Politecnico di Torino

Corso di Laurea in Ingegneria Gestionale Classe L-8
A.a. 2024/2025
Sessione di Laurea marzo 2025

Sistema di gestione per la consegna di prodotti tramite droni

Relatore:
prof. Fulvio Corno

Candidato:
Barisione Gabriele

INDICE

1. PROPOSTA DI PROGETTO

1.1 Studente proponente

s299809 Barisione Gabriele

1.2 Titolo della proposta

Sistema di gestione per la consegna di prodotti tramite droni

1.3 Descrizione del problema proposto

Il progetto si propone di affrontare un problema emergente: la gestione di flotte di droni per consegne in ambienti urbani. Il software si propone di risolvere il problema di ottimizzazione della gestione delle consegne, aiutando i droni a trovare il percorso migliore in base a parametri come capacità, distanza e consumo di risorse evitando ritardi e costi operativi elevati. L'obiettivo è sviluppare un sistema che, tramite algoritmi ricorsivi, suggerisca percorsi ottimali, riducendo al minimo i chilometri percorsi e ottimizzando l'uso dei mezzi di trasporto, con conseguente risparmio energetico e miglioramento del servizio di consegna.

1.4 Descrizione della rilevanza gestionale del problema

La rilevanza gestionale del progetto è strettamente collegata all'ottimizzazione delle risorse e alla pianificazione di percorsi ottimali, tematiche centrali nell'ambito della ricerca operativa. Grazie all'implementazione di algoritmi di ottimizzazione, il sistema sarà in grado di generare, in pochi istanti e con numero limitato di comandi, un percorso da seguire che tiene conto di priorità di consegna, distanza tra le destinazioni e dimensione di merce trasportata. Tale automazione contribuisce a migliorare l'affidabilità del servizio e un uso più efficiente delle risorse con conseguente risparmio sui costi operativi offrendo beneficio competitivo per l'azienda.

1.5 Descrizione dei data-set per la valutazione

Il database utilizzato nel progetto è assemblato da me: per la creazione del grafo sono stati usanti un numero cospiquo di civici della città di Milano presi dal sito https://dati.comune.milano.it/it/dataset/ds634-numeri-civici-coordinate : i civici di un certo quartiere andranno a costruire il grafo in cui effettuare le consegne. Data dictionary della tabella civicmilano:

- CODICE_VIA: Codice numerico identificativo del Toponimo (es. 1 per Piazza del Duomo). (numeric)
- NUMERO: Numero civico puro. (numeric)
- MUNICIPIO: Codice numerico identificativo del municipio. (numeric)
- IDMASTER: Numero sequenziale - Identificativo univoco del civico. (numeric)
- TIPO: Tipo di via (es. Strada, Piazza). (text)

- ANNCSU: Denominazione in Anagrafe Nazionale Numeri Civici e Strade Urbane del Toponimo. (es. Roma, del Duomo) (text)
- ID_NIL: Identificativo del Nucleo di Identità Locale (NIL o quartiere). (numeric)
- NIL: Nucleo di Identità Locale, o quartiere. (text)
- LONG_WGS84: Coordinata X in WGS84 - EPSG 4326. (numeric)
- LAT_WGS84: Coordinata Y in WGS84 - EPSG 4326. (numeric)

I dati dei prodotti da consegnare sono presi dal catalogo di prodotti IKEA, fonte https://www.kaggle.com/datasets/thedevastator/ikea-product.
Data dictionary della tabella ikeaproducts:
- item_id: Identificatore univoco per ogni prodotto IKEA. (numeric)
- name: Il nome del prodotto IKEA. (text)
- category: La categoria del prodotto IKEA. (text)
- price: Il prezzo del prodotto IKEA. (numeric)
- short_description: Una breve descrizione del prodotto IKEA. (text)
- depth, height, width: Le dimensioni del prodotto IKEA. (numeric)

Infine la tabella Fornitore per il dimensionamento del velivolo comprende poche righe scritte da me:
- fornitore: Azienda di trasporti che fornisce il mezzo (text)
- capienza max: Capacità massima del velivolo senza pilota in termini di spazio, per determinare quanti prodotti può trasportare. (numeric)
- prezzo fisso: Costo fisso associato al servizio, indipendente dalla distanza percorsa. (numeric)
- prezzo al chilometro: Costo variabile calcolato in base alla distanza percorsa. (numeric)

Questo dataset permette di modellare un sistema di gestione delle consegne, in cui i droni trasportano prodotti lungo uno spazio aereo. Le informazioni sugli Address consentono di determinare i target, mentre i dati sui quadricotteri e sui prodotti sono utili per gestire la capacità di carico e monitorare le spedizioni.

1.6 Descrizione preliminare degli algoritmi coinvolti

Il progetto si propone di implementare un algoritmo ricorsivo, che ha come obiettivo individuare la soluzione ottimale in termini di costo delle operazioni logistiche. L'algoritmo seguirà i seguenti passaggi:

I. Input: L'algoritmo riceve una lista dei prodotti da consegnare, con l'indicazione dei rispettivi Address di destinazione.
II. Ottimizzazione del carico: L'algoritmo esplora le combinazioni possibili per caricare i pacchi sui droni, tenendo conto della capacità massima di ciascun mezzo. Ogni ciclo deve completare la lista delle consegne rispettando i vincoli fisici dei veicoli.

III. Calcolo del percorso ottimale: Per ogni configurazione di carico, l'algoritmo calcola il percorso più efficiente in termini di distanza chilometrica. Cerca di ridurre al minimo i chilometri percorsi, evitando percorsi ridondanti o inefficaci che potrebbero aumentare i costi operativi.
IV. Valutazione e ottimizzazione dei costi: l'algoritmo tiene conto non solo delle distanze percorse, ma anche del numero di corse necessarie per completare tutte le consegne. Il costo complessivo è quindi determinato sommando il numero di chilometri percorsi e il numero di viaggi effettuati dai droni.
V. L'obiettivo principale è ridurre al minimo questi costi.

1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'interfaccia grafica dell'applicazione avrà una sezione dedicata all'impostazione dei parametri: l'utente potrà selezionare il quartiere su cui operare, l'azienda fornitrice del mezzo di trasporto (capienza massima del drone) e i relativi costi: costo fisso di ogni corsa e costo al chilometro. Un pulsante "Avvia" attiverà una funzione che genererà casualmente un certo numero di prodotti da consegnare, assegnando a ciascuno la propria destinazione. La lista delle consegne giornaliere verrà poi visualizzata a schermo. Un secondo pulsante, "Calcola percorso", eseguirà l'algoritmo ricorsivo che sulla base dei nodi da raggiungere, troverà il percorso migliore di ogni UAV (Unmanned Aerial Vehicle) stampando a video l'ordine di consegna dei prodotti insieme ad alcune statistiche di rotta: chilometri percorsi, numero di consegne eseguite, costo del viaggio.

2. DATA-SET

2.1 Struttura del database

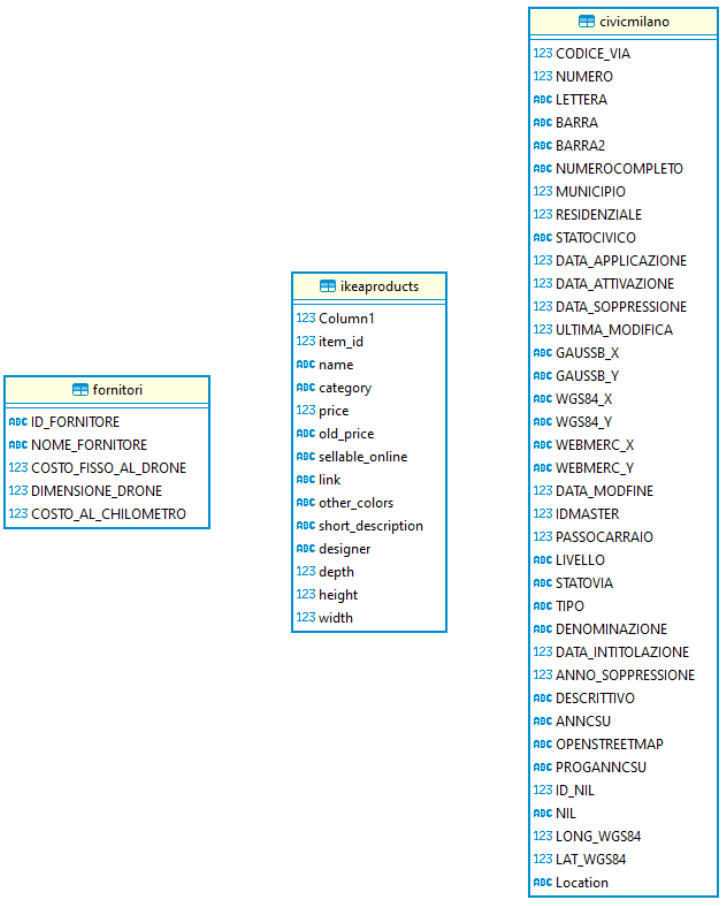
Il database su cui si basa il progetto è stato da me costruito ed è composto di tre tabelle: la prima contiene gli indirizzi della città di Milano, utilizzati per la creazione del grafo, la seconda include le informazioni relative ai prodotti destinati alla distribuzione. Infine, la terza memorizza i dati delle aziende che forniscono mezzi per il trasporto delle merci.

2.2 Raccolta dei dati

I dati sono stati raccolti importando i dataset pubblici dal sito del comune di Milano (dati.comune.milano.it) e dalla piattaforma online Kaggle (www.kaggle.com) in formato .csv. Successivamente, è stata integrata la tabella dedicata ai fornitori di veicoli per il trasporto.

2.3 Creazione del database

Il database è stato creato in mysql tramite il software DBeaver, ed è costituito da tre tabelle: una contiene gli indirizzi della città di Milano utilizzati per la creazione del grafo un'altra include le informazioni relative ai prodotti destinati alla distribuzione e infine l'ultima raccoglie i dati sulle aziende che forniscono mezzi per il trasporto delle merci. Di seguito tabelle:



3. INTERFACCIA GRAFICA

3.1 Input utente

L'utente inizia il processo selezionando un quartiere dal menu a discesa (dropdown). Questo permette di filtrare gli indirizzi di destinazione che appartengono esclusivamente al quartiere selezionato. Una volta che l'utente ha scelto il quartiere, clicca sul bottone “Crea Grafo” che costruisce un grafo completamente connesso. Ogni arco rappresenta una connessione tra due indirizzi, la distanza tra i due è il peso dell’arco.

Il passo successivo è selezionare il fornitore da un altro menu a discesa. Questo completerà in automatico i campi “Prezzo fisso di ogni drone”, “Capienza drone”, “Prezzo al chilometro” indispensabili per il calcolo dei costi.

Il bottone “crea lista consegne”, ha l'obiettivo di generare un numero casuale di consegne, massimo dieci, ciascuna associa un Product a un Address. Questo algoritmo simula la creazione di un piano di consegne, assegnando in modo casuale prodotti a indirizzi di destinazione.

3.2 Risultati applicazione

Dopo aver impostato tutti i parametri, il pulsante “cerca percorso”, avvia l’algoritmo ricorsivo di ottimizzazione. L’algoritmo analizza le possibili combinazioni di carico e i percorsi di viaggio, individuando la soluzione economicamente più vantaggiosa in termini di costo complessivo. I risultati dell’analisi vengono quindi mostrati a video nell’area dedicata.

3.3 Visualizzazione delle consegne

Per visualizzare il percorso calcolato dall'algoritmo, l’utente può premere su “Mappa”. La mappa si aggiorna, mostrando i punti di consegna e un riscontro visivo sui percorsi pianificati.

Link al video dimostrativo del software: https://youtu.be/dkSXLu3xqg8

4. ALGORITMI

4.1 Pattern MVC

Il pattern MVC (Model View Controller) è un'architettura software utilizzata per separare la logica applicativa, gestita nel file model.py, dall'interfaccia utente, implementata dal file view.py. Infine, il file controller.py collega le componenti visive alle funzioni logiche.

4.2 Pattern DAO

Il DAO (Data Access Object) è un pattern che consente di gestire facilmente il database senza esporre direttamente i dettagli della sua implementazione.
Il file DB_connect contiene la classe per connettere l'applicazione al database.
Una classe DAO si occupa di interrogare il db per quanto riguarda le informazioni presenti nelle tabelle.

4.3 Logica applicativa: classi

Nel package model sono contenute le classi che rappresentano i "protagonisti" dell'applicazione e quelle che svolgono la logica applicativa vera e propria:

- Address: informazioni relative agli indirizzi che definiscono i nodi del grafo
- Product: rappresenta i prodotti il cui volume è associato a un valore numerico tra 3 e 6 in base allo spazio che occupa per facilitare il caricamento del drone in relazione alla sua capacità.
- Fornitore:contiene i dati relativi ai costi e alla capienza dei veicoli per il trasporto

4.4 Logica applicativa: funzioni

Il pulsante "Crea Grafo" richiama la funzione creaGrafo() che riceve in input il quartiere selezionato e interroga la tabella civicmilano del database per ottenere le informazioni sugli indirizzi appartenenti a quel quartiere. Questi Questi dati vengono utilizzati per creare oggetti della classe Address quindi essere aggiunti come nodi al grafo, mentre il peso degli archi è calcolato sulla distanza geografica tra ogni coppia di nodi.

La funzione listDelivery(), chiamata dal bottone "crea lista consegne", memorizza un numero casuale intero tra 1 e 10, che rappresenta il numero di consegne da generare. Attraverso un ciclo while, la funzione seleziona casualmente un prodotto dalla lista self.listAllProdotti e un indirizzo dalla lista self.listAddressNodes, creando così una nuova coppia (prodotto, indirizzo). Questa coppia rappresenta una singola consegna.
La funzione restituisce listConsegne che accumula tutte le consegne create nella sessione corrente.

La funzione cercaPrezzoMinimo(), attivata dal bottone "cerca percorso", riceve in input la lista di consegne e i prezzi del fornitore. Restituisce in output _solBest, una lista che contiene liste di consegne e costoBest o il costo della soluzione. La funzione avvia una ricorsione, def trova_combinazioni(), che mira a esplorare tutte le possibili combinazioni di assegnazione di prodotti a diversi veicoli.

Ogni combinazione di carico viene passata alla funzione checkDimensione() che verifica che in ciascuna corsa la somma dei volumi dei pacchi non superi la capienza del drone. Se la combinazione rispetta il vincolo di capacità prosegue alle fasi successive dell'ottimizzazione. In caso contrario, la combinazione viene scartata e si passa alla successiva.

La funzione permutazioni() esplora diverse sequenze di distribuzione. Riceve in input una lista di liste, dove ogni sotto-lista rappresenta un insieme di pacchi assegnati a un drone specifico. Per ogni sotto-lista, la funzione genera tutte le possibili permutazioni dell'ordine di consegna. Successivamente, combina queste permutazioni tra i diversi droni e chiama def calcola_costi() che calcola il percorso più breve per raggiungere tutte le destinazioni utilizzando l'algoritmo Dijkstra, determina il costo della soluzione e memorizza il valore minimo nella variabile costoBest.

4.5 Esempio di combinazioni di carico e permutazioni di consegna

Consideriamo un esempio con quattro prodotti da consegnare, le possibili combinazioni di carico sono le seguenti:

- Un unico veicolo trasporta tutti i prodotti:

  {1, 2, 3, 4}

- Due veicoli con diverse distribuzioni dei prodotti:

  {1, 2, 3}, {4}
  {1, 2, 4}, {3}
  {1, 3, 4}, {2}
  {2, 3, 4}, {1}
  {1, 2}, {3, 4}
  {1, 3}, {2, 4}
  {1, 4}, {2, 3}

- Tre veicoli con diverse distribuzioni dei prodotti:

  {1, 2}, {3}, {4}
  {1, 3}, {2}, {4}
  {1, 4}, {2}, {3}
  {2, 3}, {1}, {4}
  {2, 4}, {1}, {3}
  {3, 4}, {1}, {2}

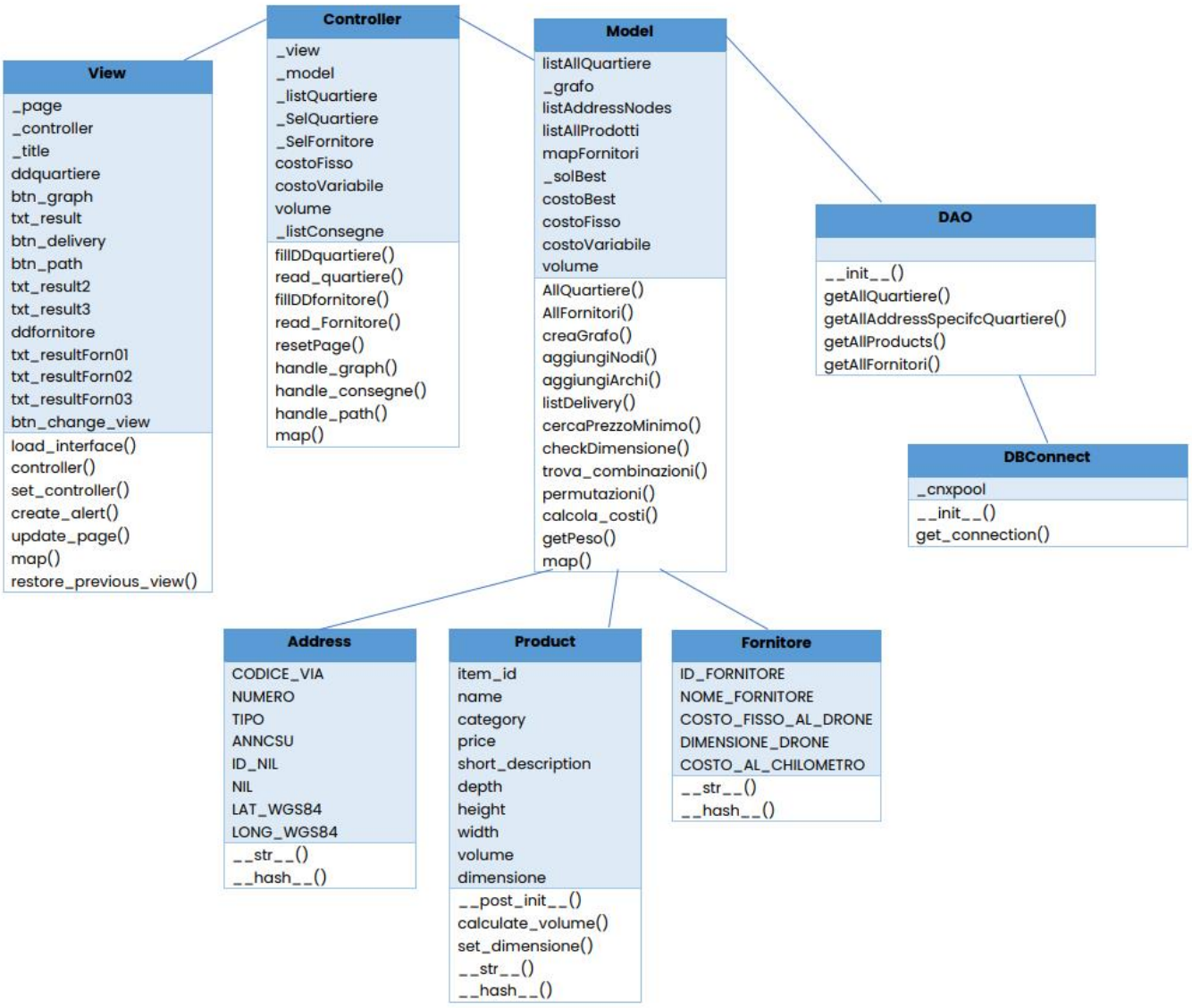
- Quattro veicoli, ciascuno con un singolo prodotto:

{1}, {2}, {3}, {4}

Ad ogni combinazione di carico viene controlla la capienza dei veicoli .
La funzione permutazioni() viene invocata per ciascuna combinazione di carico che rispetta la capienza massima dei veicoli. Il suo scopo è generare tutte le possibili sequenze di consegne all'interno di una singola corsa.
Consideriamo la combinazione di carico composta da due veicoli con le seguenti assegnazioni di prodotti: {1, 2, 3}, {4}. Le possibili permutazioni nell'ordine di consegna dei prodotti sono:

({1, 2, 3}, {4})
({1, 3, 2}, {4})
({2, 1, 3}, {4})
({2, 3, 1}, {4})
({3, 1, 2}, {4})
({3, 2, 1}, {4})

4.6 Diagramma delle classi



5. VANTAGGI E SFIDE DEI DRONI

5.1 Che cos'è un drone?

Un aeromobile a pilotaggio remoto (APR), noto comunemente come drone, è un velivolo caratterizzato dall'assenza di un pilota umano a bordo. Il suo volo è controllato da un computer a bordo del mezzo aereo oppure tramite il controllo remoto di un navigatore o pilota.
Fonte: https://it.wikipedia.org/wiki/Aeromobile_a_pilotaggio_remoto

5.2 Vantaggi ambientali dell'uso dei droni elettrici

L'uso dei droni per il trasporto delle merci presenta dei vantaggi in termini di sostenibilità rispetto ai metodi di trasporto tradizionali, come i camion alimentati a combustibili fossili. I droni, infatti, utilizzano energia elettrica, una risorsa che può essere generata da fonti rinnovabili come il solare, l'eolico e l'idroelettrico. Soprattutto su brevi distanze e per carichi leggeri, possono essere il più efficiente mezzo dal punto di vista energetico, inoltre non richiedono infrastrutture stradali dedicate.

5.3 Svantaggi e rischi

Tra gli svantaggi principali vi sono le limitazioni legate all'autonomia e alla capacità di carico, che rendono questi dispositivi adatti solo per trasporti di breve raggio e con pesi contenuti.
Un altro aspetto critico è rappresentato dalla sicurezza, sia in termini di rischio di collisione con ostacoli, altri velivoli o persone, sia per possibili cadute improvvise.
Infine, vi sono implicazioni legate alla privacy e all'accettazione sociale. L'uso di droni in ambienti urbani potrebbe provocare disagi legati al rumore e sollevare preoccupazioni sulla raccolta di dati e immagini.

5.4 Regolamentazione e normative

A livello europeo, l'impiego dei droni è regolamentato dall' European Union Aviation Safety Agency (EASA), che impone rigide regolamentazioni per garantire la sicurezza e la gestione del traffico dei velivoli a pilotaggio remoto. In Italia, l'autorità competente per la vigilanza e il controllo nel settore dell'aviazione civile è l'Ente Nazionale per l'Aviazione Civile (ENAC), che implementa le direttive europee stabilendo normative specifiche a livello nazionale.
Fonte: https://www.enac.gov.it/

6. CONCLUSIONI FINALI

6.1 Il problema del commesso viaggiatore

Dopo ogni permutazione di carico, l'algoritmo si confronta con il problema del commesso viaggiatore, che ha l'obiettivo di trovare il percorso più breve per visitare tutte le città esattamente una volta. Questo problema, classificato come NP-hard, comporta un numero di possibili percorsi in crescita esponenziale: per n città, in genere, esistono (n-1)! possibili soluzioni. In altre parole, non esiste alcun algoritmo noto in grado di risolvere il problema in tempo polinomiale per ogni istanza.

Tale complessità rende difficile l'approccio esaustivo a problemi di grandi dimensioni. Per questo motivo, si è scelto di limitare il numero delle consegne a un massimo di dieci operando su un singolo quartiere anziché servire l'intera città di Milano, evitando così di sovraccaricare il sistema. Quando le consegne sono molto numerose, le aziende di logistica si trovano a dover gestire numeri estremamente elevati. Per ovviare a questo problema, nella pratica vengono utilizzati algoritmi euristici e di approssimazione, che pur non garantendo sempre la soluzione ottimale, consentono di ottenere risultati buoni in tempi ragionevoli.

6.2 Analisi delle prestazioni e tempi di risposta

Possiamo notare che l'applicativo risponde bene agli input dell'utente, presentando tempi di reazione sufficientemente brevi. Tra i punti di forza emergono la *ricerca esaustiva*, approccio che consiste nell'esplorare tutte le possibili combinazioni tra i rami considerati validi, e l'*ottimizzazione deterministica* degli algoritmi di ricerca ricorsivi che elimina i rami non promettenti, evitando esplorazioni inutili e migliorando l'efficienza. Questo approccio è utilizzato in tecniche come il branch-and-bound.

L'analisi dei risultati della simulazione consente di ottenere una visione completa delle prestazioni dell'applicativo in contesti reali. Per concludere, l'applicativo risponde bene agli input e, con una premessa di accuratezza dei dati raccolti, risolve un problema potenzialmente reale.