Agenzia nazionale per le nuove tecnologie,
l'energia e lo sviluppo economico sostenibile

Ministero della Transizione Ecologica

Ricerca di Sistema elettrico

Manuale d'uso del software

G. Fusco, C. Colombaroni, F. Carrese, N. Isaenko

Dipartimento di Ingegneria
Civile Edile e Ambientale

Sapienza
Università di Roma

Report RdS/PTR(2020)/062

MANUALE D'USO DEL SOFTWARE

G. Fusco, C. Colombaroni, F. Carrese, N. Isaenko

Aprile 2021

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA
Piano Triennale di Realizzazione 2019-2021 - II annualità
Obiettivo: Tecnologie
Progetto: Tecnologie per la penetrazione efficiente del vettore elettrico negli usi finali
Work package: Mobilità
Linea di attività: 2.23 - Sviluppo Informatico del modulo di ottimizzazione dell'e-last mile
Responsabile del Progetto: Claudia Meloni, ENEA
Responsabile del Work package: Maria Pia Valentini, ENEA

Il presente documento descrive le attività di ricerca svolte all'interno dell'Accordo di collaborazione "*Sviluppo di algoritmi e di un sistema informatico di ottimizzazione della distribuzione urbana delle merci con veicoli elettrici.*"
Responsabile scientifico ENEA: Ing. Maria Pia Valentini
Responsabile scientifico DICEA Sapienza Università di Roma: Prof. Gaetano Fusco

Indice

Sommario

Il presente rapporto tecnico costituisce il manuale d'uso del software per la pianificazione e la gestione della distribuzione urbana delle merci con veicoli elettrici. Il software è costituito di due componenti:

- Un modulo di pianificazione, che dati in input gli ordini di consegna, ne determina l'ordine ottimale, organizza le consegne in giri e assegna i giri ai veicoli;
- Un modulo di simulazione, che simula il funzionamento in real-time del sistema e consente di verificarne la capacità d'intervento in caso di imprevisti, quindi la necessità di una ricarica imprevista (RECOVERY) o di aggiornamento dei giri in caso di imprevista congestione stradale;

Il rapporto, oltre a definire il funzionamento e l'architettura complessiva del *software* di ottimizzazione del processo distributivo delle merci con veicoli elettrici dotati di sistema di monitoraggio, fornisce la specificazione di tutti gli input necessari, la codifica dei vincoli tecnici ed operativi e la specificazione del tipo e del formato degli output prodotti.

Nel capitolo 1 vengono definite le generalità del software, gli obiettivi, i vincoli del problema e gli strumenti teorici utilizzati.

Nel capitolo 2 vengono esposti nel dettaglio gli input necessari al funzionamento dell'algoritmo, riportando il formato in cui essi vengono espressi.

Il capitolo 3 illustra il funzionamento della pianificazione, oltre alla logica dell'algoritmo di ottimizzazione, il simulated annealing, vengono descritte anche le procedure aggiuntive come la lettura dei dati, il calcolo delle matrici dei tempi di spostamento e le funzioni di verifica e di aggiornamento.

Nel capitolo 4 viene affrontata la simulazione, specificando come vengono calcolate le posizioni e i consumi dei singoli veicoli, per permettere al modulo del recovery di intervenire in caso di necessità. Il recovery si occupa anche di rilevare eventuali anomalie e permette il ricalcolo dell'itinerario affinché un veicolo riesca a terminare le dovute consegne con l'ausilio di una ricarica intermedia.

1 Architettura del Software di ottimizzazione della gestione operativa della distribuzione delle merci

Il software ha il compito di definire, anche in tempo reale, la migliore soluzione operativa delle attività di distribuzione e consegna, in relazione agli arrivi della merce da consegnare, alla disponibilità di veicoli e personale, nonché alla possibile opzione di avere una destinazione della consegna ancora non nota al momento della presa in carico. Il software risolve un Vehicle Routing Problem, un classico problema di ricerca operativa che determina i percorsi di un dato numero di veicoli per servire un dato insieme di clienti partendo da un deposito centrale. Questo tipo di problemi presenta sempre molti vincoli aggiuntivi, dipendenti dalla situazione specifica, che complicano la struttura del problema.

Nel caso analizzato vi sono:

- vincoli temporali, cioè le finestre orarie in cui si deve svolgere ciascuna consegna (quindi la durata massima del servizio), i tempi di scarico e carico merci dai veicoli, le velocità percorribili nelle strade del centro urbano;
- vincoli spaziali, correlati alla posizione del deposito e dei punti di consegna;
- vincoli legati ai veicoli elettrici utilizzati per la distribuzione, ossia le caratteristiche tecniche dei veicoli (capacità, tara, massima carica), che sono condizionati dalla presenza delle infrastrutture di ricarica lungo i percorsi per la loro autonomia (un altro vincolo spaziale approfondito più avanti nei paragrafi 3.4.1 e 4.5).

Per l'ottimizzazione della sequenza di distribuzione dei veicoli viene utilizzato un algoritmo di Simulated Annealing, che punta ad ottenere un costo generale ottimale (non obbligatoriamente il minore in assoluto, ma il minore trovato rispettando tutti i vincoli), in termini di tempo, distanza percorsa, mezzi impiegati, lavoratori impiegati o numero di rifornimenti effettuati.

Questo algoritmo, tramite la ricerca locale, partendo da un itinerario iniziale sceglie l'itinerario successivo casualmente (esplorazione random) tracciando un percorso che scarta le opzioni peggiori (itinerari con

costo maggiore, in termini di distanza dal deposito e tempo) ponderandole allo scarto del peggioramento in funzione di un parametro (temperatura) che decresce a ogni iterazione (da qui il collegamento alla ricottura del vetro). L'analisi delle scelte peggiori (ovvero dei percorsi più "costosi" per tempo e/o distanza) non viene esclusa a priori, ma le scelte peggiori hanno una probabilità di accettazione inizialmente alta e decrescente con lo scorrere del tempo, così da evitare massimi e minimi locali per avere percorsi che tendono ad una distanza minima ed un tempo minimo generale.
Il software è strutturato in due parti:

- Pianificazione: esegue l'ottimizzazione con simulated annealing e determina i giri per i veicoli della flotta indicando anche i clienti a cui ogni veicolo deve consegnare, in quale ordine e in che momento.
- Simulazione: esegue l'ottimizzazione con simulated annealing e determina i giri per i veicoli della flotta indicando anche i clienti a cui ogni veicolo deve consegnare, in quale ordine e in che momento. Visualizza la posizione dei veicoli per ogni step temporale della simulazione, calcola i consumi energetici ed è utilizzata per simulare eventi di update o di recovery. Mediante la simulazione è possibile verificare le conseguenze di imprevisti e valutare le corrispondenti soluzioni di recovery.

2 Dati di input

In questo capitolo vengono definiti e spiegati i dati necessari all'implementazione del software, nonché i parametri principali a scelta dell'utilizzatore.

2.1 Punti di consegna

I Dati riguardanti i punti di consegna (DP) sono un input del problema e sono strutturati in un file ".csv" che contiene l'ID di ciascun punto (Delivery Point ID), le coordinate geografiche dei punti di consegna (Delivery Point latitude and longitude), la quantità di merce richiesta da ognuno di loro in termini di peso e volume (Demand kg e Demand m^3), i tempi di consegna concessi per ciascuno di questi, specificati in numero di minuti dalla mezzanotte (Time Window Start and End), la data in cui viene effettuata la consegna del tipo YYYYMMDD (Date).
Il campo Delivery Point ID permette al software di posizionare i punti di consegna sul grafo. I valori definiti in questa colonna sono quindi relativi al nodo del grafo più prossimo a ciascun punto di consegna, ed è necessario che corrispondano agli identificativi dei nodi definiti nel paragrafo 2.2.
Nella tabella la prima riga indica la posizione del deposito, che avrà solo coordinate geografiche e data, mentre gli altri dati sono posti uguali a 0.

Tabella 1: tabella nel file DeliveryPoints.csv

Delivery Point ID	Delivery Point Longitude	Delivery Point Latitude	Demand_kg	Demand_m3	Time Window Start [minuti dalle 00:00]	Time Window End [minuti dalle 00:00]	Date [YYYYMMDD]
13800204878731	12.4912621	41.8995853	0	0	0	0	20210325
13800205536726	12.4425082	41.8767284	0.14	1.6	480	1080	20210325

2.2 Grafo

La rete stradale è rappresentata mediante un grafo direzionato definito dai seguenti file csv.

- Nodes.csv, con coordinate ed identificativi dei nodi del grafo. I numeri degli ID per efficienza informatica sono sostituiti da codici progressivi ID_new; dunque StartNodes ed EndNodes degli archi si riferiscono all'ID_new dei nodi.

Tabella 2: tabella nel file Nodes.csv

ID_node	xcoord	ycoord	ID_new
13800204605518.0	0.051	42.71	1
13800204627070.0	0.074	61.45	2
13800204619516.0	0.051	42.71	3

- Links.csv, contenente le velocità (variabile (Speed) di percorrenza e la lunghezza degli archi. La variabile Speed è una variabile dinamica: l'utente specifica la tabella csv con valori standard (velocità media d'arco in ciascuna fascia oraria). I valori possono poi essere aggiornati dinamicamente acquisendo dati on-line forniti da fonti esterne.

Tabella 3: tabella nel file Links.csv

StartNodes	EndNodes	Length	ID	Speed (00:00-00:59)	Speed (01:00-01:59)	Speed (… : … - … : …)	Speed (23:00-23:59)
1	2	42.71	13800054849460	50	50	…	50
1	76	61.45	13800053796191	30	30	…	30
1	2	42.71	13800054849460	50	50	…	50

2.3 *Infrastrutture di Ricarica*

Il file "colonnine.csv" contiene al suo interno una tabella con ID e coordinate delle colonnine di ricarica dei veicoli elettrici, la potenza di ricarica, e la colonna "availability" che ne indica la disponibilità (specificata come numero di stalli disponibili, valore aggiornabile dinamicamente acquisendo dati online da fonte esterna).

Tabella 4: tabella nel file ChargingStations.csv

Charging Station ID	Charging Station Longitude	Charging Station Latitude	Recharging Power [kW]	Availability [numero naturale]
13800205541264	12.324249	41.80673	25	
13800205504273	12.2801525	41.7309823	25	

2.4 *Flotta*

Il file "fleet.csv" contiene le informazioni riguardanti le caratteristiche tecniche dei veicoli utilizzati (ID, tipo di veicolo, tara del veicolo, massima carica della batteria, capacità del veicolo in chilogrammi e metri cubi, tempi di carico e scarico merci). Queste informazioni possono essere modificate dall'utente a seconda del numero e tipo di veicoli che si vuole utilizzare.

Tabella 5: tabella nel file Fleet.csv

Vehicle ID	Vehicle Type	Empty Vehicle Weight [kg]	maxCharge [kWh]	Vehicle Load Cap [kg]	Vehicle Volume Capacity [m3]	VehicleLoadTime [min]	DeliveryUnloadTime [min]
1	Midi	3000	250	1500	20	30	5
2	Mini	2000	150	1000	15	25	5

I parametri per il calcolo dei consumi sono contenuti nel file "consumptionParameters.csv", e sono specificati per tipo di veicolo (vedi Tabella 5) e per tre intervalli di percentuale di carico.

Tabella 6: tabella nel file consumptionParameters.csv

Veicle type	Load %	a	b	C	d	e
Midi	**Load < 30%**	486.6406	0.080972	1196063	1.629846	134.0512
Midi	**30% < Load < 70%**	474.497	0.07433	10230354	1.9942	146.568
Midi	**Load >70%**	503.41	-0.07251	160.5909	0	0
Large	**Load < 30%**	1.014595	-0.04122	187.4573	0.077275	75.34241
Large	**30% < Load < 70%**	0.001	2.55885	438	-0.4216	0
Large	**Load >70%**	0.003614	0.000219	-0.0000019	0	0

2.5 Parametri dell'algoritmo di ottimizzazione

I parametri necessari per l'algoritmo di ottimizzazione sono definiti in due file csv, uno relativo ai parametri dell'algoritmo di simulated annealing e un secondo per i parametri della funzione obiettivo. Per quanto riguarda i primi, vengono forniti come default nel file "annealingParameters.csv", ma si consente anche all'utente di modificarli, in quanto sono il risultato di una calibrazione rispetto a un particolare caso di test analizzato:

- Temperatura iniziale (T0)
- Temperatura finale (Tend)
- Numero di iterazioni interne (L)
- Raffreddamento (q)

Tabella 7: tabella del file annealingParameters.csv.

T0	Tend	L	Q
10^5	10^(-4)	200	0.9

La temperatura è un parametro che determina la probabilità con cui può essere accettata una soluzione peggiore di quella corrente, il numero di iterazioni interne è il numero di iterazione che viene effettuato per ogni valore di temperatura, mentre il Raffreddamento indica la percentuale di cui dovrà diminuire la temperatura iniziale affinché possa fermarsi il processo avendo raggiunto la temperatura finale. Il numero complessivo di iterazioni esterne deriva quindi dai parametri T0, Tend e q ed è dato dalla seguente equazione:

$$IT = \frac{\log(\frac{Tend}{T0})}{\log(q)}$$

I parametri della funzione obiettivo sono contenuti nel file "ParametersOF.csv". Rappresentano i pesi da attribuire alle variabili che vanno a comporre il costo della soluzione e sono i seguenti:

- Costo di esercizio del veicolo per km percorso
- Costo operativo orario (aggiuntivo rispetto a quello chilometrico)
- Costo aggiuntivo per ciascun veicolo utilizzato (ammortamento e costo conducente).

Tabella 8: tabella del file ParametersOF.csv.

km_cost	h_cost	Veh_cost
1	1	100

2.6 Parametri del modulo di Simulazione

Il file "simulation.csv" contiene una tabella con i parametri necessari all'implementazione della simulazione, ossia data, tempo di inizio e fine simulazione in minuti e il passo della simulazione (timeStep).

Tabella 9: tabella in simulationParameters.csv

Date [aaaa/mm/gg]	TimeStep [min]	simulationStart [min]	simulationEnd [min]
20210325	5	480	1080

3 Funzionamento del modulo di Pianificazione

La parte di codice riguardante la pianificazione permette di ottenere un'ottimizzazione del percorso che i veicoli dovranno eseguire per le consegne, indicando per ciascuno quali siano i clienti da servire, in quale ordine e in che orario. La versione pre-prototipale del software è sviluppata in MATLAB e si articola in diverse subroutine, corrispondenti a diversi file ".m". La pianificazione viene sviluppata nel file "main.m", che unisce tutti i codici permettendo un'unica esecuzione su MATLAB. Questo Codice richiama le funzioni che contengono gli input: "readData", "ShortestpathTA". Successivamente richiama la funzione "vrp.m" che avvia l'algoritmo di ottimizzazione.

Di seguito vengono descritti i file matlab che costituiscono il codice di pianificazione, i dati necessari per implementarli e i risultati che restituiscono. La Figura 1 illustra il diagramma di flusso che attua le seguenti operazioni:

- Vengono letti i file di input (coordinate di punti di consegna e infrastrutture di ricarica, caratteristiche della flotta, grafo) con la funzione "readData";
- Si procede al calcolo delle matrici delle distanze con "ShortestPathTA";
- Si implementa il Simulated Annealing con "vrp";
- Si ottengono le rotte;
- Viene svolta la verifica del rispetto dei vincoli con "CheckThreshold";
- Viene svolta la verifica delle avvenute consegne con "CheckTimestamp".
- Si verifica infine la presenza di eventuali aggiornamenti, in caso affermativo si ripeterà lo schema.

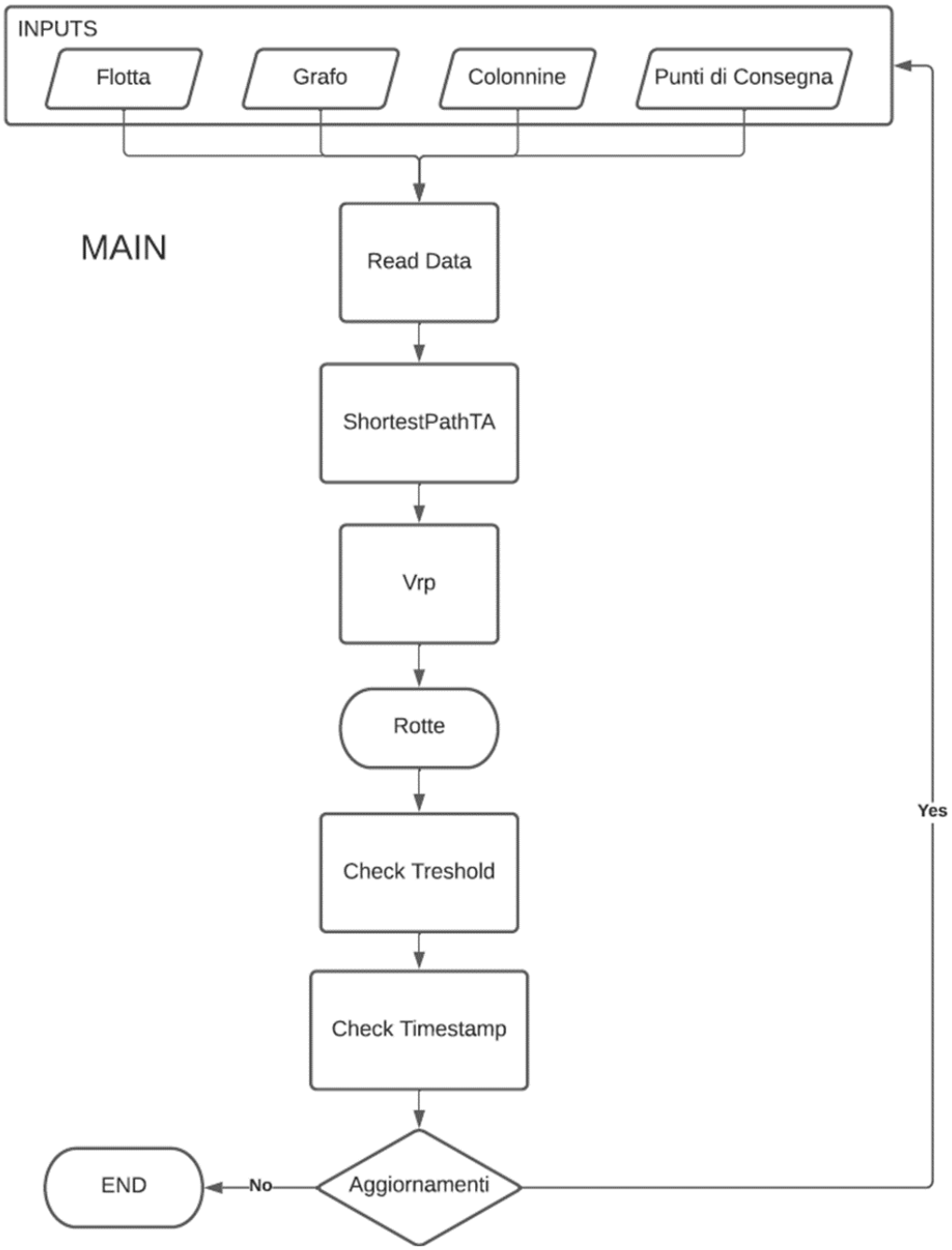

Figura 1: struttura della pianificazione e funzioni chiamate.

3.1 Lettura e Pre-Processamento dati

Per poter essere efficacemente processati dal software, i dati di input illustrati nel paragrafo precedente sono importati tramite la routine “readData.m”. Le informazioni nei file .csv vengono lette dalla funzione “readtable” e riformattate in vettori, matrici e tabelle. Le tabelle Links e Node vengono qui processate in moda da creare il grafo direzionato MATLAB

```
function [Demand,TimeWindow,DP,CS,Fleet,simPar,G] = readData()

%% Download Data
DP = readtable('../test_data/DeliveryPoints.csv');  % delivery points data
% DM = readtable('../test_data/DistanceMatrix.csv');     % distance matrix
% TT = readtable('../test_data/TravelTimes.csv');   % travel time matrix

CS = readtable('../test_data/ChargingStations.csv');     % Charging stations XY coordinates
% CSD = readtable('../test_data/ChargingStationsDistances.csv');     % Distance Matrix between delivery points and charging stations
% CSTT = readtable('../test_data/ChargingStationsTravelTimes.csv');     % Distance Matrix between delivery points and charging stations

Fleet = readtable('../test_data/Fleet.csv','HeaderLines',1);    %Electric Vehicle Fleet

simPar = readtable('../test_data/Simulation.csv','HeaderLines',1);    %Simulation Parameters

Links = readtable('../test_data/Links.csv');        % Links of Graph
Nodes = readtable('../test_data/Nodes.csv');        % Nodes of Graph
Links.EndNodes=[Links.StartNodes,Links.EndNodes];   %Joins Columns
Links.StartNodes=[];                                %Removes added column
G = digraph(Links,Nodes);                           %Creates Graph
```

Figura 2: estratto del codice del file "readData.m". Importare dati.

La routine “readData” procede a individuare ed eliminare eventuali duplicati all’interno dei dati caricati

```
%% Check, print and remove duplicates
[~, i , ~] = unique(DP.DeliveryPointID,'first');
indexToDupes = not(ismember(1:numel(DP.DeliveryPointID),i));
if sum(indexToDupes)>=1
    doubleDeliveryNodes=DP.DeliveryPointID(indexToDupes);
    DP(indexToDupes,:)=[];
    disp('More than one customer assigned to same node')
    for j=1:numel(doubleDeliveryNodes)
        fprintf('Double Node ID: %s \n',num2str(doubleDeliveryNodes(j)))
    end
end

[~, i , ~] = unique(CS.ChargingStationID,'first');
indexToDupes = not(ismember(1:numel(CS.ChargingStationID),i));
if sum(indexToDupes)>=1
    doubleCSNodes=CS.ChargingStationID(indexToDupes);
    disp('More than one Charging Station assigned to same node')
    CS(indexToDupes,:)=[];
    for j=1:numel(doubleCSNodes)
        fprintf('Double Node ID: %s \n',num2str(doubleCSNodes(j)))
    end
end
```

Figura 3: estratto del codice del file "readData.m". Rimozione duplicati.

I file di input necessari per l’esecuzione del file “readData” sono:

- DeliveryPoints.csv;
- ChargingStations.csv;
- Fleet.csv;
- consumptionParameters.csv;
- Nodes.csv;
- Links.csv:
- annealingParameters.csv;
- parametersFO.csv;
- Simulation.csv.

3.2 Calcolo delle matrici dei tempi di percorrenza

Date le coordinate dei punti di consegna (DP) e delle infrastrutture di ricarica (CS) la funzione "ShortestPathTA" procede a calcolare i cammini minimi tra queste, fornendo in output due matrici di tempi e di distanze origine-destinazione.

Gli *shortest path* vengono calcolati sulla base dei tempi di percorrenza degli archi forniti con il grafo. Si ottengono 4 strutture dati:

- Shortest path tra DP (nodi attraversati nel percorso)
- Shortest path tra DP (archi attraversati nel percorso)
- Shortest path tra DP e Charging Stations (nodi attraversati nel percorso)
- Shortest path tra DP e Charging Stations (archi attraversati nel percorso).

Queste strutture dati hanno la stessa forma delle matrici delle distanze e dei tempi di percorrenza esposte precedentemente, con la sola differenza che ogni cella della matrice, invece di contenere un unico valore contiene la lista dei nodi o degli archi necessari per congiungere ciascuna coppia di punti sul cammino minimo. Quindi l'elemento (i,j) della matrice contiene lo shortest path per andare dal punto i al punto j.

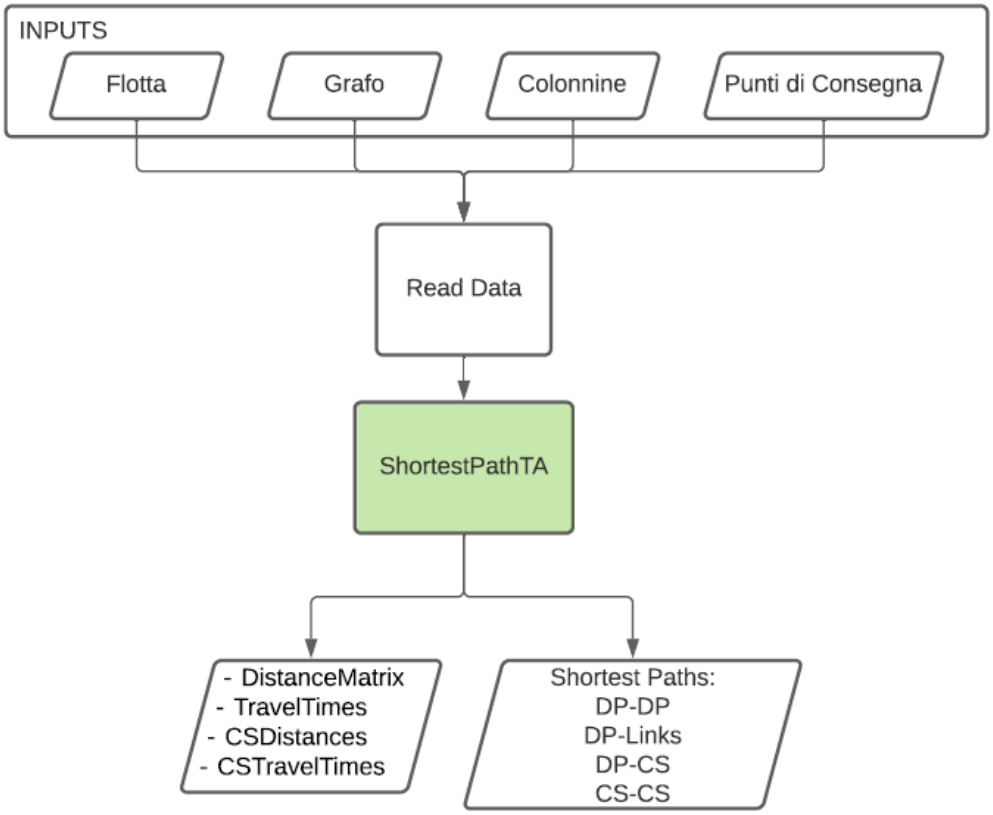

Figura 4: Output della funzione ShortestPathTA. L'Input della funzione sono le coordinate e il Grafo costruito in ReadData.

3.3 Funzionamento del modulo di Ottimizzazione

A seguito del processamento preliminare dei dati viene avviata automaticamente l'ottimizzazione, che ricerca la soluzione con migliore valore della funzione obiettivo, che consenta di avere tutte le consegne terminate col minor dispendio di energia possibile (minor numero di veicoli e operatori possibile e percorso più breve e rapido possibile).

Il file "main.m" richiama la funzione "vrp" in cui, tramite l'algoritmo di simulated annealing, viene ricercato l'itinerario dei giri più efficiente. Il processo di ottimizzazione avviene nel seguente modo:

- Si crea una soluzione che rispetti tutti i vincoli desiderati;
- Si perturba la soluzione;
- Si valuta il costo della nuova soluzione e della soluzione precedente;

- Si sceglie la soluzione da mantenere confrontando la nuova e la vecchia soluzione;
- Scelta la soluzione da mantenere si ripete il procedimento dalla perturbazione della soluzione per L volte;
- Si diminuisce la temperatura dell'algoritmo e si sceglie la soluzione migliore delle L trovate da cui ripartire nel ciclo successivo;
- Se la temperatura a fine processo è ancora superiore alla temperatura finale si ripete il processo da capo, altrimenti la soluzione scelta è il risultato dell'ottimizzazione.

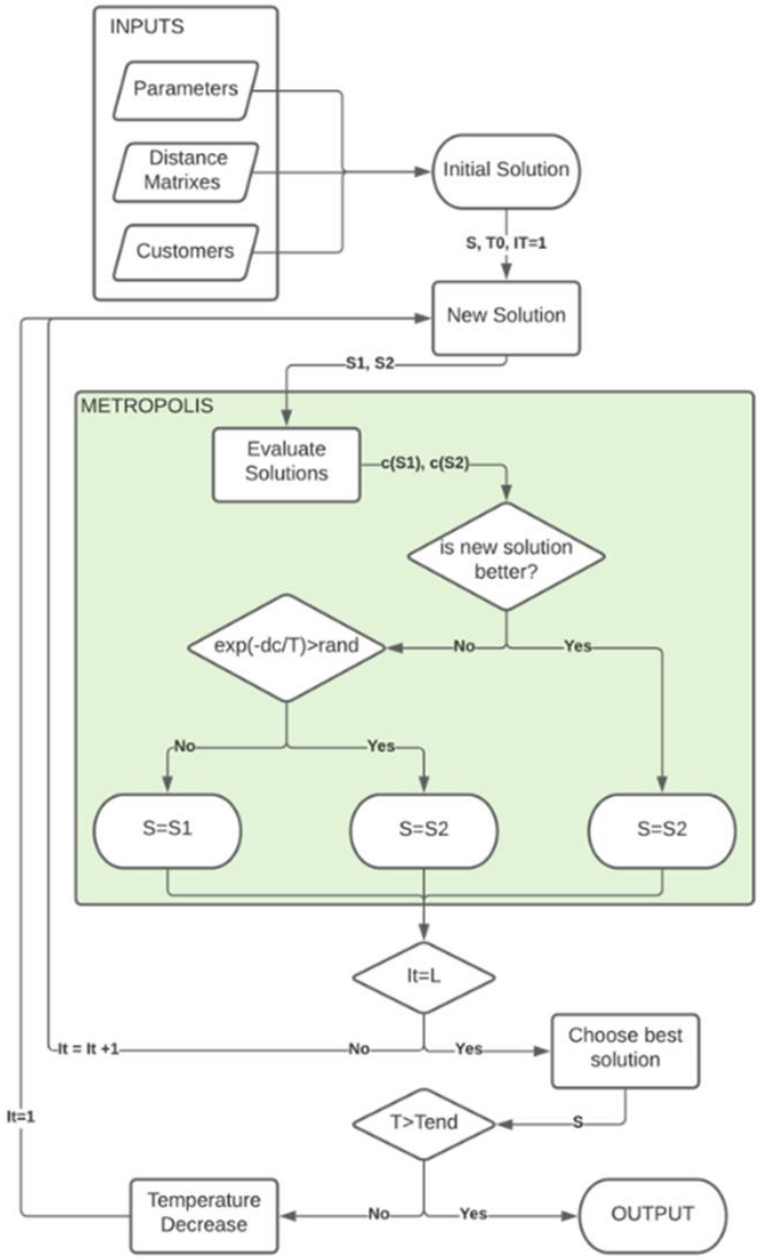

Figura 5: funzionamento dell'algoritmo di simulated annealing per l'ottimizzazione e struttura delle funzioni VRP e metropolis.

3.3.1 Soluzione Iniziale

La soluzione iniziale (S1) è la prima soluzione trovata nel file "vrp.m". Questa deve soddisfare tutte le condizioni al contorno (*saranno specificate in seguito nel paragrafo 4.3.3*) del problema e viene ottenuta

attraverso l'algoritmo; nel caso in cui le condizioni non venissero verificate l'algoritmo ripartirebbe da capo. Il file "vrp.m" definisce il codice utilizzato per l'implementazione dell'algoritmo simulated annealing e importa i parametri dell'algoritmo definiti nel file (annealingParameters.csv).

Esempio: Viene descritta, a titolo esemplificativo, una procedura di creazione della soluzione iniziale.
Si supponga di avere N consegne da effettuare denominate con una serie di numeri: 1-2-3-4-5-6-7-8-9-10. Il software prende in esame la funzione di costo per minimizzare il tempo totale percorso dai veicoli. Al primo passo genera una soluzione casuale da cui cominciare l'ottimizzazione, quindi effettua una permutazione randomica delle consegne e suddivide le consegne in itinerari di consegna (esempio di permutazione casuale delle consegne: 2-5-7-1-10-8-3-9-4-6).
Successivamente analizza le consegne una ad una partendo dalla prima della lista (quindi nell'esempio la consegna 2) e svolge le seguenti operazioni:

- inserisce uno 0 all'inizio della serie di consegne come indicazione per il deposito (punto di partenza);
- per ogni Delivery Point accumula la distanza necessaria a raggiungerlo nella variabile "disTravel", accumula il travel time nella variabile "currentTime" e la quantità di merce da consegnarvi nella variabile "delivery"; prima di passare al delivery point successivo verifica il rispetto dei vincoli;
- se la variabile "disTraveled" è maggiore alla distanza massima percorribile dal veicolo, o se la variabile "delivery" è maggiore della capacità massima del veicolo di consegna, o ancora se la variabile "currentTime" supera la finestra temporale associata alla consegna stessa, allora chiude l'itinerario di consegna corrente ponendo uno 0 prima della consegna in esame in quanto sarebbe impossibile portare a termine la consegna senza violare i vincoli. Inizia un nuovo giro di consegne e il punto di consegna in esame sarà il primo del nuovo giro;
- itera sino ad aver considerato tutti i Delivery Points.

Si ottiene una sequenza che rappresenta i giri di consegna iniziali, che dovranno essere ottimizzati (esempio: 0-2-5-7-0-1-10-8-0-3-9-4-6-0, corrispondono a tre giri di consegne).
Una illustrazione del codice è fornita in Figura 6.

```
%% Initial solution (must be a feasible solution that satisfies the constraints)
%tic
TSProute=[0,CityD(randperm(size(CityD,1)),1)']+1; %Randomly generate a TSP route excluding the tail bit
S1=ones(1,CityNum*2+1); %Initialize the VRP path S1 and allocate memory for it

DisTraveled=0;  % The distance the car has traveled
CurrentTime=CT; %Zero the vehicle's running time
delivery=0; % The car has been delivered, that is, the sum of the demand that has reached the point
k=1;
for j=2:CityNum+1
    k=k+1;   %Assign a value to the next point of the VRP path
    TTp1 = find(TravelTime(2:end,1) == S1(k-1)-1)+1; % TravelTime matrix index number 1
    TTp2 = find(TravelTime(1,2:end) == TSProute(j)-1)+1; % TravelTime matrix index number 2
    Dp1 = find(Distance(2:end,1) == S1(k-1)-1)+1; % Distance matrix index number 1
    Dp2 = find(Distance(1,2:end) == TSProute(j)-1)+1; % Distance matrix index number 2

    Dp3 = find(Distance(2:end,1) == 0)+1; % Distance matrix index number 1
    Dp4 = find(Distance(1,2:end) == TSProute(j)-1)+1; % Distance matrix index number 2
    %DisTraveled+Distance(Dp1,Dp2)+Distance(Dp3,Dp4) > Travelcon
```

Figura 6: Estratto del codice del file "vrp.m". Soluzione iniziale S1

3.3.2 Permutazione della soluzione

La permutazione della soluzione avviene tramite la funzione “Newsolution”, dove la nuova soluzione S2 viene generata a partire dalla soluzione precedente S1. “Newsolution” applica alla soluzione S1 (cioè l’itinerario in corso di ottimizzazone) una trasposizione randomica tra due punti di consegna e crea una nuova soluzione S2, un nuovo itinerario in cui le consegne avranno un nuovo ordine.
Esempio: ipotizzando S1=1-**2**-3-4-5-6-**7**-8-9-10 allora S2=1-**7**-3-4-5-6-**2**-8-9-10 è una possibile soluzione di output della routine “Newsolution”. La Figura 7 mostra un estratto del codice del file “Newsolution”.

```
function S2=NewSolution(S1)
% Input
% S1: current solution
% Output
% S2: new solution

Length=length(S1); %Get the number of original solution elements
S2=S1;

R=randperm(Length-2)+1; %Generate a random sequence of 2: CityNum+1 as the index of the position to be exchanged
S2(R(1:2))=S2(R(2:-1:1)); %transposition
```

Figura 7: Estratto del codice del file "NewSolution.m".

3.3.3 Evaluation

La funzione “Evaluation” valuta la funzione obiettivo della soluzione (corrispondente al costo associato alla soluzione) e verifica che nessun vincolo di autonomia e tempo venga superato. Viene verificato che i km totali del percorso scelto e il tempo totale per compierlo rispettino i limiti stabiliti dalla carica e dal consumo dei veicoli, e che il tempo di ciascuna consegna rientri nelle finestre temporali di consegna (time windows). Inoltre, la quantità di merce da consegnare non deve eccedere la capacità del veicolo. Nel caso in cui tali condizioni non siano verificate “Evaluation” imposta il costo della soluzione come infinito.
I pesi utilizzati per definire la funzione obiettivo sono:

- Costo di esercizio del veicolo per km percorso
- Costo operativo orario (aggiuntivo rispetto a quello chilometrico)
- Costo aggiuntivo per ciascun veicolo utilizzato (ammortamento e costo conducente).

Le finestre temporali di consegna sono rigide e supposte pari a metà giornata. Una consegna non effettuata in una fascia oraria viene assegnata alla successiva.
La Figura 8 mostra un estratto del codice “Evaluation”.

```
%Related data initialization
DisTraveled=0;  % The distance the car has traveled
CurrentTime=0; %All vehicle travel time is set to zero
delivery=0; % All vehicles have been delivered, that is, the sum of the demand that has reached the point'Œ

ttlDis=0; %The total distance of all vehicles in this scheme
ttlTime = 0;

    for i=1:length(route)-1
        if route(i)==route(i+1)
            route(i)=0;  %When the adjacent bits are all 1, the previous one is set to zero
        end
    end
    route(route==0)=[];  %Delete extra zero elements

first_delivery_index  = 2;
TTp1 = find(TravelTime(2:end,1) == 0)+1; % TravelTime matrix index number 1
TTp2 = find(TravelTime(1,2:end) == route(first_delivery_index)-1)+1; % TravelTime matrix index number 2
TTime=TravelTime(TTp1,TTp2);
TW = TimeWindow(TimeWindow(:,1) == (route(first_delivery_index)-1),2);
CurrentTime= TW - TTime; %Time to zero TimeWindow(1,2)
```

Figura 8: Estratto del codice del file "Evaluation.m".

3.3.4 Funzione Metropolis

La funzione "Metropolis" confronta la soluzione corrente (S1) con quella nuova (S2) in funzione dei valori della funzione obiettivo (che ha come variabili distanza percorsa, tempi di percorrenza, numero di mezzi impiegati e numero di operatori) per scegliere la migliore tra le due. La scelta non è deterministica, ma è soggetta ad una valutazione di natura probabilistica. Entrambe le soluzioni verranno verificate con la funzione "Evaluation".

Di seguito vengono spiegate in modo esemplificativo il processo delle soluzione intermedie di scelta. Scelto il punto iniziale coincidente con il deposito da cui cominciano i viaggi, il percorso iniziale (S1) e la temperatura inziale, durante il processo si hanno due possibilità:

1) *L'Itinerario analizzato è la scelta migliore*, cioè il valore dell'itinerario successivo è superiore a quello corrente e viene sostituito a quello corrente;

2) *L'itinerario analizzato è la scelta peggiore*, cioè il valore della funzione obiettivo è minore di quello della soluzione precedente, la scelta allora verrà approvata con una probabilità espressa in funzione dello scarto tra i due valori.

Nello specifico, come illustrato in figura 11, la soluzione S2 può essere scelta sia se è la soluzione migliore sia se rispetta l'equazione $e^{-(dC/T)} \geq rand$, dove il primo membro della disequazione è la probabilità di accettazione della nuova soluzione e il secondo membro indica un unico numero casuale (rand) distribuito uniformemente nell'intervallo (0,1). Se nessuno dei due precedenti casi è verificato, allora la soluzione migliore sarà S1.

La Figura 9 illustra un estratto del codice "Metropolis".

```
%%
[ttlDis1,ttlTime1] = Evaluation(S1,Distance,TravelTime,Demand,TimeWindow,Travelcon,Capacity,DeliveryUnloadTime);  %Calculate route length
[ttlDis2,ttlTime2] = Evaluation(S2,Distance,TravelTime,Demand,TimeWindow,Travelcon,Capacity,DeliveryUnloadTime);  %Calculate route length

%dT = ttlTime2 - ttlTime1;
% num_vehicles1 = length(find(S1 == 0))-1;
% num_vehicles2 = length(find(S2 == 0))-1;
% ttlDis1 = ttlDis1 + num_vehicles1*5;
% ttlDis2 = ttlDis2 + num_vehicles2*5;
dC = ttlDis2 - ttlDis1;   %Calculate the difference in route length
dT = 0;
if dC+dT < 0       %If the ability decreases, accept the new route
    S = S2;
    ttlDis = ttlDis2;
elseif exp(-(dC+dT)/T) >= rand   %Accept the new route with exp(-dC/T) probability
    S = S2;
    ttlDis = ttlDis2;
else  %Not accepting new routes
    S = S1;
    ttlDis = ttlDis1;
```

Figura 9: Estratto del codice di "Metropolis.m". Confronto tra le due soluzioni per determinare la migliore.

3.3.5 Informazioni aggiuntive

La funzione "TextOutput" calcola e fornisce in output le matrici ottenute come risultato dell'ottimizzazione, ossia i tempi di partenza da ogni punto di consegna/Magazzino (routeTimeS), i tempi di arrivo ad essi (routetimeE) e la distanza percorsa per raggiungerli per ciascun giro (routeDistance).

3.4 Funzione di Verifica (Check Threshold)

La funzione "checkThreshold" si occupa di verificare se uno o più itinerari superano una certa soglia di sicurezza sull'autonomia. In caso affermativo, il software rileva quali consegne portano al superamento di tale soglia e inserisce prima di esse un passaggio per una delle colonnine di ricarica. La colonnina di ricarica viene scelta minimizzando la distanza totale da percorrere per arrivare alla colonnina ed effettuare la consegna successivamente alla ricarica.

Successivamente i tempi di arrivo/partenza e distanze percorse sono aggiornati di conseguenza.

3.5 Output

Di seguito viene riportato il file "OptOut.csv" ottenuto come output del modulo software di pianificazione. OptOut.csv, riporta la sequenza temporale dei punti di consegna per ogni veicolo.

Tabella 10:Output del processo di ottimizzazione

Vehicle ID	Delivery Point ID	Arrival Time	Departure Time	Distance	SOC
		Minuti dalle 00:00	Minuti dalle 00:00	km	%
1	86616	490	510	0	90
1	83831	610	620	3.7	75

Inoltre, il file "vrp.m" fornisce automaticamente i seguenti grafici esplicativi del processo.

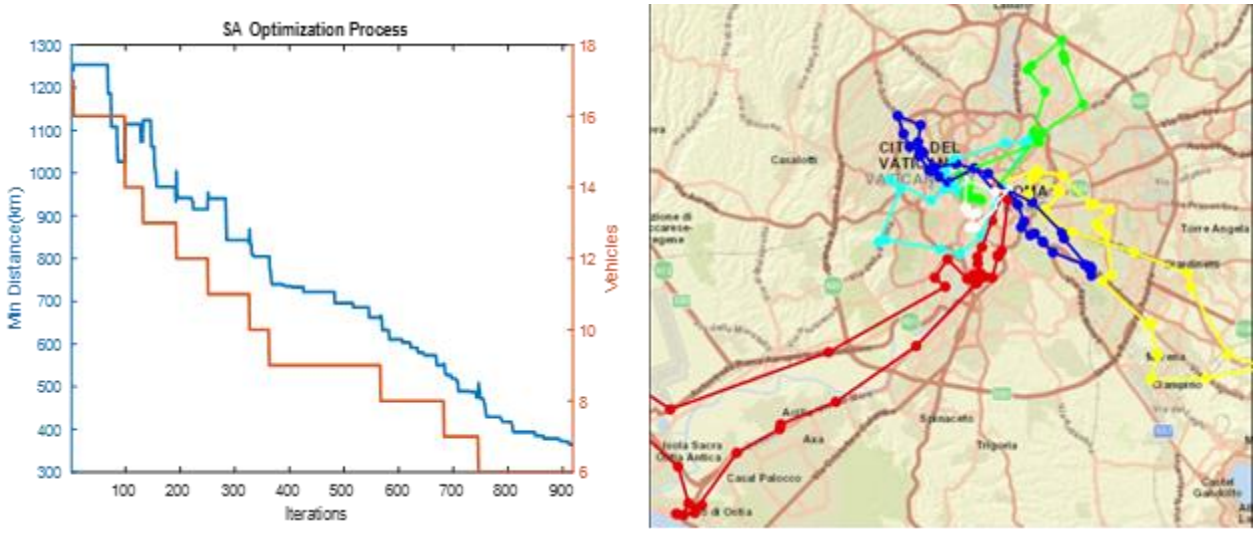

Figura 10: A sinistra grafico del processo di ottimizzazione. A destra Grafo con percorso ottimale evidenziato.

Il grafico a sinistra riporta l'andamento della distanza totale di percorrenza al variare del numero di iterazioni esterne dell'algoritmo al variare della temperatura. Si evince che man mano che le iterazioni aumentano la distanza totale percorsa dai veicoli negli itinerari diminuisce. Il grafico mostra i valori relativi alla soluzione migliore tra tutte quelle esaminate a una data temperatura. Per ogni valore di temperatura, l'algoritmo permuta la soluzione un certo numero di volte (parametro L) e sceglie la soluzione migliore tra la vecchia e la nuova permutata, per poi diminuire la temperatura finite le L iterazioni e ripetere di nuovo le L iterazioni, partendo dalla migliore di tutte quelle esaminate alla temperatura precedente.
In figura di destra sono riportati i giri risultanti dall'ottimizzazione, ciascuno contrassegnato con un colore diverso. I punti indicano i luoghi delle consegne, mentre le linee ne indicano la successione.

4 Funzioni di Monitoraggio

Completata l'ottimizzazione, il software attiva la modalità di monitoraggio durante la quale riceve informazioni sull'andamento delle consegne e verifica la presenza di possibili aggiornamenti dei dati di input.

4.1 Funzione di aggiornamento (Update)

Come precedentemente anticipato, nel file "main.m", una volta terminata l'ottimizzazione e l'eventuale inserimento di colonnine nel percorso, il codice va in pausa in attesa di un aggiornamento dei dati di input.

Nella versione finale del software, che sarà implementata nella piattaforma e-Mu, questa funzione verrà attivata automaticamente ogni volta che avverrà un aggiornamento dei dati di input.
Nella versione prototipale in ambiente di sviluppo, questa funzionalità viene attivata in simulazione dietro comando dell'utente. Digitando "update" (senza doppi apici) e premendo invio, il codice leggerà il file "update.csv" e dopo aver verificato se è necessaria una re-ottimizzazione di alcune route, procederà alla nuova ottimizzazione. Al termine della stessa, il codice andrà ancora in pausa in attesa di un comando. Digitando invece "end" (sempre senza doppi apici), il codice terminerà lasciando una scritta a conferma dell'avvenuta chiusura.
Nella versione implementata nella piattaforma, le attività svolte in simulazione saranno attuate online.
Gli aggiornamenti possono riguardare diversi aspetti: consegne incompiute, aggiunta/riduzione dei punti di consegna, cambi di posizioni dei punti di consegna o disponibilità delle colonnine di ricarica che richiedono una nuova ottimizzazione.
Per gestire gli aggiornamenti viene richiamata la routine "updateProcedure", il cui compito è quello di eseguire una nuova ottimizzazione tra le nuove consegne nel file di update e quelle affidate a veicoli che non hanno ancora lasciato il deposito. Oltre alla nuova ottimizzazione, vengono aggiornati gli output della simulazione, quali le tabelle orarie e gli itinerari di consegna e gli archi e i nodi dei singoli spostamenti all'interno del giro, in modo da avere tutte le informazioni necessarie per un eventuale recovery.

4.1.1 Check Timestamp

La funzione "checkTimestamp" verifica l'avvenuta partenza dei veicoli dal deposito al momento in cui viene l'update. Se un veicolo non ha cominciato il giro di distribuzione e non è ancora partito dal deposito, le consegne relative a quel veicolo vengono salvate e aggiunte a quelle dell'update. Questa routine permette al software di procedere nuovamente con l'ottimizzazione di tutte le consegne ancora in programma.

4.2 Recovery

La funzione "checkRecovery" è responsabile della ricerca di una stazione di ricarica per un veicolo con insufficiente autonomia nel corso dei giri di consegna.

```
currentRoute = routes{i}; % extract the route
rts = routeTimeS(i,:); % extract the row related to the current route (routeTimeS contains the times at which the vehicle depart
rts(rts == 0) = []; % delete the zeros from the array
if simulationTime <= rts(1)% the vehicle hasn't departed from the depot yet
    continue
end
sts = find(simulationTime >= rts);% sts will contain 1 for the values in rts that are lower than simulationTime and 0 otherwise.
if isempty(sts)% the vehicle hasn't departed from the depot yet
    continue
    disp('hello')
end
sts = sts(end);% I just need to know which is the last delivery point served
times = edgesTime{i}{sts};% extract from the array of the edges travel time related to current portion of route being traveled,
                          %the time passed between now and the time when the vehicle departed from the preceding delivery point
lastEdge = times + rts(sts);% rts(sts) contains the time(in minutes) when we departed from the last delivery point served
                            % times contains the cumulative time that passes
```

Figura 11: Estratto del codice di "checkRecovery.m". Questa parte del codice mostra come si interviene sulla rotta in cui il veicolo si è fermato. La partenza verrà riprogrammata da un punto diverso rispetto al deposito.

Data la posizione corrente del veicolo, rilevata in fase di monitoraggio e calcolata nel codice di "Simulation", da questa viene valutato il cammino minimo verso ogni infrastruttura di ricarica per valutare la più vicina (in termini temporali). Terminato il calcolo della stazione più vicina il percorso viene ridefinito e l'itinerario aggiornato con i tempi aggiuntivi derivanti dalla deviazione verso i punti di ricarica e permettendo così al veicolo di completare il giro previsto ed effettuare le consegne interrotte.
Le tabelle Input/Output del modulo di Recovery hanno struttura identica alle analoghe tabelle del modulo di ottimizzazione/simulazione, in quanto derivano da una ripetizione dell'ottimizzazione, attivata da una condizione di check interna al sistema, allorquando viene rilevato o un livello di carica troppo basso per un veicolo.

La figura 16 illustra un esempio di schermata online della routine “simulation” che fornisce lo stato di carica di ciascun veicolo, stimato con le funzioni di consumo energetico. Nel funzionamento reale online, il sistema può ricevere i dati dei veicoli e quando si verifica il superamento della soglia di recovery, avviare la procedura di individuazione della stazione di ricarica più vicina.

```
Simulation Time: 8:5
current charge for vehicle 3 is: 250
Simulation Time: 8:10
current charge for vehicle 1 is: 243.3331
current charge for vehicle 2 is: 247.5608
current charge for vehicle 3 is: 250
current charge for vehicle 4 is: 245.2507
current charge for vehicle 5 is: 247.0853
Simulation Time: 8:15
current charge for vehicle 1 is: 242.0871
current charge for vehicle 2 is: 245.6451
current charge for vehicle 3 is: 250
current charge for vehicle 4 is: 245.0512
current charge for vehicle 5 is: 244.3435
Simulation Time: 8:20
current charge for vehicle 1 is: 239.6575
current charge for vehicle 2 is: 244.5995
current charge for vehicle 3 is: 250
current charge for vehicle 4 is: 244.2417
current charge for vehicle 5 is: 244.2387
```

Figura 12: Esempio di Output del codice "checkRecovery.m".

4.2.1 Soglie di recovery

I parametri aggiuntivi per l’avvio del Recovery consistono nelle due soglie di attivazione, (Tabella 11), mentre la variabile “Start Recovery” è un input esterno da usare per la simulazione di un Recovery:

Tabella 11: Valori di Soglia della carica residua

Residual Charging Threshold	Travel Time Threshold
%	%

Se una delle soglie viene violata, il Recovery viene attivato e il tempo di inizio del Recovery viene registrato (variabile “Start Recovery”).

4.2.2 Input

Gli input del modulo di recovery sono i valori di input aggiornati, che nella versione prototipale in ambiente di sviluppo derivano dal processo di simulazione, ovvero la posizione e lo stato di carica dei veicoli, la disponibilità aggiornata delle stazioni di ricarica e le nuove velocità medie rilevate sugli archi stradali.
Nel Recovery, i valori della velocità vengono rilevati da una fonte esterna e aggiornati. Qualora venisse superata la soglia sulle velocità, viene automaticamente lanciata la procedura di update. La Tabella 12 illustra l’output della simulazione con le posizioni dei veicoli (latitudine, longitudine), ID arco, percentuale di arco percorso nell’istante corrente (Time), stato di carica della batteria (SOC).

Tabella 12: Valori correnti della simulazione

Vehicle ID	Time	ID arco	% percorrenza di arco	Lat	Lon	Dist	SOC

Nella Tabella 13 sono presenti i valori aggiornati del numero di stalli disponibili per la ricarica per ogni colonnina.

Tabella 13: Disponibilità delle stazioni di ricarica

Charging Station ID	Charging Station Longitude	Charging Station Latitude	Recharging Power [kW]	Availability [numeo natuale]
20056	12.324249	41.80673	25	
32931	12.2801525	41.7309823	25	

La Tabella 14 illustra i valori della tabella del grafo a seguito di un aggiornamento proveniente dall'esterno (e simulato nel processo di simulazione). La verifica di superamento della soglia sulla velocità "ThresholdSpeed" dà luogo alla procedura di update.

Tabella 14: Tabella degli archi del grafo con i campi di velocità aggiornati

StartNodes	EndNodes	Length	ID	Speed (00:00-00:59)	Speed (01:00-01:59)	Speed (… : … - … : …)	Speed (23:00-23:59)
1	2	42.71	13800054849460	50	40	…	50
1	76	61.45	13800053796191	30	20	…	30
1	2	42.71	13800054849460	50	30	…	50

4.2.3 Output

Gli output prodotti dal modulo di ottimizzazione di Recovery, sono composti dalla sequenza temporale, per ogni veicolo, dei punti di consegna ancora da effettuare ed oggetto di Recovery. La Tabella 15 ne mostra il formato con dei valori di esempio.

Tabella 15: Output del modulo di recovery

Vehicle ID	Delivery Point ID	Arrival Time [Minuti dalle 00:00]	Departure Time [Minuti dalle 00:00]	Distance [km]	SOC [%]
1	86616	490	510	0	90
1	83831	610	620	3.7	75

5 Funzionamento della Simulazione

Il modulo di simulazione permette di aggiungere alla pianificazione una simulazione del servizio, con calcolo delle posizioni dei veicoli in ogni step della simulazione, calcolo dei consumi ed eventuale recupero del veicolo in caso di guasto o di esaurimento autonomia.

Nella versione finale del software che sarà implementata nella piattaforma e-Mu queste funzioni saranno eseguite online dal monitoraggio. Gli input del monitoraggio sono generati dalla simulazione e coincidono con i suoi output.

La Simulazione viene eseguita nel file "simulation.m". La prima parte del codice è molto simile a quella di pianificazione, essendo necessario eseguire l'ottimizzazione e richiama le stesse funzioni per ottenere gli itinerari di consegna: il modulo richiama le funzioni per avere i dati di input (readData e shortestPathTA2) e per eseguire l'ottimizzazione (vrp ed associate).

Successivamente, effettua la ricostruzione dei percorsi (simulationData), avvia la simulazione e calcola le posizioni dei veicoli per ogni step visibile nel grafico in output (percorso disegnato sulla cartografia di OpenStreetMap), calcola i consumi (energyconsumption), l'eventuale necessità di recupero dei veicoli impossibilitati a continuare il percorso (checkRecovery) e gestisce gli aggiornamenti (updateProcedure).

Nella Figura 13 è illustrato il processo della simulazione nelle sue componenti.

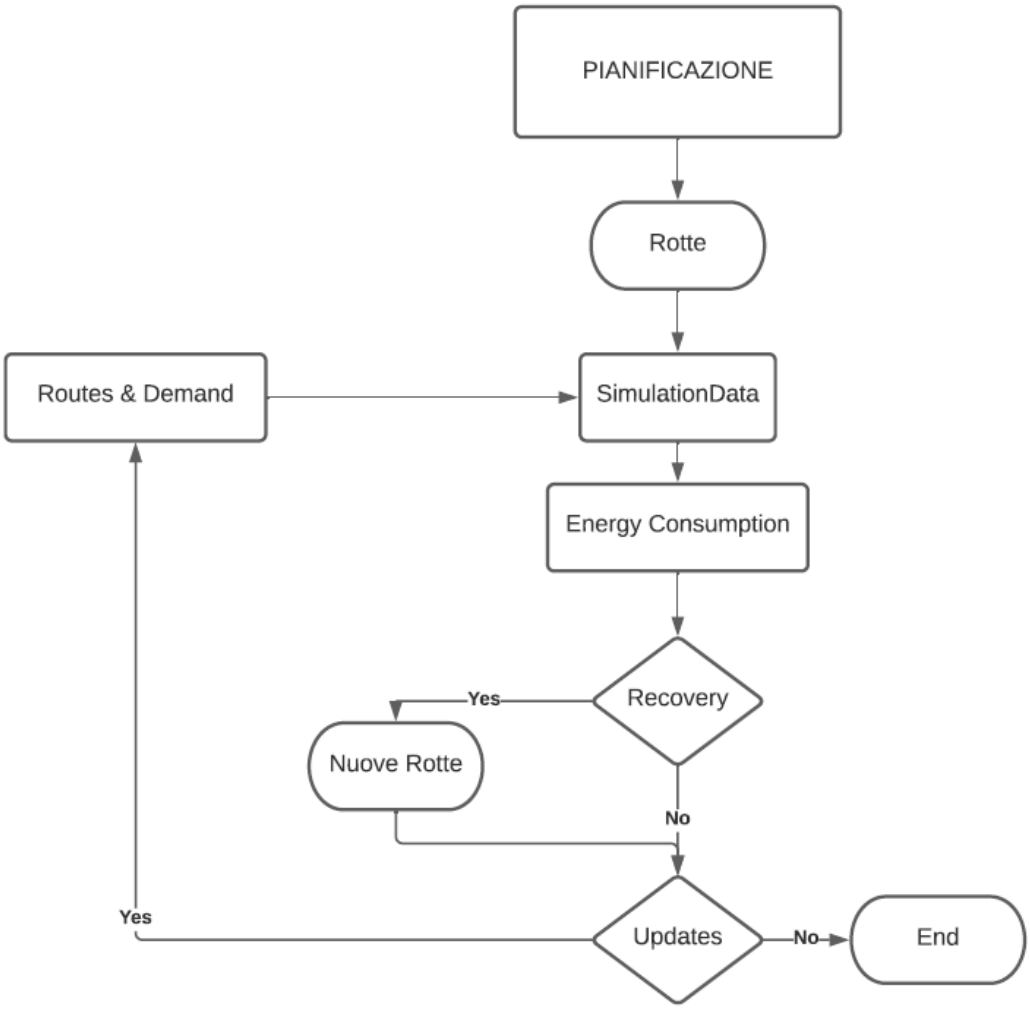

Figura 13: Funzionamento del modulo di Simulazione.

5.1 Ottimizzazione

La prima parte del codice di simulazione simula la pianificazione dei giri di consegna e ripete la procedura di ottimizzazione partendo da una soluzione iniziale per trovarne una con migliori valori della funzione obiettivo utilizzando il Simulated Annealing. Trovata la soluzione ottimale si procede alla Simulazione. La struttura e il funzionamento sono gli stessi spiegati precedentemente nel paragrafo 3.

5.2 Calcolo delle Posizioni

La funzione “SimulationData” esegue la preparazione dei dati per il codice di “simulation.m”. Utilizzando i percorsi generati da “ShortestPath”, genera due strutture dati importanti per la simulazione: edgesTime e edgesDistance. Queste due strutture contengono tempi e distanze cumulative arco per arco per ognuno degl iitinerari, suddivisi in percorsi da punto di consegna a punto di consegna. Queste strutture sono pensate per lavorare insieme alle variabili routeTimeS, routeTimeE e routeDistance (definite in “TextOutput”, quindi nella parte di Pianificazione). Ad esempio, i valori della distanza percorsa relativa ad un determinato itinerario e ad un determinato spostamento da un punto di consegna ad un altro, vanno sommati al valore corrispondente di routeDistance per determinare la distanza percorsa totale fino ad un arco prescelto. La Figura 14 illustra la pare del codice di simulazione in cui vengono salvate le sopracitate strutture dati.

```
        edgePath{1,j} = currentEdgePath; % save path in structure
    end
edgePaths{1,i} = edgePath; % save route in structure
end


graphEdges = [G.Edges{:,:}]; % extract all the data of all the edges

for i = 1:length(edgePaths) % one path for route
```

Figura 14: Estratto del codice di "SimulationData.m".

Il software di simulazione procede al calcolo delle distanze cumulative percorse dai veicoli per ogni avanzamento del tempo di simulazione. "Simulation" genera inoltre un grafico che permette di vedere per ogni step la distanza percorsa dai veicoli utilizzando la cartografia di Open Street Maps, così da poter controllare il rispetto degli orari di consegna stabiliti e quali punti di consegna vengono serviti per ogni fase.

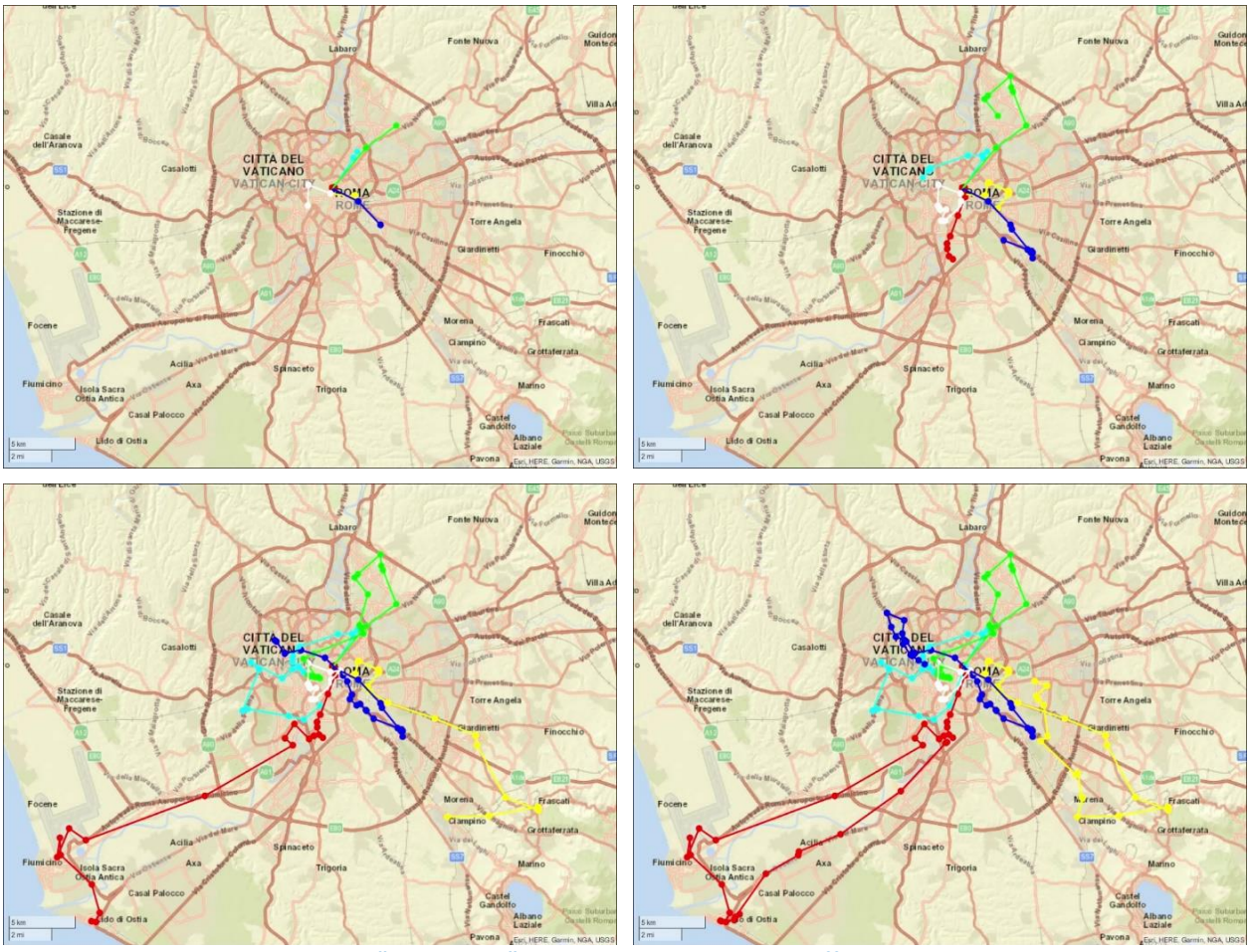

Figura 15: Output della routine "Simulation". Fotogrammi dei percorsi effettuati dai veicoli a quattro diversi step di simulazione.

5.3 Calcolo dei Consumi

Il calcolo dei consumi dei veicoli è eseguito dalla routine "EnergyConsumption", che applica le funzioni di consumo energetico con i relativi coefficienti definiti nella Tabella 16 e Tabella 17 per i veicoli di 7 tonnellate e nella Tabella 18 e Tabella 19 per i veicoli di 12.5 tonnellate.

Tabella 16: espressioni delle curve di consumo [Wh/km] COPERT per veicoli IC da 7 tonnellate

Carico del veicolo	Espressioni delle curve di consumo [Wh/km]
0% < p < 30%	$y(V) = \left(e + \left(a \cdot \exp^{-bx}\right) + \left(c \cdot \exp^{-dx}\right)\right) \cdot 12.72$
31% < p < 70%	$y(V) = \left(\left(a \cdot x^{b}\right) + \left(c \cdot x^{d}\right)\right) \cdot 12.72$
71% < p < 100%	$y(V) = \left(\left((c \cdot x^{2}) + (b \cdot x) + a\right)^{-1}\right) \cdot 12.72$

Tabella 17: coefficienti delle curve di consumo [Wh/km] COPERT per veicoli IC 7 tonnellate

Carico del veicolo	a	b	c	d	e
0% < p < 30%	1.014595	-0.04122	187.4573	0.077275	75.34241
31% < p < 70%	0.001	2.55885	438	-0.4216	
71% < p < 100%	0.003614	0.000219	-1.9E-06		

Tabella 18: espressioni delle curve di consumo [Wh/km] COPERT per veicoli IC da 7 tonnellate

Carico del veicolo	Espressioni delle curve di consumo [Wh/km]
0% < p < 30%	$y(V) = [e + (a \cdot \exp^{-bx}) + (c \cdot \exp^{-dx})] \cdot 12.72$
31% < p < 70%	$y(V) = [e + (a \cdot \exp^{-bx}) + (c \cdot \exp^{-dx})] \cdot 12.72$
71% < p < 100%	$y(V) = [c + (a \cdot \exp^{bx})] \cdot 12.72$

Tabella 19: coefficienti delle curve di consumo [Wh/km] COPERT per veicoli IC 7 tonnellate

Carico del veicolo	a	b	c	d	e
0% < p < 30%	486,6406	0,080972	1196063	1,629846	134,0512
31% < p < 70%	474,497	0,07433	10230354	1,9942	146,568
71% < p < 100%	503,41	-0,07251	160,5909		

La funzione calcola la quantità di merce ancora da consegnare per conoscere il peso totale del veicolo, quindi calcola il tempo trascorso e la distanza trascorsa dall'ultimo calcolo dei consumi (vengono calcolati distanza e tempo dello step precedente per essere sottratti al corrente) e da queste misure si deriva la velocità media, che verrà poi usata per il calcolo dei consumi energetici. Viene quindi aggiornata la carica residua del veicolo considerando 3 soglie, ad esempio: carica minore al 30 %, tra il 30 e il 70% e maggiore del 70%. Dopo la ricarica di un veicolo ad una colonnina, la carica residua viene aggiornata. La Figura 16 illustra il funzionamento della routine "energyConsumption" nell'aggiornamento del calcolo dei consumi utilizzando le funzioni definite sopra.

```
% consuption calculations
if percentage <= 30
    if vehicleWeight == 12.5
        a = 486.6406;
        b = 0.080972;
        c = 1196063;
        d = 1.629846;
        e = 134.0512;
        consumption = (e + (a*exp(-b*velocity))+(c*exp(-d*velocity)))*12.72;
    else
        a = 1.014595;
        b = -0.04122;
        c = 187.4573;
        d = 0.077275;
        e = 75.34241;
        consumption = (e + (a*exp(-b*velocity))+(c*exp(-d*velocity)))*12.72;
    end
```

Figura 16: Estratto del codice di "EnergyConsumption.m".

5.4 Output

Gli output della simulazione consistono in una tabella che contiene una riga per veicolo e per ogni istante di scrittura (definito dalla variabile TimeStep in simulationParameters.csv) con le informazioni relative a posizione, carica della batteria e distanza percorsa.
Gli output della simulazione coincidono con quelli che, nella piattaforma e-Mu, saranno gli input del monitoraggio.

Tabella 20: Output della simulazione

Vehicle ID	Time	ID arco	% percorrenza di arco	Lat	Lon	Dist	SOC

6 Conclusioni e Sviluppi Futuri

Il presente rapporto tecnico ha illustrato il funzionamento del software per ottimizzare la distribuzione urbana delle merci con veicoli elettrici, definito il contesto operativo con vincoli al problema, caratteristiche di veicoli e clienti, rete di trasporto. Il software prende in considerazione diversi vincoli operativi per rendere il software realistico: vincoli temporali, spaziali e di carico. Altri elementi di innovazione modellistica sono l'inclusione del livello di autonomia dei veicoli elettrici nell'ottimizzazione dei giri di consegna, considerando la variabilità dei tempi di percorrenza dovuti alla congestione stradale e alla distribuzione di infrastruttura di ricarica presenti sul territorio.
Oltre alla procedura di ottimizzazione per una corretta pianificazione giornaliera o di fascia oraria, è compreso nel software anche un modulo di simulazione, che permette di tener traccia dei valori previsti di carica dei veicoli e di monitorare la posizione degli stessi per assicurarsi il rispetto della tabella oraria definita in pianificazione. La simulazione è necessaria al modulo di recovery, in quanto al sorgere di un'anomalia, viene rieseguita l'ottimizzazione dell'itinerario, verificando il rispetto dei vincoli e inserendo se necessario una ricarica intermedia a una infrastruttura di ricarica.

7 Appendice

Gaetano Fusco è professore associato di Trasporti presso l'Università di Roma “La Sapienza”, dove insegna *Traffic Engineering and Intelligent Transportation Systems*. Dal 1992 al 2005 è stato ricercatore presso la stessa Università. Nel 2014 ha conseguito l'abilitazione come professore ordinario.
È esperto tecnico-scientifico del Ministero dell’Istruzione, dell’Università e della Ricerca e membro del Consiglio direttivo del Centro di Ricerca per il Trasporto e la Logistica (CTL) dell'Università di Roma “La Sapienza”. È coordinatore scientifico di numerosi progetti di ricerca finanziati da parte di società private ed enti pubblici e, In particolare, responsabile scientifico delle valutazioni di impatto sulle politiche dei trasporti della Commissione Europea, Direzione DG TREN, poi DG MOVE, svolte in un consorzio coordinato dalla società PricewaterhouseCoopers.
È autore di circa 100 pubblicazioni scientifiche sulla letteratura nazionale ed internazionale inerenti varie tematiche dei sistemi di trasporto, quali la progettazione delle reti di trasporto, la logistica e la pianificazione dei trasporti, la modellazione e la stima della domanda di trasporto, la teoria del deflusso veicolare, ma prevalentemente focalizzate sulle metodologie di applicazioni dei Sistemi di Trasporto Intelligenti, quali la regolazione semaforica, i sistemi di informazione all'utenza, la stima delle condizioni di traffico.

Chiara Colombaroni è Ricercatore e Professore aggregato di Trasporti all’Università di Roma La Sapienza, dove insegna *Programming for Transport Systems*. Nella medesima Università, ha conseguito la Laurea in Ingegneria dei trasporti nel 2003, la Laurea Magistrale in Ingegneria dei sistemi di trasporti nel 2006 e il dottorato di ricerca in Infrastrutture e Trasporti nel 2011. Nel 2013 è stata Ricercatore e Professore aggregato presso l’Università Niccolò Cusano di Roma, dove ha insegnato *Tecnica ed Economia dei Trasporti* e *Tecnologie per il Trasporto Sostenibile*.
Le sue attività di ricerca sono rivolte alla pianificazione dei trasporti e alla modellazione dei trasporti, in particolare la teoria del deflusso veicolare, i modelli comportamentali di guida, i sistemi di controllo del traffico, la progettazione di rete, la sicurezza stradale, l’ottimizzazione del posizionamento dei container nei terminal intermodali, l’uso dei Big data in mobilità.

Filippo Carrese è ingegnere dei trasporti e studente di dottorato in Infrastrutture e trasporti presso l’Università di Roma La Sapienza. La sua attività di ricerca verte sulle applicazioni del concetto di accessibilità nei sistemi di trasporto, sia in relazione alla mobilità dei passeggeri che alla distribuzione delle merci.

Natalia Isaenko è un’assegnista di ricerca con particolare attenzione alla modellazione e pianificazione dei trasporti. Ha conseguito la laurea magistrale in Ingegneria dei trasporti (2014) presso l'Università La Sapienza di Roma ed è stata insignita del premio "Excellent Graduate Student" per gli eccezionali risultati accademici. Dopo la laurea è entrata a far parte del Dipartimento dei Trasporti dell'Università di Roma “La Sapienza” come borsista nell'ambito di un progetto multidisciplinare riguardante l'applicazione dell'analisi del traffico online e dei metodi per la previsione del traffico. Ha completato la sua tesi di dottorato “Metodi spaziotemporali per previsioni di traffico a breve termine” in Infrastrutture e Trasporti nel 2019 presso l'Università La Sapienza di Roma dove è tutor didattico per studenti stranieri. Durante il suo dottorato di ricerca è stata coinvolta in diversi progetti di ricerca, relativi alla modellazione del traffico, alla previsione del traffico e alla calibrazione dei modelli di domanda di mobilità.