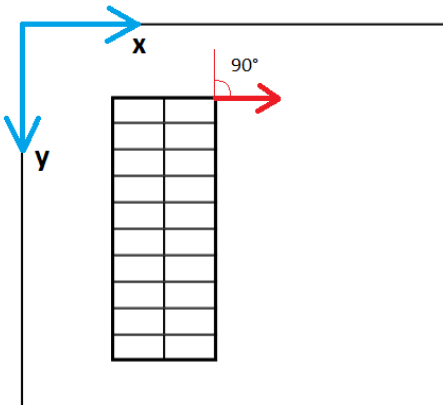

Figura 3.12: Sistema di riferimento

Posizione e orientamento dello scaffale si possono rappresentare in vari modi, a seconda del punto di riferimento che si vuole prendere e dell'angolo impostato (figura 3.13). A partire dal versore di orientamento allo scaffale, si generano poi tutti i vettori unitari delle campate che contiene (figura 3.14).

Dati due punti $A(x_1, y_1)$ e $B(x_2, y_2)$, la distanza calcolata è quella euclidea:

$$AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Considerando una data campata e una certa tratta, il problema riguarda l'intersezione fra due rette, più precisamente:

- La prima retta a è passante per il punto P di posizione della campata e per quello $P'(x_2, y_2)$ ottenuto a partire dalla posizione della campata $P(x_1, y_1)$ più uno spostamento infinitesimo lungo l'asse X e Y, dato dalle relative componenti del vettore normale $\hat{v} = (v_x, v_y)$:

$$\begin{cases} x_2 = x_1 + v_x \cdot \epsilon \\ y_2 = y_1 + v_y \cdot \epsilon \end{cases}$$

  dove ϵ è un numero piccolo vicino allo 0.

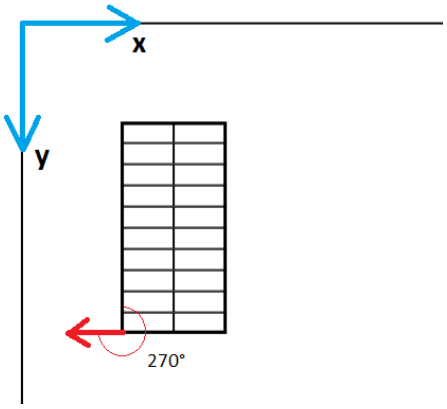

Figura 3.13: Posizione e orientamento di uno scaffale

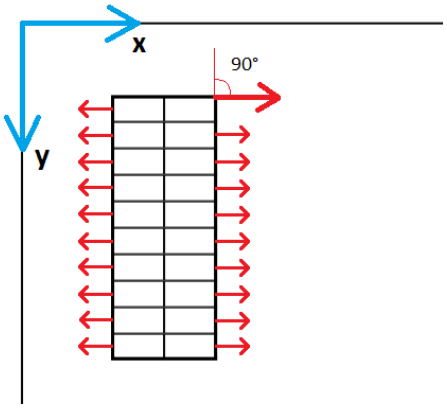

Figura 3.14: Orientamento delle campate

- La seconda retta b è quella passante per i punti estremi della tratta in esame definita dall'utente.

Avendo ora ottenuto le due rette a e b, si procede al calcolo della loro intersezione.

Data l'equazione implicita della retta:

$$Ax + By + C = 0$$

e dati i parametri:

$$\begin{cases} A = y_1 - y_2 \\ B = x_2 - x_1 \\ C = x_1 \cdot y_2 - y_1 \cdot x_2 \end{cases}$$

dove (x_1, y_1) e (x_2, y_2) sono due generici punti appartenenti alla retta, l'intersezione $P_{int}(x_{int}, y_{int})$ fra due rette

$$Ax + By + C = 0$$

e

$$A'x + B'y + C' = 0$$

vale:

$$\begin{cases} x_{int} = \frac{B'(A'C - AC') - C'(A'B - AB')}{A'(A'B - AB')} \\ \\ y_{int} = \frac{AC' - A'C}{A'B - AB'} \end{cases}$$

La rappresentazione grafica è riportata in figura 3.15.

A questo punto, l'algoritmo memorizza i valori di distanza fra P e i $P_{int,n}$ trovati, dove $P_{int,n}$ è il punto di intersezione riguardante la tratta n-esima, andando poi ad inserire effettivamente nella rete dei percorsi il collegamento fra la campata e la tratta raggiungibile che ha il valore di distanza minore ($P^*_{int,n}$).

Una tratta risulta non raggiungibile da una campata se il loro collegamento viene intersecato da un altro scaffale (figura 3.16). Per rilevare gli ostacoli è stato quindi necessario considerate anche eventuali intersezioni fra la retta a e i lati degli scaffali; se la distanza fra il punto P e $P^*_{int,m}$, dove $P_{int,m}$ è il punto di intersezione fra la retta a e il segmento di ostacolo m-esimo, e $P^*_{int,m}$ è quello più vicino

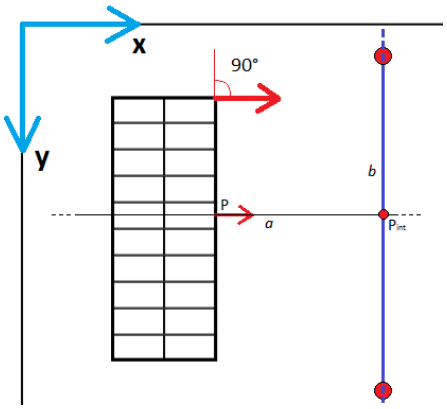

Figura 3.15: Calcolo dell'intersezione fra la retta della campata e del percorso principale

a P, è minore di $P^*_{int,n}$, allora il punto P della campata risulta non raggiungibile.

Trovato un $P^*_{int,n}$, viene inserito nella rete di percorsi un nodo (`Node`) in tale posizione (se non esiste già), modificando di conseguenza la struttura dei `Link` per collegarlo con il `LocationNode` in P e mantenendo coerente le direzioni delle tratte impostate dall'utente. Il nodo in P^* viene poi etichettato come aggiunto dal sistema e per questo non sarà poi selezionabile come nodo deposito in fase di pianificazione dei percorsi.

In tal modo, vengono collegate tutte le campate possibili con nuove tratte, andando a formare una rete di nodi e collegamenti che viene utilizzata nel passo successivo.

3.5.4 Algoritmo di generazione del percorso

Una volta generato il modello del passo precedente, anch'esso eventualmente modificabile manualmente dall'utente, esso può essere utilizzato per la generazione del percorso da compiere nel raggiungere le campate scelte.

A questo punto, a partire da questo grafo, si sceglie di far generare al sistema un **grafo ausiliario completo** su di esso, ovvero:

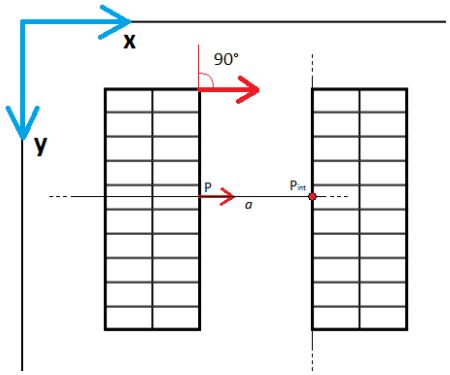

Figura 3.16: Calcolo dell'intersezione fra la retta della campata e di uno scaffale di ostacolo

si produce un grafo che ha un nodo ausiliario (`AuxiliaryNode`) per ogni `LocationNode` e per ogni possibile nodo deposito del grafo iniziale e che contiene un **arco ausiliario** (`AuxiliaryLink`) per ogni coppia di nodi ausiliari (ove sussistano i requisiti di raggiungibilità fra le coppie). Un **arco ausiliario** è formato dagli archi che che realizzano il percorso di costo minimo fra i nodi ausiliari estremi, può attraversare più nodi del grafo iniziale e ha per costo la somma dei costi degli archi che contiene. Un arco (`Link`) ha infatti associato un costo pari alla distanza euclidea tra i suoi due nodi estremi. In tal modo, si ha un grafo che rivela il percorso minimo per andare da un qualsiasi `LocationNode` della rete ad un altro. L'algoritmo per la produzione di questo grafo ausiliario è relativo al *shortest path problem* (2.3.7).

Per sviluppare questa funzionalità, si implementa una struttura dati ulteriore, la classe `DijkstraNode`, utile allo svolgimento dell'*algoritmo di Dijkstra*. Questa è la parte che risulta più gravosa dal punto di vista computazionale, coinvolgendo la totalità di nodi della rete e considerando la complessità di $O(n^2)$. L'algoritmo di Dijkstra viene svolto per ogni nodo ausiliario; dato n il numero di questi nodi, ad ogni iterazione si ottengono $n - 1$ archi ausiliari che connettono il nodo in questione a tutti gli altri.

A questo punto, disponendo già di tutti i percorsi minimi fra

i punti, dato l'insieme delle campate (ovvero, dei `LocationNode`) da raggiungere e il nodo deposito, non resta fare altro che mettere in ordine gli archi ausiliari che collegano i vari nodi, per ottenere l'esito dell'ottimizzazione richiesta. Questa parte si risolve mediante l'algoritmo euristico *nearest neighbor* (2.3.5), di complessità $O(n^2)$. Questo algoritmo non opera sulla totalità del grafo ausiliario, ma solo sulla parte necessaria, estraendo solo i nodi selezionati dall'utente e i relativi archi ausiliari.

A livello di visualizzazione per l'utente, per visualizzare quale percorso prendere per andare da un punto all'altro, è sufficiente evidenziare gli archi contenuti da quelli ausiliari, potendo usufruire sia della distanza complessiva del circuito, sia della distanza di ogni singola tratta punto a punto.

Questo tipo di suddivisione algoritmica consente di avere, a fronte di una fase di configurazione computazionalmente più pesante nella costruzione del grafo ausiliario, dei tempi di risposta molto più brevi nel mostrare il risultato all'utente, una volta inserito l'input delle campate da raggiungere. Infatti, si è analizzato come il numero delle riconfigurazioni a livello di layout e di percorsi sia trascurabile rispetto al numero delle pianificazioni richieste sullo stesso layout, dunque è stato considerato ragionevole optare per una scelta di questo tipo.

3.5.5 Scelte per visualizzazione e interazione utente

L'interazione tra software e utente avviene mediante dispositivi di input e output. L'output avviene mediante lo schermo, rappresentando graficamente il layout del magazzino e la rete di percorsi. D'altro canto, l'utente invia input al sistema mediante mouse e tastiera: tramite appositi bottoni può impartire varie azioni di input per inserire i dati del layout (scaffali e percorsi principali). Per favorire la configurazione grafica del magazzino, si decide di sfruttare il *drag-and-drop*, dimodoché gli oggetti presenti possano essere spostati con il mouse. Per gli scaffali (e le relative campate), è prevista una maschera per impostarne le proprietà in maniera precisa, mentre per inserire nodi e percorsi è prevista solo la modalità tramite mouse. Per aiutare il disegno ortogonale delle tratte, si inserisce una

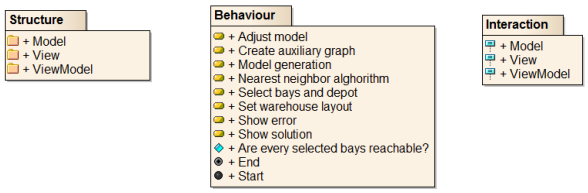

Figura 3.17: Architettura di progetto

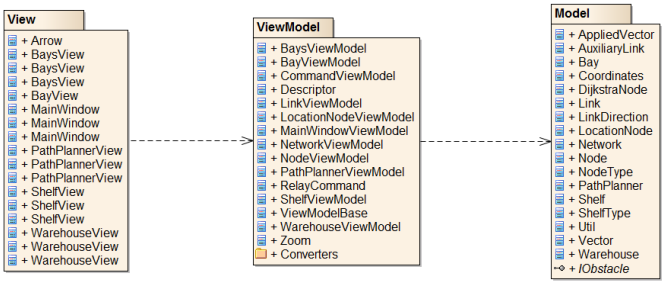

Figura 3.18: Structure

funzionalità che rende perpendicolari le tratte che si discostano di poco dall'esserlo. Infine, per supportare l'utente nella visualizzazione del layout, si sceglie di aggiungere le funzionalità di *zoom-in* e *zoom-out*.

3.5.6 Architettura

Date tutte le scelte e considerazioni di cui sopra, vengono presentati i diagrammi `UML` dell'architettura di progetto. Come per l'architettura logica, si riporta il modello del progetto sulle tre dimensioni di struttura, comportamento e interazione (figura 3.17).

Per quanto riguarda la struttura, si applica il pattern *Model-View-ViewModel*, seguendo le considerazioni fatte nella sezione 3.5.1 (figura 3.18).

Il *model* è rappresentato in figura 3.19. La struttura del modello rimane molto simile a quella dell'analisi, fermo restando le entità

aggiunte per implementare la soluzione.

In figura 3.20 è presente invece il diagramma del comportamento; anch'esso risulta molto simile a quello creato in fase di analisi, con in più qualche dettaglio, ad esempio al tipo di algoritmo svolto nella fase di generazione del percorso.

Nel diagramma delle interazioni di figura 3.21 si vogliono rappresentare solo le macro-interazioni fra `View`, `ViewModel` e `Model`. Un diagramma completo, in questo contesto, sarebbe stato altrimenti di poca comprensibilità. A seguito di un'interazione dell'utente con la *view*, si innescano a cascata dei comportamenti a livello del *view-model* e infine altri eventuali comportamenti nel *model* con un possibile aggiornamento dei dati. Al termine delle operazioni (al ritorno dalla chiamata a funzione, nel primo caso, oppure al lancio di un evento del modello, nell'ultimo caso), ciò comporta la notifica, da parte del *view-model* del cambiamento dei dati nel modello alla *view*, dimodoché essa possa aggiornarsi. Nel caso al centro, dopo un'interazione utente con la *view*, vengono invece aggiornati solo i dati di visualizzazioni appartententi al *view-model*, senza toccare il modello.

Dopo aver realizzato l'architettura di progetto, si svolge la fase di implementazione mediante l'ambiente di sviluppo *Microsoft Visual Studio 2010 Premium*. Nella sezione successiva si riportano una serie di schermate, come esempio dell'esecuzione del software.

3.6 Esecuzione

In questa sezione si riporta una sequenza di immagini che rappresentano un esempio di esecuzione del sistema software sviluppato.

Dopo aver inserito un insieme di scaffali, il layout di presenta come in figura 3.22.

Selezionando uno (o più) scaffali, mediante il tasto sinistro del mouse, si aprono altre viste per modificarne le proprietà (figura 3.23).

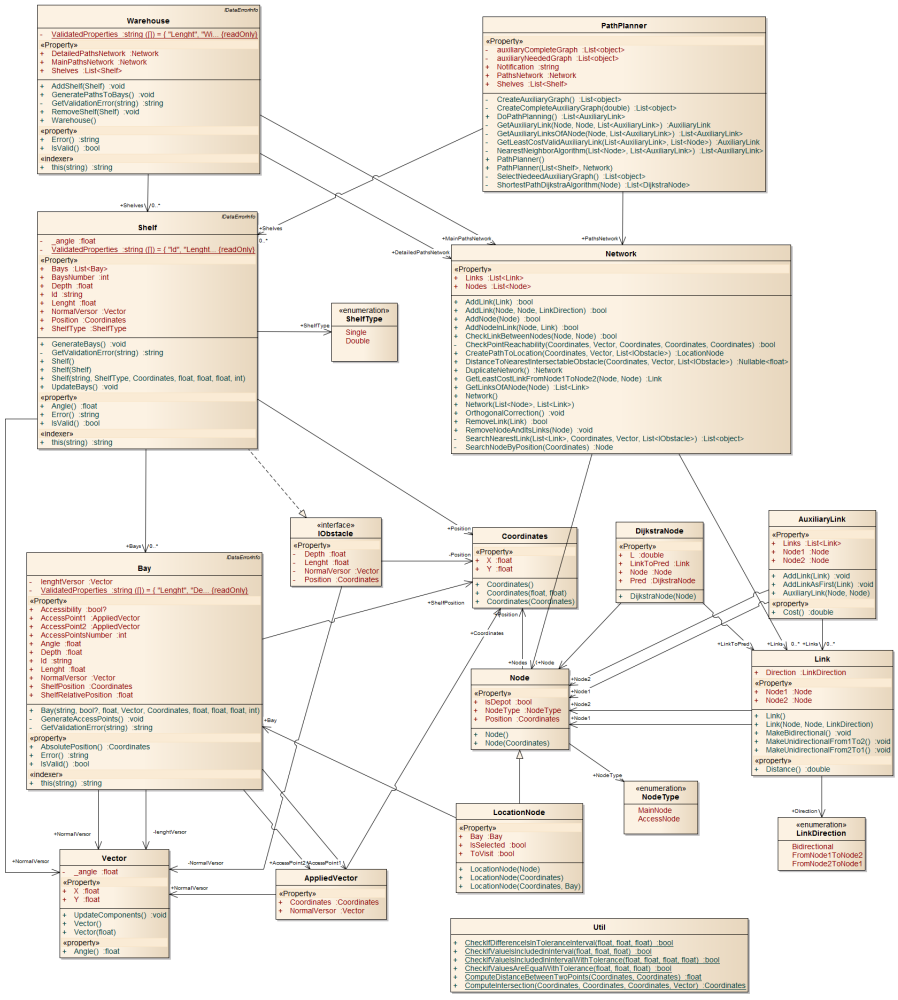

Figura 3.19: Model

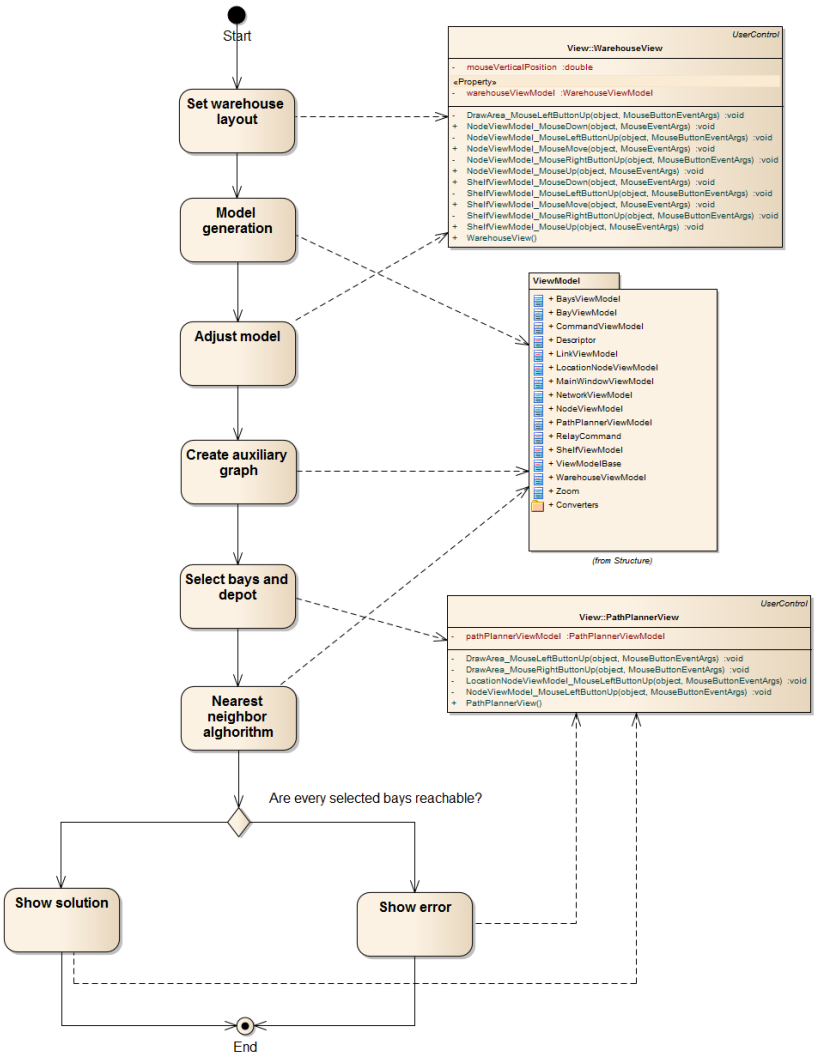

Figura 3.20: Behaviour

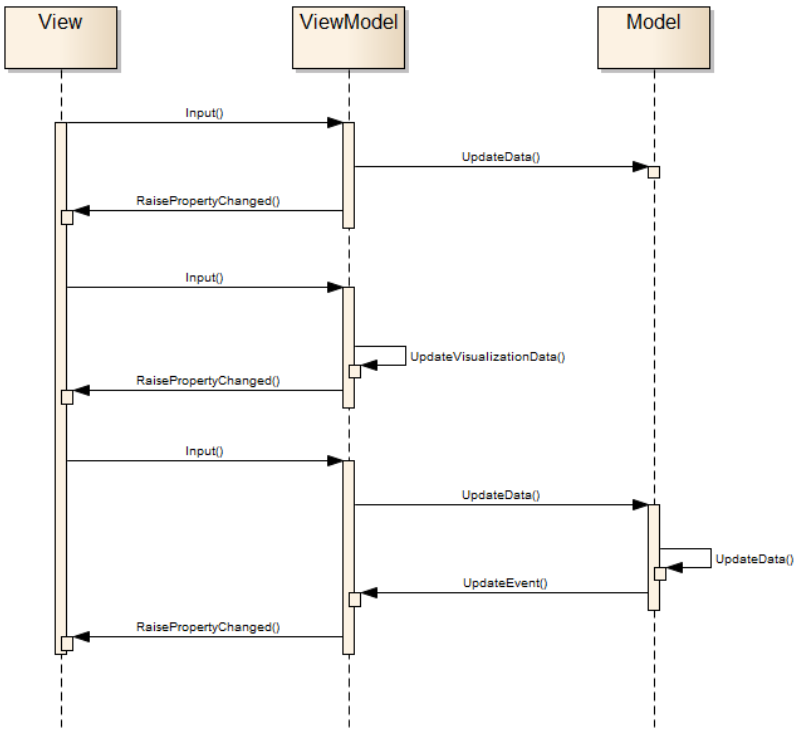

Figura 3.21: Interaction

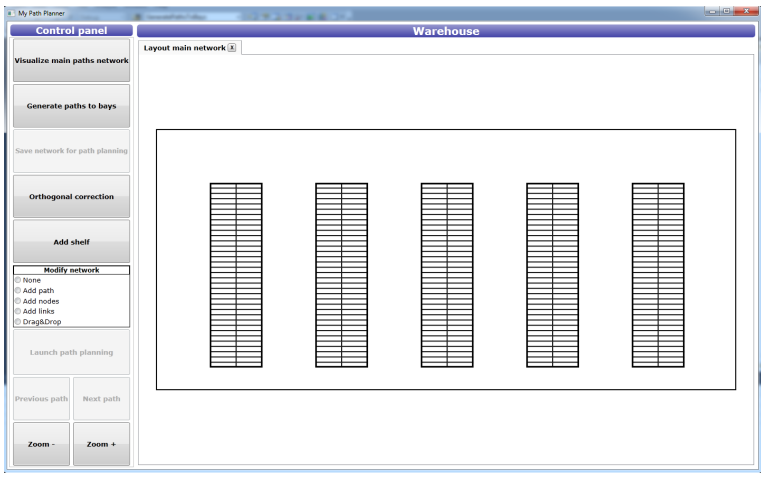

Figura 3.22: Layout di magazzino con scaffali

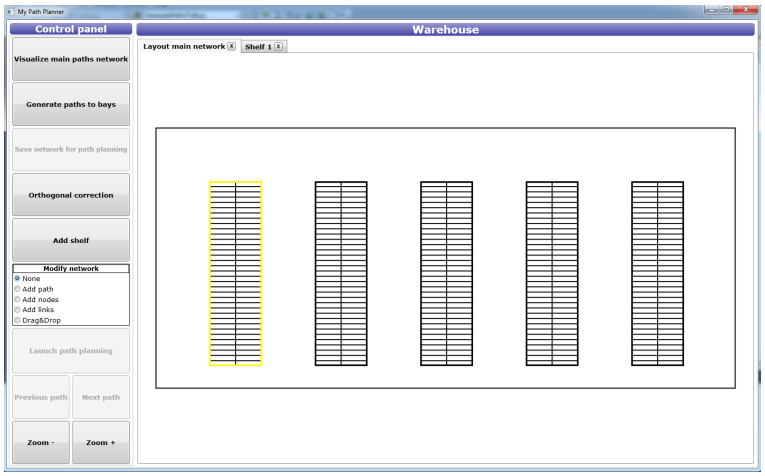

Figura 3.23: Scaffale selezionato

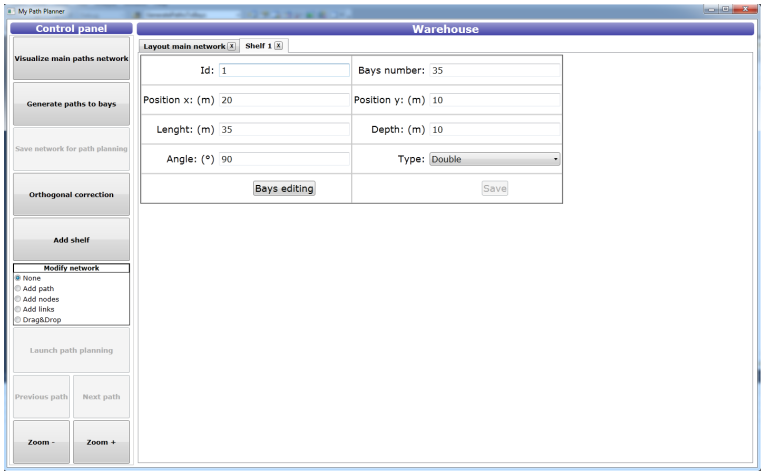

Figura 3.24: Dati dello scaffale

La vista che mostra le proprietà di uno scaffale, tutte modificabili, è quella nella figura 3.24. Inoltre, è possibile modificare le proprietà di ogni campata singolarmente, premendo il bottone *Bays editing*.

I dati e la disposizione degli scaffali possono essere modificati a piacimento, fino a creare la configurazione desiderata (figura 3.25), anche sfruttando la funzionalità di drag&drop. Inoltre, premendo il tasto destro del mouse su di uno scaffale lo si ruota di 90°. Invece, premendo i tasti *del* o *backspace* si eliminano dal layout gli elementi correntemente selezionati.

A questo punto, è possibile aggiungere una serie di tratte principali, inserendo nodi o link singolarmente, oppure direttamente una loro successione. Selezionando *Add path*, infatti, alla pressione del tasto sinistro del mouse si aggiunge un nodo e un link a quello inserito al click precedente (che rimane evidenziato di colore giallo). Per cambiare il nodo selezionato, al quale verrà collegato quello inserito, è sufficiente cliccare su un altro nodo con il tasto destro del mouse. Cliccando con il tasto destro del mouse su di un collegamento fra nodi gli si cambia il senso di percorrenza, da bidirezionale a monodirezionale, scegliendo come orientare la freccia. Anche in questo caso, alla pressione dei tasti *del* o *backspace* si eliminano gli elementi selezionati. Il risultato è in figura 3.26.

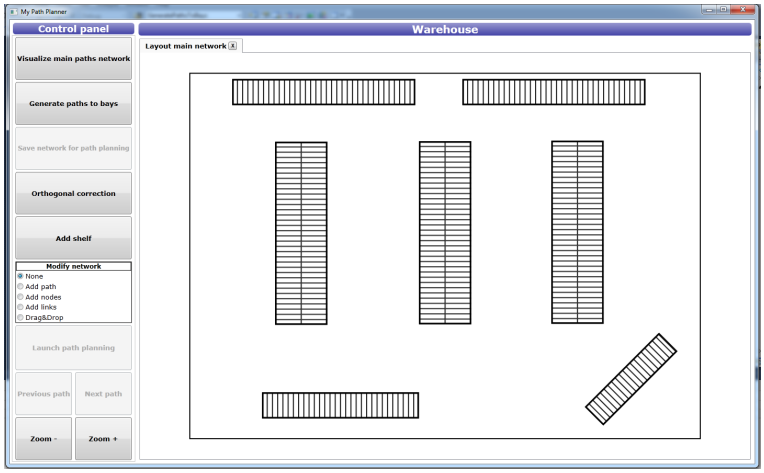

Figura 3.25: Layout di magazzino con scaffali

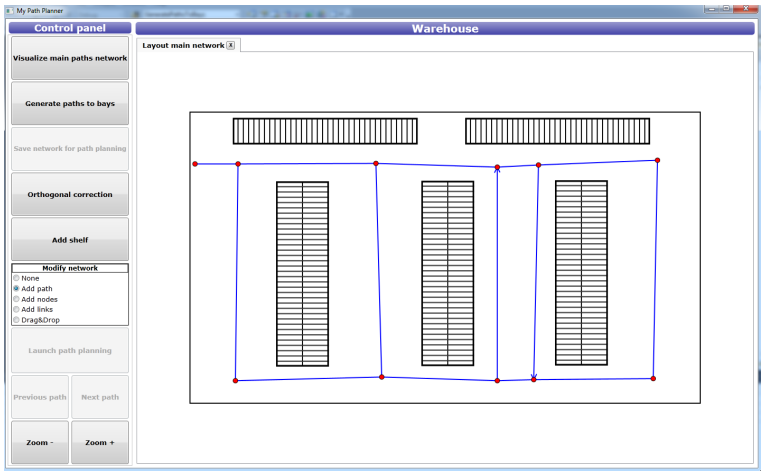

Figura 3.26: Layout di magazzino con scaffali e percorsi principali

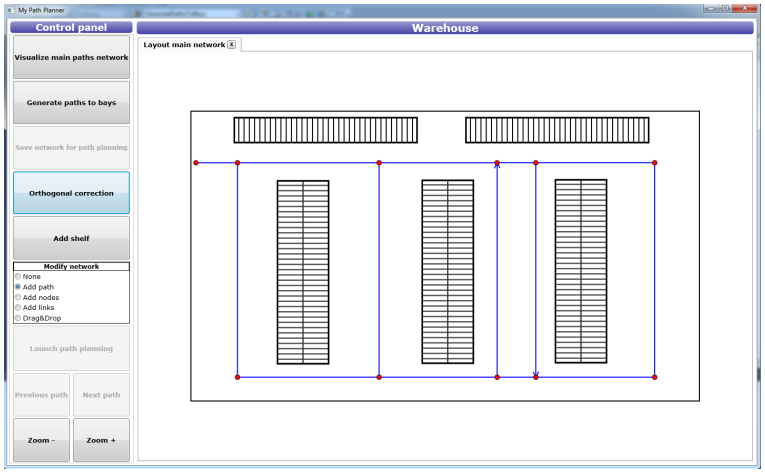

Figura 3.27: Layout di magazzino con scaffali e percorsi principali corretti

Avendo inserito i percorsi mediante il mouse risulta difficile disegnarli in maniera ortogonale; per sopperire a questo problema si può utilizzare il bottone *Orthogonal correction*, che rende le tratte come in figura 3.27.

A questo punto, cliccando sul bottone *Generate paths to bays* si procede alla generazione del modello dei percorsi, che collega ogni campata alla tratta raggiungibile più vicina. Ogni campata collegata alla rete di percorsi viene poi evidenziata di verde, mentre viene colorata di rosso in caso opposto (fig. 3.28).

È ora possibile, dopo altre eventuali modifiche che l'utente può apporre a nodi e tratte, utilizzare questo modello per calcolare i percorsi da compiere durante le operazioni di movimentazione interna. Cliccando sul bottone *Save network for path planning* viene creato il grafo ausiliario come descritto nella sezione 3.5.4.

Quando il programma ha terminato la fase di calcolo, si presenta la vista che permette di selezionare i nodi delle campate e il nodo deposito (fig. 3.29). Questa interfaccia mostra tutti gli scaffali e i possibili nodi selezionabili; si ricorda che è possibile selezionare solo un nodo deposito e un numero di locazioni da raggiungere maggiore di zero.

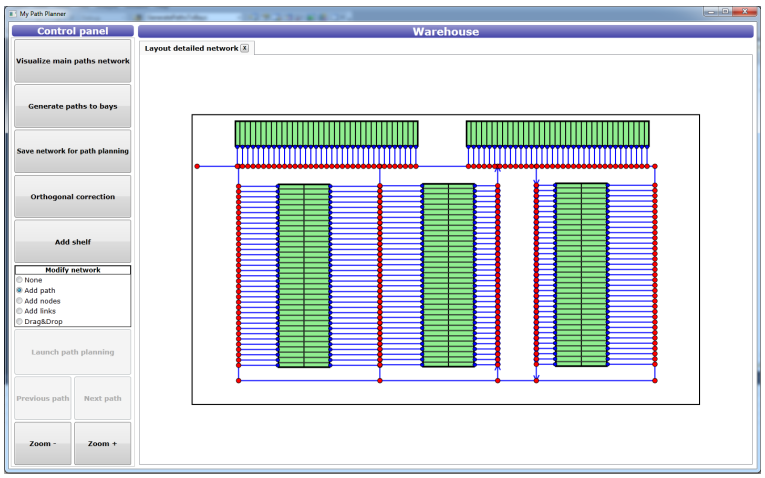

Figura 3.28: Layout dopo la generazione del modello

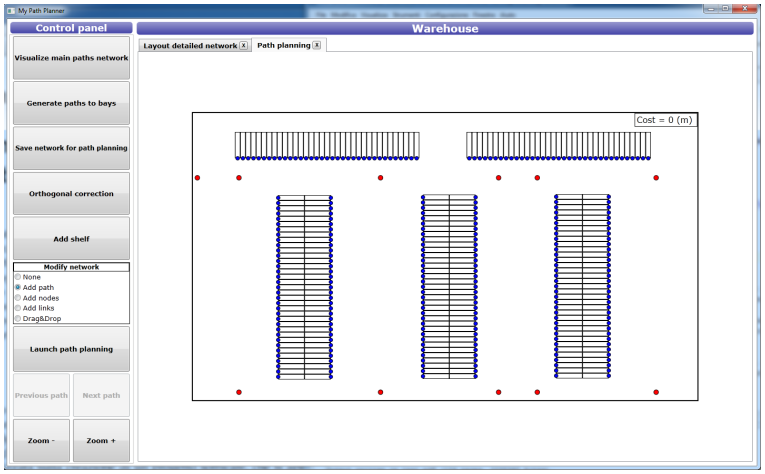

Figura 3.29: Vista per la pianificazione del percorso

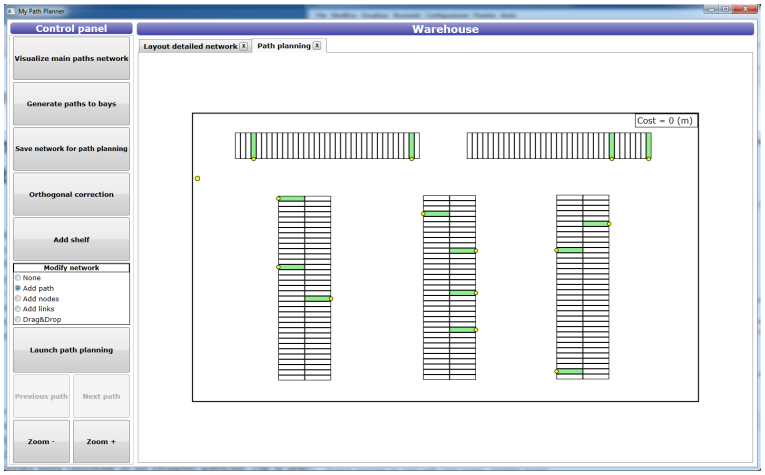

Figura 3.30: Campate e deposito selezionati

Si selezionano un certo numero di campate e il nodo deposito, come in figura 3.30.

Premendo il bottone *Launch path planning* si avviano i successivi calcoli sulla sequenza di tratte da compiere per attraversare tutte le locazioni e tornare al nodo deposito, come deciso nella sezione 3.5.4. In maniera quasi istantanea, il programma restituisce il risultato, evidenziando le tratte da percorrere, con un etichetta che ne indica l'ordine e il costo totale del circuito, misurata in metri (fig. 3.31).

Premendo i pulsanti *Previous path* e *Next path* è possibile evidenziare le singole tratte punto a punto, dalla prima (che ha come nodo iniziale il nodo deposito) all'ultima (che ha come nodo finale il nodo deposito). Un esempio è in figura 3.32.

Ora, deselezionando un nodo qualsiasi, è possibile procedere ad una nuova selezione e calcolo di percorso, oppure si può tornare a modificare il layout della mappa principale o secondare e ripetere tutte le operazioni descritte sopra.

Si mostra ora il caso in cui non è possibile collegare certe locazioni alla rete dei percorsi, perché le relative tratte sono assenti o intralciate da altri scaffali. Partendo dal layout principale di figura 3.33, si genera il modello di rete dei percorsi dettagliato.

Il risultato della generazione è in figura 3.34; le campate del lato

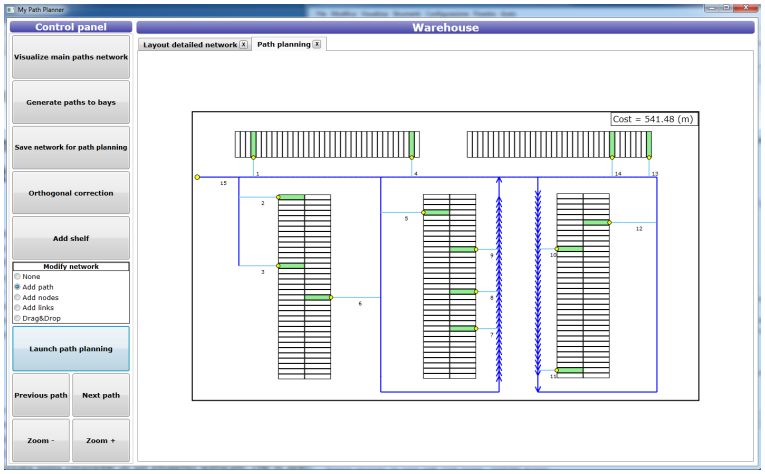

Figura 3.31: Risultato del calcolo del percorso

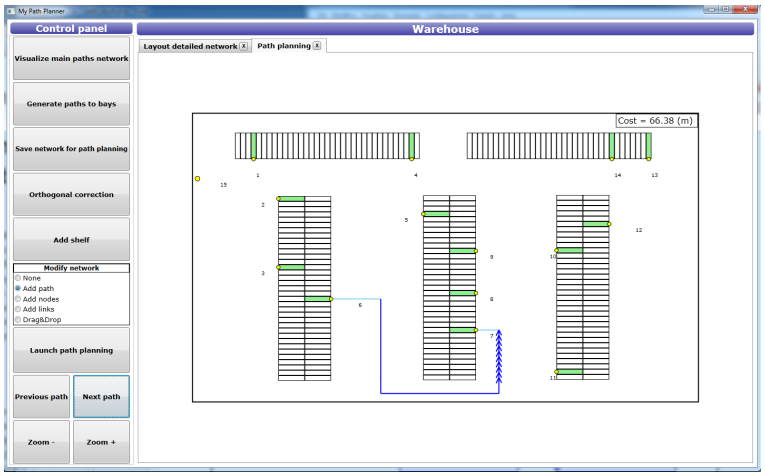

Figura 3.32: Tratta punto a punto del risultato

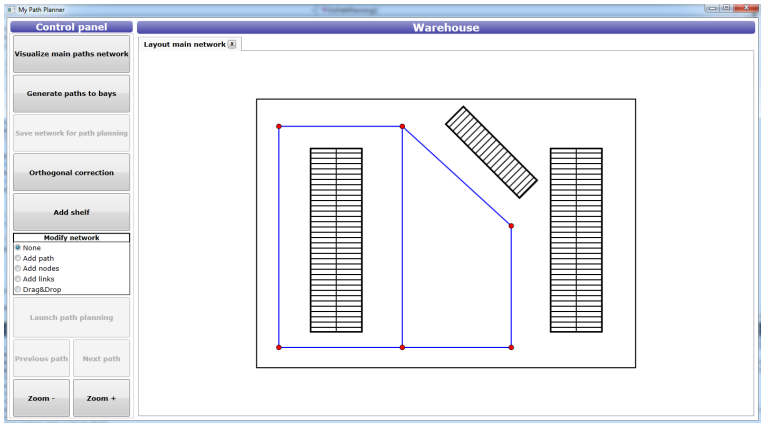

Figura 3.33: Layout di magazzino

destro dello scaffale più a destra risultano non collegate in quanto non è stata trovata alcuna tratta di fronte ad esse. Invece, le campate del lato sinistro dello stesso scaffale, nella parte alta, non sono accessibili in quanto intralciate dallo scaffale posto in obliquo davanti ad esse.

Un altro caso che è possibile modellare, riguarda quello in cui vi sono scaffali molto lunghi che sono attraversabili al centro mediante un sottopassaggio. Ciò è rappresentabile mediante i percorsi inseriti nella figura 3.35.

Generando la rete dettagliata di percorsi, si ha il risultato di figura 3.36; nel caso in cui le campate sotto le quali vi è costruito un passaggio non siano accessibili, è possibile modificare manualmente la rete generata, rimuovendo i nodi delle campate inaccessibili e i relativi collegameti, dimodoché non siano selezionabili e utilizzabili nella fase successiva (vedi figura 3.37).

3.7 In sintesi

In questo capitolo si è riportata la descrizione del processo di sviluppo che ha portato all'implementazione di un supporto software per la pianificazione e ottimizzazione dei percorsi all'interno di un ma-

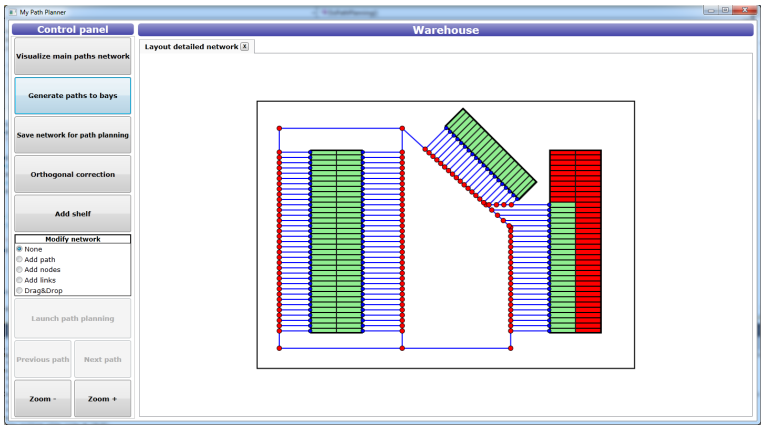

Figura 3.34: Rete di percorsi con campate non collegate

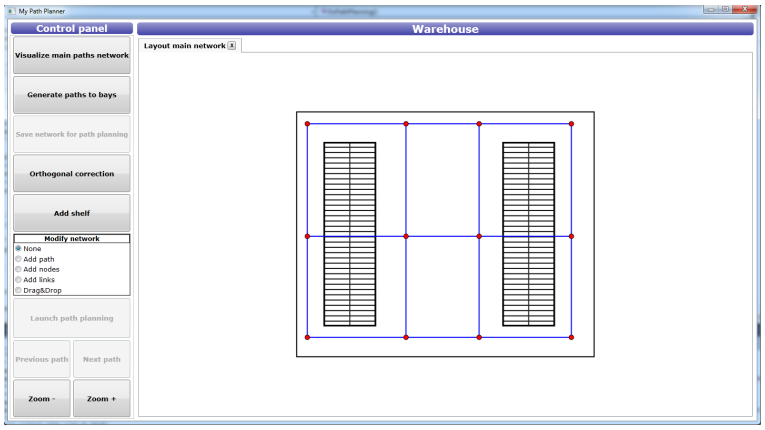

Figura 3.35: Layout con sottopassaggio

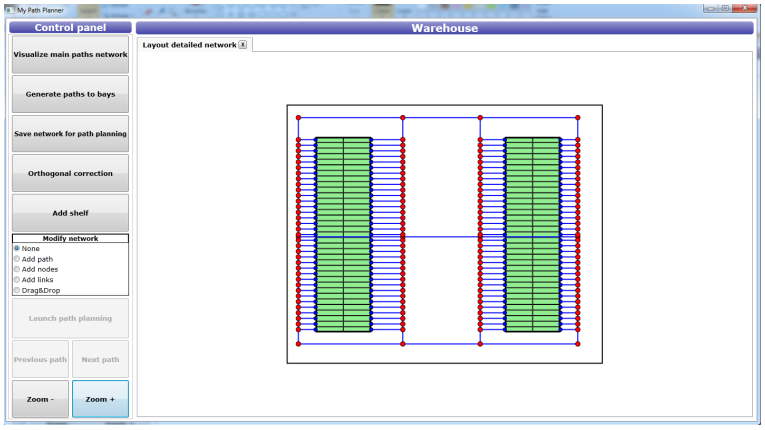

Figura 3.36: Rete di percorsi con sottopassaggio

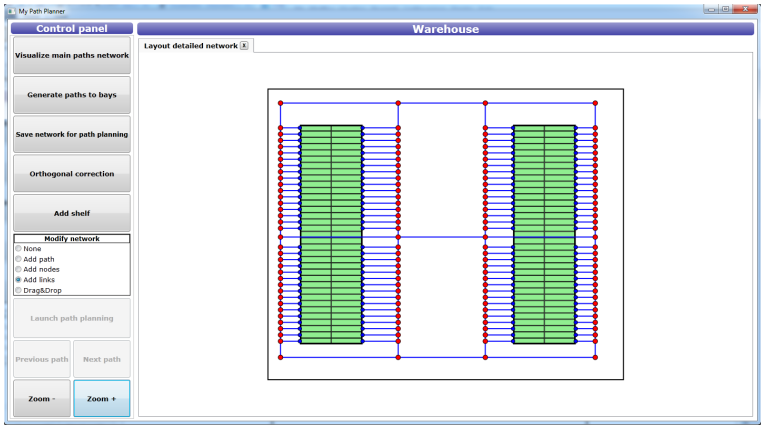

Figura 3.37: Rete di percorsi corretti dall'utente

gazzino. A partire dai requisiti del problema, si è continuato con la presentazione dell'analisi, definendo i punti cardine del problema, per poi passare al progetto della soluzione, scegliendo la tecnologia da usare, e infine alla sua implementazione.

Al termine di una serie di processi di sviluppo a spirale è stato valutato, in accordo col tutor aziendale, di aver raggiunto un risultato accettabile in un prototipo che è in grado di rappresentare e soddisfare un insieme sufficiente di casi d'uso.

Capitolo 4

Risultati

4.1 Introduzione

In questo capitolo si riportano le analisi e i commenti relativi all'applicazione del software presentato al capitolo precedente, prima riportando i tempi di calcolo utilizzando layout con un numero di scaffali crescente e poi su di un magazzino reale. In particolare, si riporterà la disposizione degli scaffali e dei percorsi principali, per poi far generare il modello dettagliato dei percorsi al programma e simulare un'operazione di movimentazione interna, con una missione di prelievo che coinvolge un certo insieme di locazioni da raggiungere.

4.2 Analisi dei tempi

In questa sezione si vuole riportare l'andamento dei tempi di calcolo al variare del numero di scaffali (e quindi di nodi) presenti nella rete.

Vengono analizzati i tempi della generazione del modello (di 3.5.3) al variare del numero di campate presenti e degli algoritmi di Dijkstra e *nearest neighbor* (di 3.5.4) al cambiare del numero di nodi della rete su cui agiscono.

I tempi impiegati nella generazione del modello sono stati considerati come trascurabili, in quanto in tutte le prove effettuate sono

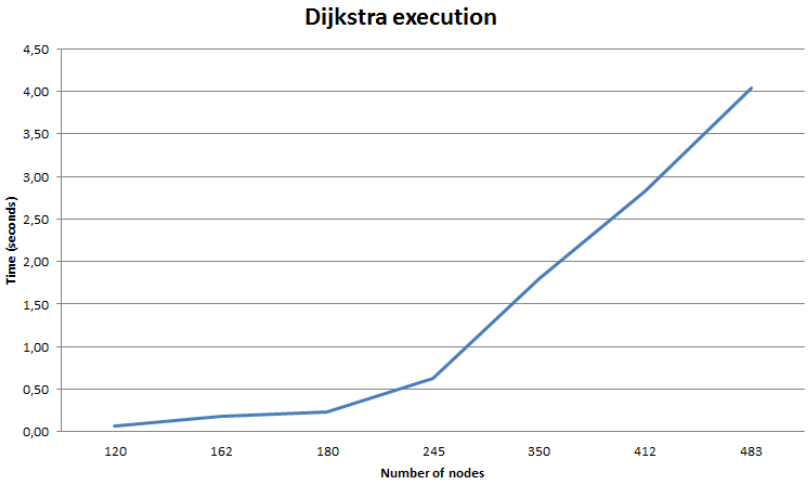

Figura 4.1: Tempi di esecuzione dell'algoritmo di Dijkstra

sempre risultati inferiori al secondo. Stessa considerazione vale per i tempi di calcolo dell'algoritmo *nearest neighbor*.

La fase più significativa è invece quella della generazione del grafo completo. Dato m come somma dei `LocationNode` e dei possibili nodi deposito (ovvero quelli inseriti manualmente dall'utente) e dato n il numero di nodi totali della rete ($n = m+$ i nodi aggiuntivi inseriti in fase di generazione del modello), si sono lanciate diverse esecuzioni che hanno avuto come esito i tempi registrati nei grafici seguenti.

I tempi di calcolo dipendono dal numero di nodi del modello precedente e dall'hardware che esegue il programma. In questo caso, l'esecuzione è stata avviata compilando i sorgenti per piattaforme `x86` e su un sistema con Windows 7, processore Intel Core 2 Duo P8700 2.53 GHz e 4 GB di RAM DDR3.

Per calcolare i tempi dell'esecuzione del grafo 4.1 si è lanciato l'algoritmo su di una rete con numero di nodi presenti nell'asse delle X per m iterazioni. Il tempo ottenuto è quello calcolato dalla media su quello totale. Si nota come, confrontandolo con il grafico 4.2, l'andamento sia simile; infatti, l'algoritmo di Dijkstra è a complessità $O(n^2)$

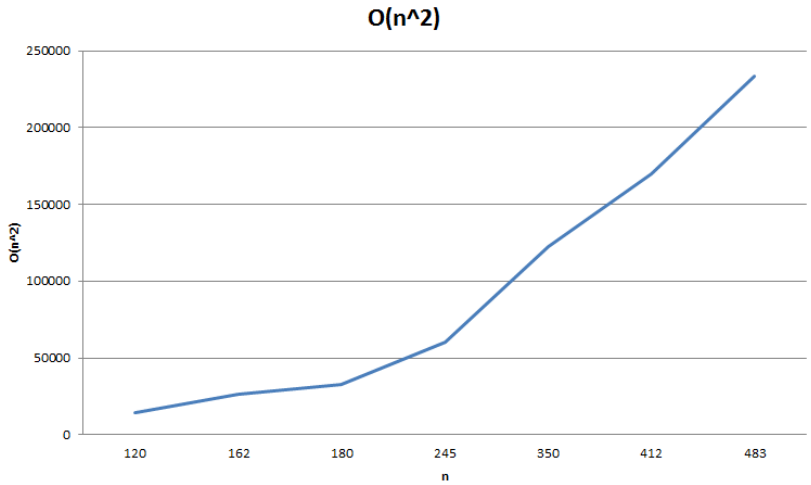

Figura 4.2: O(n^2)

Per ultimo, si riporta il grafico che rappresenta l'andamento dei tempi nella costruzione del grafo ausiliario; si ricorda che per elaborare tale grafo, è necessario eseguire l'algoritmo di Dijkstra per m volte, ogni volta su una rete di n nodi. Sull'asse delle ascisse si riporta il valore di $m \cdot n^2$.

4.3 Uno scenario reale

Lo scenario che si vuole prendere in esame è quello di un magazzino reale nel quale vengono realizzate missioni di deposito o prelievo su varie locazioni mediante carrello elevatore. La struttura appartiene agli stabilimenti di un cliente di *Onit* ed è formata da 21 scaffali per un totale di 271 campate.

Come prima cosa si riporta il layout del magazzino nel programma (fig. 4.4), dopodiché si inseriscono i percorsi principali nei corridoi, come in figura 4.5.

A questo punto, facendo generare i percorsi alle campate, si ottiene il risultato di figura 4.6. Si aggiunge, inoltre, un nodo in

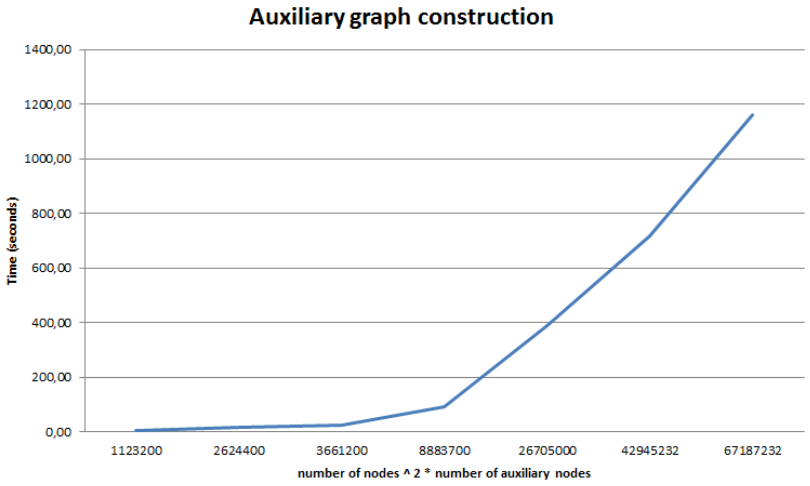

Figura 4.3: Tempi di costruzione del grafo ausiliario completo

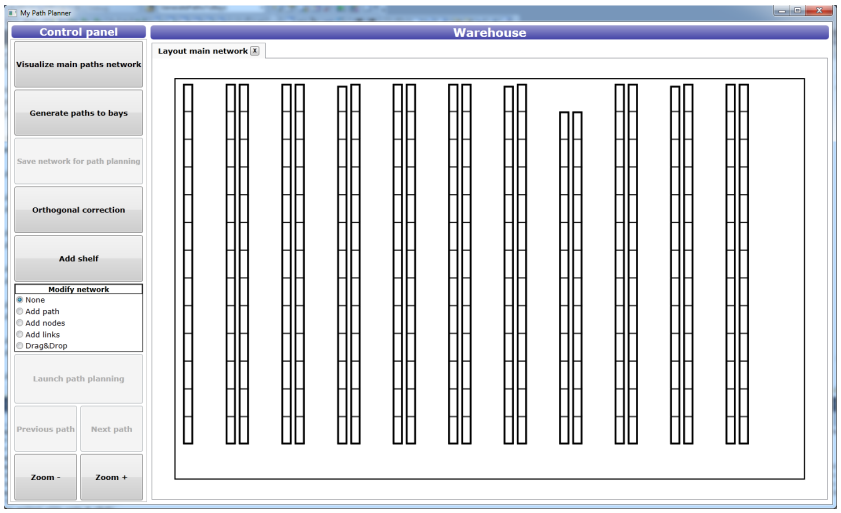

Figura 4.4: Layout del magazzino di un cliente di Onit

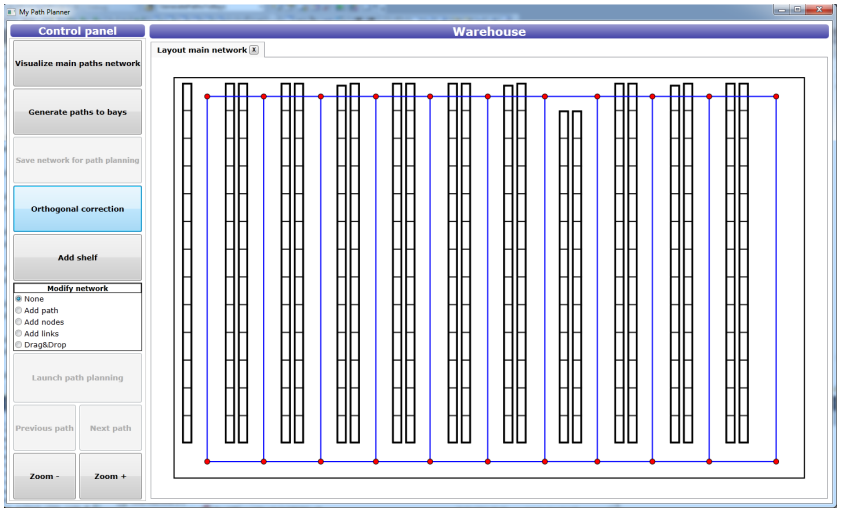

Figura 4.5: Layout del magazzino con percorsi principali

alto al centro, da considerarsi quello di ingresso e uscita dopo una missione di movimentazione nel magazzino (il cosiddetto deposito).

Ora si può utilizzare la rete di nodi creata per generare il grafo ausiliario completo necessario al *nearest neighbor* della fase successiva. Cliccando sul bottone *Save network for path planning*, dopo circa 12 minuti e 30 secondi, appare la schermata di figura 4.7, calcolata con 266 nodi ausiliari e 409 nodi totali.

Procedendo con una selezione di 15 campate e del nodo deposito (fig. 4.8), per un totale di 16 nodi, lanciando la pianificazione del percorso viene prodotto, quasi istantaneamente, il risultato di figura 4.9, con la sequenza delle tratte punto a punto da percorrere per portare a termine la missione.

4.4 In sintesi

In questo capitolo si sono mostrate alcune analisi sui tempi effettuate sul programma realizzato. Si è mostrato come la fase computazionalmente più onerosa sia quella che calcola il grafo ausiliario completo sulla rete, come previsto. I tempi di attesa possono comunque

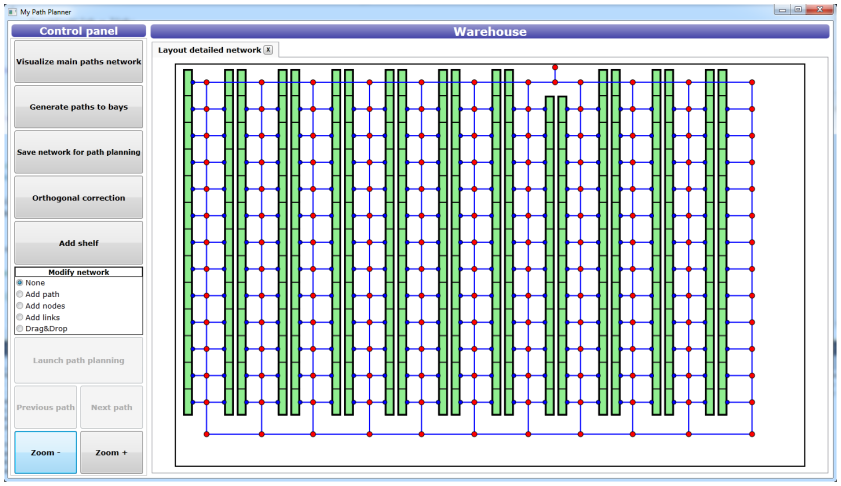

Figura 4.6: Layout del magazzino dopo la generazione del modello

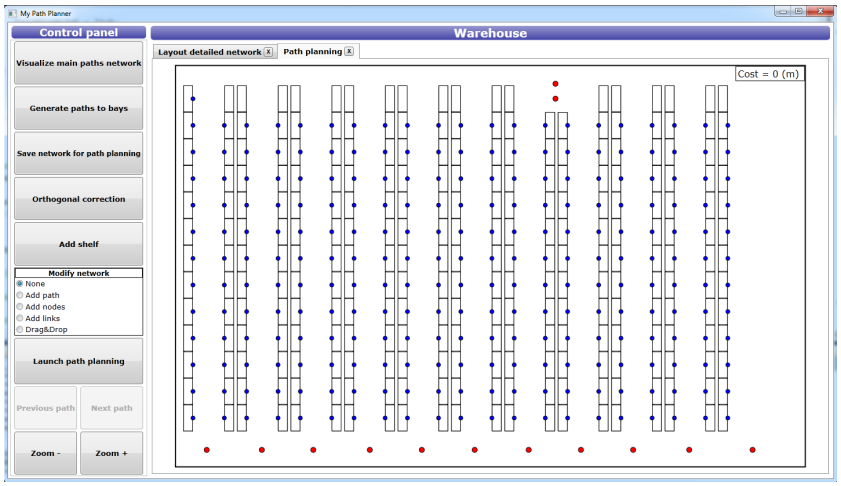

Figura 4.7: Interfaccia per selezione delle locazioni da visitare e del deposito

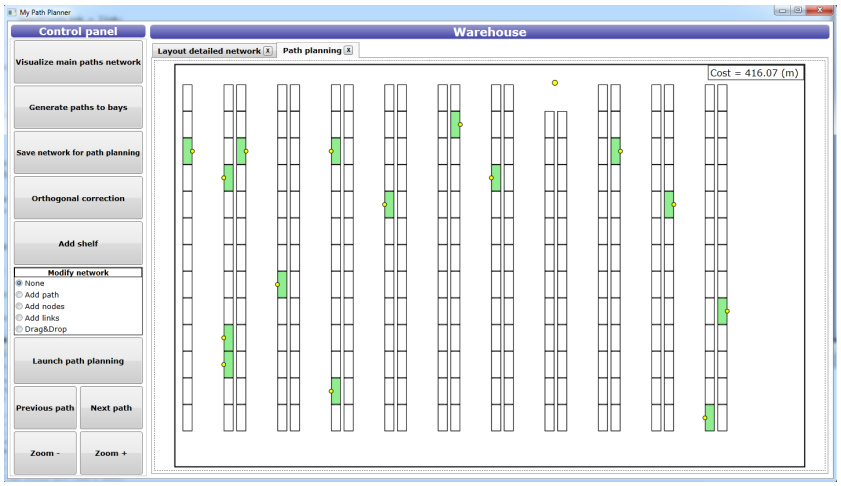

Figura 4.8: Selezione di 15 campate e del nodo deposito

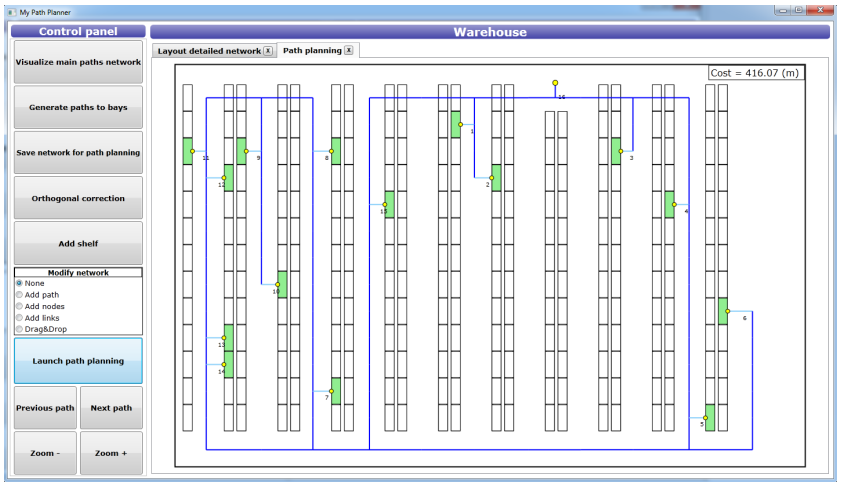

Figura 4.9: Risultato della pianificazione del percorso

essere compatibili con un utilizzo del programma in uno scenario reale, in quanto questa fase di configurazione, fatta una volta per tutte, permette poi di avere dei tempi di risposta istantanei nella pianificazione vera e propria del singolo percorso. Ciò è stato testato anche sul layout di un impianto di stoccaggio reale, con risultati soddisfacenti.

Capitolo 5

Conclusioni

Durante l'attività che ha portato alla stesura di questa tesi ho compiuto un percorso di apprendimento e di integrazione delle conoscenze pregresse, le quali sono state sfruttate per sviluppare il prototipo di un software utile alla pianificazione e ottimizzazione dei percorsi per la movimentazione interna nei magazzini industriali.

Ho svolto questa esperienza grazie a un'opportunità di tirocinio per tesi in azienda, svoltasi presso *Onit Group s.r.l.*, mediante la quale ho potuto frequentare regolarmente l'ambiente di lavoro per circa 3 mesi, dal 25 Giugno 2012 al 12 Settembre 2012, per un complessivo di circa 345 ore.

I capitoli riportati nella tesi hanno ripercorso, a grandi linee, l'ordine cronologico con il quale è proseguita l'attività di studio, prima, e di sviluppo software, poi.

Nel primo capitolo ho riportato una panoramica sulla logistica, descrivendone l'importanza generale per poi focalizzarmi sui magazzini industriali nelle loro caratteristiche principali; riguardo questo argomento, ho anche potuto sfruttare esperienze dirette mediante alcune visite all'azienda *Orogel* di Cesena, azienda leader in Italia nella produzione dei surgelati e cliente di *Onit*, dunque un esempio importante nell'ambito di studio. Ho potuto così approfondire la conoscenza sul dominio del problema.

Nel secondo capitolo mi sono focalizzato sulla ricerca di letteratura riguardante la modellazione dei magazzini e gli algoritmi di ottimizzazione dei percorsi. Ho prodotto una sintesi di quelli che

ho ritenuto più rilevanti e utili per affrontare, in prima battuta, il problema della pianificazione dei percorsi.

Dopo aver recuperato e sintetizzato la letteratura riportata ai capitoli precedenti, nel terzo ho illustrato il processo di sviluppo che ha portato alla realizzazione del software riguardante la pianificazione e l'ottimizzazione dei percorsi. A partire da un insieme di requisiti iniziali e a seguito di una serie di confronti con il tutor aziendale Claudio Gambetti, si è deciso quali altri requisiti poter aggiungere e sviluppare, in un processo a spirale, fino al conseguimento di un risultato soddisfacente.

Nel quarto e ultimo capitolo, infine, ho applicato il software realizzato in uno scenario di magazzino reale, osservandone i risultati e analizzandone le prestazioni in termini di tempi di calcolo. Per fare questo, ho riportato il layout di un impianto di stoccaggio di un cliente di **Onit** nel software per simulare una movimentazione di merce riguardante diverse locazioni. Ho appurato che il programma, una volta terminata la fase di configurazione, che risulta la più onerosa, calcola i percorsi in tempi soddisfacenti.

Possibili sviluppi futuri possono riguardare sia un'estensione dei casi d'uso risolvibili dal programma, molto numerosi in questo ambito, sia un miglioramento delle parti già presenti, come ad esempio sfruttare il multi-threading per la fase di calcolo del grafo completo, oppure ampliare gli algoritmi utilizzabili per la generazione dei percorsi, oltre al *nearest neighbor*, in funzione della qualità della soluzione che si vuole ottenere.

Nel complesso, le fonti di conoscenza dalle quali ho attinto per svolgere la tesi si rifanno principalmente alla ricerca operativa, nei primi due capitoli, e all'ingegneria del software, nel terzo capitolo. Inoltre, grazie al periodo di tirocinio in *Onit*, ho potuto accedere a tutta una serie di conoscenze collaterali, ad esempio riguardo il linguaggio *Visual C#* e l'utilizzo delle librerie di classi *WPF* appartenenti al *Framework .NET*. Sebbene non fossi coinvolto nell'implementazione di un prodotto, ho avuto la possibilità di poter osservare dall'interno la software house. Ciò mi ha permesso di avere un riscontro *sul campo* del processo di sviluppo del software nel mondo lavorativo, osservando le interazioni fra i colleghi e, a volte, quelle con il cliente.

In conclusione, valuto questa esperienza come positiva per due motivi principali: il primo riguarda il risultato della tesi, cioè l'aver realizzato un prototipo da cui si può prendere spunto per integrare nuove funzionalità ad un prodotto di gestione magazzino già esistente. Si noti che l'alternativa all'utilizzo di un programma come quello che ho sviluppato è di inserire manualmente a sistema tutti i nodi e gli archi dei percorsi, il che richiederebbe tempi nettamente superiori (circa un ordine di grandezza). Il secondo motivo deriva dal fatto che questa esperienza mi ha permesso di inserirmi in maniera stabile in *Onit*, con un impiego nella *business unit automazione*, che è responsabile degli sviluppi software e della manutenzione sugli impianti automatici di *Orogel*.

Per queste ragioni, mi auspico che le occasioni di tirocinio in azienda siano maggiormente utilizzate per rendere più naturale il passaggio fra il mondo universitario e quello lavorativo, diminuendo il divario fra i due.

Desidero infine ringraziare il relatore *prof.* Daniele Vigo, *Onit Group* per l'opportunità concessami e in particolare il correlatore *ing.* Claudio Gambetti e tutta l'*area industria* per la formazione ed il supporto fornito durante il periodo di svolgimento della tesi.

Bibliografia

[1] Robert B. Handfield, Ernest L. Nichols : *Introduction to supply chain management*, 1999, Prentice Hall

[2] Daniele Vigo: *IT-Based Management of Logistics 2*, 2010, Metodi e Modelli per il Supporto alle Decisioni LM, Università di Bologna

[3] Antonio Comi: *Logistica e trasporto delle merci*, Università degli Studi di Roma Tor Vergata

[4] Prof. Ing. Riccardo De Carlini: *Logistica Industriale - Capitolo VI: I Magazzini Industriali*, Ingegneria Gestionale, Università degli Studi di Napoli Federico II

[5] Gianpaolo Ghiani, Gilbert Laporte, Roberto Musmanno: *Introduction to Logistics Systems Planning and Control*, Wiley, 2004

[6] Alexander Mandel, Jan Kappallo, Wassili Sabelfeld, Christian Reinhardt, Markus Puchta: *Method and apparatus for path planning and distance calculation*, 2009, United States, Patent Application Publication

[7] Daniele Vigo: *Algoritmi euristici*, 2010, Metodi e modelli per il supporto alle decisioni LM, Università di Bologna

[8] Daniele Vigo: *Algoritmi metaeuristici*, 2010, Metodi e modelli per il supporto alle decisioni LM, Università di Bologna

[9] Daniele Vigo: *Teoria dei grafi - Cammini a costo minimo*, 2010, Ricerca operativa LA, Università di Bologna

[10] Antonio Natali: *Ingegneria dei sistemi software LM*, 2010, Università di Bologna

[11] MSDN Library: *.NET Framework 4*, http://msdn.microsoft.com/it-it/library/w0x726c2(v=vs.100).aspx

[12] MSDN Library: *WPF*, http://msdn.microsoft.com/it-it/library/ms754130(v=vs.100).aspx

[13] MSDN Library: *Using the Model-View-ViewModel Pattern*, http://msdn.microsoft.com/it-it/library/hh821028.aspx

[14] Josh Smith: *WPF Apps With The Model-View-ViewModel Design Pattern*, http://msdn.microsoft.com/it-it/magazine/dd419663.aspx