Alma Mater Studiorum · Università di Bologna

Scuola di Scienze

Dipartimento di Informatica – Scienza e Ingegneria

Corso di Laurea in Informatica

ANALISI E SVILUPPO DI CONSEGNA CON DRONI E TRASPORTO PUBBLICO

Relatore:
Angelo Trotta

Presentata da:
Andreea Scrob

III Sessione

Anno Accademico 2022-2023

Niente è davvero
difficile se lo si divide
in tante piccole parti.

Sommario

Il crescente sviluppo del settore degli acquisti a distanza ha reso imperativa l'identificazione di nuove strategie per la consegna di pacchi, al fine di accostarle al metodo tradizionale che impiega veicoli a combustione.
Sì è scelto quindi di utilizzare i droni. Da un'analisi approfondita di vantaggi e svantaggi, si è individuato come problema chiave: la breve autonomia della batteria. Al fine di superare questa limitazione, è stata esaminata e sperimentata un'innovativa soluzione basata sull'utilizzo di mezzi pubblici per la ricarica e per l'avvicinamento alla destinazione.
Il presente studio approfondisce dettagliatamente le tempistiche coinvolte in questo nuovo approccio, confrontandole con quelle relative alla consegna esclusivamente aerea. I risultati indicano che l'adozione di mezzi pubblici per la ricarica contribuisce significativamente a ridurre i tempi complessivi di consegna, offrendo un'alternativa promettente alle limitazioni intrinseche della durata delle batterie dei droni.

Indice

Introduzione **1**

1 Letteratura **3**

1.1 Perché impiegare i droni per le consegne? 4

1.2 Problematiche 5

1.3 Risvolti futuri 6

2 System model **7**

2.1 Dataset delle linee del trasporto pubblico 8

2.2 Informazioni preliminari 9

2.3 Funzioni di supporto 12

2.3.1 Creare il grafo 13

2.3.2 Distanza aerea 14

2.3.3 Ricerca dei cammini minimi 15

2.3.4 Funzioni ausiliari 21

3 Modelli di spedizione dei pacchi **27**

3.1 Algoritmo solo in volo 27

3.2 Algoritmo solo in autobus 29

3.3 Algoritmo misto 30

4 Valutazione **33**

4.1 Analisi dei algoritmi di ricerca 33

4.1.1 Analisi di Alg_fly 33

4.1.2 Analisi di Alg_bus 35
4.1.3 Analisi di Alg_mix 36
4.1.4 Confronto 37
4.2 Analisi dell'orario di partenza 38

5 Conclusioni **41**
5.1 Vantaggi 41
5.2 Svantaggi 41
5.3 Studi futuri 42

Glossario e acronimi **45**

Bibliografia **47**

Ringraziamenti **49**

Elenco delle tabelle

2.1 Estratto dal file agency.txt. 8
2.2 Estratto dal file calendar_dates.txt. 9
2.3 Estratto dal file routes.txt. 10
2.4 Estratto dal file stop_times.txt. 11
2.5 Estratto dal file stops.txt 11
2.6 Estratto dal file trips.txt. 12

Elenco delle figure

2.1 Rappresentazione di una porzione di grafo 14
2.2 Immagine stampa dalla funzione **fn_air_distance** che calcola la distanza aerea tra la fermata 7114 e 305. 15
2.3 Diagramma di Flusso dell'algoritmo di Dijkstra modificato. 18
2.4 Rappresentazione della I parte del codice di **calculate_bus_time**. . . 24
2.5 Rappresentazione della II parte del codice di **calculate_bus_time**. . 24

3.1 Diagramma di flusso del funzionamento dell'Algotitmo Misto 32

4.1 Velocità della consegna calcolata da Alg_fly. 34
4.2 Velocità della consegna calcolata da Alg_bus. 35
4.3 Velocità della consegna calcolata da Alg_mix. 36
4.4 Confronto delle velocità di consegna dei tre algoritmi. 37
4.5 Confronto delle velocità di consegna dei tre algoritmi in base all'orario di partenza. 39
4.6 Confronto delle velocità di consegna dei tre algoritmi nella giornata del 17 Novembre 2023. 40

Introduzione

Nella società moderna, l'acquisto online è diventato una pratica comune e tutti siamo consapevoli delle sfide legate ai tempi di spedizione, spesso dipendenti da vari fattori. Tradizionalmente i pacchi vengono affidati a corrieri umani che si spostano utilizzando veicoli per consegnare gli acquisti direttamente a casa nostra. Tuttavia, stiamo assistendo a una trasformazione nell'approccio alla consegna, specialmente in paesi come l'Inghilterra, dove piccoli robot autonomi vengono impiegati per consegnare cibo a domicilio, particolarmente efficaci per distanze relativamente brevi.

Ma cosa succederebbe se spingessimo ulteriormente i confini dell'innovazione nella consegna di pacchi? Un'opzione intrigante è rappresentata dall'impiego di droni come mezzi di consegna. Si immagini un futuro in cui, anziché affidarci esclusivamente a veicoli e corrieri tradizionali, si potrebbe sfruttare la versatilità e la rapidità di droni per rendere le consegne ancora più efficienti.
L'adozione della capacità dei droni di superare ostacoli e percorrere distanze più velocemente rispetto ai veicoli tradizionali apre la strada a consegne più rapide e meno vincolate dal traffico stradale.

L'impiego di droni per la consegna non solo potrebbe ridurre i tempi di attesa, ma potrebbe anche abbattere i costi operativi e ridurre l'impatto ambientale. L'efficienza e la sostenibilità di tale approccio potrebbero rappresentare un passo significativo verso il futuro della logistica e delle consegne, portando a un'evoluzione nella nostra esperienza quotidiana di ricevere pacchi direttamente nelle nostre case.

La sfida primaria nell'implementare la consegna tramite droni risiede nella limitata autonomia delle batterie, che riduce la portata delle consegne a distanze relativamente brevi, dell'ordine di poche decine di chilometri. Questo aspetto rappresenta un ostacolo significativo all'efficienza e all'estensione delle operazioni di consegna con droni.

Per affrontare questo problema, la presente tesi si concentrerà su un approccio innovativo che cerca di superare tale ostacolo proponendo l'utilizzo delle reti di autobus, in particolare quelle gestite da Tper a Bologna, come punti di ricarica per i droni.
Si condurrà un'analisi approfondita dei percorsi dei trasporti pubblici, inclusi gli orari e le frequenze delle fermate, al fine di strutturare al meglio gli itinerari delle linee e ottimizzare i tempi di ricarica per i droni. L'obiettivo è massimizzare l'efficienza e ridurre al minimo il tempo di fermo, consentendo ai droni di prolungare la propria autonomia e coprire distanze più estese.

1 | Letteratura

Come dice il nome, gli aeromobili a pilotaggio remoto (APR), o in inglese *unmanned aerial vehicle* (UAV), noti anche come droni, sono veicoli in grado di volare senza la presenza di un umano a bordo [1].
Questi possono essere distinti in diverse categorie in base alle caratteristiche specifiche; tuttavia, ci concentreremo sulla modalità di pilotaggio. In questo caso esistono tre tipologie principali di droni:

1. a **guida assistita**, questi sono spesso utilizzati nell'ambito fotografico e cinematografico. Un operatore umano controlla il drone per catturare immagini o video da diverse prospettive.

2. **semi-autonomi**, questi sono in grado di eseguire alcune operazioni autonomamente ma richiede l'intervento umano per compiti più complessi o situazioni impreviste. Ad esempio, possono seguire un percorso programmato autonomamente e necessitare della supervisione umana in caso di ostacoli imprevisti.

3. **totalmente autonomi**, questo genere di droni è in grado di operare completamente in modo autonomo senza l'intervento umano durante la fase di volo. Sono dotati di avanzati sistemi di intelligenza artificiale e sensori per evitare ostacoli, pianificare percorsi e prendere decisioni autonome. Possono essere impiegati in attività come sorveglianza, monitoraggio ambientale o consegne autonome.

Il nostro studio si concentrerà proprio su quest'ultima categoria, in particolare per quanto riguarda le spedizioni.

1.1 Perché impiegare i droni per le consegne?

Al giorno d'oggi, il modo più comune per trasportare pacchi è utilizzare veicoli come furgoni o camion. Tuttavia, questo approccio spesso comporta tempi di consegna molto lunghi a causa della limitata disponibilità di mezzi rispetto alla costante crescita del volume di merci o al traffico presente sulle strade.

L'impiego di **aeromobili**[1] offre numerosi vantaggi, come la riduzione della congestione del traffico [2], diminuendo la presenza di camion sulle strade, e la capacità di raggiungere aree con infrastrutture stradali insufficienti.
Si può immaginare l'enorme aiuto che potrebbero fornire in scenari di disastri naturali, dove il movimento dei veicoli su ruote è impedito, o quanto faciliterebbero lo scambio di medicinali nelle zone di guerra.

Uno dei settori che potrebbe trarre maggior vantaggio è quello alimentare, ad esempio i ristoranti, che potrebbero ridurre le distanze in modo più veloce, consentendo al cibo di arrivare probabilmente più caldo. Non solo, ma anche il settore medico ne trarrebbe beneficio, ad esempio attraverso lo scambio di sangue o la consegna più rapida di farmaci e vaccini [3].

Inoltre, uno sviluppo interessante nel settore dei trasporti riguarda l'esplorazione dell'utilizzo di droni per lo spostamento di passeggeri. A tal proposito, Uber stretto una collaborazione con Pipistrel [4], un'azienda rinomata nel campo dell'aviazione leggera. Pipistrel è nota per la produzione di velivoli innovativi e ha recentemente manifestato interesse nella realizzazione di droni per il trasporto di persone. Questo progetto potrebbe rivoluzionare il concetto di mobilità urbana, offrendo una soluzione rapida e efficiente per il trasporto individuale.

[1] o *aircraft* sono veicoli in grado di volare grazie al supporto dell'aria. Contrappongono la forza di gravità utilizzando la potenza statica o dinamica di un profilo alare, o, nel nostro caso, sfruttando la spinta verso il basso generata dai motori. Esempi comuni di aeromobili sono gli elicotteri, i droni e gli aquiloni.

1.2 Problematiche

La prima problematica che può sorgere riguarda il posizionamento del pacco in modo tale da non compromettere l'equilibrio di volo del drone. Sebbene possa sembrare intuitivo posizionare il carico al di sopra, studi recenti [5] hanno dimostrato che, se il pacco è eccessivamente grande, può ostacolare il flusso d'aria delle eliche, generando una resistenza significativa. Da ciò deriva la conclusione che i pacchi dovrebbero essere posizionati al di sotto del velivolo, con l'ulteriore accorgimento che le dimensioni non devono superare quelle dei quattro rotori[2], al fine di evitare uno sbilanciamento eccessivo.

Un'altra problematica da considerare è quella della sicurezza, che viene affrontata in diversi aspetti: dalla sicurezza in volo del drone, alla sicurezza di agenti esterni, come animali o esseri umani [6], fino alla sicurezza durante l'atterraggio [7]. Si è consolidata la consapevolezza, che per garantire la sicurezza del drone e di ciò che lo circonda, è fondamentale dotarlo di protezioni attorno alle eliche.

Infine, affrontiamo la problematica principale che caratterizza questa tipologia di veicoli: la limitata capacità della batteria. Tale limitazione impedisce l'utilizzo del drone su distanze molto lunghe, soprattutto quando è richiesto un peso di carico considerevole, poiché l'aumento del peso comporta un incremento del consumo energetico [8]. Una soluzione potrebbe prevedere stazioni di ricarica rapida distribuite strategicamente, consentendo ai droni di effettuare brevi soste per la ricarica o addirittura la sostituzione della batteria. Questo approccio garantirebbe una continuità operativa senza prolungati tempi di inattività.

Ai fini del funzionale, si sta considerando la possibilità d'implementare un sistema di autenticazione sia da parte del cliente che del drone, in quanto fondamentale per garantire sicurezza e tracciabilità. Il cliente potrebbe autenticarsi tramite un'applica-

[2] si riferisce a una parte di un dispositivo che ruota attorno a un asse centrale. Nell'ambito dei droni e degli aeromobili, il termine si riferisce generalmente alle pale di un'elica o di un'ala rotante che genera la forza necessaria per il sollevamento e il movimento del veicolo.

zione mobile, o un altro mezzo di identificazione, consentendo al drone di consegnare il pacco solo al destinatario autorizzato. D'altra parte, il drone dovrebbe essere dotato di un sistema di autenticazione proprio per garantire che la consegna avvenga solo a persone autorizzate, evitando il rischio di frodi o furto di pacchi.

1.3 Risvolti futuri

Superate queste problematiche iniziali, l'impiego di droni dovrebbe effettivamente portare a vantaggi significativi. La velocizzazione delle tempistiche di spedizione sarebbe innegabile, con la possibilità di effettuare consegne più rapide rispetto ai tradizionali mezzi di trasporto. L'adozione di stazioni di ricarica rapida contribuirebbe anche a mantenere un flusso operativo continuo, riducendo al minimo i tempi di fermo.

In termini di costi, l'utilizzo di droni potrebbe risultare più economico rispetto ai mezzi di trasporto tradizionali, specialmente considerando la riduzione dei costi legati al carburante e al personale.
Dal punto di vista ambientale, contribuirebbe a una diminuzione delle emissioni inquinanti rispetto ai veicoli a combustione interna. Per non escludere la silenziosità dei droni rispetto ai veicoli tradizionali che potrebbe ridurre l'impatto acustico nelle aree urbane.

In conclusione, superare le sfide iniziali e implementare soluzioni efficaci potrebbe portare a un uso esteso di droni per la consegna, offrendo numerosi benefici in termini di velocità, costi, sostenibilità ambientale e riduzione del rumore.

2 | System model

Si sono già anticipate le modalità di smistamento di pacchi per gli acquisti da remoto, esplorando l'ipotesi di integrare i droni nel meccanismo di trasporto per ridurre i tempi e migliorare il servizio. Tuttavia, uno dei principali ostacoli a questa innovativa idea è la limitata capacità delle batterie. Questa restrizione impone una distanza massima che i droni possono coprire prima di dover tornare per la ricarica. Affrontare questa sfida diventa cruciale per rendere il sistema di consegna con droni più pratico e funzionale.

L'approccio intrapreso nel nostro studio si è focalizzato su una soluzione progressista per superare la limitata autonomia di questi aeromobili, sfruttando il trasporto pubblico come elemento chiave del processo di ricarica e consegna.
L'idea centrale consiste nell'utilizzare gli autobus pubblici gestiti da Tper come punti di appoggio per stazioni di ricarica, consentendo così ai droni di estendere la loro copertura e di avvicinarsi più efficacemente alle destinazioni finali.
La collaborazione con il trasporto pubblico si basa su un approccio sinergico: anziché introdurre nuove infrastrutture di ricarica fisse, dedicate ai droni, sparse per la città, sfruttiamo la già esistente rete di autobus come base di supporto, diventando così punti strategici dove i droni possono atterrare e collegarsi automaticamente per ricaricare le loro batterie.
Un aspetto fondamentale di questo approccio è stata la selezione accurata degli autobus su cui i droni si appoggiano. Questa selezione non è casuale; al contrario, è mirata a garantire che, durante la fase di ricarica dei droni, essi si avvicinino gradualmente alla destinazione di consegna.

2.1 Dataset delle linee del trasporto pubblico

Durante la fase introduttiva del nostro progetto ci si è focalizzati principalmente sull'analisi dei materiali di partenza. I dati chiave impiegati riguardano i percorsi e gli orari degli autobus.
Attraverso il sito ufficiale della Tper, eseguendo una selezione per *servizio su gomma*, abbiamo ottenuto una serie di file testo di rilevanza per la nostra indagine, in particolare quelli classificati sotto la denominazione "**gommagtsbo**". Questi file costituiranno la base principale del nostro studio. Per fornire ulteriori dettagli sulla continuazione dell'analisi, di seguito ne sono elencati alcuni:

- **agency.txt**: contiene informazioni relative all'azienda Tper, come l'id, il nome, il sito web, ecc... .

agency_id	agency_name	agency_url
TPERBO	TPER spa	https://www.tper.it

agency_timezone	agency_lang	agency_phone
Europe/Rome	it	051 290290

Tabella 2.1: Estratto dal file agency.txt.

- **calendar_dates.txt**: riportato nella tabella 2.2, è fondamentale nel controllare se un determinato "trip_id", dopo aver ricavato il corrispondente "service_id", è disponibile in un determinato giorno.

- **routes.txt**: raffigurato nella tabella 2.3 da cui possiamo convertire i nome delle linea degli autobus avendo il loro numero, detto anche "route_id". Le ultime tre colonne non sono riportate, in quanto irrilevanti ai fini della ricerca.

- **stop_times.txt**: come possiamo vedere nella tabella 2.4, viene spesso utilizzata per ottenere il "trip_id" degli autobus che passano nel orario di interesse, "departure_time" alla fermata di riferimento, indicata dallo "stop_id". Si noti che la

service_id	date	exception_type
1521_1000000018845	20231004	1
1521_1000000018846	20231004	1
1521_1000000018847	20231004	1
⋮	⋮	⋮
1521_1000000019431	20240122	1
1521_1000000019431	20240123	1
⋮	⋮	⋮
1521_1000000020439	20240601	1
1521_1000000020443	20231001	1

Tabella 2.2: Estratto dal file calendar_dates.txt.

colonna relativa al "arrival_time" non verrà mai menzionata e, di conseguenza, mai utilizzata nello studio in quanto è identica a quella del "departure_time".

- **stops.txt**: questo file è riportato nella tabella 2.5 contiene tutti gli id univoci delle fermate, i loro corrispondenti nomi e le locazioni geografiche. Le ultime colonne sono per "location_type" e "parent_station" che non verranno utilizzate nello studio.

- **trips.txt**: infine abbiamo il file che associa il "route_id" con "service_id" con "trip_id", funge da collante con i file precedenti.

2.2 Informazioni preliminari

Ne giunge che a Bologna sono attive 239 linee di autobus, ciascuna identificata da un numero specifico che, nelle tabelle, è riportato sotto la voce "**route_id**" e corrisponde al numero dell'autobus, come ad esempio l'autobus 20, il quale ha come "**route_long_name**" *San Biagio - Casalecchio di Reno - Pilastro*. Come i bolognesi sapranno, in quanto linea molto usata, il numero 20 passa molto di frequente, nei orari di punta anche ogni 5 minuti. Dato che distinguerli con il "route_id" non sarebbe efficiente, a ciascun viaggio è associato un "**trip_id**" univoco. Inoltre, per

route_id	agency_id	route_short _name	route_long_name	...	...	...
101	TPERBO	101	Bologna - Castel San Pietro Terme - Imola	...	...	...
103	TPERBO	103	Piancaldoli - Castel San Pietro Terme	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮
20	TPERBO	20	San Biagio - Casalecchio di Reno - Pilastro	...	...	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮
98	TPERBO	98	Bologna - Castel Maggiore	...	...	...
99	TPERBO	99	Bologna - Medicina - Lugo	...	...	...

Tabella 2.3: Estratto dal file routes.txt.

assicurarci che un determinato "**trip_id**" sia attivo in una data specifica, utilizziamo il "**service_id**", che tramite la tabella 2.2, ci permette di verificare se c'è una corrispondenza.

Nel corso dello studio, è stata adottata la scelta di monitorare attentamente i punti di partenza e di arrivo, ancorandoli alle fermate degli autobus. In un contesto ipotetico, consideriamo, ad esempio, lo *Stadio Renato Dall'Ara* come punto di partenza e il dipartimento di Informatica come destinazione. In questa situazione, utilizziamo le fermate degli autobus più vicine a questi luoghi come punti di partenza e di arrivo per i droni.
Ecco un esempio dei dati di elaborazione:

- partenza: stop_id = **7114** *Stadio* (idealmente in corrispondenza al deposito dei droni);

trip_id	arrival_time	departure_time	stop_id	stop_sequence
1521_9030970	15:00:00	15:00:00	8	1
1521_9030970	15:01:00	15:01:00	410	2
1521_9030970	15:03:20	15:03:20	301	3
⋮	⋮	⋮	⋮	⋮
1521_9151851	06:52:00	06:52:00	410	24
⋮	⋮	⋮	⋮	⋮
1521_9267261	24:08:40	24:08:40	243026	31
1521_9267261	24:10:00	24:10:00	28558	32

Tabella 2.4: Estratto dal file stop_times.txt.

stop_id	stop_name	stop_lat	stop_lon	...	...
1	STAZIONE CENTRALE	44.50576575	11.34317589	...	
10	PORTA GALLIERA	44.50444281	11.3463873	...	
⋮	⋮	⋮	⋮	⋮	⋮
50617	BEVILACQUA STELLONI	44.75907714	11.24685142	...	
50618	BEVILACQUA STELLONI	44.75911273	11.24675804	...	
⋮	⋮	⋮	⋮	⋮	⋮
99010	VILLA SPADA	44.48874504	11.31613943	...	
99013	CASAGLIA 31	44.48468919	11.31428903	...	

Tabella 2.5: Estratto dal file stops.txt

- destinazione: stop_id = **305** *Porta San Donato* (idealmente in corrispondenza del dipartimento di Informatica);

La maggior parte delle volte si farà riferimento alle fermate, non tramite il loro "stop_name", quindi il loro nome, ma bensì tramite il "stop_id", in quanto univoco rispetto al primo.

route_id	service_id	trip_id	trip_ headsi- gn	direction _id	shape_id
1	1521_1000000019461	1521_9051539		0	1019639013
1	1521_1000000019461	1521_9051540		0	1019639013
⋮	⋮	⋮	⋮	⋮	⋮
25	1521_1000000019720	1521_9050904		1	3689130834
25	1521_1000000019720	1521_9050905		1	3689130834
⋮	⋮	⋮	⋮	⋮	⋮
99	1521_1000000019857	1521_9064531		1	1724660597
99	1521_1000000019857	1521_9064532		1	233390887

Tabella 2.6: Estratto dal file trips.txt.

2.3 Funzioni di supporto

Fin dall'inizio è stato evidente che dovevamo individuare una struttura dati efficiente per calcolare i percorsi degli autobus, dato che l'analisi iterativa non risultava ottimale. Dopo un'approfondita valutazione, si è scelto di creare un grafo orientato utilizzando i dati disponibili.
Per lo studio è stato quindi sviluppato un programma in **Python** e principalmente sono state utilizzate le seguenti librerie:

- **Pandas**: per aprire e leggere agevolmente i file di testo del dataset;
- **Networkx**: per sfruttare le funzioni disponibili per la creazione, gestione e lettura del grafo;
- **Matplotlib**: per poter visualizzare il grafo e creare i grafici relativi alla parte di analisi.

2.3.1 Creare il grafo

Come evidenziato nel seguente codice, il processo inizia aprendo e leggendo i file *stops.txt* e *stop_times.txt*. Successivamente, viene creato un grafo **G**, che in questo caso comprende 6614 nodi e 30963 archi. I nodi del grafo rappresentano le fermate degli autobus e sono distinti in base ai loro identificatori univoci **stop_id** e **stop_name**. Questi nodi sono interconnessi tramite archi, i quali sono identificati dai valori di **trip_id** e **departure_time**. Pertanto, il grafo offre una rappresentazione chiara e strutturata delle relazioni spazio-temporali tra le fermate degli autobus nel sistema di trasporto pubblico.

```
# Leggi i dati dai file CSV
stops_df = pd.read_csv('stops.txt')
stop_times_df = pd.read_csv('stop_times.txt')

# Inizializza un grafo diretto
G = nx.MultiDiGraph()
#Aggiunge i nodi
for index, row in stops_df.iterrows():
    G.add_node(row['stop_id'], stop_name=row['stop_name'])

prev_stop_id = None
prev_trip_id = None
#Aggiunge gli archi
for index, row in stop_times_df.iterrows():
    trip_id = int(row['trip_id'])
    departure_time = row['departure_time']
    stop_id = int(row['stop_id'])

    if prev_stop_id is not None and
    stop_id != prev_stop_id and trip_id == prev_trip_id:
        G.add_edge(prev_stop_id, stop_id,
        trip_id=prev_trip_id, departure_time=departure_time)

    prev_stop_id = stop_id
    prev_trip_id = trip_id
```

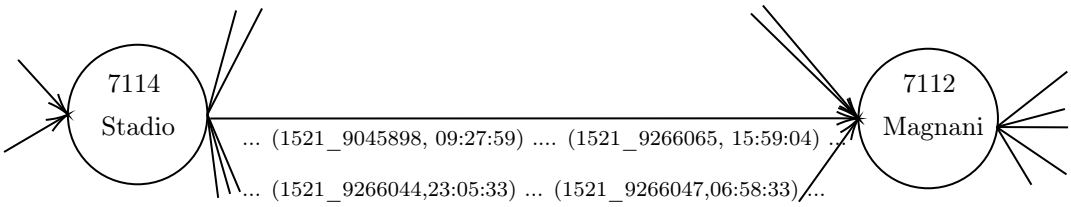

Figura 2.1: Rappresentazione di una porzione di grafo

Nella figura 2.1 troviamo una porzione del grafo delle linee degli autobus di Bologna. I nodi 7114 e 7112 sembrerebbero apparentemente essere collegati da un unico arco; tuttavia, un'analisi più attenta rivela una serie di coppie sottostanti a esso. Ciascuna di queste coppie identificherebbe un arco che collega i suddetti nodi. Quindi, ciò che in inizialmente sembrerebbe essere un singolo arco potrebbe rivelarsi essere una sovrapposizione di centinaia di essi. Per approfondire, analizziamo alcune di queste coppie:

- (1521_9045898, 09:27:59), identifica l'autobus, con il **reoute_id**, 14 che parte alle 09:27:59, nei fine settimana, dalla fermata 7114, verso la 7112;
- (1521_9266065, 15:59:04), identifica l'autobus 21 che parte alle 15:59:04, nei giorni festivi, dalla fermata 7114 verso la 7112;
- (1521_9266044, 23:05:33), identifica l'autobus 21B che parte nel orario notturno 23:05:33, nei giorni festivi, dalla fermata 7114 verso la 7112;

2.3.2 Distanza aerea

Una funzione cruciale per lo sviluppo di uno degli algoritmi che esamineremo nel prossimo capitolo è quella dedicata al calcolo della distanza aerea, **fn_air_distance**, tra due fermate degli autobus. In particolare, questa funzione prende in input la fermata di partenza e la fermata di arrivo, estraendo le rispettive coordinate dal file *stops.txt*. Successivamente, tali coordinate vengono utilizzate per determinare la distanza in linea retta tra le due fermate, contribuendo così alla precisione e all'efficacia dell'algoritmo in esame. Per migliorare l'esperienza utente, si è integrato l'utilizzo di **Mapbox** per

la visualizzazione su mappa di tutte le informazioni.

Nella figura 2.2, è possibile visualizzare l'output generato dal metodo. Dopo aver fornito in input la fermata di partenza (**7114**) e quella di arrivo (**305**), la funzione identifica queste fermate sulla mappa attraverso stelle blu. Successivamente, calcola la distanza in linea retta tra di esse, evidenziando graficamente il tragitto in rosso. In questo caso la distanza risultante è di **3.6045071525060237 Km**.

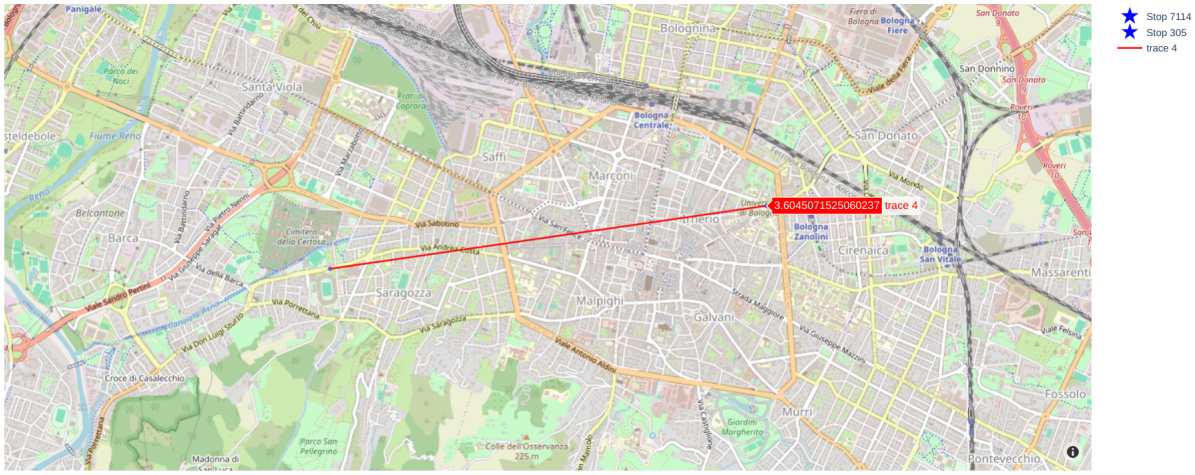

Figura 2.2: Immagine stampa dalla funzione **fn_air_distance** che calcola la distanza aerea tra la fermata 7114 e 305.

2.3.3 Ricerca dei cammini minimi

Nell'ambito delle considerazioni decisionali sulla scelta delle tecnologie di sviluppo, si è valutato l'utilizzo della libreria **Networkx**, focalizzandosi in particolare sulle sue funzionalità integrate per gli algoritmi di **Dijkstra** [9] e **Bellman-Ford** [10]. Tuttavia, dopo aver eseguito alcuni tentativi di utilizzo di questi algoritmi su il grafo creato, si sono riscontrate alcune limitazioni che li rendevano inefficaci per le nostre esigenze.

Nel dettaglio si è notato che l'approccio generale degli algoritmi di Dijkstra e Bellman-Ford restituiva il percorso più breve tra un nodo di origine e uno di destinazione, senza

considerare alcuni fattori essenziali per il nostro contesto specifico. Ad esempio, non era considerata la disponibilità variabile degli archi in base all'orario o alla data. Inoltre, la restituzione di un array di nodi richiedeva ulteriori elaborazioni per ottenere tutte le informazioni desiderate, come gli orari associati alle fermate, gli autobus coinvolti e i corrispondenti **route_id**.

Di conseguenza, si è optato per reimplementare un algoritmo di ricerca dei cammini minimi da zero che ci desse maggiore flessibilità nelle specifiche richieste. Inoltre, abbiamo adottato una metodologia di calcolo dei pesi degli archi che si adattava meglio alle nostre necessità, garantendo una maggiore aderenza ai vincoli del sistema.

Calcolo dei pesi degli archi

La funzione **calculate_edge_weight** è progettata per calcolare il peso di un arco, tenendo conto della disponibilità di un servizio in una data specifica. Questo è il cuore del nostro algoritmo di Dijkstra, in quanto ci permette di verificare le nostre esigenze temporali.

Innanzitutto, la funzione recupera il **service_id** associato al **trip_id** corrente, per permetterci di controllare quando un arco è disponibile in una data specifica.
Se il servizio non è disponibile, la funzione assegna come perso infinito, rendendolo praticamente indesiderato per la selezione dei percorsi.

In seguito viene presa in considerazione la differenza di orario, tra quello di partenza e quello di arrivo; in primo luogo escludere quelli con una differenza d'orario superiore a un'ora e quelli che, ovviamente, partono prima del nostro arrivo.

Infine, se i **trip_id** associati all'arco corrente e a quello precedente coincidono, la differenza di tempo è considerata come il peso dell'arco. Ciò implica che l'arco rappresenta un segmento continuo del viaggio all'interno dello stesso viaggio o percorso.
Tuttavia, se i **trip_id** sono diversi, la funzione assegna un peso più elevato all'arco. Questo viene fatto per favorire gli archi con lo stesso **trip_id**, incoraggiando la conti-

nuità all'interno dello stesso autobus.

In sostanza, questa funzione racchiude le considerazioni legate alla disponibilità del servizio, alle differenze di tempo e alla continuità del viaggio, producendo un peso dell'arco adatto per il calcolo del percorso più veloce.

Algoritmo di Dijkstra

L'algoritmo di **Dijkstra** è un algoritmo classico utilizzato per trovare il percorso più breve tra due nodi in un grafo ponderato e orientato. Inizia con una distanza infinita per tutti i nodi tranne il nodo di partenza, che ha distanza 0. Ad ogni passo, come mostrato anche nel diagramma di flusso della figura 2.3[1], seleziona il nodo con la distanza minima, esplora i suoi vicini e aggiorna le distanze se un percorso più breve è trovato. Il processo continua finché tutti i nodi sono stati visitati o la destinazione è raggiunta. Questo algoritmo è ampiamente usato in applicazioni come reti di computer, sistemi di trasporto e logistica, dove è necessario determinare il percorso ottimale con il minor costo possibile.

Come già anticipato, ci sono numerose librerie che lo implementano e permettono di utilizzarlo applicandolo al proprio grafo, ma nessuna rispettava le richieste di studio, per questo motivo si è deciso d'implementare nuovamente l'algoritmo. Le strutture dati utilizzate, riportate qui sotto, sono quelle che solitamente si trovano in un generico algoritmo di Dijkstra e che vengo utilizzate anche nel progetto di studio:

- **dizionario**: per memorizzare le distanze minime (**distances**), dai nodi di partenza a tutti gli altri nodi nel grafo, e i nodi padre (**previous_nodes**), ovvero i nodi precedenti a quello considerato, associandoli informazioni come anche il **trip_id** e il **departure_time** .
- **coda di priorità**: è essenziale per selezionare in modo efficiente il nodo con la distanza minima durante ogni iterazione. Spesso si ricorre all'utilizzo di una

[1]In nero troviamo le parti dell'algoritmo di Dijkstra e in verde le parti aggiunte per adattarlo alle richieste del programma.

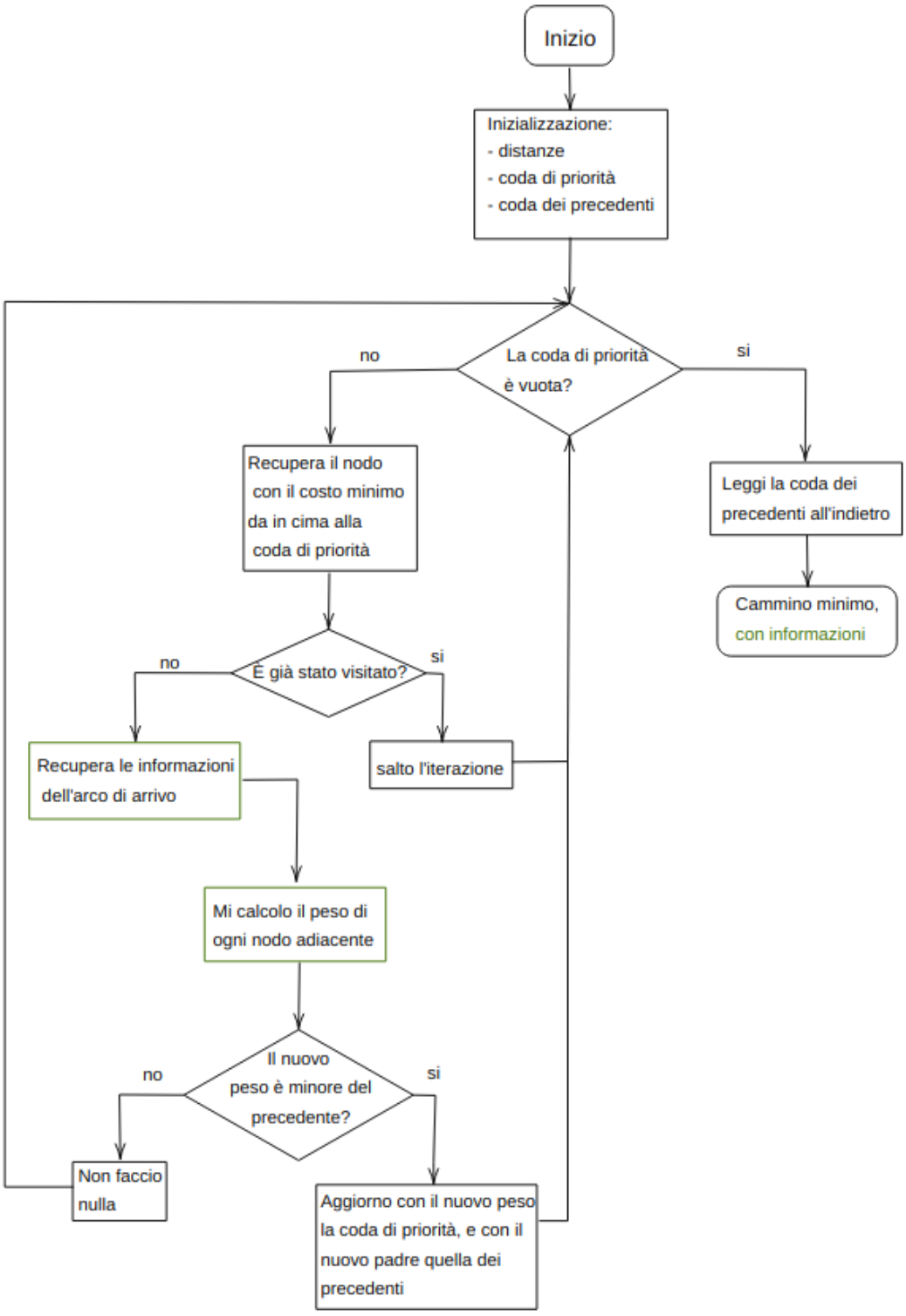

Figura 2.3: Diagramma di Flusso dell'algoritmo di Dijkstra modificato.

heap, una struttura dati che consente di aggiornare i pesi all'interno della coda in modo efficiente. In Python non è possibile riprodurre fedelmente questa funzionalità, ma è possibile inserire lo stesso nodo con un peso maggiore nella coda, il che può causare la presenza di nodi duplicati con pesi diversi.

- **manuale**: per attenuare il problema del punto precedente si fa uso di un **set** che tiene traccia dei nodi già visitati, assicurando che ciascun nodo venga elaborato una sola volta.

```
def dijkstra_modified(graph, source_node, destination_node,
    departure_time, data):
     distances = {node: float('inf') for node in graph.nodes}
     previous_nodes = {node: {'node': None, 'departure_time': None, '
    trip_id': None} for node in graph.nodes}

     distances[source_node] = 0
     priority_queue = []

     for node in distances:
         priority_queue.append((distances[node], node))
     heapq.heapify(priority_queue)

     seen = set()
     #Continua ...
```

Listing 2.1: Inizializzazione delle strutture dati.

Nella figura 2.3 sono evidenziate in verdi le parti che sono state aggiunte rispetto la versione tradizionale, eccone le specifiche:

- **informazioni temporali**: come abbiamo già ampiamente anticipato, questa implementazione tiene conto delle informazioni temporali, come l'orario di partenza.

- **informazioni di tragitto**: quando un arco viene scelto, per il cammino minimo, si tiene traccia del trip_id associato. Questo è importante per noi poiché così possiamo risalire all'autobus al quale il drone deve appoggiarsi.

- **considerazione dei viaggi futuri**: attraverso l'utilizzo di una funzione ausiliaria, denominata **find_future_trip**, siamo in grado di determinare l'identificativo (trip_id) del viaggio programmato nelle vicinanze di un orario specificato. Questa capacità di individuare il viaggio più prossimo consente di calcolare con precisione il momento in cui il drone dovrebbe prendere un particolare autobus. Inoltre, questa funzione espande significativamente le opzioni di percorsi disponibili, offrendo una maggiore flessibilità nella scelta delle traiettorie da considerare.

- **percorso minimo**: come ultima cosa stampa il percorso trovato con tutte le informazioni richieste. Eccone un esempio:

```
Nodo di partenza: 7012 Nodo di arrivo: 305
Orario di partenza: "08:50:40" Data di partenza: 2023/10/11
Percorso piu' breve: [7102, 710, 704, 602, 401, 405, 301, 303,
    305]
Lunghezza del percorso piu' breve: 8

Trip ID con orario di partenza "08:52:46" alla fermata 7012-
    ANDREA COSTA: 1521_9255515
La linea che devi prendere e': 89
Fermata 710-PORTA SANT'ISAIA: Orario di partenza 08:54:46

Trip ID con orario di partenza 08:58:00 alla fermata 704-PIAZZA
    MALPIGHI: 1521_9045387
La linea che devi prendere e': 20
Fermata 602-UGO BASSI: Orario di partenza 09:00:00
Fermata 401-SAN PIETRO: Orario di partenza 09:03:00
Fermata 405-VIII AGOSTO: Orario di partenza 09:05:39
Fermata 301-SFERISTERIO: Orario di partenza 09:07:00
Fermata 303-IRNERIO: Orario di partenza 09:08:02
Fermata 305-PORTA SAN DONATO: Orario di arrivo 09:10:10
```

Concludendo osservando il costo computazionale di **dijkstra_modified**, che non si discosta dal costo del algoritmo di Dijkstra tradizionale che utilizza la coda di priorità. La complessità temporale di **dijkstra_modified** è di $\mathbf{O((V+E)* \log(V))}$, dove **V**

è il numero di vertici nel grafo e **E** è il numero di archi.
La parte iniziale dedicata all'inizializzazione delle strutture dati, richiedono *O(V)*, per **distances** e **previous_nodes**, e *O(V*log(V))* operazioni per la coda di priorità.

Il ciclo principale, che viene eseguito finché la coda di priorità non è vuota, in ogni iterazione:

- estrai il nodo con la minima distanza dalla coda prioritaria. Questa operazione richiede *O(log(V))*.
- per ogni arco uscente dal nodo corrente, calcola il peso dell'arco e aggiorna le distanze e i nodi precedenti se il nuovo percorso è più breve. Questa parte richiede *O(E * log(V))* operazioni in totale per tutte le iterazioni.

Come ultimo passaggio abbiamo la ricostruzione del percorso più breve, che richiede in nel caso peggiore *O(V)* operazioni.

2.3.4 Funzioni ausiliari

In questo paragrafo anticipiamo alcune funzioni che saranno discusse nei capitoli successivi, al fine di favorire una maggiore comprensione e abitudine da parte del lettore.

Per cominciare, introduciamo le caratteristiche del drone che sono state rilevanti per lo studio.
Le variabili sotto esame sono:

- **capacità massima** della batteria che solitamente la troviamo indicata in Joule (**J**).
- **consumo in volo**
  - **da scarico**: quando il drone vola senza trasportare alcun pacco, in una situazione ideale come quella di studio, in cui non vengono prese in considerazione intemperie meteorologiche e ostacoli, il consumo che effettua Llaeromobile è costante.

  - **da carico**: quando il drone effettua il trasporto del pacco, la situazione differisce dal primo caso in quanto il drone deve gestire un peso maggiore, comportando un consumo energetico più elevato. Nello studio è stato costantemente considerato il limite massimo di consumo energetico possibile, corrispondente al trasporto del pacco più pesante che il drone è in grado di gestire.

  Esse vengono misurate in Watt (**W**).

- **potenza della base di ricarica**: per convenzione tutte le basi di ricarica, sia quelle a terra che quelle sugli autobus, saranno considerate nello studio con la stessa potenza, misurate anche esse in Watt (**W**).

- **velocità di volo** del drone, indicata in Chilometri-orari (**km/h**).

Vediamo ora le funzioni che incontreremo nel corso di questo paper.

cal_fly_time

Questa funzione calcola il timpo, in ore (**h**), necessario per percorre una certa distanza a una determinata velocità, secondo la seguente formula:

$$t = \frac{d}{v} \tag{2.1}$$

La distanza, indicata in chilometri (**km**), è calcolata con l'ausilio della funzione che determina la distanza aerea, di cui abbiamo già parlato.

max_time_fly

Utilizzata per calcolare il tempo massimo, in ore, che un drone può volare con una determinata capacità di batteria. Viene calcolato nel successivo modo:

$$t = \frac{e}{c} \tag{2.2}$$

Considerando che il consumo può variare in base al fatto che il drone stia trasportando o meno un pacco, la funzione accetta anche un parametro booleano per gestire questa condizione. Inoltre, nel caso in cui l'energia (capacità della batteria) sia indicata in

Joule, prima di calcolare il tempo viene trasformata in Wattora (**Wh**), seguendo la conversione:

$$1J = \frac{1}{3600}Wh \tag{2.3}$$

cal_recharge_time

Anche questo caso parliamo di tempo, ma in termini di quanto ci mette un drone a completare la ricarica di un ciclo di batteria. Prima di tutto si calcola quanta energia manca per raggiungere il limite massimo, ovvero:

$$\Delta cap = capMassima - capAttuale \tag{2.4}$$

Successivamente il tempo impiegato per completare il ciclo di ricarica:

$$t = \frac{\Delta cap}{p} \tag{2.5}$$

Anche in questo caso le **capacità** dovranno essere trattate in Wattora e la **potenza** della base di ricarica in Watt, cosi il tempo risulterà in ore.

calculate_bus_time

Nel contesto del nostro sistema, la funzione che stiamo esaminando è fondamentale per calcolare il tempo totale durante il quale il drone si trova sui mezzi di trasporto pubblico. Inizialmente, la funzione si occupa di trovare il percorso minimo utilizzando l'algoritmo di Dijkstra che abbiamo precedentemente implementato. Una volta determinato il percorso ottimale, la funzione procede a determinare il tempo necessario per ricaricare completamente la batteria del drone, la quale si scarica durante l'attesa del autobus in quanto aspetta in volo.

Questo periodo di tempo corrisponde al tempo che il drone trascorre sulla base di ricarica del bus, individuato da Dijkstra. Una volta raggiunta la ricarica completa, la funzione si interrompe restituendo l'informazione relativa alla successiva fermata, la **last_stop** e l'orario di arrivo corrispondente, **arrive_time**. Questo approccio consente di determinare con precisione quanto tempo il drone effettivamente passa a bordo degli

autobus, che è leggermente superiore al tempo di ricarica e che sarà successivamente sommato al tempo di attesa che per prendere l'autobus. L'obiettivo è ottimizzare l'utilizzo delle linee di autobus disponibili, massimizzando il tempo di volo del drone e riducendo al minimo i tempi di attesa.

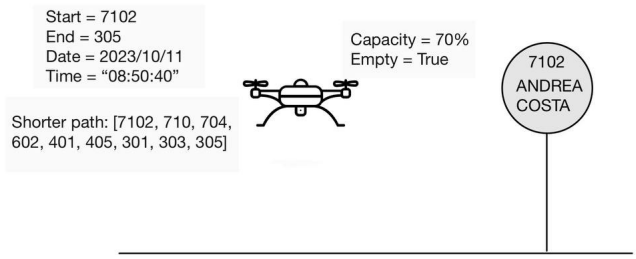

Figura 2.4: Rappresentazione della I parte del codice di **calculate_bus_time**.

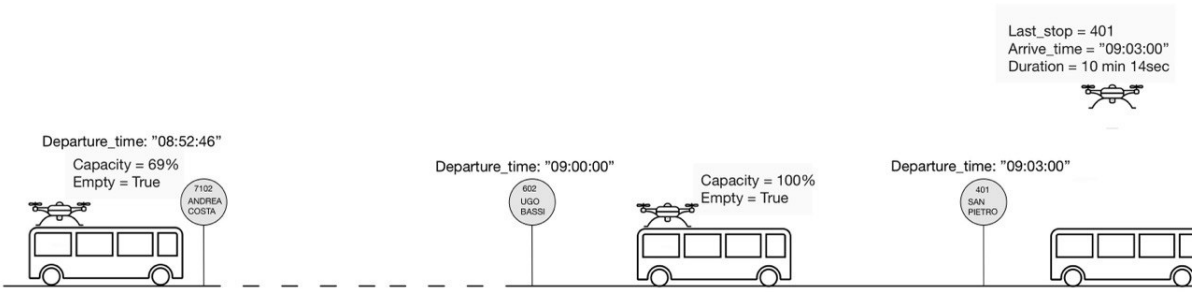

Figura 2.5: Rappresentazione della II parte del codice di **calculate_bus_time**.

La figura 2.4 è un esempio allampanate di quello che fa la funzione **calculate_bus_time**. Vediamo uno scenario in cui il drone di parte dalla fermata 7102, e deve arrivare alla 305, in data 11 Ottobre 2023 alle ore 08:50:40 e ha una batteria carica al 70%. La variabile **empty** settata a *True* indica che non sta trasportando il pacco.
La funzione calcola i percorso più corto indicato nell'immagine con **shorter path**.

Nella figira 2.5, vediamo invece una rappresentazione di quello che succede nella seconda parte del codice, ovvero dice al drone quando deve scendere, perché la batteria è completamente carica. I dati in output sono **last_stop** e **arrive_time** che corrisponde all'orario in cui arriva, che ci permette di calcolare la durata del viaggio (**duration**). Questo metodo è rappresenterà il nucleo del **Alg_mix**, che vederemo nel paragrafo 3.3.

find_future_trip

Nell'algoritmo di Dijkstra, ci troviamo inizialmente con un orario e una fermata di partenza, senza conoscere il **route_id** o il **trip_id**, poiché quest'ultima è un'etichetta degli archi, non dei nodi. Di conseguenza, l'identità dell'autobus in arrivo è sconosciuta, limitando la nostra capacità di calcolare i pesi degli archi, in quanto le informazioni sugli archi in uscita dipendono da quelli in ingresso.
La funzione svolge un ruolo cruciale, permettendoci di esaminare gli archi in ingresso al nodo di partenza, filtrandoli per l'orario più vicino a quello fornito in input e la data di partenza. In questo modo, otteniamo l'orario di arrivo desiderato e le informazioni necessarie per il calcolo dei pesi degli archi in uscita. Fondamentalmente questa funzione fornisce la capacità di determinare quali autobus stanno per giungere alla fermata di partenza, agevolando il calcolo efficiente del percorso ottimale nell'ambito dell'algoritmo di Dijkstra.

3 | Modelli di spedizione dei pacchi

Per uno studio più approfondito, sono stati scritti tre algoritmi per calcolare il tempo di spedizione con il drene. Ognuno di questi si verifica in una delle seguenti casistiche: solo in volo, solo in autobus e misto tra le precedenti.

3.1 Algoritmo solo in volo

In questa circostanza, l'algoritmo **Alg_fly** calcola il tempo necessario per la spedizione se il drone dovesse effettuarla esclusivamente in volo. Lo studio è stato condotto considerando una situazione ideale, in cui il drone parte con la capacità massima e quando la batteria si scarica si dirigere verso la base di ricarica, che idealmente è sottostante alla posizione in cui si trova.

Input:

```
start = 7102 #id della fermata di partenza
end = 305 #id della fermata di arrivo
empty = True #indica se il drone e' vuoto o e' carico con il pacco
cap_battery = 32 #capacita' attuale della batteria in Wh
max_capacity = J_in_Wh(130000) #capacita' massima della batteria in
   Watt-ora(Wh)
con_c = 150 #consumo in volo carico in watt (W)
con_e = 120 #consumo in volo scarico in watt(W)
base_power = 40 #potenza di ricarica della base presente sul autobus
   in watt (W)
speed = 10 # velocita' di volo in cholometri-orari(km/h)
```

```
def Alg_fly(start, end, empty, cap_battery, max_capacity, con_c, con_e
    , base_power, speed):
  fly_time = cal_fly_time(start, end, speed)
  max_time = max_time_fly(cap_battery, empty, con_c, con_e)
  if fly_time>max_time:
    # Calcolo quante volte il tempo di volo sta nel tempo massimo
    num_recharge_cycles = fly_time / max_time
    # Arrotondo per difetto, poiche' non posso effettuare una frazione
    di ricarica
    num_recharge_cycles = math.floor(num_recharge_cycles)
    # Calcolo il tempo di ricarica totale
    total_recharge_time = num_recharge_cycles * cal_recharge_time(
   max_capacity, 1, base_power)
    # Restituisco il tempo di volo + tempo totale di ricarica
    return fly_time + total_recharge_time
  else:
    return fly_time
```

Listing 3.1: Alg_fly

Alg_fly al suo interno ha due variabili temporali importanti:

- **fly_time**: rappresenta il tempo necessario per percorrere la distanza tra i punti di partenza e destinazione. Questo valore è calcolato mediante la funzione di supporto **cal_fly_time**, vista nel Paragrafo 2.3.4, che considera la velocità del drone indicata da **speed**.

- **max_time**: indica il tempo massimo che il drone può volare con la capacità della batteria presente al momento della partenza.

L'if condizionale all'interno dell'algoritmo è progettato per gestire la situazione in cui il tempo necessario per il drone per raggiungere la destinazione in volo (**fly_time**) supera il tempo massimo di volo consentito (**max_time**). In altre parole, se il drone non può coprire l'intera distanza in volo con la sua attuale capacità di batteria, entra in gioco la gestione delle ricariche.

Se la condizione **fly_time** $>$ **max_time** è verificata, significa che il drone non può

volare per tutto il tempo necessario senza esaurire la batteria. A questo punto, l'algoritmo calcola quanti cicli di ricarica sono necessari per coprire l'intera distanza. Questo viene fatto dividendo il tempo di volo previsto per il tempo massimo consentito. L'arrotondamento per difetto di questa divisione rappresenta il numero intero di cicli di ricarica necessari.

Successivamente viene calcolato il tempo totale di ricarica moltiplicando il numero di cicli di ricarica per il tempo di ricarica necessario per ciascun ciclo. Questo tiene conto della capacità massima della batteria e della potenza della base di ricarica.

Output:
il tempo necessario per raggiungere la destinazione in volo:

```
Durata del tragitto: 1 ora, 24 minuti, 00 secondi
```

Il risultato restituito dall'algoritmo è la somma del tempo di volo effettivo e del tempo totale di ricarica. Questo rappresenta il tempo complessivo necessario per coprire la distanza desiderata, tenendo conto della ricarica intermedia. In questo modo, il drone può effettuare una serie di voli e ricariche per affrontare distanze più lunghe rispetto alla sua autonomia di volo singola.

3.2 Algoritmo solo in autobus

In questa seconda situazione ipotetica, l'utilizzo esclusivo dell'autobus da parte del drone potrebbe sembrare un'opzione limitante. Tuttavia, è importante sottolineare che questo algoritmo potrebbe rivelarsi estremamente prezioso in contesti diversi o in caso di malfunzionamenti del drone. In particolare, potrebbe essere impiegato in scenari in cui l'accesso al drone è compromesso o in cui l'utilizzo del drone stesso potrebbe non essere il mezzo più efficiente.

L'implementazione di un algoritmo di questo tipo diventa fondamentale ai fini del-

la ricerca, poiché consente di comprendere a fondo le tempistiche coinvolte nelle varie fasi del processo. Esaminando attentamente il modo in cui l'autobus viene utilizzato come supporto o sostituto al drone, è possibile acquisire dati preziosi sulla durata, l'efficienza e la praticità di tale approccio in diverse circostanze. Inoltre, l'analisi dettagliata delle tempistiche può contribuire a identificare punti critici in cui potrebbe essere necessario ottimizzare o adattare l'algoritmo.

Dal punto di vista pratico, questo algoritmo, è un richiamo alla funzione di Dijkstra che abbiamo scritto (**dijkstra _modified**), ovvero calcola la durata del tragitto. Infatti l'output sarà analogo a quello visto nel paragrafo 2.3.3 nella sezione dedicata Algoritmo di Dijkstra, con l'aggiunta della dicitura:

```
Durata del tragitto: 0 ore, 17 minuti, 20 secondi
```

3.3 Algoritmo misto

La terza circostanza considera una situazione che combina le due circostanze precedenti. Prevede, infatti, che che durante una consegna il drone calcoli quando debba prendere l'autobus e quando invece debba volare. Il codice relativo, denominato **Alg_mix**, si basa su quello di **Alg_fly**, infatti vengono calcolati il tempo necessario per volare dalla posizione di partenza alla destinazione (**fly_time**) e il tempo massimo di volo con la batteria attuale (**max_time**).

```
def Alg_mix(empty, cap_battery, start, end, time, max_capacity,
    base_power, con_c, con_e, tot_time):
     # Calcolo il tempo che servirebbe a raggiungere la destinazione
    volando
     fly_time = cal_fly_time(start, end, speed)
     # Calcolo il tempo massimo di volo con la batteria disponibile
     max_time = max_time_fly(cap_battery, empty, con_c, con_e)

     # Controllo se e' possibile coprire l'intero tragitto in volo
     if fly_time > max_time:
         # Se il volo non e' sufficiente, utilizzo l'autobus
```

```
        single_time, departure_time, last_stop = calculate_bus_time(
time, start, end, max_capacity, cap_battery, base_power)
        # Aggiorno il tempo totale con il tempo trascorso sull'autobus
        tot_time += single_time
        # Se non e' l'ultima fermata, richiamo ricorsivamente Alg_mix
        if last_stop != end:
            Alg_mix(empty, max_capacity, last_stop, end,
departure_time, max_capacity, base_power, con_c, con_e, tot_time)
    else:
        # Se e' possibile coprire l'intero tragitto in volo, aggiorno
il tempo totale con il tempo di volo
        tot_time += fly_time

    return tot_time
```

Listing 3.2: Alg_mix

A cambiare è quello che succede quando non è in grado di percorre il tragitto totalmente in volo, ovvero quando questo

```
if fly_time > max_time:
```

vine soddisfatto.
In presenza di tale scenario, l'algoritmo viene richiamato in modo ricorsivo.

Per semplificare l'illustrazione, esaminiamo cosa accade attraverso un Diagramma di flusso, mostrato in Figura 3.1.
La condizione dell'istruzione "if" è determinata dal primo nodo decisionale che gestisce sia il caso base che quello ricorsivo. Il caso base è diretto: restituisce il tempo totale del tragitto. Nella situazione in cui il tragitto fosse così breve, fin dalla prima chiamata dell'algoritmo, da poter essere percorso interamente in volo, allora il valore restituito è il tempo di volo.
La complessità aumenta leggermente quando la risposta alla domanda è negativa. In tal caso, la funzione **calculate_bus_time**, descritta nel paragrafo 2.3.4 tra le funzioni ausiliarie, viene invocata e mantiene il drone sugli autobus fino a quando la batteria è completamente carica. A quel punto restituisce la fermata in cui il drone dovrebbe scendere insieme al tempo di arrivo associato. Questi valori diventano la fermata di

partenza, con il relativo tempo, che vengono quindi passati come argomenti a Alg_mix.

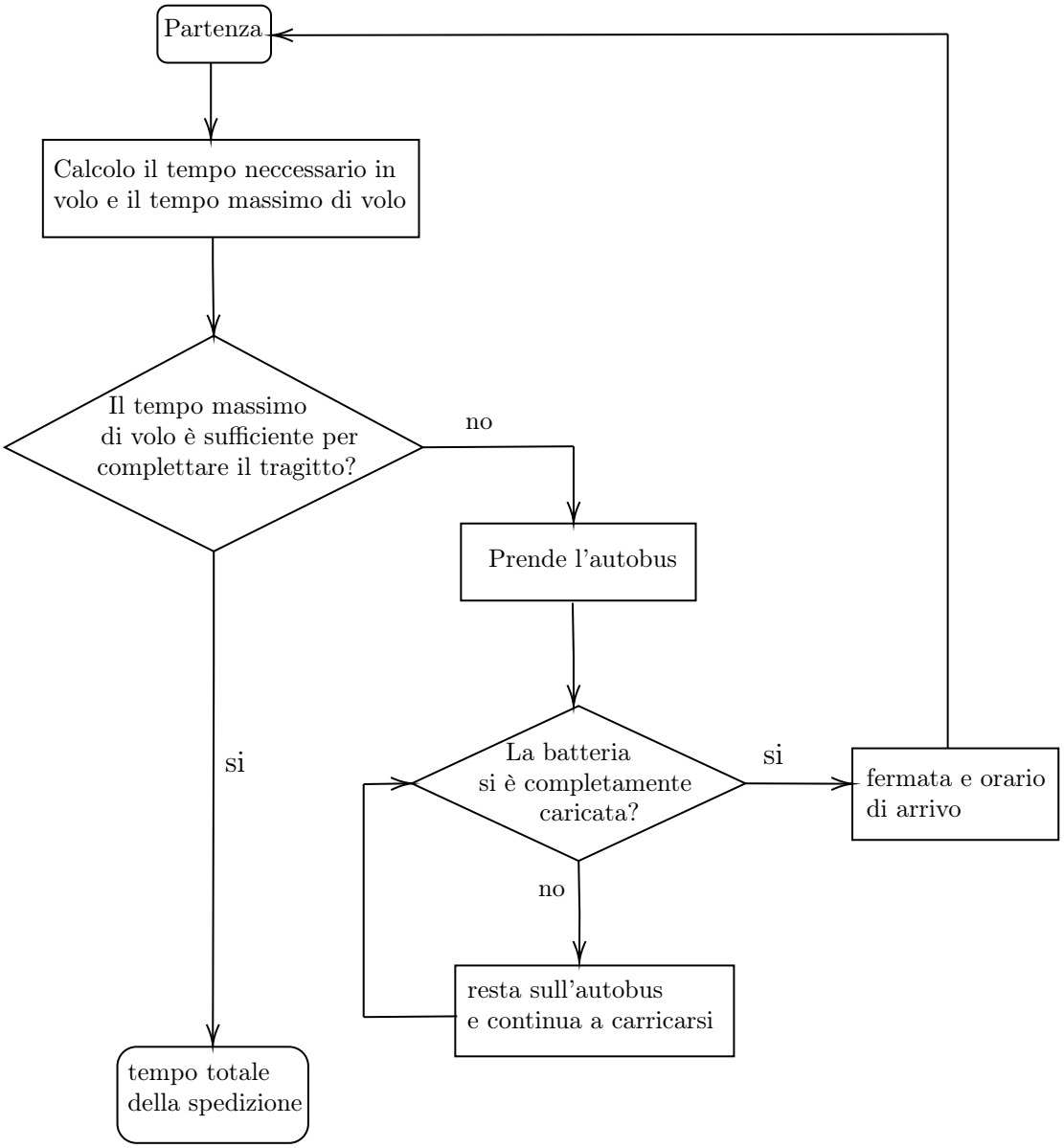

Figura 3.1: Diagramma di flusso del funzionamento dell'Algotitmo Misto

4 | Valutazione

Per valutare in modo più dettagliato le performance degli algoritmi presenti nel capitolo 3, abbiamo preso in esame vari casi studio, concentrandoci su uno dei più significativi: la tratta **7120-9215**. Esaminiamola attentamente: si immagini, che la nostra azienda di consegna pacchi si trovi nelle vicinanze della fermata 7120, conosciuta come *Certosa*, mentre l'abitazione del cliente si trova vicino alla fermata 9215, denominata *Roma*. La data di consegna è fissata per il 17/11/2023, con l'orario previsto di partenza del drone alle 12:30:00.

La distanza aerea tra le due fermate e di 7.29km.
Per il drone utilizzato, si considera una batteria con una capacità massima di 130 kJ, che viaggia a una velocità costante di 10km/h e consuma 120W da carico. Le basi di ricarica presenti a terra e sui mezzi pubblici hanno la stessa potenza, pari a 40W.

Procediamo all'analisi delle tre opzioni disponibili: consegna tramite drone in volo, autobus o un approccio misto. L'obbiettivo è determinare la migliore opzione in termini di tempistiche di consegna.

4.1 Analisi dei algoritmi di ricerca

4.1.1 Analisi di Alg_fly

Nella figura 4.1 possiamo osservare il grafico relativo ai dati raccolti da **Alg_fly**. In primo luogo, si nota un andamento ricorrente, attribuibile all'autonomia limitata del

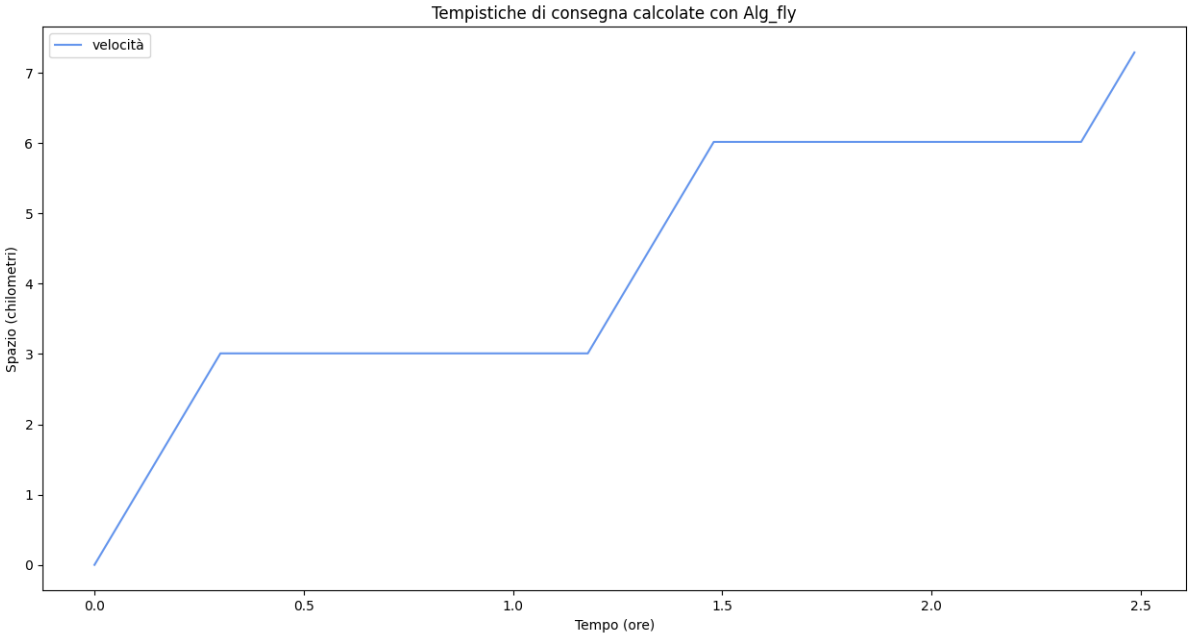

Figura 4.1: Velocità della consegna calcolata da Alg_fly.

drone, che è di 18 minuti. Di conseguenza, il drone può percorrere al massimo 3 km prima di doversi fermare e ricaricarsi. La porzione di grafico che mostra un andamento non crescente indica il momento di ricarica, la quale, in proporzione al tempo di volo, risulta essere molto lunga, durando circa 50 minuti. Considerando che vengono effettuate due soste per la ricarica, che in questo caso equivale al 70% del tempo totale di spedizione.

Si potrebbe migliorare la situazione con l'utilizzo di postazioni più potenti, cercando così di dimezzare il tempo necessario per la ricarica, riducendolo al 54% rispetto al tempo totale di consegna, ben il 16% in meno di prima.

Un'alternativa, più significativa, potrebbe essere la sostituzione della batteria, procedura che, rispetto alla ricarica, ha una tempistica irrilevante arrivando a ridurre quasi a zero i tempi inutilizzati, riducendo la durata totale della consegna a 44 minuti, ovvero 30% in meno della situazione iniziale.

4.1.2 Analisi di Alg_bus

Sull'algoritmo in questione, non ci sono da fare molte considerazioni, se non che la distanza è aumenta, per ragioni note, mentre il tempo totale del percorso è diminuito rispetto alla situazione precedente. Nella figura 4.2 è presente il grafo relativo ai da-

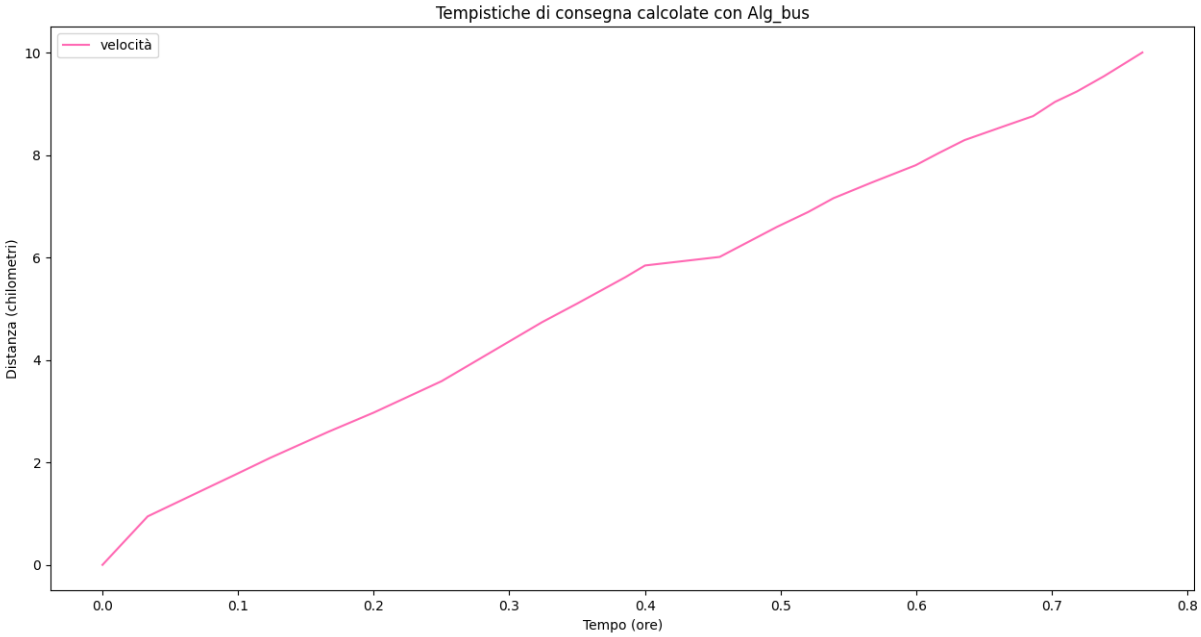

Figura 4.2: Velocità della consegna calcolata da Alg_bus.

ti raccolti da **Alg_bus** con una distanza totale di 10 km e un tempo totale di 46 minuti.

Tuttavia, ciò che riteniamo importante e non emergente dal grafico sono la quantità di cambi necessari e le tempistiche di attesa di essi. Da uno studio accurato dei dati affiora la presenza di due diversi trip_id per questo percorso, indicando quindi solo un cambio.
Per quanto riguarda il tempo di attesa, esso è ridotto a qualche minuto, evidenziato nel grafico in prossimità delle 0,4 ore, dove si osserva un rallentamento rispetto l'andamento consueto.

In una situazione reale, questa riduzione al minimo del tempo di attesa potrebbe risultare rischioso nel caso in qui il primo autobus dovesse fare ritardo nell'arrivare alla fermata di cambio. Attualmente l'algoritmo non gestisce questo tipo di circostanze, ma potrebbe presentare uno spunto per uno sviluppo futuro.

4.1.3 Analisi di Alg_mix

Nella figura 4.3 esaminiamo l'andamento dell'ultimo algoritmo, **Alg_mix**. A primo impatto ci può sembrare simile al precedente, ma in realtà lo è solo nei primi 24 minuti del percorso. Infatti, quando si avvicina alla distanza, abbastanza da poterla effettuare in volo, abbandona l'autobus e percorre autonomamente gli ultimi 3 km.

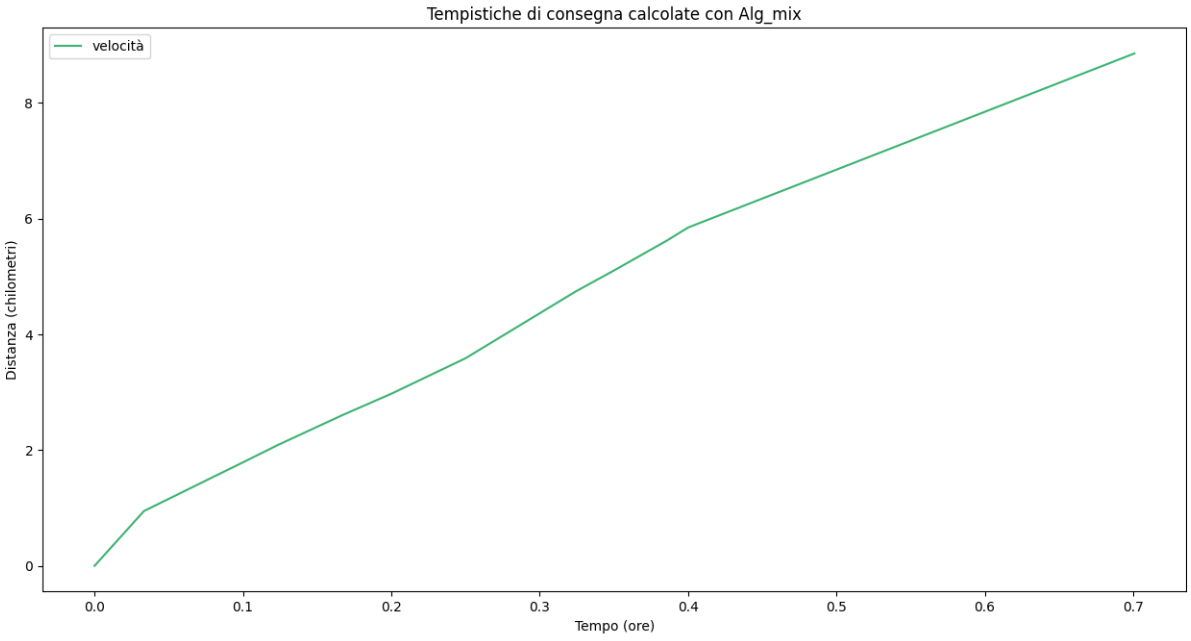

Figura 4.3: Velocità della consegna calcolata da Alg_mix.

4.1.4 Confronto

Possiamo ora confrontare i tre grafici per giungere a una valutazione finale. A prima vista, notiamo che l'utilizzo esclusivo del drone risulta svantaggioso, anche nel caso in cui si raddoppi la potenza di carica.

Resta quindi da esaminare le differenze tra l'approccio misto e quello degli autobus.

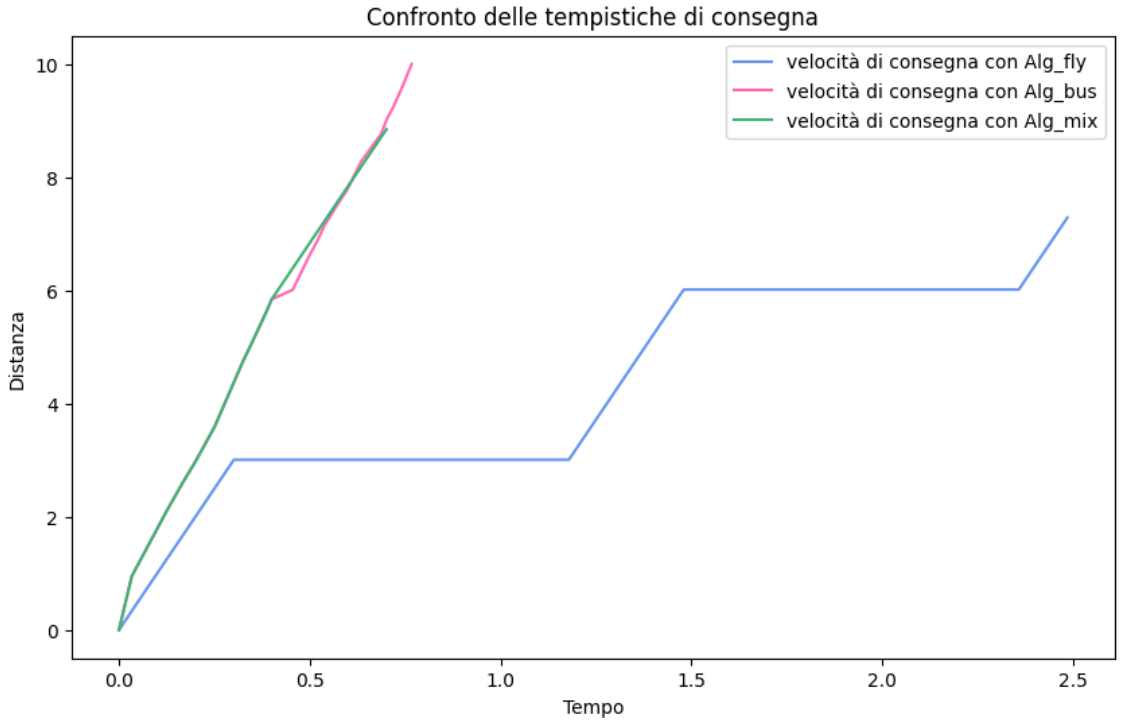

Figura 4.4: Confronto delle velocità di consegna dei tre algoritmi.

In termini di tempistiche, risulta più vantaggioso adottare la modalità mista, anche se di 5 minuti, ma risulta comunque più veloce. La discrepanza diventa più significativa se confrontiamo le distanze. Nonostante l'utilizzo della modalità mista comporti un leggero allungamento rispetto la prima modalità, quella in volo, si dimostra comunque più conveniente.

Un'ultima considerazione da menzionare riguarda la coincidenza del momento di cambio, di modalità di spostamento, nel grafico verde, che possiamo intuire essere nell'ulti-

mo tratto, quando cambia l'inclinazione. Notiamo che ciò si verifica nello stesso punto del cambio autobus nel grafico rosa.

4.2 Analisi dell'orario di partenza

Nel paragrafo precedente, lo studio si è concentrato su un orario di partenza specifico, le 12:30:00, scelto deliberatamente poiché rappresenta un momento di punta in cui gli autobus passano più frequentemente. Nonostante ciò, si è pero deciso di analizzare la stessa situazione ogni cinque minuti, dall'orario di lavorativo, dalle 8:00 fino alle 18:00. Nella figura 4.5 possiamo osservare una media di quello che succede nei orari prescelti.

Anche questa volta si ha la conferma che la modalità che impiega solo il volo non è vantaggiosa. Per quanto riguarda l'opzione, che sfrutta solo l'ausilio dell'autobus, emerge come più efficiente negli orari in cui gli autobus passano molto frequentemente, come alle 13:00 e alle 18:00, soprattutto, osservando i dati, risulta vantaggioso nella prima parte dell'ora. Al contrario non è consigliato sfruttare l'autobus nei orari di prima mattina e di primo pomeriggio.

Nell'ultima figura 4.6 abbiamo la media giornaliera dei valori calcolati nella figura 4.5, che ancora una volta conferma che la modalità mista sia, di poco, più efficiente delle altre.

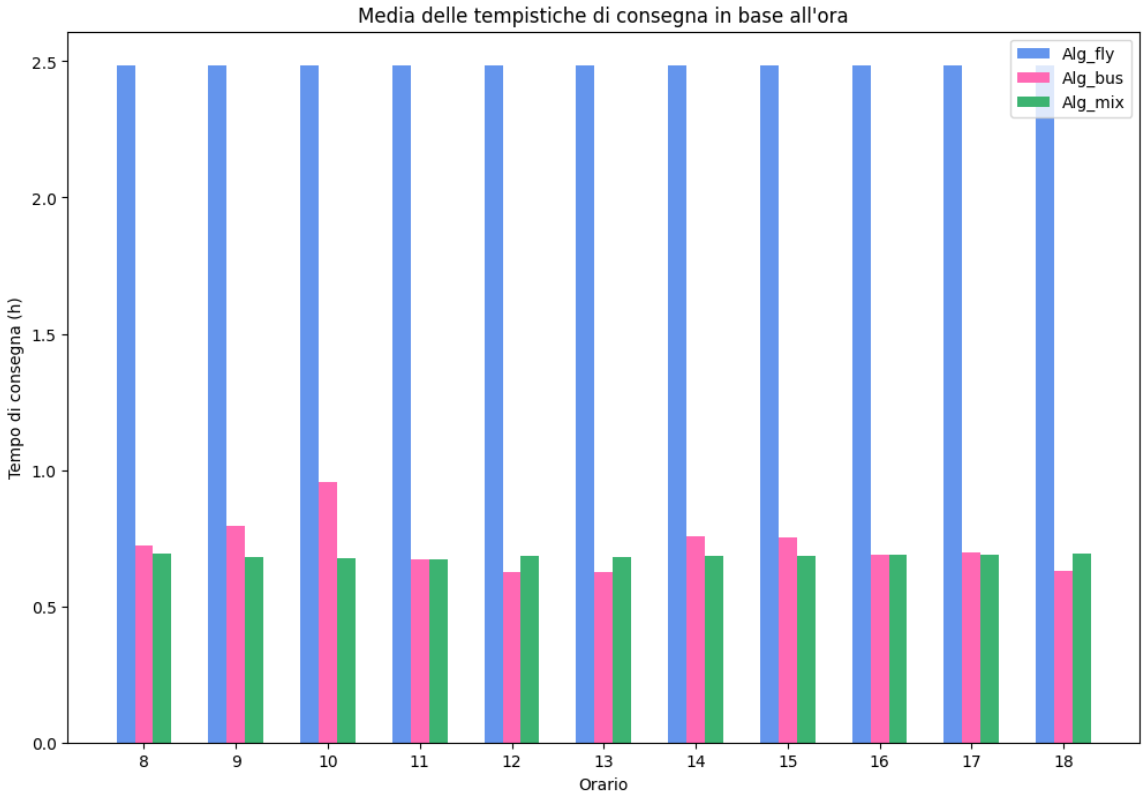

Figura 4.5: Confronto delle velocità di consegna dei tre algoritmi in base all'orario di partenza.

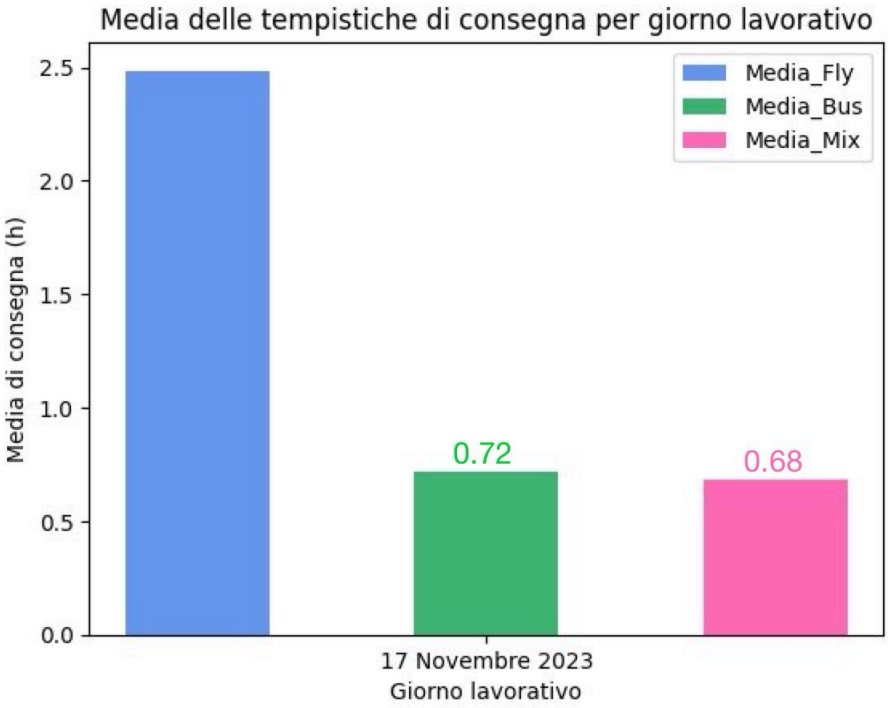

Figura 4.6: Confronto delle velocità di consegna dei tre algoritmi nella giornata del 17 Novembre 2023.

5 | Conclusioni

Come si è visto, nel corso della tesi ci sono numerosi vantaggi e svantaggi nell'utilizzare i droni per la consegna dei pacchi.

5.1 Vantaggi

Dalle nostre ricerche è emerso che fare affidamento solamente sulla modalità di volo dei droni per le spedizioni non risulta vantaggioso, principalmente a causa della scarsa durabilità della batteria. Anche in scenari ideali, come la vasta presenza di basi di ricarica sparse per la città, la situazione non mostra un miglioramento.

Ne è giunto però, che sfruttare gli autobus che si dirigono nella stessa direzione del drone, come basi di ricarica, accorci drasticamente le tempistiche.

Si sono anche visti i motivi per il quale usare i droni come sostituti di veicoli tradizionali sia più vantaggioso. La loro mancanza di emissioni li rende ecologici e sostenibili, contribuendo a ridurre l'impatto ambientale. Infine, l'utilizzo di droni consente di estendere la copertura nelle aree in cui la rete stradale è limitata o assente, ampliando così la portata dei servizi di spedizione.

5.2 Svantaggi

Ciò non toglie che ci siano anche degli svantaggi.
Il più scontato è che i droni sono sensibili alle condizioni meteorologiche avverse, quali

venti forti o condizioni climatiche particolari, che possono compromettere la loro capacità di volo. Ciò impone limiti temporali e ambientali alla loro operatività, riducendo l'affidabilità del sistema in situazioni atmosferiche sfavorevoli.

Un ulteriore inconveniente riguarda la vulnerabilità dei droni a ostacoli presenti nel loro percorso, come la presenza di cavi elettrici, la quale può non essere facilmente individuata dai droni, e la possibilità di collisioni con uccelli o piccioni, che potrebbero volare nelle immediate vicinanze, costituisce un rischio aggiuntivo per l'integrità dei droni.

Inoltre, esistono normative rigide, come quelle relative all'altitudine massima di volo e alle aree proibite, possono limitare la flessibilità e la copertura delle spedizioni con droni, specialmente in contesti urbani densamente regolamentati.

5.3 Studi futuri

Sono individuabili molteplici ambiti di miglioramento al fine di rendere il sistema più adatto alle condizioni reali. In primo luogo, è opportuno considerare l'ottimizzazione dell'esperienza utente, introducendo la capacità di calcolare la localizzazione geografica dei punti di partenza e quello di arrivo. Tale implementazione consentirebbe al drone di consegnare il pacco direttamente al destinatario, superando la limitazione attuale legata alla prossimità delle fermate, migliorando così significativamente l'esperienza dell'utente.

Un passo successivo consisterebbe nell'ottimizzazione dell'algoritmo di *routing*, attualmente basato su Dijkstra modificato, il quale, sebbene efficiente per grafi di dimensioni limitate, si rivela inadeguato per reti di trasporto estese come quella di Bologna. Si propone l'adozione di un algoritmo più performante, come A* (A-Star)[1], in grado di gestire con maggiore efficienza contesti di grandi dimensioni.

[1] è un algoritmo di ricerca su grafi come quello di Dijkstra, ma con la differenza che utilizza un stima euristica per selezionare il nodo da visitare.

Tuttavia, anche con un algoritmo ottimizzato, sussiste il rischio di ritardi, o addirittura, di mancata presenza degli autobus, eventi attualmente non contemplati dagli algoritmi studiati. Si suggerisce, quindi, l'implementazione di un meccanismo di ricalcolo del percorso nel caso in cui, entro un intervallo di tempo prestabilito, il mezzo di trasporto assegnato non risulti disponibile.

Un ulteriore miglioramento potrebbe derivare dall'evoluzione dell'**Alg_fly**, per consentire la gestione simultanea di più droni. Si ipotizza la distribuzione capillare di droni sull'area di copertura, permettendo uno scambio di pacchi tra droni in movimento, prima che questi si scarichino e si fermino a ricaricarsi, minimizzando così i tempi morti associati alla ricarica.

Una caratteristica essenziale da implementare sarebbe la considerazione delle condizioni meteorologiche locali. L'analisi del meteo potrebbe influenzare la decisione riguardante la scelta del percorso o addirittura la modalità di spedizione, permettendo di adattarsi a situazioni climatiche avverse.

Infine, si potrebbe integrare l'intelligenza artificiale per ottimizzare la selezione dei percorsi e delle modalità di spedizione in modo dinamico, tenendo conto di fattori come il traffico, e integrare tutte le migliorie sopra elencate. Questa implementazione consentirebbe al sistema di adattarsi in tempo reale alle variazioni delle condizioni operative.

Glossario e acronimi

Glossario

c consumo. 22

cap capacità. 23

capAttuale capacità attuale della batteria. 23

capMassima capacità massima della batteria. 23

d distanza. 22

e energia. 22

p potenza della base di ricarica. 23

t tempo. 22, 23

v velocità. 22

Acronimi

APR aeromobili a pilotaggio remoto. 3

UAV unmanned aerial vehicle. 3

Bibliografia

[1] Wikipedia. *Aeromobile a pilotaggio remoto*. [Online; consultato il 22-01-2024]. 2023. URL: https://it.wikipedia.org/wiki/Aeromobile_a_pilotaggio_remoto.

[2] Giulio Attenni et al. "Drone-Based Delivery Systems: A Survey on Route Planning". In: *IEEE Access* 11 (2023), pp. 123476–123504. DOI: 10.1109/ACCESS.2023.3329195.

[3] J.E. Scott e C.H. Scott. "Drone Delivery Models for Healthcare". In: *Proceedings of the 50th Hawaii International Conference on System Sciences*. Hilton Waikoloa Village, HI, apr. 2017, pp. 3297–3304. DOI: 10.24251/HICSS.2017.399. URL: https://doi.org/10.24251/HICSS.2017.399.

[4] Eitan Frachtenberg. "Practical Drone Delivery". In: *Computer* 52.12 (2019), pp. 53–57. DOI: 10.1109/MC.2019.2942290.

[5] Przemyslaw Kornatowski et al. "Downside Up:Rethinking Parcel Position for Aerial Delivery". In: *IEEE Robotics and Automation Letters* PP (mag. 2020), pp. 1–1. DOI: 10.1109/LRA.2020.2993768.

[6] Przemyslaw Mariusz Kornatowski et al. "A Morphing Cargo Drone for Safe Flight in Proximity of Humans". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4233–4240. DOI: 10.1109/LRA.2020.2993757.

[7] Longhao Qian, Silas Graham e Hugh H.-T. Liu. "Guidance and Control Law Design for a Slung Payload in Autonomous Landing: A Drone Delivery Case

Study". In: *IEEE/ASME Transactions on Mechatronics* 25.4 (2020), pp. 1773–1782. DOI: 10.1109/TMECH.2020.2998718.

[8] Babar Shahzaad et al. "Resilient composition of drone services for delivery". In: *Future Generation Computer Systems* 115 (2021), pp. 335–350. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0167739X19331735.

[9] Wikipedia. *Algoritmo di Dijkstra*. [Online; consultato 18-11-2023]. 2023. URL: https://it.wikipedia.org/wiki/Algoritmo_di_Dijkstra.

[10] Wikipedia. *Algoritmo di Bellman-Ford*. [Online; consultato il 18-11-2023]. 2023. URL: https://it.wikipedia.org/wiki/Algoritmo_di_Bellman-Ford.

Ringraziamenti

Innanzitutto, vorrei ringraziare Angelo Trotta per aver creduto in me e avermi supportata.
Vorrei esprimere la mia gratitudine a quelli di Hyperting per avermi dato la possibilità di vedere una realtà diversa da quella universitaria.

Grazie di cuore a mamma e papà, che in questi anni sono stati i miei fan numero uno.
Un ringraziamento speciale va ad Alex e Melissa per tutto l'amore che mi hanno dimostrato e per la pazienza che hanno avuto con me. Grazie ai miei parenti che, pur essendo lontani, mi hanno sempre sostenuto in tutto quello che facevo.
Un sentito ringraziamento ad Ade, che non mi ha lasciata solo un attimo negli ultimi 10 anni.
Un grazie particolare a Luca, che mi è sempre stato vicino e mi ha aiutato a crescere; senza di te, probabilmente avrei mollato al primo anno.
Ringrazio Chiara che pur lontana mi è sempre stata vicina, in qualsiasi situazione.
Grazie a Federica per avermi offerto più volte una spalla su cui piangere e per avermi fatto sentire il suo supporto con solo uno sguardo.

Desidero ringraziare Apo, che con il suo spirito salentino ha saputo rallegrare le mie giornate e, con la sua premura, ha impedito che commettessi tanti errori.
Grazie a Erik, che con un "ora lo risolviamo" mi risolevava le giornate. Grazie a Emanuele che nell'ultimo periodo mi ha insegnato tanto.
Un sentito ringraziamento a Emi, che nonostante ci vedessimo una volta al mese, è

sempre stato presente; ti prego non portarmi più al Hemingway.
Grazie ad Alejandro per tutti le notti passate a studiare insieme e per avermi insegnato un vasto vocabolario.
Grazie a Tia, che non mi ha mai lasciato abbandonata a me stessa.
Grazie a Gaia, senza la sua disponibilità mi sarei persa fin dal primo giorno.
Grazie a Pa per avermi insegnato a essere estroversa; senza di te, non conoscerei metà delle persone di questa lista.
Grazie a Foxy per avermi mostrato che tutto è possibile, con le 500 ore passate in chiamata per spiegarmi programmazione.
Grazie a Dan per tutte le serate passate a parlare della vita.
Grazie a Mehala per insegnarmi tanto ogni volta che ti vedo.
Grazie a Michi, Renato, Roberta e Teresa per aver reso il lavoro un posto migliore.

Grazie a coloro che ci sono sempre stati, Guys del Ranzani, abbiamo passato dei momenti bellissimi che conservo con gelosia nel mio cuoricino.
Un sentito ringraziamento ai ragazzi del bunker che rallegrate il mio presente.
Infine, vorrei ringraziare coloro che rimarranno nella mia vita nonostante tutto.