POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

AI Powered Smart Logistics

Logistica per l'industria 4.0 supportata da Intelligenza Artificiale

Relatore
prof. Paolo Garza

Candidato
Giuseppe Distefano
matricola: 256663

Anno accademico 2020-2021

Ringraziamenti

Ringrazio tutta la mia famiglia, i miei amici e chi mi è stato vicino per il supporto che mi è stato dato in un periodo pesante e difficile per tutti.

Ringrazio inoltre il relatore per questo lavoro, il professor Paolo Garza, per i consigli ricevuti nella realizzazione di questa tesi.

Indice

Capitolo 1

Introduzione

Con l'avvento dell'**Industria 4.0** i leader nel settore delle catene logistiche (o delle *"Supply Chain"*) hanno l'obiettivo di garantire qualità, consegne nei tempi stabiliti, disponibilità dei prodotti e costi contenuti, monitorando e mitigando un numero sempre maggiore di potenziali eventi dannosi, assicurandosi che i fornitori e i componenti dei prodotti soddisfino le aspettative dei clienti.
La natura di questi eventi comprende sia eventi comuni (come condizioni climatiche, ritardi nelle consegne o difetti di produzione) che eventi di importanza maggiore (come instabilità politica, disastri naturali o l'instabilità finanziaria di un fornitore), che possono essere sia interni che esterni alla Supply Chain stessa.

Con il passare degli anni e l'avanzamento tecnologico le catene logistiche si sono dovute evolvere in **reti di catene logistiche** (*Supply Chain Network* o "SCN"), con una struttura non più lineare o sequenziale ma complessa, rappresentabile con una rete. In modo da poter soddisfare gli obiettivi precedentemente menzionati, le SCN devono essere progettate e aggiornate costantemente.
Da questa esigenza nasce il problema della *Supply Chain Network Design* (SCND), ossia la progettazione efficiente di SCN, in grado di ottimizzare il flusso di operazioni che coinvolgono tutti gli enti che partecipano alla realizzazione e alla distribuzione di un prodotto.

Obiettivo della tesi è la realizzazione di un *Proof of Concept* di uno strumento di progettazione di SCN, che analizza dati di tipo storico, in tempo reale e di previsione provenienti da una azienda del settore Automotive, per realizzare la SCN ottimizzata sfruttando le potenzialità dell'Intelligenza Artificiale.

In questo studio verranno complessivamente utilizzate:

- Tecniche di Clustering appartenenti al Machine Learning non supervisionato (DBSCAN), applicate secondo una tecnica ricorsiva per raggruppare in modo efficace i clienti sul territorio di distribuzione.
- Strumenti appartenenti a Google come la piattaforma open-source OR-Tools, usata per la risoluzione di una variante complessa del *Vehicle Routing Problem* (VRP) in modo da ottenere la distribuzione efficiente dei prodotti
- Le API di Google Maps per la realizzazione delle mappe geografiche di distribuzione basate su strade reali e il servizio *Directions* per ottenere le indicazioni stradali da usare per le tratte.
- I servizi meteo offerti dal collettivo *OpenWeather* per realizzare una gestione della distribuzione che evita condizioni meteo sfavorevoli.

Di questo strumento ne verranno realizzate quattro versioni, di complessità via via crescente, numerate da 1 a 4, più una "versione 0", che non sfrutta l'approccio di clustering ricorsivo, realizzata a scopo esemplificativo per mostrare l'impatto del clustering in termini di qualità della soluzione.

Nello scenario di distribuzione di partenza sono presenti molteplici stabilimenti di produzione, molteplici prodotti da distribuire e veicoli con capacità fissa. A questo scenario verrà aggiunta prima la possibilità di passare attraverso punti intermedi, noti come centri distribuzione, poi verranno realizzate mappe su strade reali. Infine, verrà aggiunto il supporto alla gestione di eventi esterni alla Supply Chain forniti tramite una fonte esterna.

Per quanto concerne i capitoli successivi:

- Il capitolo 2 rappresenta una introduzione di concetti importanti per la comprensione del testo, con relativo contesto storico e letterario.
- Nel capitolo 3 verranno affrontati temi come il DBSCAN e il VRP in maniera più approfondita e tecnica.
- Il capitolo 4 presenterà le 5 versioni dello strumento di SCND con enfasi sul come sono state realizzate.
- Nel capitolo 5 verranno presentati e discussi i risultati relativi alle varie versioni mettendoli a paragone.
- Il capitolo 6 agirà da conclusione, fornendo anche spunti di miglioramento.

Capitolo 2

L'industria 4.0 e il *Vehicle Routing Problem*

2.1 L'industria 4.0

L'Industria 4.0 rappresenta il processo di automazione delle industrie tramite l'integrazione di tecnologie definite "abilitanti" o *smart*.

A volte chiamata "Quarta rivoluzione industriale", l'Industria 4.0 nasce nel 2011 da un progetto facente parte di una strategia economica della Germania - che riguardava l'automazione e l'ammodernamento del proprio settore manifatturiero - e presentato nella sua versione finale nel 2013. I princìpi dietro l'Industria 4.0 arriveranno in Italia a partire dal 2016 con il progetto "Impresa 4.0".

Questi princìpi si basano sull'idea di *smart factory*, che include al suo interno l'uso di:

- Tecnologie di interconnessione atte alla collaborazione tra strumenti, macchine ed operatori, in grado di processare enormi quantità di dati.

- Infrastrutture informatiche atte all'integrazione di sistemi e strutture che devono essere trasparenti nel fornire informazioni chiare sulle decisioni.

- L'implementazione di sistemi atti ad assistere gli operatori nel prendere decisioni, svolgere task rischiosi e a risolvere i problemi.
- L'uso di sistemi sempre più performanti e con consumi minori.
- L'uso di sistemi cyberfisici che tramite tecnologie come l'Internet-of-Things permettono ai sistemi di comunicare e di prendere decisioni decentralizzate e in maniera automatica.

Per quanto concerne le tecnologie abilitanti, di seguito sono raffigurate le 9 tecnologie secondo il Piano nazionale Industria 4.0 (2017-2020) del Ministero dello Sviluppo Economico Italiano (MISE) [1].

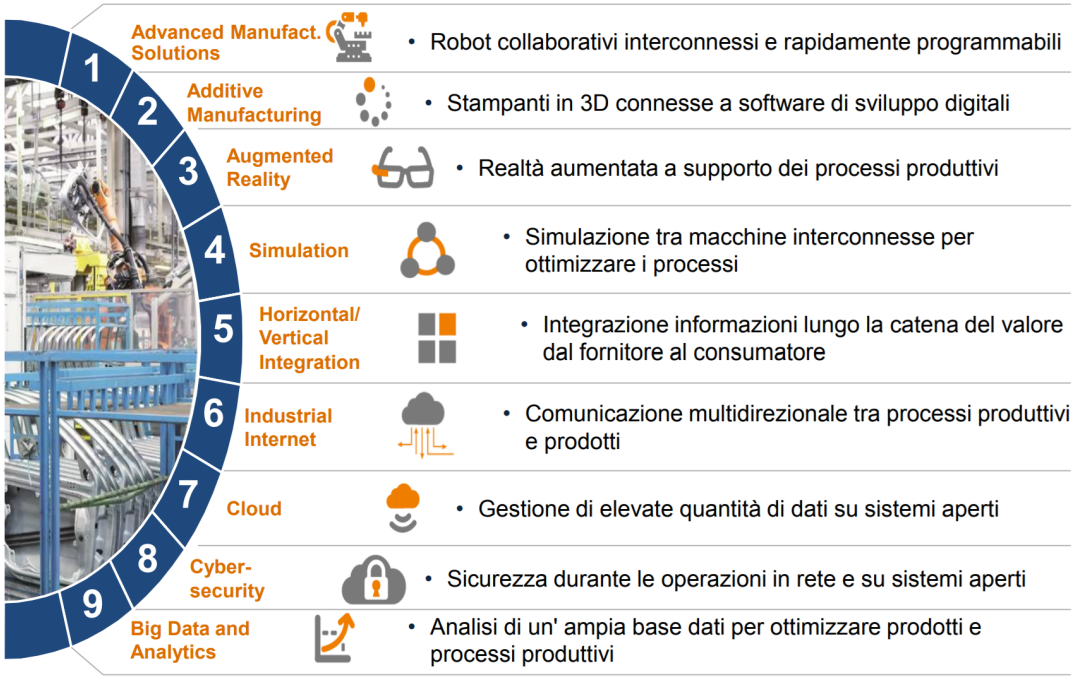

Figura 2.1. Le 9 tecnologie abilitanti secondo il MISE

Basandoci su questa categorizzazione, le tecnologie abilitanti di interesse per questa tesi sono:

- *Advanced Manufacturing Solutions*: L'impiego di sistemi di produzione avanzati, interconnessi e modulari per garantire scalabilità e prestazioni migliori.

- *Simulations*: L'utilizzo di simulazioni per l'ottimizzazione dei processi.
- *Horizontal/Vertical Integration*: L'implementazione di sistemi integrati che permettono lo scambio di informazioni tra gli enti del processo produttivo sia in orizzontale che in verticale.
- *Industrial Internet*: Scambio di dati e informazioni tramite internet sia all'interno che all'esterno dell'azienda.
- *Cloud*: Sistemi di Cloud Computing usati per archiviazione di informazioni e analisi dei dati.
- *Big Data and Analytics*: Tecnologie per l'analisi di grandi moli di dati per ottimizzare il processo produttivo e realizzare previsioni.

In conclusione, l'Industria 4.0 rappresenta la tendenza verso l'automazione e il *data-exchange* nelle tecnologie e nei processi manifatturieri tramite l'implementazione di sistemi cyberfisici, Internet-of-Things (IoT), Cloud Computing e **Intelligenza Artificiale**.

2.2 L'Intelligenza Artificiale

Il termine Intelligenza Artificiale (IA) venne coniato a Stanford dal professor John McCarthy nel 1956. Con questo termine si possono identificare quei sistemi che approssimano, replicano e infine migliorano il processo di ragionamento tipico dell'uomo [2].
Sono state realizzate tante definizioni di IA nel corso degli anni. Quella utilizzata per questo studio vuole l'intelligenza artificiale come la scienza del realizzare programmi e macchine in grado di:

- *Agire umanamente*, in modo da fornire una risposta indistinguibile da quella fatta da un umano.
- *Pensare umanamente*, ossia seguire un processo logico che ricorda quello umano.
- *Pensare razionalmente*, ossia seguire una linea di pensiero "corretta" (un concetto che risale ad Aristotele).
- *Agire razionalmente*, in modo da raggiungere il miglior risultato o, laddove vi sia incertezza, il miglior risultato tra quelli possibili.

Andando più nel dettaglio, un ente che agisce razionalmente (agente razionale) deve massimizzare il raggiungimento di obiettivi predefiniti. Questi devono essere espressi in termini di *utilità* del risultato e rappresentati come quantità misurabile. Poiché si misura solo l'utilità di un risultato, ossia le decisioni prese, la razionalità non riguarda il processo di ragionamento utilizzato. Essere razionale quindi significa massimizzare l'utilità attesa, vincolata solo dalla potenza di calcolo della macchina.
Sfruttando queste definizioni, l'intelligenza artificiale si può anche intendere come "razionalità computazionale" [3].
Occorre definire come un agente razionale possa "imparare" qualcosa. Diremo che un agente razionale *impara* quando il risultato di una sua azione, misurato tramite un certo valore P, su una certa operazione T, migliora di un valore di esperienza E [4].

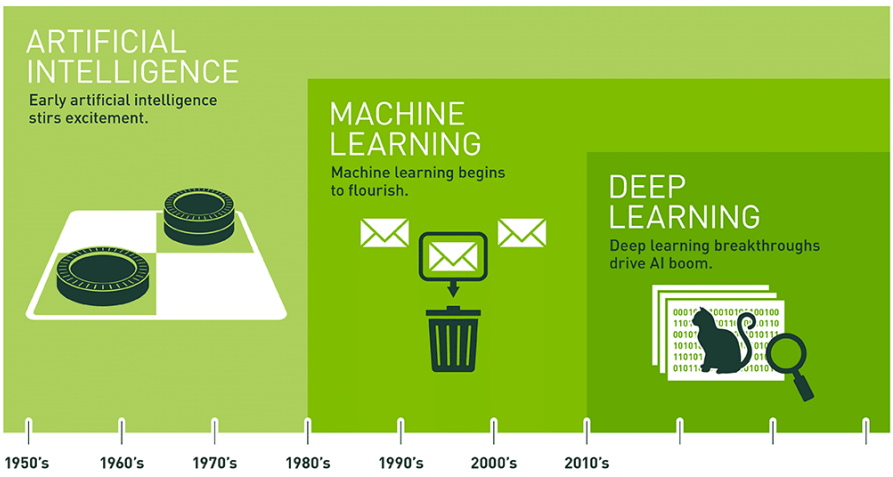

Figura 2.2. Rappresentazione visiva dell'evoluzione dell'IA. Fonte: Nvidia [5]

Legato al concetto di intelligenza artificiale vi è quello di *Machine Learning* (ML), un sottoinsieme dell'IA e un'area di studio dell'informatica dedicata allo sviluppo di sistemi e algoritmi atti a valutare e categorizzare dati (solitamente appartenenti a un certo dominio) per trovare ed estrarre schemi, o *pattern*, al loro interno. Il risultato di questi algoritmi è una decisione o *insight*. Questi permettono di migliorare l'abilità del sistema di avere un comportamento efficace, adattandosi alle nuove circostanze.

Con una prima categorizzazione possiamo dividere i modelli di ML in supervisionato e non supervisionato. Nel primo caso il modello predittivo

(l'algoritmo) è già a conoscenza di dati correlati di etichetta (ossia l'obiettivo della predizione) e sulla base di questi cerca di fare previsioni su dati senza attributo etichetta. In poche parole il modello viene allenato su dati dove il risultato è noto per poi essere usato su dati dove il risultato non è noto. In questo ramo del ML si identificano i problemi legati a classificazione di un attributo discreto o regressione legata ad attributi continui.
Il ML non supervisionato invece non ha a che fare con dati già dotati di etichetta. Qui i modelli cercano di estrarre similarità insite negli stessi dati in modo da poterli raggruppare o categorizzare. Questi modelli sono molto utili in caso di dati con dimensionalità molto elevata, dove le similarità tra i dati potrebbero non essere evidenti. Problemi di ML non supervisionato sono ad esempio legati a *clustering*, associazione tra dati (usati per identificare sequenze) e riduzione della dimensionalità (usati nei problemi di Big Data Visualization).
Lo studio collegato a questa tesi sfrutta un algoritmo di clustering noto come **DBSCAN** (*Density-Based Spatial Clustering of Applications with Noise*), applicato però in maniera ricorsiva. Verranno fornite più informazioni a riguardo nel capitolo 2.

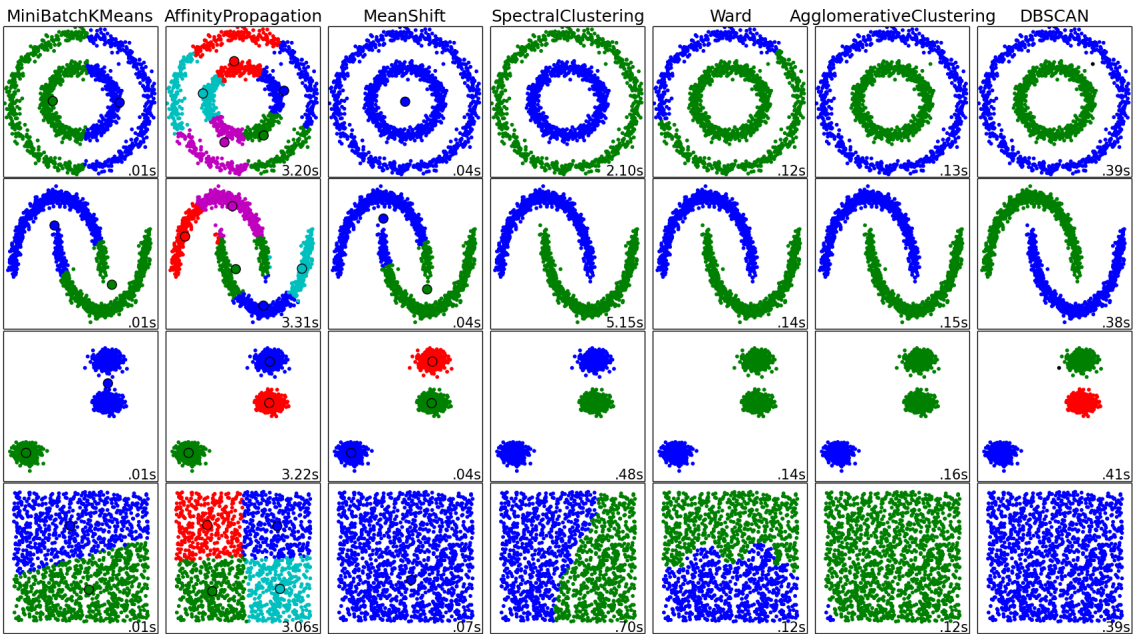

Figura 2.3. Esempi di output per diversi algoritmi di clustering su uno stesso dataset Fonte: scikit-learn

Andando più in profondità, è presente un ulteriore sottoinsieme del Machine Learning, noto come *Deep Learning* (DL), che viene realizzato tramite l'impiego di Reti Neurali, atte a imitare la struttura e il funzionamento dei neuroni umani. Questi sistemi apprendono e si aggiornano in maniera continuativa all'arrivare di nuove informazioni. In queste reti ogni connessione può essere regolata in modo da assegnare una importanza, o un peso, maggiore o minore a un certo attributo, per migliorare la qualità del risultato.
Come tecnologia l'IA, oltre a essere un campo discusso anche da filosofi oltre che scienziati (per via dei suoi aspetti etici), è stata caratterizzata da tempi di alti e bassi in termini di importanza nel corso degli anni, con una alternanza tra scoperte minori e maggiori che hanno portato l'industria a periodi di avvicinamento o di diffidenza verso l'IA. Oggi l'IA è presente in ogni campo e in molti aspetti della vita quotidiana.
Ricollegandoci però alla logistica, da un paio di anni a questa parte questo settore ha iniziato la sua transizione a una industria supportata dall'Intelligenza Artificiale. Grazie alle sue capacità nel *problem solving* essa ha le potenzialità di migliorare le attività logistiche sotto ogni punto di vista, estendendo l'efficienza umana per quanto concerne competenze, qualità e velocità, eliminando lavori mondani e di routine, permettendo alla forza lavoro logistica di concentrarsi solo sui lavori che hanno un impatto sui processi della **Supply Chain** [2].

2.3 La *Supply Chain* e le sfide ad essa collegate

La catena logistica, o filiera o *Supply Chain* (SC), rappresenta l'insieme di tutte le entità atte alla fornitura di un prodotto o servizio ai clienti.
Le filiere "classiche" possono essere rappresentate come un processo lineare o sequenziale, con alcune di queste che possono discostarsi da questo modello per permettere il rientro di certi prodotti nella catena di fornitura (ad esempio usati o danneggiati) per il loro completo riutilizzo o di alcune delle loro parti. A seguito del rapido avanzamento tecnologico, tuttavia, le catene logistiche si sono dovute adattare e trasformare in breve tempo in **reti** di catene logistiche, con strutture che coinvolgono diversi gradi di connettività e interdipendenza tra più entità.
Come accennato nel sommario ci si riferirà alle reti di catene logistiche con *Supply Chain Network* o SCN.

Una SCN può includere al suo interno entità quali:

- **Persone** (comprensivo di risorse umane e clienti finali)
- **Organizzazioni**
- **Fornitori**
- **Processi interni** (usati per supporto alla SC)
- **Disponibilità di risorse**
- **Luoghi**

  Stabilimenti di produzione

  Magazzini per lo stoccaggio

  Centri di distribuzione

  Punti vendita
- **Servizi logistici** (spesso appartenenti a terze parti)
- **Mezzi di trasporto** (via terra, rotaie, mare o aereo)
- **Mercati di destinazione**
- **Normative fiscali e di tassazione**

Considerando lo scenario di business attuale, che di anno in anno diventa sempre più complesso, le aziende che operano nel settore delle SC (sia a livello locale che globale) si ritrovano a dover consegnare i loro prodotti o servizi con livelli di qualità maggiore e costi ridotti, tenendo in considerazione un insieme di rischi ed eventi di natura disruptiva crescente. Affrontare questa sfida, sempre in continua evoluzione, porta a tenere costantemente aggiornata la SCN, la cui ottimizzazione diventa quindi un obiettivo primario.

Per quanto concerne gli eventi dannosi per la catena di fornitura, questi si possono classificare in eventi endogeni (interni alla SCN) ed esogeni (esterni alla SCN). Esempi di questi sono:

- **Eventi endogeni**

  Ritardi nelle consegne

  Difetti di produzione

  Problemi con un fornitore (instabilità economica, mancanza di materiali, investigazioni legali)

- **Eventi esogeni**

  Condizioni meteo avverse

  Disastri naturali

  Scioperi dei lavoratori

  Instabilità civile o politica di uno stato in cui deve transitare una spedizione

In questo scenario, vista la complessità dello schema di una SCN, le aziende hanno progressivamente investito risorse per realizzare le reti che meglio supportano i loro processi logistici.
Questo ha dato vita al problema noto come *Supply Chain Network Design*, ossia la progettazione efficiente di SCN, che ha portato all'utilizzo di modelli matematici per ottimizzarle. Più in particolare, per quanto riguarda la consegna ottimale dei prodotti ai clienti tramite un servizio logistico, si è visto l'impiego di modelli e algoritmi di ottimizzazione combinatoria sui grafi, tipici della ricerca operativa, poiché è possibile rappresentare la SCN come un grafo connesso pesato composto da nodi e archi che rappresentano rispettivamente luoghi/impianti (con le proprie capacità/disponibilità) e i relativi flussi di prodotti.

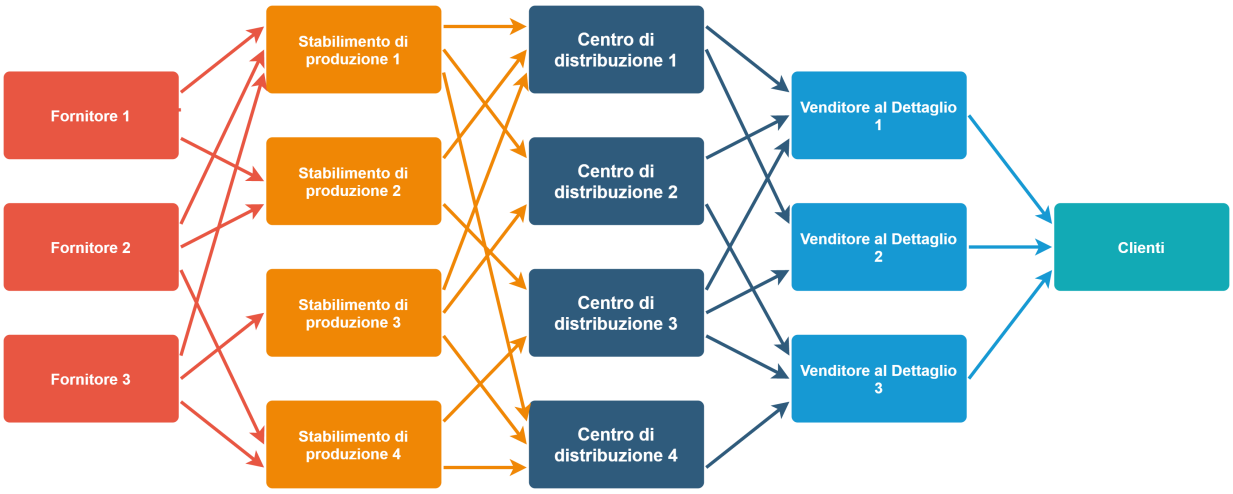

Figura 2.4. Esempio di Supply Chain Network

Realizzando però una SCN con l'impiego di questi modelli, non è mai presente una soluzione ottima né un metodo perfetto per costruire tutte le reti. È sempre presente un compromesso tra aspetti come efficienza, costi e

risposta ai rischi. Più nel dettaglio, il tipo di problemi adatto alla gestione della logistica appartengono alla famiglia dei ***Vehicle Routing Problems*** (VRP), il cui tema verrà affrontato nel prossimo paragrafo. L'enfasi sul problema logistico è legato al fatto che esperti suggeriscono che l'80% dei costi nella SC è dato dalla posizione degli stabilimenti e dal flusso dei prodotti che parte da essi [2].
Uno dei modi per affrontare questa sfida è l'utilizzo congiunto di IA e di risoluzione di VRP nella progettazione di SCN. Le aziende in questo settore si ritrovano in una posizione tale da poter solo beneficiare dalle potenzialità dell'IA. Questo per via delle grandi moli di dati (strutturati e non) generati giornalmente, che l'IA può usare per trovare dipendenze che un umano potrebbe non trovare, oppure la crescente dipendenza delle aziende da una infrastruttura di rete, oppure ancora per ridurre i tempi per operazioni ripetitive che appartengono ai settori di contabilità, finanza, legale o risorse umane. Si può quindi utilizzare l'IA per ridefinire pratiche aziendali, prendere decisioni e agire in maniera reattiva o proattiva, risparmiando tempo e incrementando la produttività generale dell'azienda [2].
Nel caso obiettivo di questa tesi si sfrutterà l'IA, più in particolare il ML non supervisionato, e la risoluzione di varianti del VRP per analizzare dati storici, dati in tempo reale (es. condizioni meteo) e dati frutto di stima o previsione (piani di produzione e commerciali) per la realizzazione della SCN ottimizzata, che porta gli operatori logistici a prelevare, trasportare e consegnare i prodotti ai clienti seguendo le rotte ottimali.

2.4 Il *Vehicle Routing Problem* (VRP) e le sue varianti

Come accennato nel paragrafo precedente, il VRP rappresenta quella famiglia di problemi di distribuzione e trasporto che rispondono all'esigenza di trovare il set di rotte migliori per far sì che un insieme di veicoli, ognuno con una certa capacità, possa consegnare uno o più prodotti, realizzati in uno o più stabilimenti o depositi, a un certo insieme di clienti, per poi tornare al punto di partenza. La natura di questo problema lo rende sempre attuale e applicabile in qualsiasi contesto. Esso presenta diversi tipi di ottimizzazione possibile a seconda della sua applicazione, ad esempio:

- minimizzare i costi di trasporto (in termini di distanza percorsa o carburante consumato)
- minimizzare il tempo di viaggio
- minimizzare il numero di veicoli usati
- massimizzare i profitti
- minimizzare le penalità in caso di servizio con qualità inferiore

Essendo alla base un problema di ottimizzazione appartenente alla Programmazione lineare intera (branca della Ricerca operativa), l'ottimizzazione si può esprimere tramite una funzione obiettivo (lineare a più variabili) da massimizzare/minimizzare, correlata da una serie di vincoli di uguaglianza o disuguaglianza lineari oltre all'avere una o più variabili che possono assumere valori soltanto interi.
La distribuzione tra depositi/stabilimenti e clienti viene realizzata tramite un grafo connesso dove i nodi rappresentano depositi e clienti e gli archi rappresentano le strade che li connettono. Questi ultimi hanno dei costi associati relativi a lunghezza o tempo di percorrenza, che può variare a seconda del tipo di veicolo.
Il VRP è una generalizzazione del problema del commesso viaggiatore (*Traveling Salesman Problem* o TSP), dove è presente un solo veicolo con capacità infinita, che rappresente il commesso viaggiatore, che deve trovare l'itinerario migliore (inteso come più breve o più economico) per attraversare tutte le destinazioni.
Di questo problema sono state realizzate nel corso degli anni molte varianti e specializzazioni, tra cui:

- VRP multi-deposito (MD-VRP)
- VRP multi-prodotto (MP-VRP)
- VRP con finestre temporali (VRP-TW)
- VRP con veicoli dotati di capacità (CVRP)
- VRP con prelievi e/o consegne (VRP-PD)
- VRP con flotta mista, ossia veicoli eterogenei (MF-VRP)
- VRP a due livelli, o *two-echelon* (2E-VRP)

É anche possibile avere problemi con aspetti relativi a più varianti contemporaneamente. Il caso studiato in questa tesi, nella sua versione più completa e complessa è un **MD-MP-2E-CVRP-TW**, quindi sono presenti più depositi, più prodotti da distribuire, due livelli di distribuzione (con strutture intermedie tra stabilimenti e clienti), veicoli con capacità fissa e infine finestre temporali entro il quale distribuire i prodotti.
Il *Vehicle Routing Problem* è un problema la cui complessità è di tipo NP-hard, come provato dalla pubblicazione di Lenstra e Rinnooy Kan nel 1981 [6], ossia è un problema difficile non deterministico la cui risoluzione avviene in tempi polinomiali. Viene inoltre provato che i metodi di risoluzione esatti, riescono a trovare soluzioni solo entro i 150 nodi [6]. L'elevata complessità di questa famiglia di problemi li porta ad avere un insieme di soluzioni ammissibili limitato, con alcune di queste che potrebbero essere calcolate in tempi non umanamente possibili.
Come molti problemi di ottimizzazione combinatoria (nel nostro caso, programmazione lineare intera), il VRP presenta uno spazio delle soluzioni convesso, che porta i metodi di risoluzione ad avvicinarsi alla soluzione ottima, con la possibilità però di restare "bloccati" in una soluzione intermedia, nota come minimo locale, che potrebbe essere "accettabile" oppure lontana da quella ottima.
Per questo motivo, e anche per via del fatto che i casi reali di VRP sono anche più complessi di quelli teorici, si tendono a implementare delle metodologie euristiche o metaeuristiche per sfuggire ai minimi locali, che oggi rappresentano le soluzioni allo stato dell'arte. Alcune tra queste sono:

- Algoritmi Genetici, che si ispirano al concetto di selezione naturale
- *Guided Local Search*, solitamente considerata la migliore metaeuristica per i VRP
- *Tabu Search*
- Algoritmi delle colonie di formiche o *Ant Colony Optimization*, ispirati dal comportamento delle formiche
- *Simulated Annealing*, tecnica probabilistica per approssimare l'ottimo di una funzione

Le tecniche metaeuristiche sono oggi preferite nel caso di applicazioni molto grandi con vincoli complessi. Nel caso in esame per questa tesi è stato usato il *Guided Local Search*. Nel prossimo capitolo verranno affrontati alcuni di questi temi in maniera più dettagliata.

Capitolo 3

Scopo della ricerca e strumenti utilizzati

L'obiettivo di questo capitolo è dare informazioni riguardo lo scopo della tesi, fornire un approccio più tecnico e dettagliato sui metodi e algoritmi usati e quali strumenti applicativi sono stati scelti per svolgere il lavoro.

3.1 Il caso in esame e i dati utilizzati

Come detto in precedenza, lo scopo della tesi è la realizzazione della SCN ottimizzata tramite l'applicazione di tecniche di clustering appartenenti al ML non supervisionato e l'impiego di risoluzione di problemi di VRP. Per la realizzazione dell'esperimento sono stati utilizzati dei dati che provengono da una azienda del settore automotive europeo, con alcune modifiche ad-hoc per avere dataset con informazioni coerenti.

Lo schema in figura 3.1 mostra in forma basilare lo schema per la realizzazione della SCN. Verranno realizzate quattro diverse versioni di questa soluzione, di complessità via via crescente e con caratteristiche aggiuntive, in modo da fornire una soluzione sempre più completa e prossima a un caso reale. Tutte seguiranno pressoché questo schema, con eventuali differenze documentate successivamente.

Tutte le soluzioni sono state realizzate in Python e testate sulla piattaforma Google Colab.

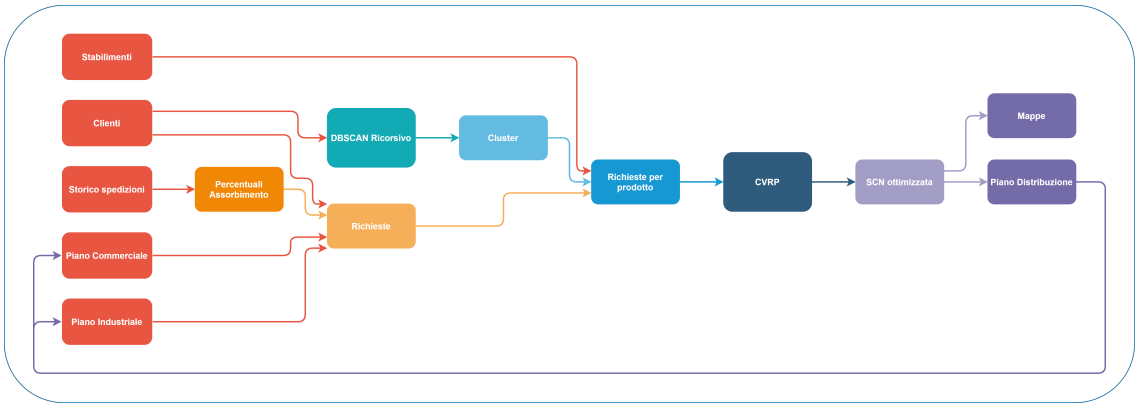

Figura 3.1. Schema base per realizzare la SCN ottimizzata

In questo schema i dati in input (in rosso in figura) vengono forniti dall'azienda che vuole realizzare la SCN. Essi sono:

- **Stabilimenti**: lista degli stabilimenti dell'azienda, comprensivi di coordinate geografiche e zona di mercato.
- **Clienti**: lista dei clienti dell'azienda, comprensivi di coordinate geografiche e zona di mercato. Rappresentano i punti vendita o concessionarie.
- **Storico delle spedizioni**: documentazione riguardo le spedizioni passate.
- **Piano commerciale**: documento che definisce i piani di vendita, ossia quanti veicoli di un certo tipo si vogliono vendere in un certo mercato, per un dato orizzonte temporale. Rappresentano dati di previsione.
- **Piano industriale**: documento che definisce i dati di produzione, ossia quanti veicoli di un certo tipo produce ogni stabilimento, per un dato orizzonte temporale. Rappresentano dati di previsione.

A partire dallo storico delle spedizioni, tramite analisi dei dati, si ricavano le **percentuali di assorbimento**, che stabiliscono quanti veicoli devono essere distribuiti per ogni zona di mercato dentro il mercato di destinazione, sulla base della percentuale di volume di produzione assorbito.
Occorre far notare che i concetti di zona di mercato e mercato di destinazione sono diversi. All'interno di un certo mercato di destinazione possono esserci infatti più zone, che identificano e coprono una certa area. Ad esempio nel

mercato italiano si potrebbe stabilire una zona per ogni provincia che supera una certa superficie in km^2.

Per il caso in esame è stato considerato più comodo realizzare delle percentuali di assorbimento ad-hoc, quindi viene omessa da ognuna delle quattro soluzioni l'elaborazione delle percentuali di assorbimento a partire dallo storico delle spedizioni. È comunque possibile implementare la realizzazione delle percentuali di assorbimento a partire dallo storico spedizioni analizzando quest'ultimo e ricavando, sulla base delle zone di mercato dove si trovano i punti vendita e sulla base dei volumi di produzione di un dato prodotto, qual è la percentuale di volume prodotto che viene distribuita in una certa zona (esempio: la zona di mercato X assorbe il 12% della produzione totale del veicolo Y).

I dati di clienti, percentuali di assorbimento e i piani commerciale e industriali vengono a questo punto utilizzati per realizzare le **Richieste**, che raccolgono tutte le richieste di distribuzione stabilite dal piano commerciale divise per ogni zona all'interno del relativo mercato secondo le percentuali di assorbimento. Le zone vengono stabilite dai clienti.
Al contempo i dati relativi ai clienti vengono dati come input a uno dei due "moduli" cardine di questo schema, ossia quello che realizza il Clustering tramite il **DBSCAN ricorsivo**, che verrà approfondito nel paragrafo 3.2. In uscita da questo modulo abbiamo i **Cluster**, ossia i clienti raggruppati geograficamente e ottimizzati in modo da non avere gruppi troppo densi.
Una volta ottenuti i cluster, questi vengono usati insieme a richieste e stabilimenti per realizzare le **Richieste per prodotto**, in modo da raggruppare all'interno dello stesso cluster le richieste appartenenti allo stesso prodotto. Queste richieste vengono passate al secondo modulo cardine dello schema, ossia quello che risolve il problema di **CVRP** (in realtà una sua variante complessa), approfondito nel paragrafo 2.3. In questo modulo, e in tutte le varianti che sono state realizzate, verrà utilizzata la piattaforma **Google OR-Tools**.
In uscita si avrà finalmente la **SCN ottimizzata**, che contiene al suo interno il set migliore di rotte per distribuire tutti i prodotti, per tutti gli stabilimenti a tutti i clienti. Essa rappresenta la soluzione ottima per lo scenario di distribuzione. Ci si aspetta che, una volta immessi in un contesto reale, più ci si avvicini alla soluzione ottima, migliori saranno i risultati complessivi. La SCN ottimizzata verrà utilizzata per due scopi. Il primo è realizzare delle **Mappe** che mostrano dettagliatamente le tratte dei veicoli (una mappa per ogni stabilimento). Il secondo è la realizzazione del **Piano distribuzione**,

che rappresenta un documento che definisce i volumi di auto da distribuire da ogni stabilimento distinti per prodotto (comprensivo di marca, modello e allestimento), modalità di trasporto e mercato di destinazione, il tutto secondo un certo orizzonte temporale, che può essere mensile o annuale.

Ogni riga nel piano distribuzione possiede le seguenti informazioni:

- Codice Mercato
- Zona Mercato
- Veicolo (Marca, Modello, Allestimento)
- Stabilimento
- Quantità Assorbita

Occorre infatti notare che l'intera SCN realizzata è relativa a un certo orizzonte temporale, ossia potrebbe essere utilizzata per realizzare il modo migliore di distribuire i prodotti per il mese N+1 sulla base dei dati in input relativi al mese N, tra i quali sono presenti anche i piani di previsione per il mese N+1. Questo documento, che quindi rappresenta un dato di previsione, è di vitale importanza perché verrà idealmente utilizzato per la formulazione dei piani commerciali e industriali successivi.

3.2 Il DBSCAN e la sua applicazione in modo ricorsivo

Il DBSCAN, acronimo di *Density-based spatial clustering of applications with noise* è uno degli algoritmi di clustering più famosi e comuni utilizzati nell'ambito del ML non supervisionato. Pur essendo stato proposto nel 1996 [7] continua a essere oggi uno dei più citati nelle pubblicazioni scientifiche, tra le quali questo lavoro di tesi.
In generale l'obiettivo del clustering è: dato un insieme di dati ignoti, si vuole separarli o raggrupparli in modo che gli oggetti considerati simili, secondo una certa misura di distanza, finiscano negli stessi gruppi e oggetti diversi finiscano in gruppi diversi. In questi raggruppamenti sono presenti oggetti che non è possibile collocare da nessuna parte. Questi sono gli *Outlier* o rumore. Il clustering è in grado di evidenziare sia gli oggetti simili che anche il rumore. Il tipo di analisi che viene effettuata è quindi di tipo descrittivo.

Lo stesso processo viene realizzato dal DBSCAN che, dato un set di nodi rappresentati come punti in uno spazio, cerca di raggruppare i nodi vicini in

uno stesso cluster, marcando come rumore (da qui il *noise* nel nome) quei nodi che si trovano troppo lontani dal loro vicino più prossimo. Nella seguente figura è rappresentata la visualizzazione dell'output di un DBSCAN.

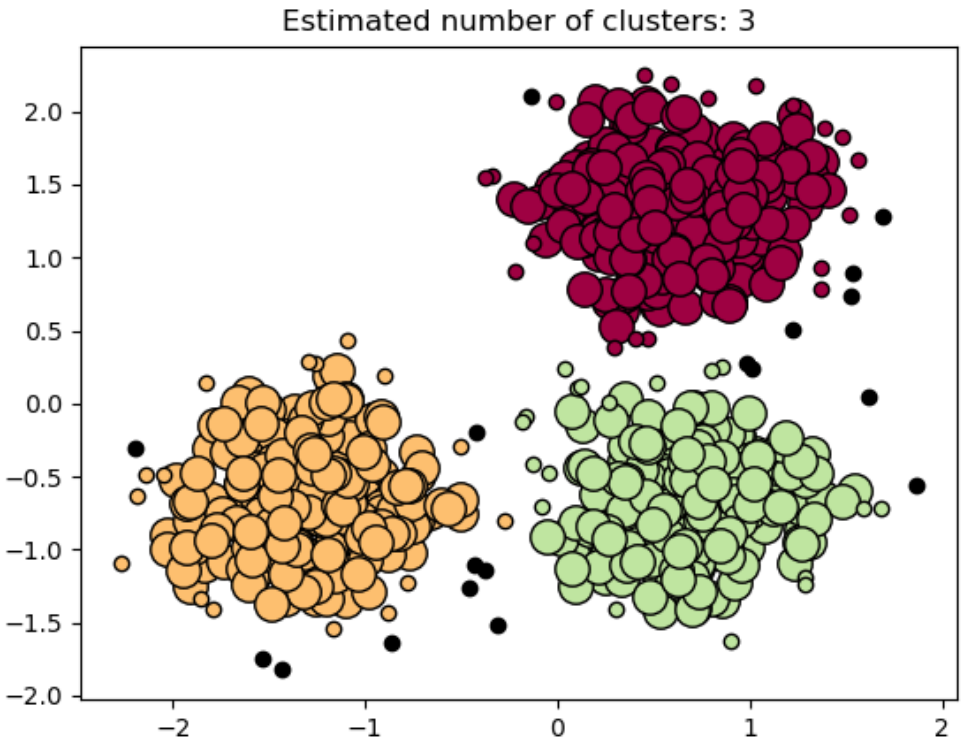

Figura 3.2. Esempio di clustering con DBSCAN su un dataset. I punti in nero rappresentano il rumore. Fonte: scikit-learn

Il suo successo è dovuto principalmente al fatto che:

- Richiede solo due parametri: il numero minimo di nodi richiesto per realizzare un cluster ($minPts$) ed ε (eps), che rappresenta la distanza massima tra due nodi per considerarli nello stesso "vicinato". Due nodi sono considerati vicini se la loro distanza è minore o uguale a ε. Quest'ultimo è il parametro più importante del DBSCAN.

- A differenza di un altro famoso algoritmo di clustering, il k-means, non richiede il numero di cluster da realizzare a priori.

- Può realizzare cluster dalla forma arbitraria, quindi cluster separabili non-lineari, impossibili da ottenere con il k-means.

Sulla base dei due parametri, $minPts$ ed ε, vengono classificati i punti in un dataset come segue:

- ***Core point***: se sono presenti almeno $minPts$ punti (incluso il punto in esame) entro un raggio ε.
- ***Border point***: se è raggiungibile da un *core point* e sono presenti meno di $minPts$ punti entro un raggio ε.
- ***Outlier* o rumore**: se non è un *core point* e non è raggiungibile da nessun *core point*.

Con raggiungibilità di un punto si intende l'essere nel vicinato di un *core point*, quindi essere a distanza minore o uguale di ε da esso.
Affinché due punti si possano definire raggiungibili è necessario scegliere la misura di distanza. Laddove solitamente viene scelta la distanza euclidea, nel caso in esame è necessaria una misura diversa.
Visto che i punti nel dataset rappresentano latitudini e longitudini, viene scelta la distanza **ortodromica** (che rappresenta la linea più breve per congiungere due punti su una sfera), che viene calcolata tramite la **formula dell'emisenoverso**, che calcola la distanza tra due punti a partire dalle loro latitudini e longitudini. Dati due punti P_1 e P_2, la distanza secondo la formula dell'emisenoverso è data da:

$$d(P_1, P_2) = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

dove P_1 ha come latitudine e longitudine la coppia (φ_1, λ_1), P_2 ha come latitudine e longitudine la coppia (φ_2, λ_2) e r rappresenta il raggio medio della Terra (6371.0088 km).

Tornando al DBSCAN, esso possiede però uno svantaggio, ossia che tutti i cluster devono avere lo stesso raggio, il cui valore che dipende da ε. Questo porta ad avere nella stessa applicazione dell'algoritmo cluster della stessa dimensione ma che possiedono al loro interno troppi nodi o pochissimi nodi. Facendo un esempio pratico si supponga di avere nella stessa soluzione un cluster con 1000 nodi e uno con solo 2 nodi. Questa divisione in cluster, soprattutto se applicata a un VRP, risulta essere inefficiente.

Per avere una suddivisione dei clienti in cluster efficiente in modo da poterlo applicare a un VRP, occorre una soluzione che permette di realizzare cluster di raggio diverso e dove non è presente una grande disparità nella densità dei cluster, in modo da evitare cluster con troppi nodi rispetto ai restanti.
Il DBSCAN ricorsivo (cui ci si riferirà successivamente come **RDBSCAN**) è una soluzione di clustering atta a risolvere i CVRP proposta nel 2018 dal collettivo composto da K.Bujel, F. Lai, M. Szczecinski, W. So e M. Fernandez [8], che riesce proprio in questo intento. Essa sfrutta come applicativo atto alla risoluzione del VRP la piattaforma Google OR-Tools, che è la stessa usata in questo studio.
Realizzato per risolvere un CVRP-TW (ossia un VRP con limiti di capacità dei veicoli e finestre temporali) esso realizza un primo clustering con DBSCAN cercando il raggio che massimizza il numero medio di nodi nei cluster e successivamente per ogni cluster con numero di nodi sopra una soglia stabilita come parametro, si applica ricorsivamente il DBSCAN in modo da "rompere" questi cluster in gruppi di cluster più bilanciati, ognuno con un numero di nodi sotto la soglia prestabilita.
Questa soluzione realizza inoltre cluster con raggio differente, portando a cluster con raggio più grande in regioni meno dense e cluster con raggio più piccolo in regioni più dense in modo che contengano un numero di nodi simile ma con raggio diverso.

Per quanto concerne le prestazioni, la pubblicazione ha messo a paragone le seguenti soluzioni:

- Google OR-Tools senza clustering
- DBSCAN + Google OR-Tools
- RDBSCAN + Google OR-Tools

L'applicazione dell'RDBSCAN ha portato a un modello in grado di lavorare con molti più nodi rispetto alle altre due versioni, con una riduzione del 61% dei tempi di esecuzione. Questo avviene perché la divisione più efficiente in cluster realizzata da RDBSCAN limita il numero di soluzioni da analizzare per OR-Tools, riducendo quindi il tempo di esecuzione. Un grafico che rappresenta quanto detto è presente in figura 3.3. Questa soluzione consuma inoltre meno memoria rispetto a soluzioni senza clustering e con DBSCAN "standard".

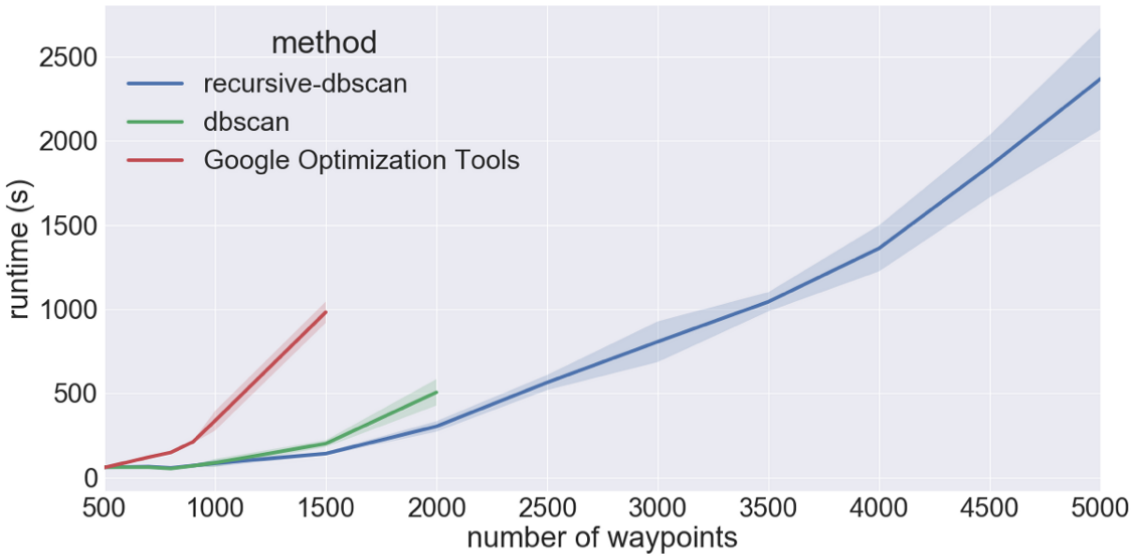

Figure 2: Plot of runtimes for Route Optimization methods

Figura 3.3. Grafico che mostra le differenze nei tempi di esecuzione per le tre soluzioni al variare di numero di nodi. Fonte: [8]

Per quanto concerne il calcolo delle distanze totali, le soluzioni trovate usando RDBSCAN risultano peggiori del 3.6% rispetto alle soluzioni ottenute usando solo OR-Tools, mentre il DBSCAN standard risulta molto inefficiente con distanze peggiori del 120% in più rispetto al solo OR-Tools. Allo stesso modo si nota per RDBSCAN un utilizzo del 6.4% in più di veicoli rispetto alle soluzioni trovate con il solo utilizzo di OR-Tools.

Questo peggioramento però è soltanto apparente, in quanto dovuto al fatto che RDBSCAN non permette ai veicoli di essere assegnati a ordini che appartengono a cluster differenti, laddove invece OR-Tools pur di ottimizzare le distanze tratta tutti i clienti come appartenenti allo stesso gruppo, anche se la distanza che li separa è molto grande. Questo può portare a consegne fatte da un solo veicolo verso clienti separati da distanze molto grandi secondo uno schema che in un contesto reale non risulta ottimale. Le soluzioni con RDBSCAN sono quindi molto più efficienti per situazioni prossime a casi reali.

I risultati sono visibili nella tabella sottostante, proveniente dalla pubblicazione originale.

Metric	OR-tools	Recursive-DBSCAN	DBSCAN
Total Distance	**0.0%**	+3.6%	+120%
Runtime	0.0%	**-61%**	-57%
#busy vehicles	**0.0%**	+6.4%	+150%

Figura 3.4. Fonte: [8]

Per quanto concerne le performance, infine, sia OR-Tools che il DBSCAN standard portano al crash del programma dovuto a utilizzo di RAM eccessivo per soluzioni con numero di clienti maggiore a 2000, laddove RDBSCAN permette di trovare soluzioni con un numero di clienti fino a 5000.

La pubblicazione non fornisce una implementazione ufficiale dell'RDBSCAN, che verrà invece realizzata e allegata a questo lavoro di tesi (visionabile in Appendice A).

L'applicazione dell'RDBSCAN nelle quattro soluzioni realizzate nei capitoli successivi, ha bisogno di una serie di parametri stabiliti a priori, che sono:

- Dimensione massima del cluster
- Dimensione minima del cluster
- Numero minimo di cluster
- Raggio minimo (espresso in metri)
- Raggio massimo (espresso in metri)

La particolarità di alcuni di questi parametri è che possono essere definiti da un esperto di dominio, come ad esempio le dimensioni massime e minime dei cluster e il loro raggio massimo di estensione.

3.3 L'approccio al CVRP

Il tipo di problema comune alle quattro versioni affrontate nei capitoli successivi è un **MD-MP-CVRP**, acronimo di *Multi-Depot, Multi-Product, Capacitated Vehicle Routing Problem*, che verrà affrontato nel corso di questo paragrafo. La formulazione matematica del problema è la seguente.

Si consideri $G(V, E)$ un grafo orientato (o Digrafo), con $V = D \cup C$, dove:

- $|D|$ rappresenta il numero di stabilimenti o depositi, con $D = \{1, 2, ..., |D|\}$
- $|C|$ rappresenta il numero di clienti o punti vendita, con $C = \{|D| + 1, |D| + 2, ..., |D| + |C|\}$
- $E = \{(i, j)/i, j \in E, i \neq j\}$ rappresenta l'insieme di archi che corrispondono ai collegamenti tra le coppie di nodi.

Si stabilisce inoltre che:

- Ogni collegamento è caratterizzato da una distanza $d_{i,j}$ non negativa tra il nodo i-esimo e il nodo j-esimo.
- Ogni cliente o punto vendita $c \in C$ possiede un insieme non nullo di richieste relative a uno o più prodotti $dm_{i,p}$, dove i rappresenta l'i-esimo cliente e p rappresenta il prodotto.
- $P = \{1, 2, ..., |P|\}$ rappresenta l'insieme dei prodotti distribuiti.
- I prodotti vengono realizzati in uno o più stabilimenti $d \in D$.
- Per la distribuzione dei prodotti ai clienti viene sfruttata una flotta di $M = \{1, 2, ..., |M|\}$ veicoli.
- Ogni veicolo $m \in M$ possiede una portata o capacità massima Q_i con $i = \{1, ..., M\}$.

L'obiettivo del problema è minimizzare la distanza complessiva percorsa soddisfando tutte le richieste dei clienti, partendo e tornando ai depositi.

Si considerino inoltre le seguenti assunzioni per la risoluzione del problema:

- Ogni veicolo che parte da uno stabilimento deve terminare la sua tratta nello stesso stabilimento.

- La richiesta totale di una tratta non deve eccedere la capacità del veicolo.
- Ogni richiesta relativa a un prodotto di un dato cliente viene servita da uno o più veicoli a seconda della capacità di questi ultimi.

Passando dalla formulazione teorico-matematica al caso pratico, occorre fare delle precisazioni riguardo:

- Cosa viene usato per realizzare la soluzione.
- Il tipo di veicoli.
- La realizzazione delle richieste per capacità.
- Come viene messo in pratica l'aspetto multi-prodotto del problema.

Per quel che riguarda il primo punto, la risoluzione di questi problemi è affidata alla piattaforma open-source Google OR-Tools, una suite di software dedicata ai problemi di ottimizzazione combinatoria tipici della ricerca operativa.

Per quanto concerne il tipo di veicoli utilizzato, essi rappresentano una delle limitazioni imposte in questo studio. In un caso di SCN reale ci si aspetta di distribuire i prodotti secondo più mezzi di trasporto che possono viaggiare su strada, ferrovia, via aria e via mare, ognuno con diverse capacità di trasporto. Poiché implementare una soluzione del genere richiederebbe tipologie di collegamenti diversi e tratte specifiche per ogni mezzo di trasporto (ad esempio solo rotte via mare per mezzi navali), è stata scelta come tipologia di flotta per questo lavoro di tesi solo quella che viaggia su strada.
È comunque possibile realizzare una soluzione che comprende flotta eterogenea (via strada, ferrovia e mare) con collegamenti specifici, ma occorrerebbe scomporre il problema realizzando una soluzione per ogni tipo di flotta (quindi un VRP per la distribuzione su strada, uno per la distribuzione via mare, ecc.), che aumenta la complessità del problema generale.
Nelle quattro versioni realizzate nei capitoli successivi a questo i veicoli utilizzati sono bisarche, autotreni a due piani per il trasporto su strada delle auto dallo stabilimento al punto vendita, dove è stata considerata una capacità di trasporto di 10 macchine per bisarca.

Per quanto concerne il terzo punto, le richieste di una certa auto che superano in numero la capacità di una bisarca vengono divise sulla base della suddetta capacità. Facendo un esempio pratico e supponendo di avere un

cliente con una richiesta di 43 auto (uguali) verranno realizzate 4 richieste da 10 auto (la capacità della bisarca) più una richiesta da 3 auto. Quest'ultima richiesta potrà essere associata a un'altra consegna (fino a un massimo di altri 7 veicoli) provenienti da un'altra destinazione nello stesso cluster.

Infine, approfondendo il quarto punto, è necessario far notare che la piattaforma Google OR-Tools non è in grado di risolvere i VRP multi-prodotto, che è invece proprio una delle particolarità delle soluzioni proposte. Per risolvere ciò, una volta calcolate le richieste (tramite i dati di clienti, percentuali di assorbimento e i piani commerciale e industriale), si calcolano per ogni cluster le richieste divise per prodotto. Questo permette di dividere un VRP per la distribuzione di N prodotti di natura diversa in N VRP, uno per ogni prodotto da distribuire. Questo approccio migliora di molto la scalabilità della soluzione e l'uso delle risorse a tempo di esecuzione da parte di OR-Tools.

Capitolo 4

Le soluzioni proposte

4.1 "Versione 0": Risoluzione del MD-MP-CVRP senza clustering

Questa versione è stata realizzata a scopo esemplificativo per porre un confronto qualitativo con le versioni che sfruttano RDBSCAN. Ciononostante, le parti che sono in comune con le versioni successive verranno introdotte in questo paragrafo.

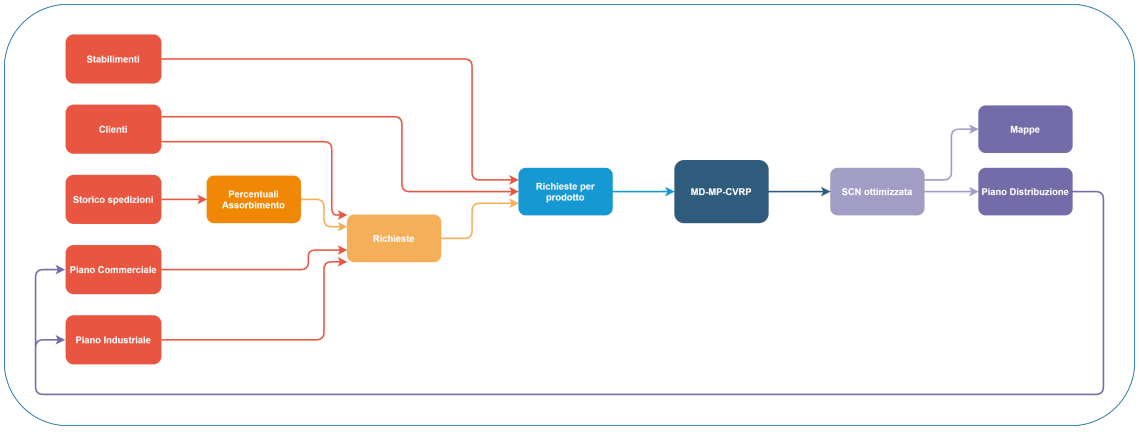

Figura 4.1. Schema riassuntivo della versione 0

I dati in input (relativi a stabilimenti, clienti, storico spedizioni e piani commerciale e industriale), in rosso nello schema sopra, vengono forniti tramite file in formato .xlsx (Excel), mentre i dati in output saranno un piano

distribuzione in formato .xlsx e una serie di mappe di distribuzione in formato .html, una per ogni stabilimento presente.
Occorre ricordare che per il caso in esame sono state realizzate delle percentuali di assorbimento ad-hoc, quindi viene omessa da ogni versione l'elaborazione delle percentuali di assorbimento a partire dallo storico delle spedizioni. Le percentuali di assorbimento vengono fornite come file in formato .xlsx.
La versione 0, come tutte le altre versioni, è stata realizzata in Python ed eseguita sulla piattaforma Google Colab, che permette l'*hosting* dello script Python su una macchina che offre 12.72 GB di RAM.
In modo da permetterne l'esecuzione sono stati utilizzati i seguenti pacchetti:

- **numpy** e **scipy**: necessari per quasi ogni elaborazione in ambito scientifico su Python
- **pandas**: per lavorare su dati strutturati, come file .xlsx
- **haversine**: per sfruttare la formula dell'emisenoverso
- **ortools**: per la risoluzione del VRP
- **gmplot**: per la visualizzazione su mappe di Google Maps

A partire dai dati relativi a Clienti, Percentuali di assorbimento e i piani commerciale e industriale vengono realizzate le richieste, che sono nella forma:

$$((Q_1, P_1), [lat_1, long_1]), ((Q_2, P_2), [lat_2, long_2]), ..., ((Q_n, P_n), [lat_n, long_n])$$

dove:

- Q_i e P_i rappresentano le unità richieste e il relativo prodotto.
- lat_i e $long_i$ rappresentano latitudine e longitudine.

Si supponga ad esempio la richiesta:

$$((9, 12), [40.499, -3.673]), ((32, 14), [40.499, -3.673]), ((3, 24), [42.6, -5.7333])$$

Qui si nota che i primi due punti possiedono lo stesso valore di latitudine e longitudine, questo perché il cliente a quella coordinata fa una richiesta per due tipi di prodotti. In questo esempio il primo cliente [40.499, -3.673] vuole 9 unità del prodotto 12 e 32 unità del prodotto 14, mentre il secondo cliente [42.6, -5.7333] desidera 3 unità del prodotto 24. Queste richieste

vengono suddivise in primo luogo per prodotto, associando ad esse uno o più stabilimenti di produzione, che verranno usati come depositi quando passati in input a OR-Tools, ottenendo:

(9,12), prodotto dallo stabilimento S_1

(32,14), prodotto dallo stabilimento S_2

(3,24), prodotto dallo stabilimento S_3

A questo punto, prima dell'inizializzazione di OR-Tools, vengono divise le richieste sulla base della capacità dei veicoli, che nel nostro caso è 10. Si avranno quindi tante richieste a capacità massima pari al multiplo della capacità dei veicoli più prossimo alla richiesta, più una richiesta con la rimanenza. Per far sì che OR-Tools possa distribuire questi prodotti, viene inoltre "replicato" il cliente un numero di volte pari alle richieste divise per capacità. Le richieste diventeranno quindi:

(9,12), prodotto dallo stabilimento S_1

(10,14), (10,14), (10,14), (2,14), prodotto dallo stabilimento S_2

(3,24), prodotto dallo stabilimento S_3

In questo caso, il cliente che richiede 32 unità del prodotto 14 verrà trattato da OR-Tools come 4 clienti con richieste 10, 10, 10 e 2 rispettivamente, posizionati alle stesse coordinate geografiche. Ci si aspetta che per questo cliente verranno allocati 4 veicoli solo per la distribuzione del 14-esimo prodotto.

Per quanto concerne l'allocazione dei veicoli, questi vengono inizializzati sulla base di un parametro definito dall'utente e associati al particolare set di richieste, e non allo stabilimento in sé. Nel caso in esame vengono allocati 150 veicoli per set di richieste. Questo porta inoltre a stabilire che nessuno di questi set deve superare le 1500 unità di veicoli (150 veicoli * 10 unità trasportabili) altrimenti OR-Tools non riuscirà a trovare una soluzione.

NOTA: È stato scelto un valore alto per questo parametro poiché verranno servite le richieste di tutti i clienti nello scenario di distribuzione contemporaneamente. Questo valore scenderà nelle versioni successive una volta inserito il clustering.

Allocati i veicoli, si replicano gli stabilimenti un numero di volte pari al numero di prodotti, in modo da permettere a OR-Tools di gestire più veicoli che partono dallo stesso stabilimento. Questo è necessario altrimenti per la natura del VRP, partirà sempre un solo veicolo che dovrà soddisfare tutte le richieste.
A questo punto tramite stabilimenti replicati e clienti nella richiesta si realizza, con le loro coordinate geografiche, la **matrice delle distanze**, che documenta in forma matriciale la distanza tra ogni coppia di nodi. Anche in questi casi viene usata la formula dell'emisenoverso per calcolare una distanza che rappresenta pressocché quella in linea d'aria tra ogni coppia di punti. Seguendo l'esempio, una volta divise le richieste e allocati i veicoli, si inizializza infine OR-Tools per risolvere 3 CVRP (o MD-CVRP nel caso in cui un prodotto venga realizzato in più stabilimenti). Poiché l'esempio rappresenta un caso banale, verranno sfruttati 6 veicoli per distribuire tutti i prodotti.

Dal risultato di OR-Tools si ottiene la distribuzione ottimale. Da questo vengono salvate le informazioni relative a tutte le tratte trovate, che verranno usate per la realizzazione del piano distribuzione e delle mappe html, discusse nel prossimo capitolo.
Nelle seguenti tabelle sono presenti i parametri utilizzati.

Tabella parametri generali:

Nome parametro	Valore
Numero di stabilimenti	12
Numero di clienti	60

Tabella parametri per Google OR-Tools:

Nome parametro	Valore
Capacità del veicolo	10
Numero di veicoli per set di richieste	150
Strategia per il calcolo della soluzione	PATH_CHEAPEST_ARC (*)
Tempo per il calcolo della soluzione	5 secondi
Metaeuristica	GUIDED_LOCAL_SEARCH

(*): La strategia PATH_CHEAPEST_ARC a partire dal nodo di partenza della tratta, lo connette al nodo che realizza il segmento con costo minore, per poi estendere la tratta iterando sull'ultimo nodo aggiunto.

4.2 Versione 1: Risoluzione del MD-MP-CVRP con RDBSCAN

In questo paragrafo viene affrontata la prima delle quattro versioni che sfruttano RDBSCAN realizzate in questa tesi. Lo schema riassuntivo è il seguente:

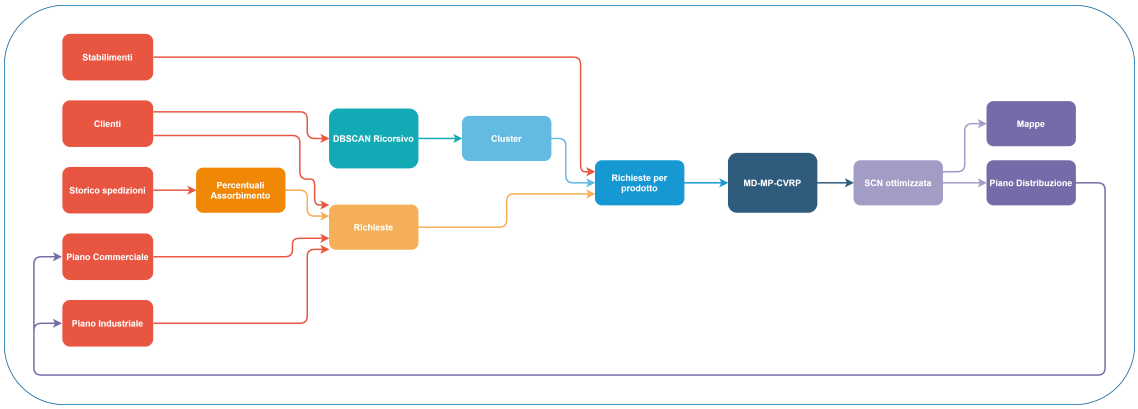

Figura 4.2. Schema riassuntivo della versione 1

Lo schema ricalca nella fase iniziale quanto già visto per la versione 0. Ai pacchetti utilizzati viene aggiunto il pacchetto **sklearn** per accedere all'implementazione del DBSCAN fornita da *scikit-learn*.

A partire da questa versione e per tutte quelle successive, viene applicato l'RDBSCAN al set di clienti, in modo da poterli suddividere in cluster.

Una rappresentazione in pseudocodice dell'RDBSCAN è la seguente:

```
function RDBSCAN(waypoints, minRadius, maxRadius,
 minClusterSize, maxClusterSize, minNumberOfClusters):
    bestClusters = [] ;
    bestAverageClusterSize = 0 ;
    meanEarthRadius = 6371008.8 // raggio medio della Terra
     in metri ;
    finalClusters = [] ;
    while minRadius < maxRadius do
        radius = (minRadius + maxRadius) / 2;
        eps = radius / meanEarthRadius ;
        clusters = DBSCAN(eps, algoritm = 'ball_tree', metric =
         'haversine', min_samples = minClusterSize, waypoints) ;
        numberOfClusters = length(clusters) ;
        if numberOfClusters < minNumberOfClusters then
            maxRadius = radius -1 ;
        else
            minRadius = radius +1 ;
            if averageClusterSize > bestAverageClusterSize then
                bestAverageClusterSize = averageClusterSize ;
                bestClusters = clusters ;
            end
        end
    end
    if lenght(bestClusters) == 0 then
        return 'NO SOLUTION FOUND' ;
    end
    for cluster ∈ bestClusters do
        if lenght(cluster) > maxClusterSize then
            // chiamata ricorsiva ;
            newClusters = RDBSCAN(cluster, minRadius, maxRadius,
             minClusterSize, maxClusterSize, minNumberOfClusters) ;
            finalClusters = finalClusters ∪ newClusters ;
        else
            finalClusters = finalClusters ∪ cluster ;
        end
    end
    return finalClusters ;
```

Algorithm 1: Pseudocodice del RDBSCAN

Ottenuti i cluster tramite RDBSCAN viene realizzato in output un primo grafico 2D, visibile nei due esempi in figura 4.3 e 4.4, con la distribuzione dei nodi nei relativi cluster. Per mostrare in maniera chiara l'appartenenza dei punti ai cluster è stato scelto di rappresentare i cluster con un solo nodo all'interno di una piccola area circolare, i cluster con due nodi racchiusi in un'area rettangolare arrotondata in prossimità dei nodi e per i restanti cluster una rappresentazione poligonale che sfrutta la funzione *ConvexHull* appartenente al package **SciPy**.

Occorre notare che ogni punto nei grafici sottostanti rappresenta una coordinata geografica. Infatti l'asse x rappresenta il valore della latitudine e l'asse y quello relativo alla longitudine.

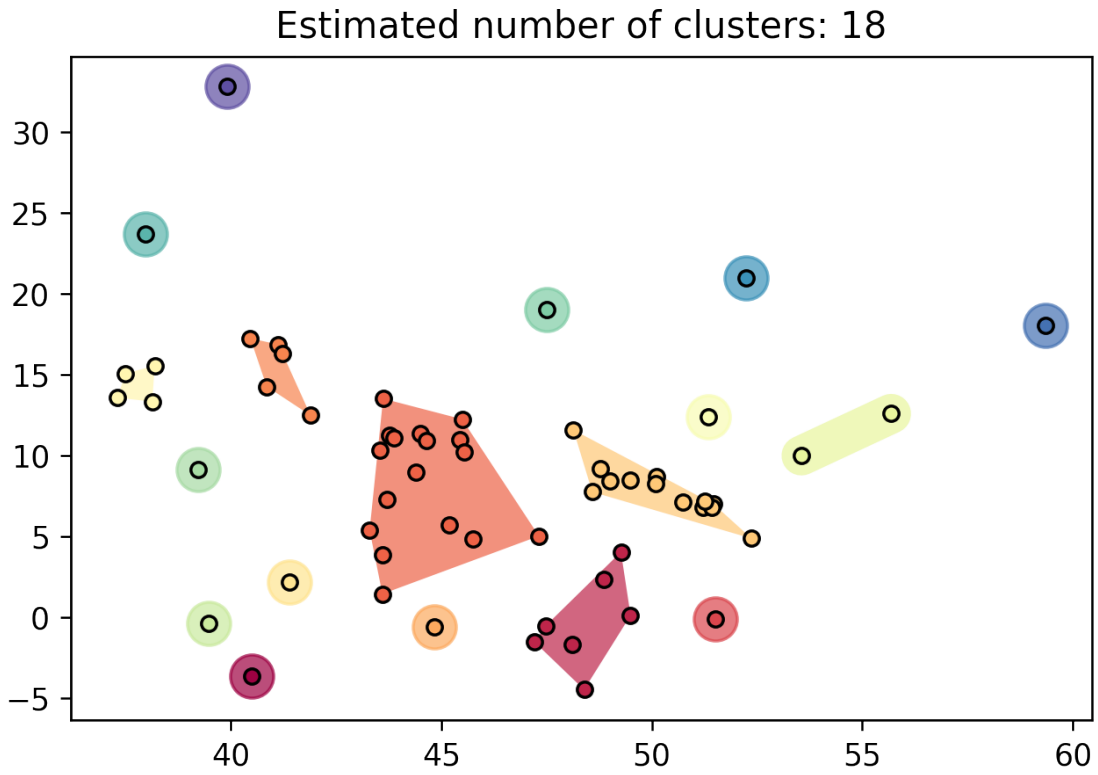

Figura 4.3. Esempio di output grafico dell'RDBSCAN, con maxClusterSize = 20 e maxRadius = 300km

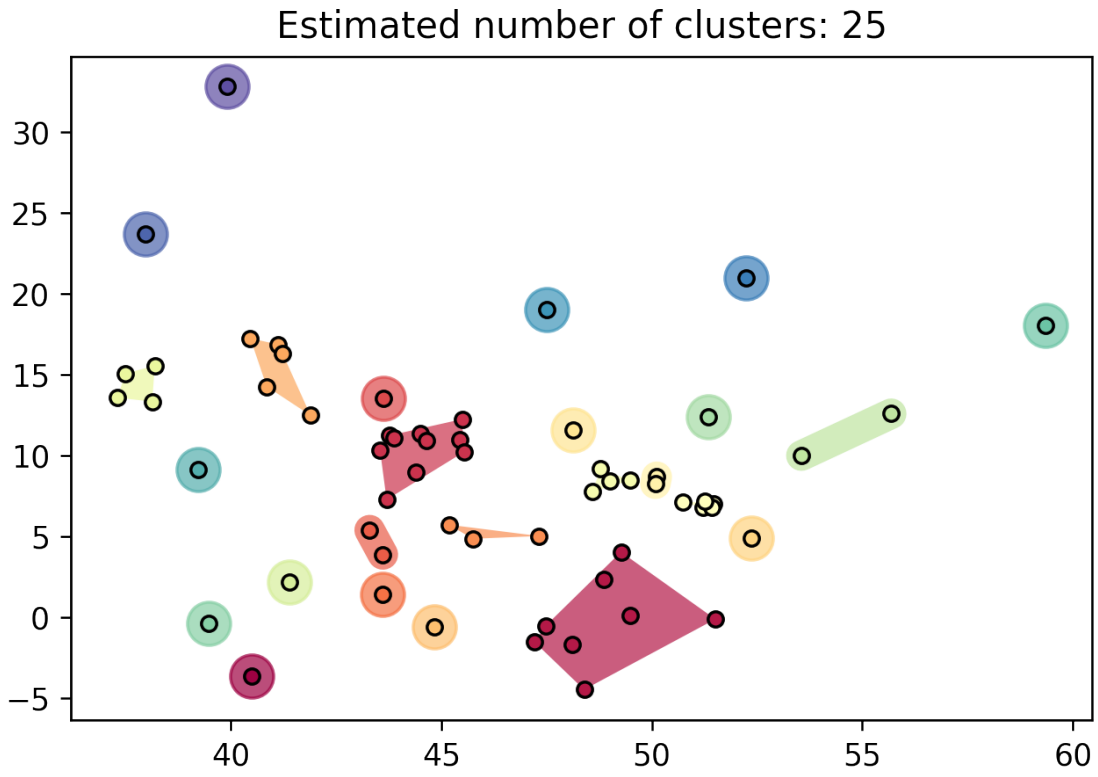

Figura 4.4. Esempio di output grafico dell'RDBSCAN, con maxClusterSize = 10 e maxRadius = 300km

Subito dopo questa fase si lavorerà con le richieste relative a un dato cluster che, come per il paragrafo precedente, saranno nella forma:

$((Q_1, P_1), [lat_1, long_1]), ((Q_2, P_2), [lat_2, long_2]), ..., ((Q_n, P_n), [lat_n, long_n])$
dove:

- Q_i e P_i rappresentano le unità richieste e il relativo prodotto.
- lat_i e $long_i$ rappresentano latitudine e longitudine.

Da questo punto in poi sia lo schema che il tipo di elaborazioni effettuate ricalca quanto visto nella versione 0, ottenendo a seguito della risoluzione del CVRP la SCN ottimizzata, da cui vengono ricavati il piano distribuzione e le mappe html.

Una particolarità presente nelle versioni realizzate in questo lavoro è il supporto all'**esecuzione parallela** nella risoluzione dello scenario di distribuzione. Infatti è possibile ridurre notevolmente i tempi di esecuzione facendo risolvere a un set di thread (in condizioni ideali un thread per ogni cluster) una istanza di OR-Tools associata a un cluster, facendo sì che alla fine i risultati vengano raggruppati (o restituiti a un thread "padre") e con questi vengano realizzate mappe html e piano distribuzione.

I parametri utilizzati sono:

Tabella parametri generali:

Nome parametro	Valore
Numero di stabilimenti	12
Numero di clienti	60

Tabella parametri per RDBSCAN:

Nome parametro	Valore
Dimensione massima del cluster	20
Dimensione minima del cluster	1
Numero minimo di cluster	3
Raggio minimo (in metri)	1
Raggio massimo (in metri)	250000

Tabella parametri per Google OR-Tools:

Nome parametro	Valore
Capacità del veicolo	10
Numero di veicoli per set di richieste	60
Strategia per il calcolo della soluzione	PATH_CHEAPEST_ARC (*)
Tempo per il calcolo della soluzione	5 secondi
Metaeuristica	GUIDED_LOCAL_SEARCH

4.3 Versione 2: Miglioramento della versione 1 con struttura a due livelli (MD-MP-2E-CVRP)

Nella versione 2 viene modificato lo schema precedente realizzando quanto segue:

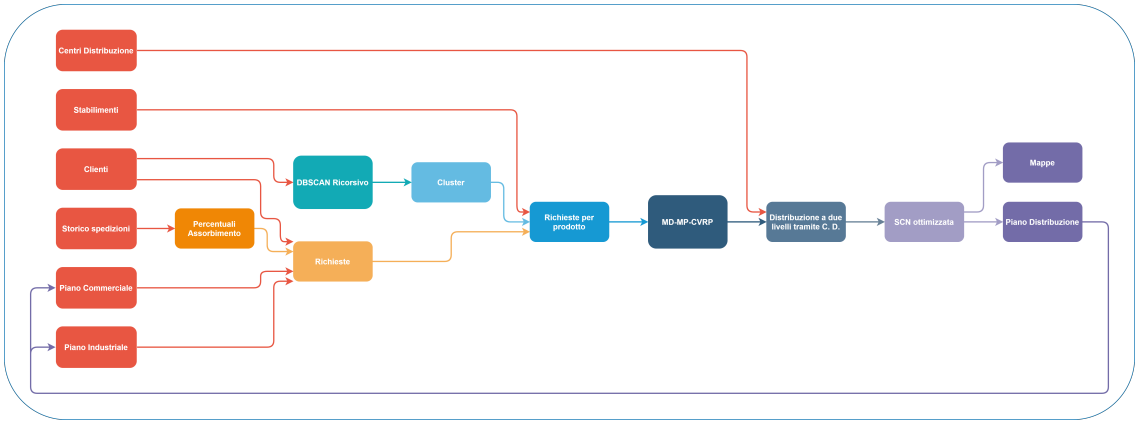

Figura 4.5. Schema riassuntivo della versione 2

La prima cosa che si nota è la presenza di un nuovo dato di input che rappresenta l'elenco dei **centri distribuzione**, ossia punti di stoccaggio, di snodo, o di rifornimento intermedi da cui passare una volta essere partiti da uno stabilimento prima del giro di consegne dei prodotti. Essi vengono passati come input tramite un file in formato .xlsx.
Il procedimento sfruttato è lo stesso della versione 1 ma la SCN ottimizzata non viene ottenuta col risultato dell'MD-MP-CVRP, bensì viene applicato a quest'ultimo una ulteriore elaborazione in modo da ottenere la SCN ottimizzata. La scelta del centro distribuzione corretto viene fatta tramite la seguente formula:

$$CD_{best} = cd \in CD / \min(d(p, cd_j) + d(cd_j, K))$$

dove:

- $p \in P$ rappresenta lo stabilimento che produce il prodotto di interesse.
- $cd_j \in CD$ rappresenta il generico centro distribuzione.

- K rappresenta la coordinata media dei clienti interessati al prodotto.

Fissato lo stabilimento che realizza il prodotto, per ogni set di clienti dentro un cluster interessato a questo, viene calcolata la coordinata media (o centroide), e viene scelto come centro distribuzione quello che minimizza la distanza tra stabilimento e centroide passando per il centro distribuzione stesso.
Ciò che si vuole realizzare con questa versione è quindi una struttura a due livelli (dall'inglese *two-echelon*) dove sono presenti due livelli di tratte, ossia quelle che vanno dagli stabilimenti ai centri distribuzione e quelle che vanno dai centri distribuzione ai set di clienti.
Imporre un vincolo del genere però potrebbe portare a realizzare tratte inefficienti. Per fare un esempio pratico si consideri una situazione in cui stabilimento e cliente sono geograficamente vicini, mentre il centro distribuzione si trova lontano da essi. Seguendo il vincolo così come è stato definito si realizza la tratta *stabilimento* → *centro distribuzione* → *cliente*, che risulta essere inefficiente sotto ogni punto di vista.
Per ovviare a questo problema è stato scelto di rilassare questo vincolo implementando un valore di soglia, definito tramite parametro passato dall'utente (o stabilito da un esperto di dominio), con il quale viene preferito il passaggio attraverso il centro distribuzione o il viaggio diretto dallo stabilimento al cliente. Una descrizione in pseudocodice del vincolo implementato è il seguente:

function *PassaggioTramiteCD(clienti, tratte, CD_{best}, soglia)*:
    stabilimento = tratte[0] `// primo elemento della tratta` ;
    for *tratta* ∈ *tratte* **do**
        distanzaDiretta = d(stabilimento, primoCliente);
        distanzaTramiteCD = d(stabilimento, CD_{best}, primoCliente);
        if *distanzaTramiteCD* < *distanzaDiretta* + *soglia* **then**
            `// Si preferisce il passaggio tramite centro distribuzione` ;
        else
            `// Si preferisce il viaggio diretto` ;
        end
    end

Algorithm 2: Pseudocodice per il passaggio attraverso i centri distribuzione

Occorre notare che in questo pseudocodice la funzione $d()$ per il calcolo della distanza rappresenta per questa versione la funzione dell'emisenoverso.

I parametri utilizzati, raggruppati nelle seguenti tabelle, sono:

Tabella parametri generali:

Nome parametro	Valore
Numero di stabilimenti	12
Numero di centri distribuzione	21
Numero di clienti	60

Tabella parametri per RDBSCAN:

Nome parametro	Valore
Dimensione massima del cluster	20
Dimensione minima del cluster	1
Numero minimo di cluster	3
Raggio minimo (in metri)	1
Raggio massimo (in metri)	250000

Tabella parametri per Google OR-Tools:

Nome parametro	Valore
Capacità del veicolo	10
Numero di veicoli per set di richieste	51
Soglia per il passaggio dal centro distribuzione	100 (in km)
Strategia per il calcolo della soluzione	PATH_CHEAPEST_ARC
Tempo per il calcolo della soluzione	5 secondi
Metaeuristica	GUIDED_LOCAL_SEARCH

4.4 Versione 3: Miglioramento della versione 2 con utilizzo di API Google

Lo schema della versione 2 viene modificato nel seguente modo:

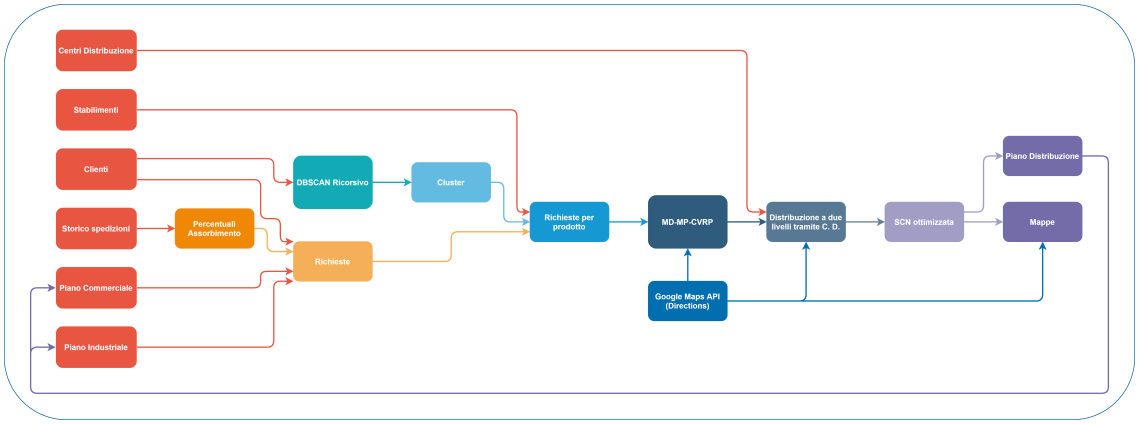

Figura 4.6. Schema riassuntivo della versione 3

Sebbene non vi siano modifiche per quanto riguarda il tipo di elaborazioni effettuate per ottenere la SCN ottimizzata, si nota per la versione 3 l'implementazione delle API di Google Maps, fornite dalla Google Cloud Platform. Più in particolare è stato sfruttato il servizio *Directions* per il calcolo di ogni distanza, che quindi non sarà più quella in linea d'aria ottenuta tramite la formula dell'emisenoverso. Come si può vedere nello schema, le API di Google Maps vengono utilizzate nei moduli relativi a:

- **MD-MP-CVRP**, dove:

  La matrice delle distanze viene calcolata sulla base della distanza su strada tra ogni coppia di nodi

- **Distribuzione 2E tramite centri distribuzione**, dove:

  Il calcolo del centro distribuzione migliore e l'eventuale passaggio tramite esso sfrutta la distanza su strada

- **Mappe**, dove:

  Vengono realizzate delle tratte reali per ogni mappa di distribuzione

L'unica distanza che continua a sfruttare la formula dell'emisenoverso resta quella legata all'RDBSCAN, che divide i clienti sulla base delle loro coordinate geografiche e non sulla base della distanza su strada che li separa.

Per sfruttare *Directions*, come ogni altro servizio non gratuito di Google, è necessaria una sottoscrizione alle risorse Cloud di Google e una chiave per accedere ad esse. Una particolarità del servizio Directions è la presenza di una versione "avanzata", chiamata *Advanced Directions*, utile per fornire, a costo maggiore, ulteriori informazioni, avvertimenti e rotte migliori in caso di traffico o strade bloccate. Seguendo l'obiettivo di realizzare un *Proof of Concept*, non è stato scelto di usare la versione avanzata, ma bensì quella "standard".

Occorre comunque notare che la versione avanzata risulterebbe utile in tutte le sezioni che sfruttano le API Google e soprattutto la sezione mappe. Come miglioramento del tutto teorico del programma presentato, si potrebbero ad esempio calcolare le mappe tramite un modello a richiesta, in modo da averle sempre aggiornate in tempo reale. Questo porterebbe alla gestione di una prima categoria di eventi esogeni per la *Supply Chain* come quelli relativi a traffico o congestioni stradali, o dissesti naturali che rendono non percorribili certi tratti stradali.

I parametri utilizzati sono raggruppati nelle seguenti tabelle, e sono:

Tabella parametri generali:

Nome parametro	Valore
Numero di stabilimenti	12
Numero di centri distribuzione	21
Numero di clienti	60

Tabella parametri per RDBSCAN:

Nome parametro	Valore
Dimensione massima del cluster	20
Dimensione minima del cluster	1
Numero minimo di cluster	3
Raggio minimo (in metri)	1
Raggio massimo (in metri)	250000

Tabella parametri per Google OR-Tools:

Nome parametro	Valore
Capacità del veicolo	10
Numero di veicoli per set di richieste	51
Soglia per il passaggio dal centro distribuzione	25 (in km)
Strategia per il calcolo della soluzione	PATH_CHEAPEST_ARC
Tempo per il calcolo della soluzione	5 secondi
Metaeuristica	GUIDED_LOCAL_SEARCH

4.5 Versione 4: Miglioramento della versione 3 con eventi esterni e finestre temporali (MD-MP-2E-CVRP-TW)

La versione 4 rappresenta l'ultima variante del programma realizzato. Lo schema viene modificato come segue:

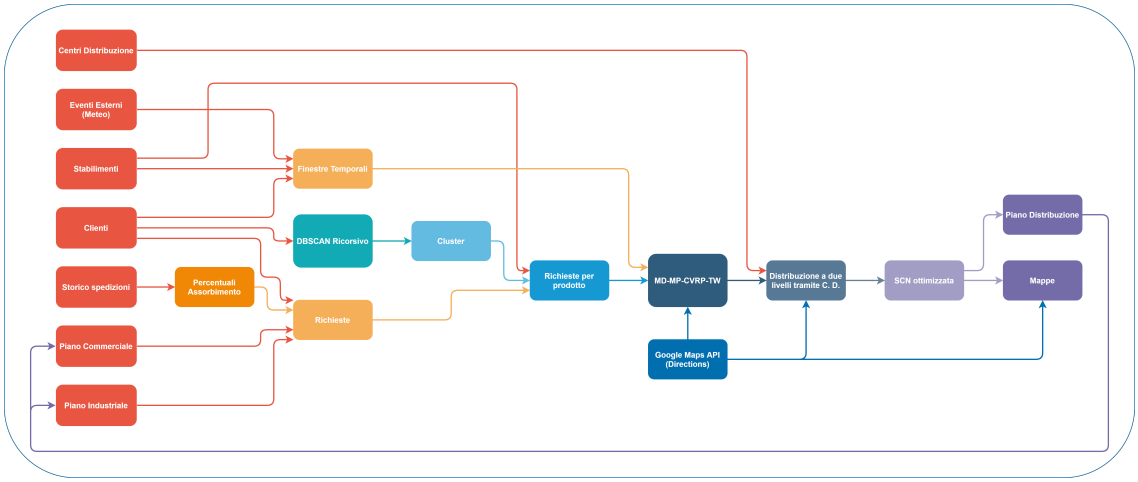

Figura 4.7. Schema riassuntivo della versione 4

Un primo aspetto da notare è la presenza di un nuovo dato di input rappresentato dagli **eventi esterni**. Il loro inserimento tra i dati iniziali permette alla SCN di gestire e reagire agli eventi esogeni alla Supply Chain discussi nel paragrafo 2.3. Eventi di questo tipo appartengono a quella famiglia di eventi che non sono predicibili nel processo di realizzazione e distribuzione di un prodotto e che possono variare dalle condizioni meteo a dissesti di origine naturale, fino a scioperi dei lavoratori o instabilità civile o politica.
Per la realizzazione di questo *Proof of Concept* si è scelto di gestire un solo tipo di evento esterno, con l'idea di applicare una logica "incrementale" in modo da permettere il monitoraggio e la gestione di eventi esterni di natura diversa, per far sì che la SCN possa rispondere prontamente e correttamente a ogni situazione.
Per ricevere le informazioni necessarie alla gestione di questi eventi è necessaria una fonte esterna, che può essere un particolare *feed* oppure un servizio offerto all'azienda.

Andando nel dettaglio, per questo lavoro di tesi vengono gestiti i soli eventi legati alle **condizioni meteo**. Si è scelto di utilizzare i servizi offerti dal collettivo *OpenWeather*, un team di esperti di IT e data scientist che dal 2014 forniscono tramite una piattaforma open-source, dati meteo storici, attuali e di previsione tramite una API [9], in modo simile a come Google offre il servizio *Directions* introdotto con la versione 3.
I dati ottenuti, provenienti eventualmente da fonti eterogenee, vengono processati e analizzati per realizzare le **finestre temporali**, che rappresentano il secondo aspetto da notare nello schema, la cui realizzazione e funzionamento verrà descritta con un esempio nel corso di questo paragrafo.
Le finestre temporali verranno passate poi in input al CVRP, che diventa un MD-MP-CVRP-TW. Inserendo i vincoli imposti dalle finestre temporali, la versione 4 sposta l'obiettivo principale dell'ottimizzazione offerta da OR-Tools (e definita nel paragrafo 3.3) dalla minimizzazione della distanza percorsa alla **minimizzazione dei tempi di viaggio**, realizzando tutte le consegne all'interno delle finestre temporali. In questo scenario la matrice delle distanze diventa ora una matrice dei tempi di percorrenza.
L'obiettivo finale del programma resta sempre la realizzazione del piano distribuzione, che stavolta ha un **orizzonte temporale** fissato a una 8 giorni (in modo coerente all'orizzonte temporale fornito dal servizio meteo). È stato scelto un tale intervallo di tempo poiché non ha senso calcolare previsioni meteo per intervalli di tempo superiori, dove l'inaccuratezza della previsione diventa elevata.

La procedura per la realizzazione delle finestre temporali, e come queste vengono utilizzate da OR-Tools, è la seguente:

In prima istanza si interroga tramite la API il servizio *OpenWeather* e si ottengono le previsioni meteo relative al gruppo di 8 giorni per ogni luogo che appartiene all'elenco dei clienti (i punti vendita) e agli stabilimenti di produzione. Ogni previsione relativa a un luogo viene fornita su scala giornaliera.

A partire da questi risultati vengono filtrati solo quelli che sono associati a quelle condizioni meteo considerate avverse. Supponendo di mappare i giorni con i numeri nell'intervallo [0, 7] (estremi inclusi), si consideri di aver ottenuto i seguenti risultati per un certo luogo X:

$(2, Temporale), (4, Neve), (5, Neve)$
dove 2, 4 e 5 corrispondono alle giornate di Mercoledì, Venerdì e Sabato.

Occore notare come il tipo di condizione avversa può essere stabilita da un esperto di dominio, in modo da stabilire secondo una politica aziendale quale condizione meteo è da considerare avversa o no. I codici [10] relativi alle condizioni avverse utilizzate nella versione 4 sono visibili nelle tabelle dei parametri a fine paragrafo.

A questo punto si convertono gli intervalli relativi ai giorni in secondi e si associano ai relativi luoghi. Seguendo i dati dell'esempio e considerando il valore di secondi in un giorno (86400), si ottiene:

$[X, [(172.800, 259.200), (345.600, 432.000), (432.000, 518.400)]]$

Si arriva quindi alla creazione dell'istanza di OR-Tools, a cui viene associata una finestra temporale complessiva di 8 giorni convertita in secondi, quindi l'intervallo [0, 691200].

Prima della risoluzione del modello di distribuzione però, vengono recuperate tutte le finestre temporali relative a condizioni meteo avverse nei luoghi presenti nel cluster e vengono rimosse dalla finestra temporale usata.

Seguendo i dati dell'esempio si ottiene per il luogo X una finestra temporale di consegna dei prodotti che evita negli 8 giorni in esame le giornate di Mercoledì, Venerdì e Sabato per problemi legati al meteo.

In un contesto generale si ottiene in risultato un istanza di MD-MP-CVRP-TW che cerca di realizzare un modello di distribuzione per servire tutti i clienti nei cluster evitando le finestre temporali in cui sono presenti condizioni meteo avverse. Occorre notare come la gestione degli eventi esterni non va a dipendere dalla natura stessa dell'evento, poiché questi vengono convertiti in generici intervalli di tempo da evitare nella finestra di consegna dei prodotti. Questo porta alla logica "incrementale" discussa prima, poiché permette la gestione di eventi di natura diversa trattandoli allo stesso modo.

Andando oltre i nuovi aspetti presenti in questa versione, il resto del procedimento per ottenere la SCN ottimizzata è lo stesso della versione 3.

I parametri utilizzati sono raggruppati nelle seguenti tabelle, e sono:

Tabella parametri generali:

Nome parametro	Valore
Numero di stabilimenti	12
Numero di centri distribuzione	21
Numero di clienti	60

Tabella codici condizioni avverse OpenWeather:

Codice	Descrizione
2xx	Temporale
3xx (esclusi 300, 301, 310)	Pioviggine
5xx (escluso 500)	Pioggia
6xx	Neve
7xx	Condizioni atmosferiche

Tabella parametri per RDBSCAN:

Nome parametro	Valore
Dimensione massima del cluster	20
Dimensione minima del cluster	1
Numero minimo di cluster	3
Raggio minimo (in metri)	1
Raggio massimo (in metri)	250000

Tabella parametri per Google OR-Tools:

Nome parametro	Valore
Capacità del veicolo	10
Numero di veicoli per set di richieste	51
Soglia per il passaggio dal centro distribuzione	100 (in km)
Strategia per il calcolo della soluzione	PATH_CHEAPEST_ARC
Tempo per il calcolo della soluzione	5 secondi
Metaeuristica	GUIDED_LOCAL_SEARCH

Capitolo 5

I risultati

5.1 Risultati per la "versione 0"

Dal risultato dell'esecuzione di OR-Tools si ottiene la distribuzione ottimale secondo i parametri passati, visti nelle tabelle del paragrafo 4.1. Ciò che viene mostrato nella finestra di esecuzione di Google Colab è:

```
Total distance:  1138634m, number of travels:  614
Total load distributed:  5865
Average load per vehicle:  9.552, over a maximum of:  10
Load efficiency:  95.52%
Maximum load in a route:  1027
Optimal number of vehicles per request:  103
Distribution plan created
```

Le prime due stringhe stabiliscono che, secondo questo schema di distribuzione, è stata percorsa una distanza totale di 1138.634km con 614 viaggi, per distribuire tutti i prodotti, pari a 5865 unità. La terza e la quarta stringa stabiliscono che i veicoli impiegati (nel nostro caso, le bisarche), hanno effettuato la distribuzione con un carico medio di 9.552 prodotti (nel nostro caso, automobili) su una capacità massima di 10, e una **efficienza di carico** del 95.52%.
Con efficienza di carico si definisce il rapporto tra la capacità di carico occupata durante la distribuzione e la capacità massima cumulativa della flotta. In questo caso è data dal rapporto tra 5865 e 6140 (614 veicoli con capacità di 10 unità).

La quinta e la sesta stringa propongono una possibile ottimizzazione sul numero di veicoli per set di richieste sulla base della tratta con carico maggiore nell'intera distribuzione. Questo valore nel caso relativo alla versione 0 è 103.

Occorre notare che in output viene usato il termine *"travels"* (viaggi) invece che *"vehicles"* (veicoli). Sebbene OR-Tools calcoli la soluzione supponendo che ogni veicolo parta allo stesso istante temporale da ogni stabilimento, questi in un contesto reale rappresentano invece il numero di tratte percorse per distribuire i prodotti secondo lo schema ottimizzato.

In uno scenario reale ci si aspetta inoltre che al rientro dei veicoli nei rispettivi stabilimenti, essi possano caricarsi nuovamente e distribuire altri set di prodotti su tratte non ancora avviate, servendo quindi un'altra richiesta.

A questo punto vengono realizzate una serie di mappe geografiche in formato .html, di cui due esempi sono mostrati sotto, in figura 5.1 e 5.2.

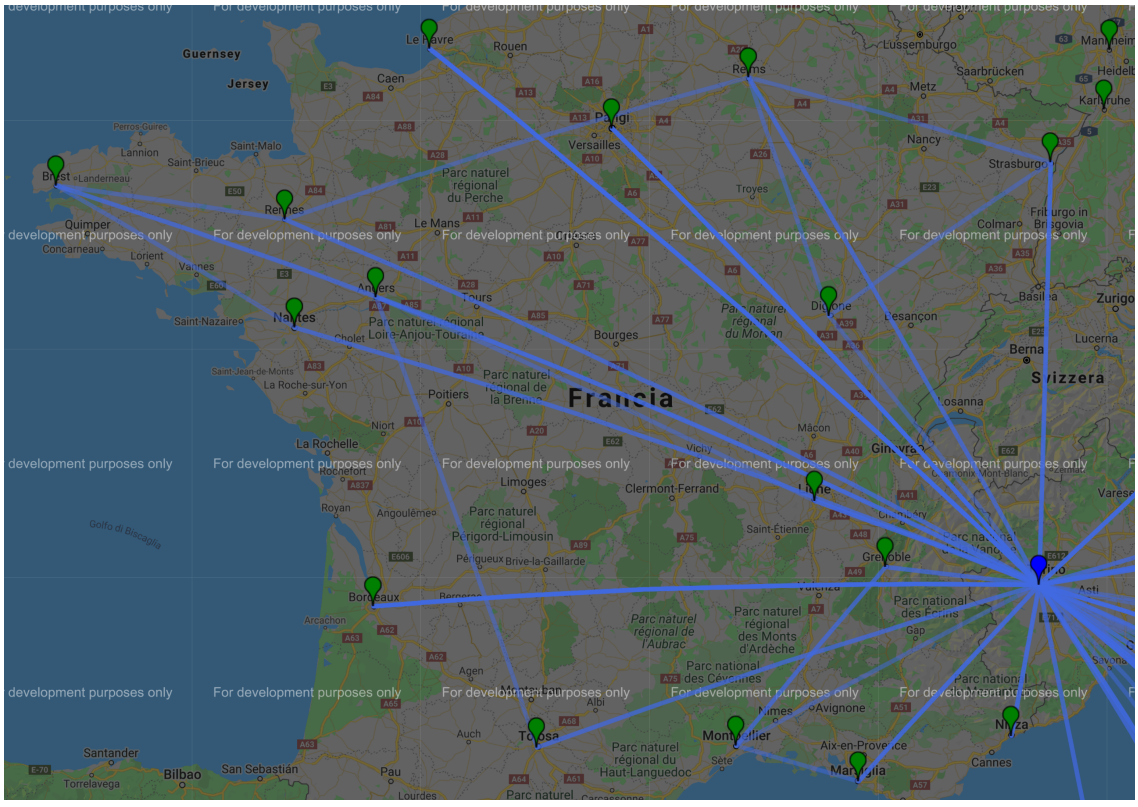

Figura 5.1. Scenario di distribuzione per la versione 0 #1

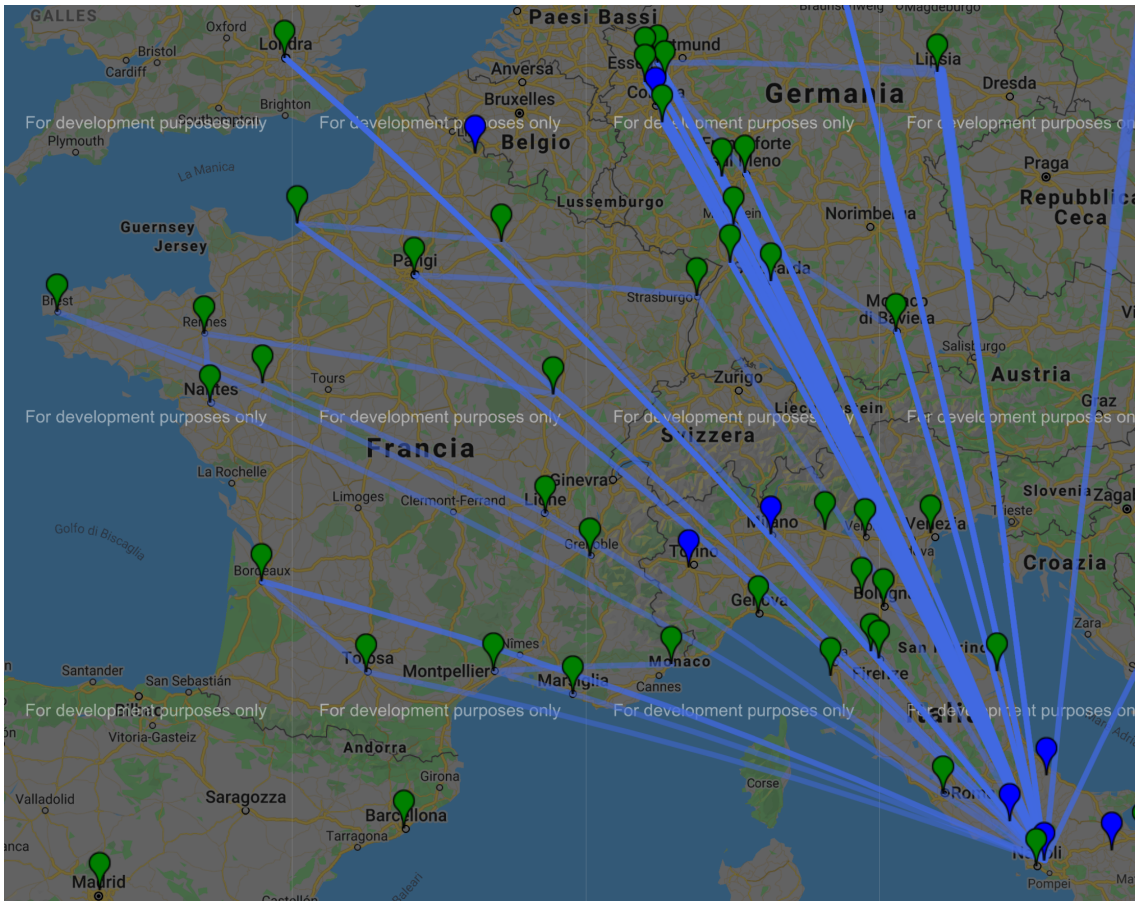

Figura 5.2. Scenario di distribuzione per la versione 0 #2

Queste mappe vengono realizzate sfruttando il pacchetto **gmplot**, ed evidenziano stabilimenti (punti blu scuro), clienti (punti verdi) e tutte le rotte che vanno dagli stabilimenti verso tutti i clienti che richiedono i prodotti. In ogni mappa l'ultimo segmento della tratta, ossia quella che dall'ultimo punto di consegna porta allo stabilimento di partenza viene omessa per facilitarne la leggibilità.
In queste si può notare che oltre alla presenza di tratte che trasportano a capacità massima una serie di prodotti direttamente al cliente, alcune di queste realizzano consegne in più punti.
L'ultima stringa in output notifica della creazione del file che documenta il **piano distribuzione**, realizzato in formato .xlsx. Esso stabilisce i volumi di prodotti da distribuire da ogni stabilimento distinti per prodotto (che nel nostro caso sono auto distinte per marca, modello e allestimento), modalità

di trasporto e mercato di destinazione. Il piano distribuzione viene inoltre realizzato secondo un certo orizzonte temporale, che può essere mensile o annuale. Per le versioni da 0 a 3 il tipo di orizzonte temporale è ininfluente.

Idealmente il piano distribuzione verrà utilizzato per realizzare i piani commerciale e industriale relativi a orizzonti temporali successivi, oltre a influire sulle future spedizioni, i cui dati verranno usati per realizzare le future percentuali di assorbimento.

5.2 Risultati per la versione 1

Prima che venga inizializzato OR-Tools, il primo risultato ottenuto è quello dell'RDBSCAN, che mostra la seguente suddivisione dei clienti in cluster:

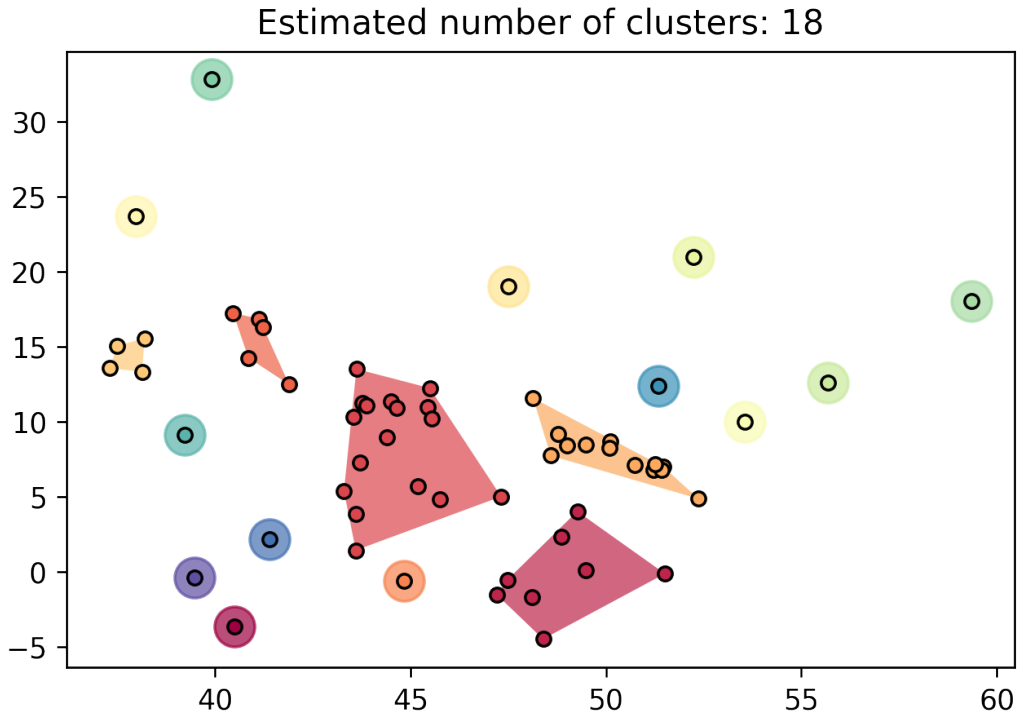

Figura 5.3. Risultato grafico dell'RDBSCAN

In questo grafico l'asse delle x e l'asse delle y rappresentano rispettivamente la latitudine la longitudine dei clienti nello scenario di distribuzione. Una volta terminata la simulazione dello scenario di distribuzione tramite OR-Tools, ciò che viene mostrato in output è:

```
Total distance:  1172521m, number of travels:  634
Total load distributed:  5865
Average load per vehicle:  9.251, over a maximum of:  10
Load efficiency:  92.51%
Maximum load in a route:  506
Optimal number of vehicles per request:  51
Distribution plan created
```

In questo schema di distribuzione ottimizzato si raggiunge una consegna di tutti i prodotti ai clienti sfruttando 634 viaggi, percorrendo una distanza totale di 1172.52 km, con una efficienza di carico del 92.51%.

Mettendo questi risultati a paragone con quelli della versione 0, che non sfrutta RDBSCAN, si nota come questi siano leggermente peggiori. Più nel dettaglio, sono stati impiegati il 3.257% di veicoli in più rispetto alla soluzione senza clustering e sono stati percorsi il 2.976% di metri in più. Il tutto con una efficienza di carico il 3.01% minore rispetto alla versione 0.

Ci si chiede il perché di queste prestazioni minori. La risposta è da ricondurre a quanto già scritto nel paragrafo 2 del capitolo 3. Secondo la pubblicazione dell'RDBSCAN infatti, esso non permette ai veicoli di essere assegnati a ordini che appartengono a cluster differenti, laddove OR-Tools ottimizza le distanze trattando tutti i clienti, anche se molto distanti secondo la mappa di distribuzione, come raggiungibili da uno stesso veicolo per consegnare prodotti.

Si consideri ad esempio lo scenario di distribuzione che appare nella mappa in figura 5.1, relativo alla versione 0. In esso si può notare una particolare rotta che realizza consegne di prodotti in circa 6 città nel territorio francese molto distanti fra di loro. In una situazione reale è improponibile chiedere a un solo veicolo di coprire tutte quelle distanze.

Inoltre la pubblicazione fa notare il limite massimo di nodi per ottenere una soluzione sfruttando soltanto OR-Tools, che è pari a 2000 nodi, mentre con l'utilizzo congiunto di OR-Tools e RDBSCAN questo limite aumenta fino a 5000 nodi, che è da favorire per situazioni prossime allla realtà di molte grandi aziende.

Per questi motivi, sebbene i risultati siano solo apparentemente leggermente peggiori rispetto alla versione senza clustering, le soluzioni con RDBSCAN sono molto più efficienti per simulazioni di casi di distribuzione reali.

In figura 5.4, 5.5 e 5.6 sono presenti tre esempi di mappe di distribuzione che sfruttano RDBSCAN.

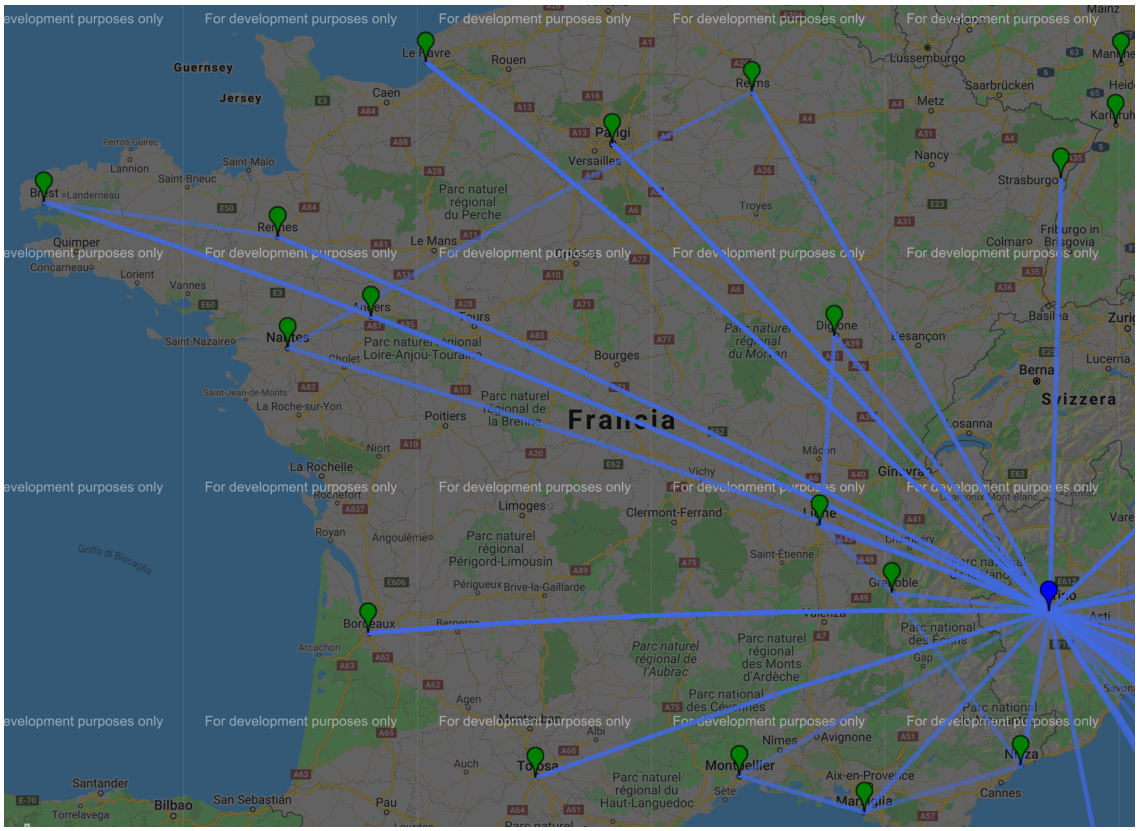

Figura 5.4. Scenario di distribuzione per la versione 1 #1

Mettendo a confronto i risultati della mappa in figura 5.4 con quelli in figura 5.1 si nota come l'utilizzo dell'RDBSCAN ha portato alla divisione del territorio francese in un set di 2 grandi cluster che vengono serviti separatamente per la realizzazione delle consegne.

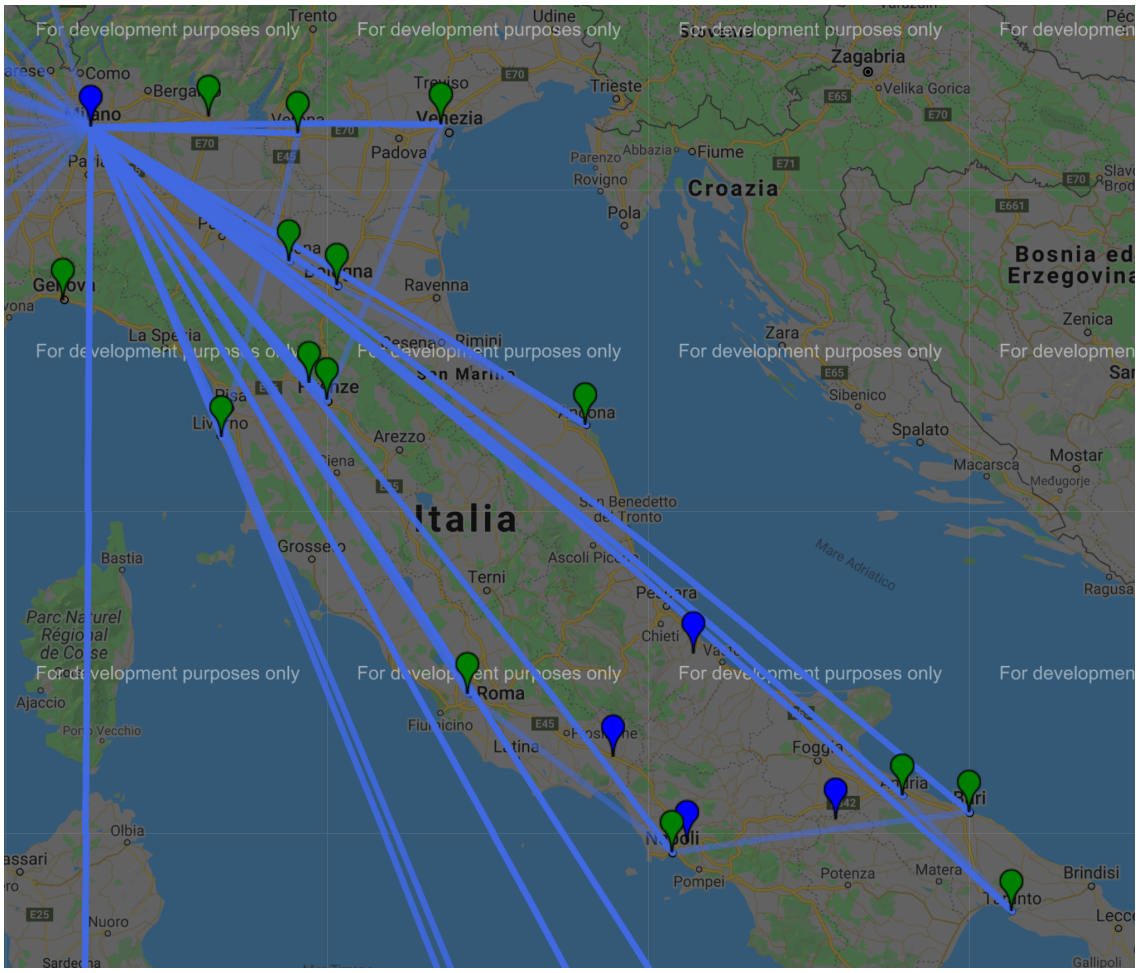

Figura 5.5. Scenario di distribuzione per la versione 1 #2

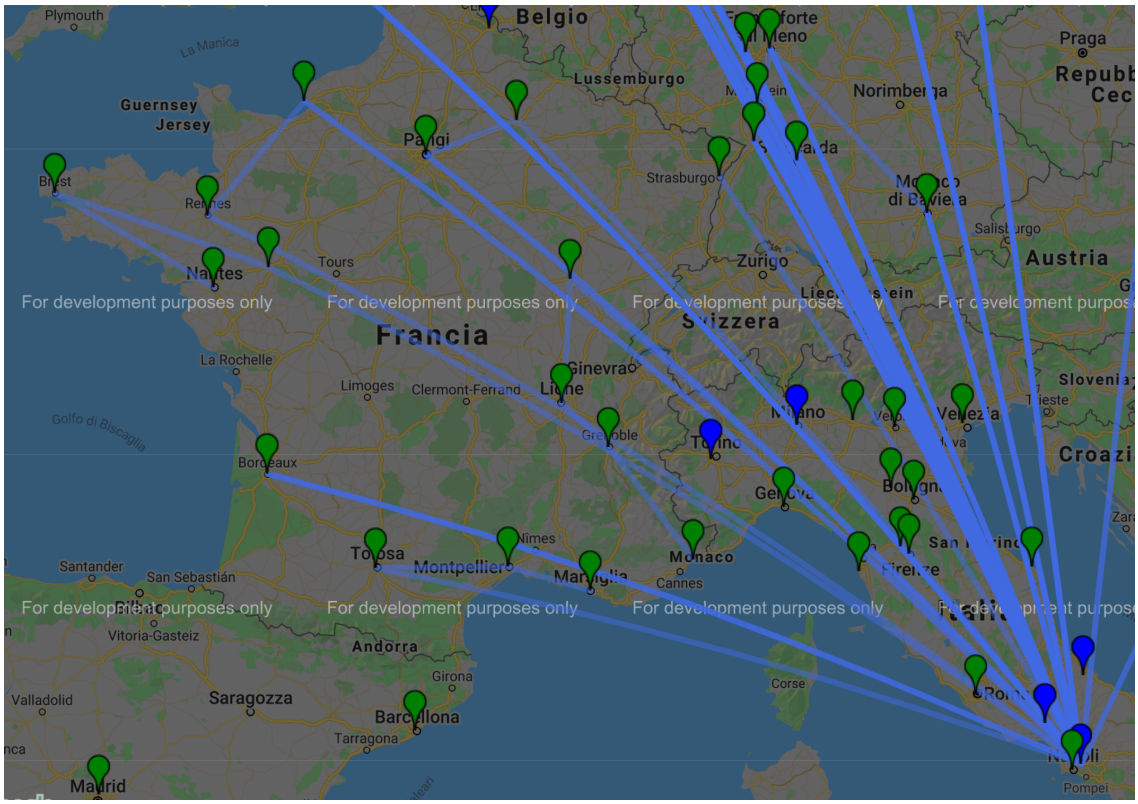

Figura 5.6. Scenario di distribuzione per la versione 1 #3

5.3 Risultati per la versione 2

Per quanto concerne questa versione la visualizzazione grafica del risultato dell'RDBSCAN è la stessa già vista in figura 3.4 (dovuto al fatto che i parametri non sono cambiati dalla versione precedente).
Non vi sono miglioramenti in termini di efficienza di carico poiché a partire dalla versione 1 è stato raggiunto il valore di efficienza di carico massimo per tutte le versioni con RDBSCAN che minimizzano la distanza percorsa, ossia il 92.51%, sfruttando 634 viaggi per distribuire un totale di 5865 prodotti. Solo la versione 4, che non minimizza la distanza percorsa, otterrà una efficienza maggiore, nel paragrafo 5.5.

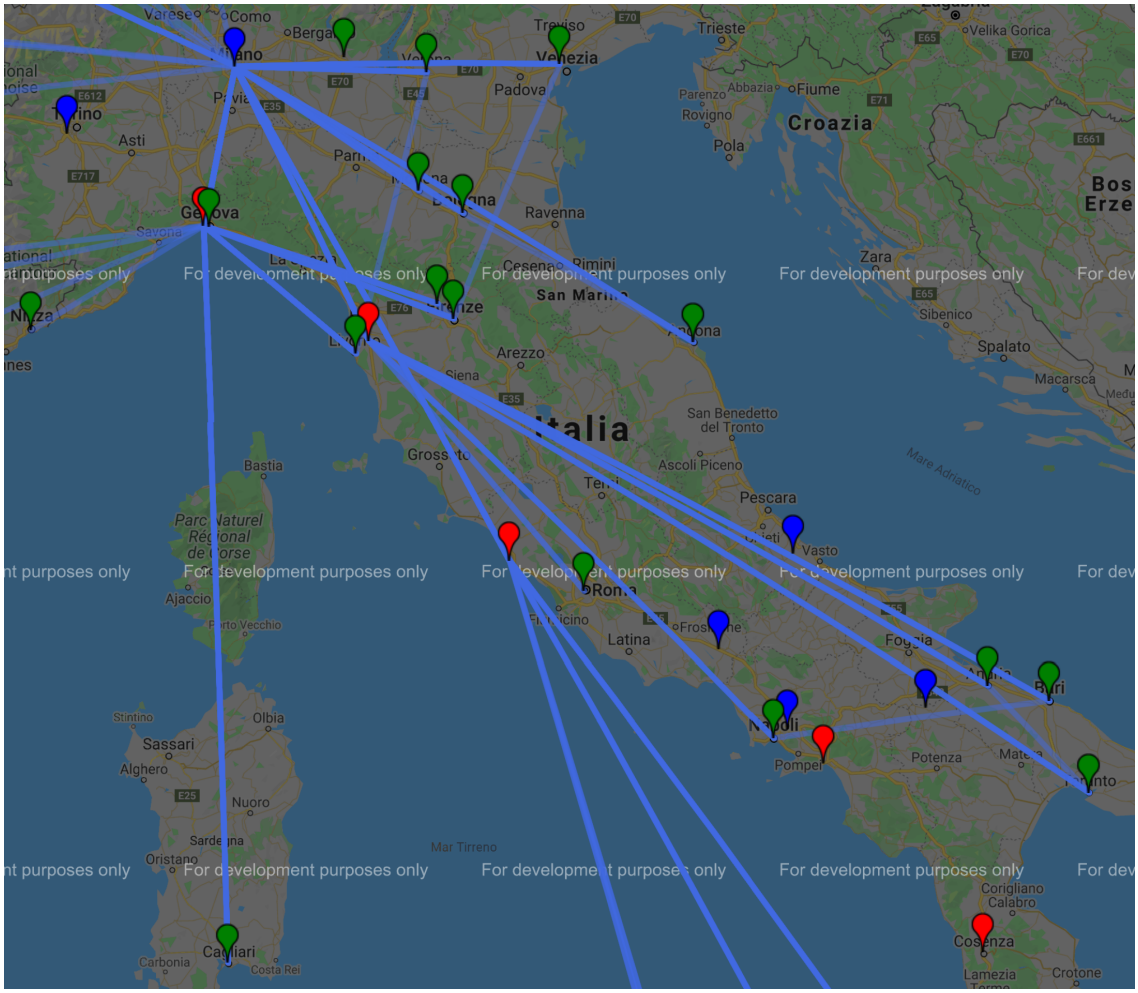

Figura 5.7. Scenario di distribuzione per la versione 2 #1

Osservando questa mappa di distribuzione si nota come la presenza e la scelta dei centri distribuzione può portare alla preferenza di un tipo di trasporto diverso da quello su strada per determinati segmenti di tratta.
Altre due mappe di distribuzione sono le seguenti:

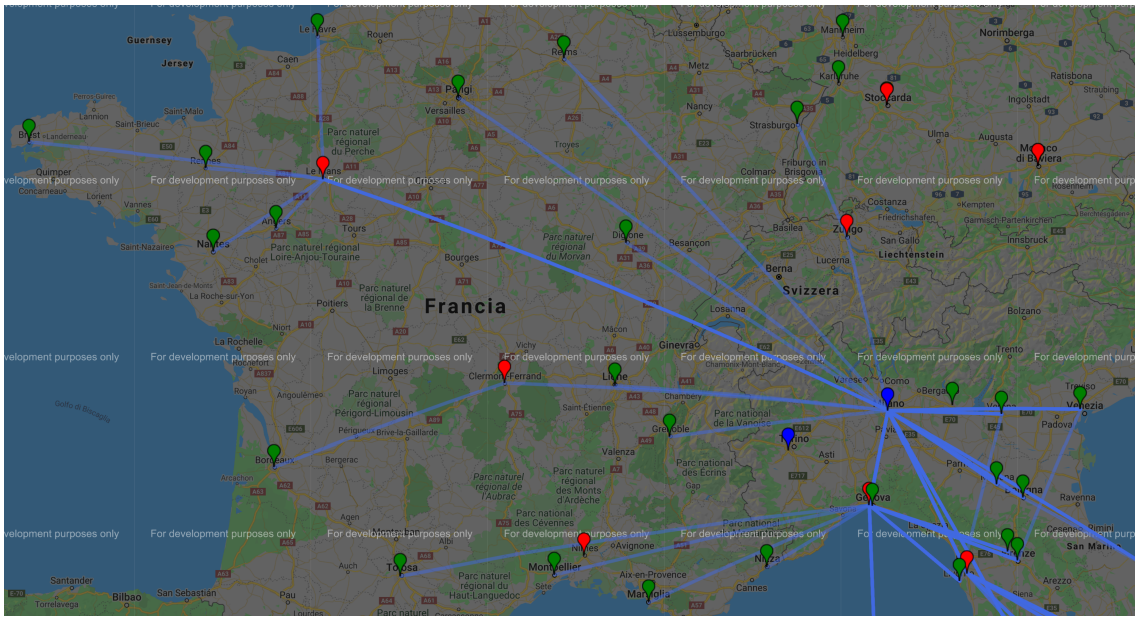

Figura 5.8. Scenario di distribuzione per la versione 2 #2

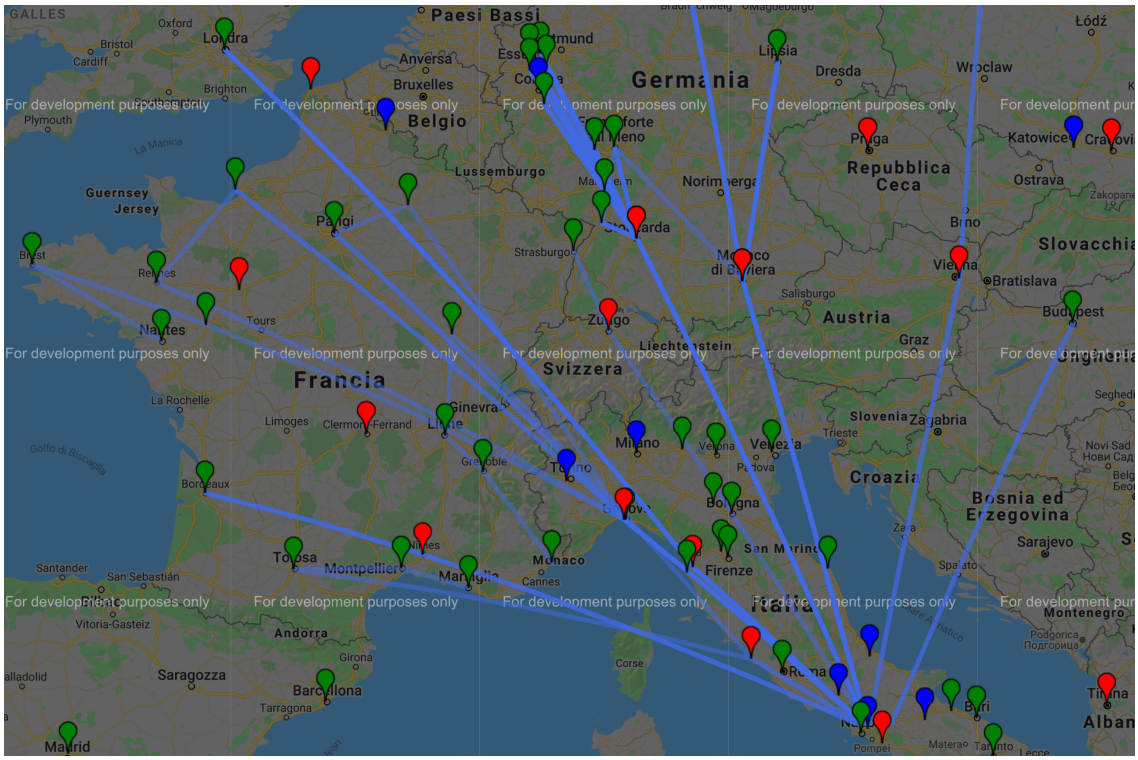

Figura 5.9. Scenario di distribuzione per la versione 2 #3

Analizzando le mappe di distribuzione ottenute in risultato si nota inoltre come molte tratte passano prima per i centri distribuzione mentre per alcune consegne si preferisce il viaggio diretto. Sebbene, per via del passaggio attraverso un luogo intermedio, la distanza percorsa per tratta sia maggiore rispetto alla versione 1, questa può essere una soluzione conveniente per far rispettare delle politiche logistiche.

5.4 Risultati per la versione 3

I risultati relativi a questa versione portano a delle mappe che sfruttano strade reali. Ci si aspetta che le distanze percorse siano molto maggiori di quelle delle versioni precedenti dovuto al fatto che non sono più distanze "in linea d'aria", quindi si va verso una distribuzione più vicina a un caso reale, che però ha un ulteriore livello di ottimizzazione offerto dal servizio *Directions* di Google. Alcuni esempi di scenari di distribuzione sono rappresentati nelle seguenti mappe:

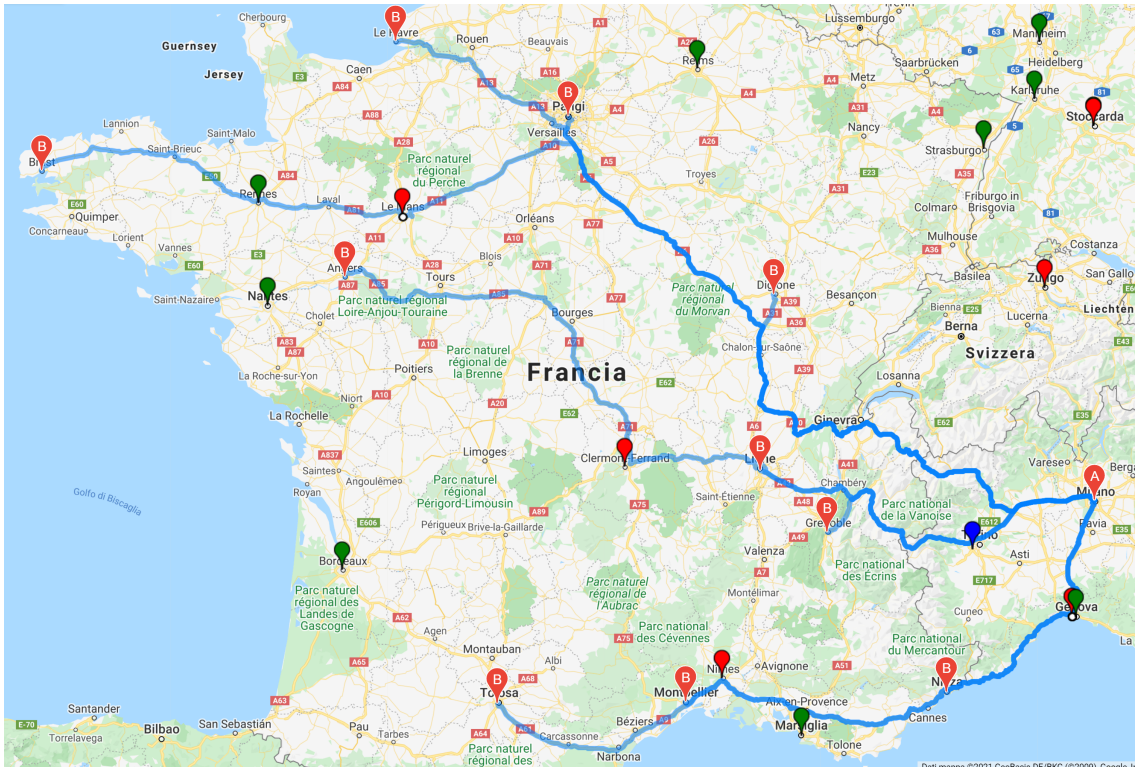

Figura 5.10. Scenario di distribuzione per la versione 3 #1

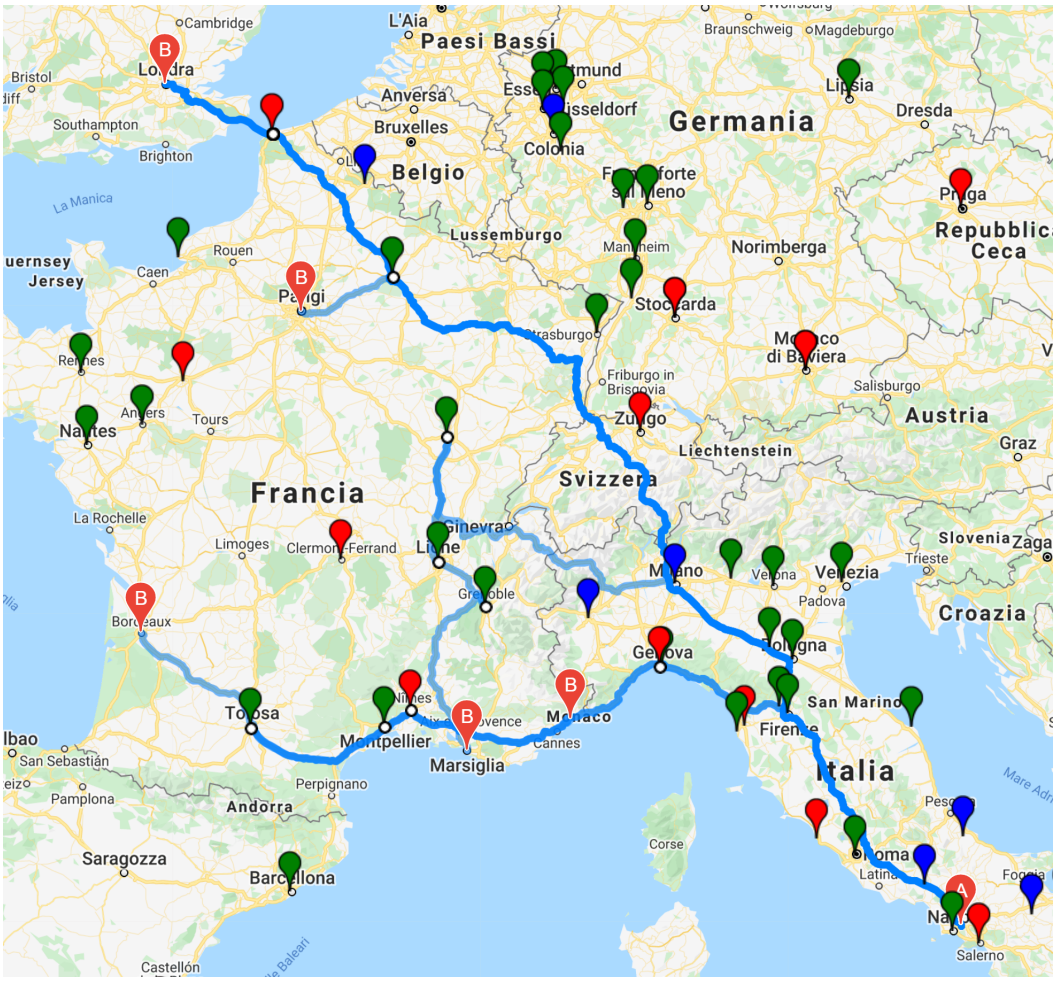

Figura 5.11. Scenario di distribuzione per la versione 3 #2

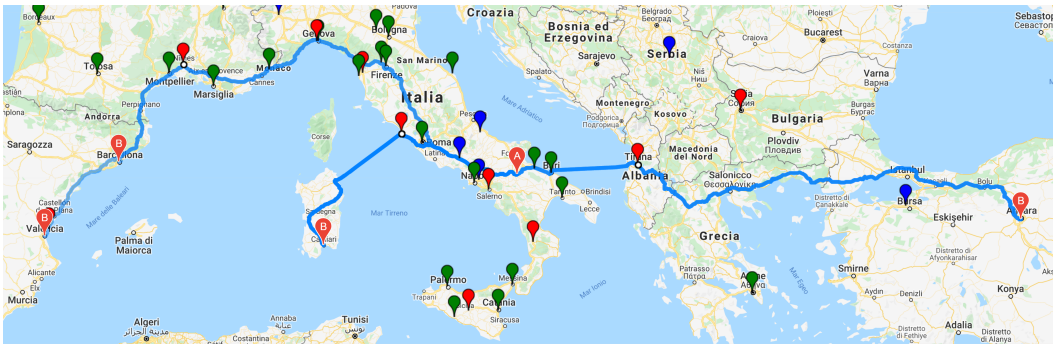

Figura 5.12. Scenario di distribuzione per la versione 3 #3

Un primo aspetto che si nota osservando le figure 5.11 e 5.12 è che, sfruttando *Directions*, per alcune tratte viene effettuata la distribuzione mediante traghetto.

Un importante miglioramento rispetto alla versione precedente è però una migliore scelta del passaggio attraverso i centri distribuzione, che non si basa più sulla distanza in linea d'aria bensì quella percorsa su strada, che porta a ottimizzare di molto le rotte di distribuzione. Alcuni esempi di questa ottimizzazione sono illustrati di seguito:

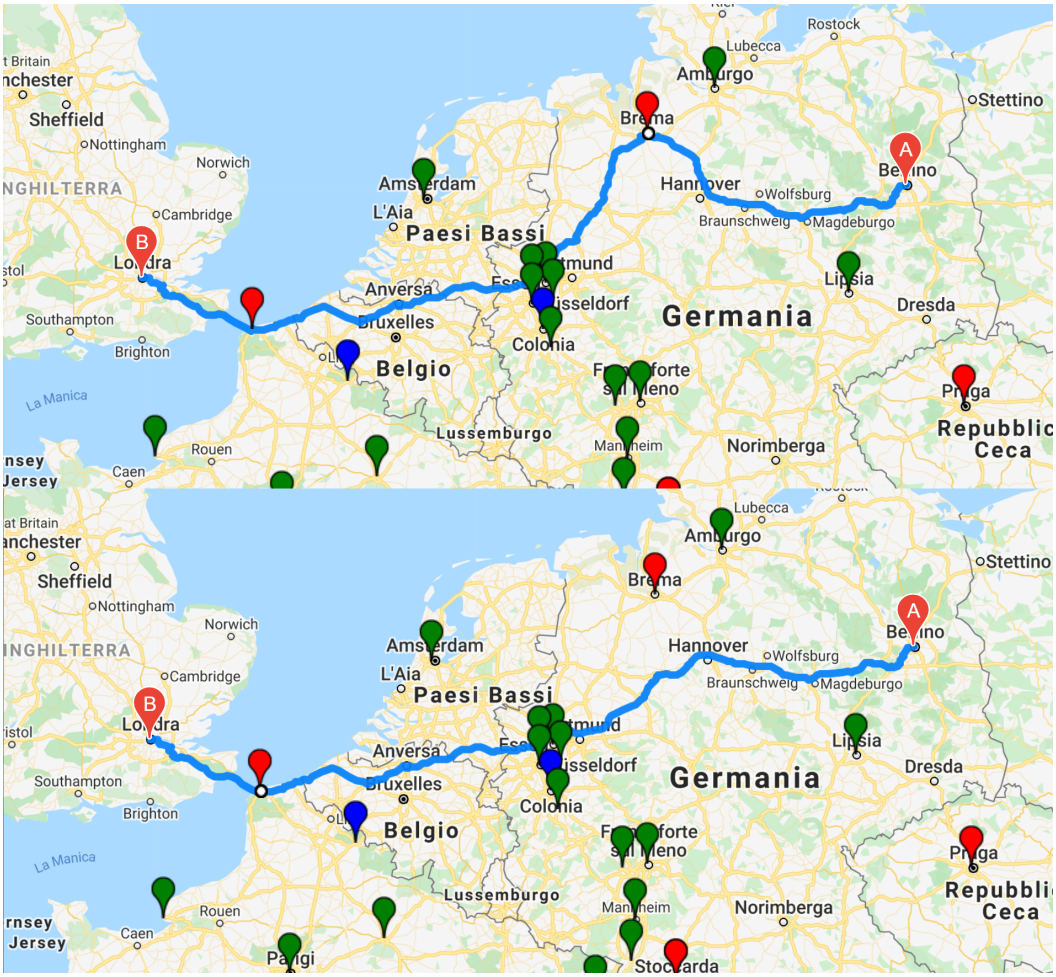

Figura 5.13. Confronto tra versione 2 e versione 3 #1

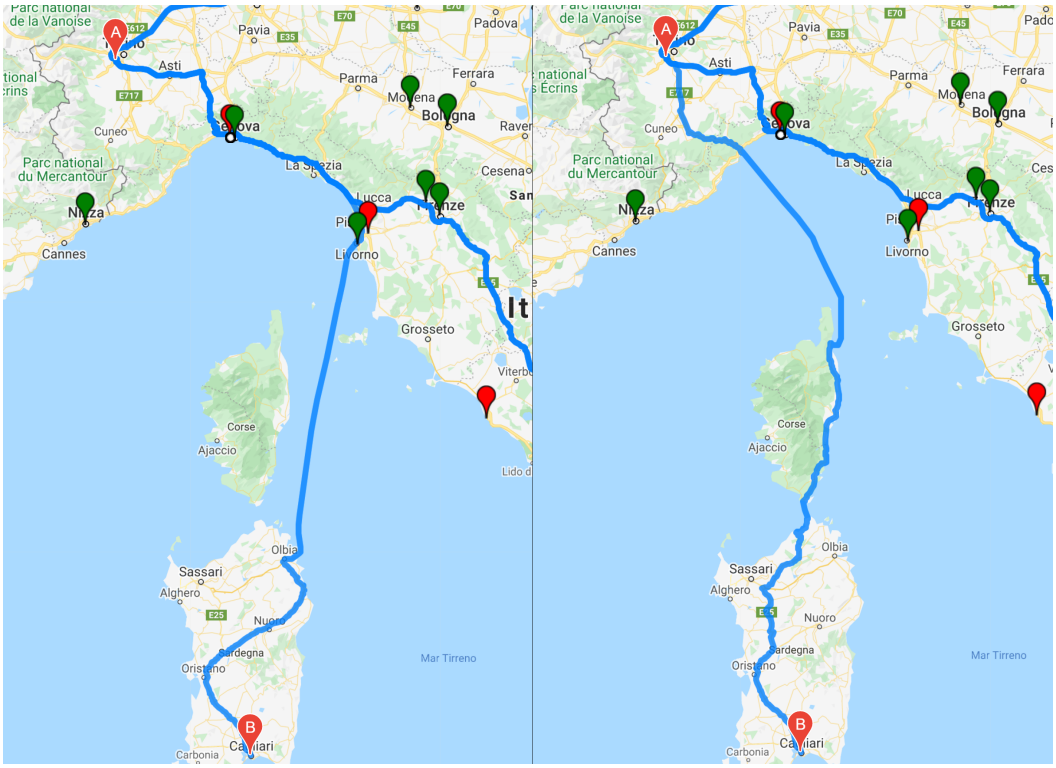

Figura 5.14. Confronto tra versione 2 e versione 3 #2

In queste immagini sono presenti da una parte un risultato della versione 2 al quale è stato applicato il servizio *Directions* solo per la realizzazione della mappa, mentre dall'altra parte è presente il risultato della versione 3, che applica i servizi di Google ogni volta che viene richiesta una distanza. Risulta evidente l'ottimizzazione del tragitto.

Più nel dettaglio, nel caso particolare della figura 5.13 (dove è presente il risultato della versione 2 in alto e quello della versione 3 in basso), si nota come nella versione 3, per la realizzazione della tratta Berlino - Londra, venga scelto come centro distribuzione attraverso cui passare quello di Calais e non quello di Brema scelto nella versione 2, che porta a una tratta con percorrenza minore.

Nella figura 5.14 invece (che mostra il risultato della versione 2 a sinistra e quello della versione 3 a destra), si nota come nella tratta Torino - Cagliari della versione 3, venga preferito il viaggio diretto invece che il passaggio

attraverso il centro distribuzione localizzato a Genova presente nella versione 2, oltre a una partenza tramite un traghetto diverso, che porta anche in questo caso a una tratta con percorrenza minore rispetto all'altra versione.

Quest'ultima figura fa emergere però un aspetto particolare. Per realizzare la tratta Torino - Cagliari ottimizzando la distanza percorsa, OR-Tools e *Directions* scelgono di far passare parte della tratta attraverso la Corsica.
Questa scelta, apparentemente poco "ottimale", accade perché il programma attuale non tiene conto di aspetti legati alla costificazione delle tratte, che porterebbero ad esempio ad evitare percorsi che passano attraverso un altro stato, a cui corrispondono diverse leggi e tassazioni, che potrebbero inficiare negativamente su aspetti come il costo complessivo della tratta.
In maniera del tutto teorica, aspetti legati alla costificazione, e quindi alla minimizzazione dei costi di ogni operazione, verranno trattati nel capitolo conclusivo, nella parte legata agli aspetti migliorabili.

5.5 Risultati per la versione 4

L'ultima versione realizza una distribuzione non più basata sulla minimizzazione delle distanze ma sulla minimizzazione dei tempi di percorrenza su strada, soddisfando tutte le richieste nel rispetto delle finestre temporali associate sia a stabilimenti che a clienti. Le finestre stabiliscono gli intervalli di tempo in cui non sono presenti condizioni meteo avverse alle coordinate geografiche dello stabilimento o dei clienti.
Due esempi di tratte ottenute da OR-Tools sono:

```
Route for vehicle 0
0 Demand(0) Window(0.0, 5.23) -> 55 Demand(3) Window(0.77, 6.0) ->
61 Demand(3) Window(0.88, 7.11) -> 54 Demand(3) Window(1.01, 7.24)
-> 0 Available load(1) Window(1.77, 5.23)
Time of the route:  1.77 days / Load of the route:  9

Route for vehicle 40
40 Demand(0) Window(0.0, 6.85) -> 59 Demand(3) Window(1.0, 7.17)
-> 57 Demand(4) Window(1.2, 7.37) -> 66 Demand(3) Window(1.31, 7.48)
-> 40 Available load(0) Window(1.83, 6.85)
Time of the route:  1.83 days / Load of the route:  10
```

Si consideri ad esempio la prima tratta, dove il veicolo contrassegnato con 0 effettua la seguente consegna:

$0\,(stabilimento) \rightarrow 55\,(3prodotti) \rightarrow 61\,(3prodotti) \rightarrow 54\,(3prodotti) \rightarrow 0\,(stabilimento)$.

Ogni luogo nella tratta ha associata una finestra temporale all'interno del quale il veicolo deve transitare, sia per consegnare i prodotti che per partire e tornare allo stabilimento di partenza. Ad esempio nella destinazione 54 non è possibile consegnare i prodotti negli intervalli (0.0, 1.01) e (7.24, 8.0) poiché in quegli istanti o sono presenti condizioni meteo avverse alla consegna oppure la bisarca non è ancora arrivata a destinazione.

Il veicolo arriva alla destinazione 55 0.77 giorni (circa 18 ore e 30 minuti) dopo l'inizio del periodo di 8 giorni (l'orizzonte temporale prefissato), per giungere alla destinazione 61 in 0.11 giorni (circa 2 ore e 40 minuti), arrivare alla destinazione 54 in 0,13 giorni (circa 3 ore e 7 minuti) per poi tornare allo stabilimento dopo 1,77 giorni (42 ore 30 minuti) dalla partenza, consegnando un totale di 9 veicoli.

Seguendo lo stesso ragionamento, nella seconda tratta viene realizzata una consegna di 10 veicoli, quindi una bisarca a pieno carico, in 1,83 giorni (circa 44 ore).

Andando al risultato della simulazione, una volta terminata l'esecuzione di OR-Tools, viene mostrato il seguente output:

```
Number of travels:  628
Total load distributed:  5865
Load efficiency:  93.39%
Solution achieved with optimal number of vehicles per request
```

Si nota un risultato migliore rispetto alle versioni precedenti che sfruttano RDBSCAN, ossia tutte tranne la versione 0 che, come accennato nel paragrafo 5.1, rappresenta un caso troppo lontano dalla realtà. Si ottiene una distribuzione con 628 viaggi totali (contro i 634 delle versioni precedenti) e una efficienza di carico del 93.39% (contro il 92.51% delle versioni precedenti).

Nella realizzazione delle mappe di distribuzione, si nota un differente raggruppamento delle rotte, legato al fatto che vengono utilizzati meno veicoli rispetto alle altre versioni. Sebbene non vi siano evidenti differenze dalle mappe di distribuzione della versione 3, in queste mappe viene assicurata la consegna di tutti i prodotti ai punti di destinazione nel rispetto delle finestre temporali dei clienti. Alcuni esempi sono mostrati di seguito:

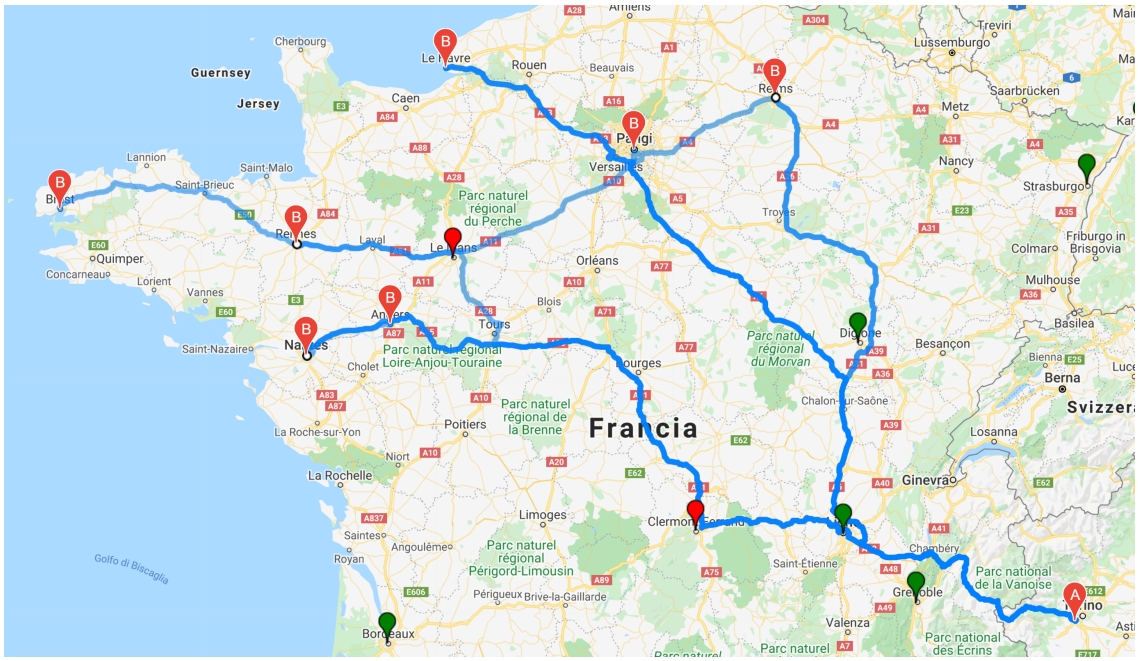

Figura 5.15. Scenario di distribuzione per la versione 4 #1

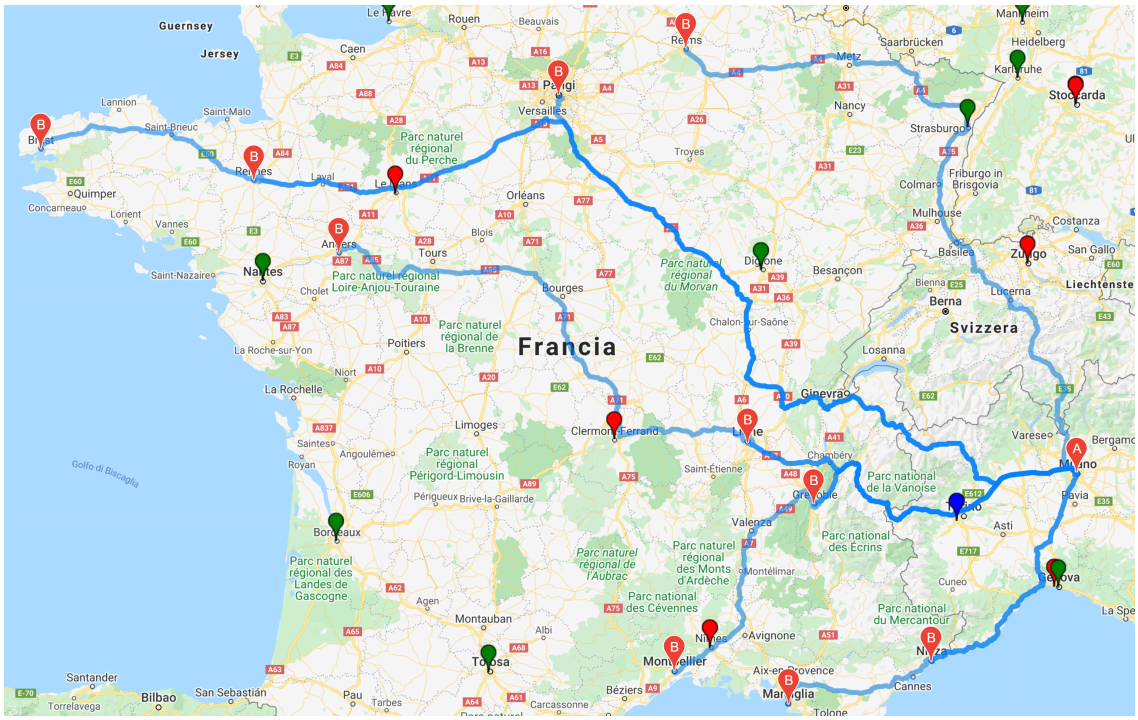

Figura 5.16. Scenario di distribuzione per la versione 4 #2

Capitolo 6

Conclusioni

Arrivati al capitolo finale, con questo lavoro di tesi è stato possibile realizzare delle simulazioni di scenari di distribuzione che, aumentando di complessità, si sono via via avvicinati a casi reali, ottimizzando aspetti come la distanza percorsa o il tempo percorso dai veicoli su strada, arrivando a reagire a eventi non previsti dalla catena di produzione o dal servizio logistico, sfruttando comunque meccanismi legati al Machine Learning non supervisionato e quindi al ramo dell'Intelligenza Artificiale.
L'idea di coniugare Intelligenza Artificiale e la *Supply Chain Network* è presente sul mercato ormai da tempo, al punto che per gli esperti nel campo noto come *Supply Chain Management*, viene considerata una necessità. Questa unione però si scontra spesso con l'elevata complessità (in termini di realizzazione e manutenzione) delle soluzioni di IA e soprattutto con il processo di adattamento delle sfide della SC con tecniche legate ad esempio al Machine Learning o al Deep Learning.
Sebbene negli ultimi anni ci si è sempre più spinti verso l'adattare queste tecniche per una gestione sempre più efficiente ed efficace, questa idea al momento resta principalmente un campo di ricerca molto discusso e in continua e costante evoluzione, che quindi non è ancora arrivato alla "soluzione definitiva".
La tesi proposta realizza con la sua ultima versione una simulazione abbastanza complessa (che supporta l'esecuzione in parallelo), con uno scenario di distribuzione ottimale in una situazione dove sono presenti molteplici stabilimenti di produzione, molteplici prodotti da distribuire, la possibilità di passare attraverso punti intermedi (i centri distribuzione) e la capacità di rispondere a eventi esterni alla Supply Chain forniti tramite una fonte esterna (nel caso in esame un feed meteo), il tutto sfruttando veicoli con capacità

limitata che percorrono strade reali.

Questo scenario, seppur complesso e in grado di rispondere a esigenze sempre attuali, riesce a gestire molti aspetti della logistica e della distribuzione di un prodotto, ma si può ulteriormente migliorare in modo da poter simulare scenari più prossimi alla realtà, in un ambiente dominato da eventi inattesi e molta incertezza. Questo *Proof of Concept* infatti, non è stato realizzato per essere un qualcosa di "monolitico", ma è stato realizzato con un ottica "modulare" in modo da rendere possibile l'inserimento di tecnologie per simulare e gestire altri aspetti tipici di uno scenario di distribuzione reale, per portare questo sistema ad essere ancora più robusto e soprattutto per aumentare la sua capacità di "decidere" in maniera proattiva.

L'obiettivo di questo capitolo è anche quello di documentare tutti i limiti di questo esperimento e quali sono gli aspetti migliorabili. Nei due paragrafi successivi si discuterà rispettivamente prima di tutti quegli aspetti migliorabili relativi ai lavori già fatti e poi di quegli aspetti che non sono stati trattati nella realizzazione dell'esperimento, fornendo tecniche e modi per implementarli.

6.1 Aspetti migliorabili: limiti interni

Una prima limitazione è legata a ciò che OR-Tools può ammettere come variabili per la risoluzione dello scenario di distribuzione. Nel caso si voglia impostare e risolvere problemi di ricerca operativa (come i VRP) di natura molto complessa, che vanno oltre le capacità di OR-Tools, la stessa Google consiglia di usare un altro strumento (non appartenente a Google) chiamato Gurobi, che è uno strumento professionale le cui capacità però hanno un costo.
Alcuni esempi di limiti di OR-Tools sono l'incapacità di trattare il rientro dei veicoli nei rispettivi stabilimenti alla fine di una tratta di consegna per permettere a questi di poter ripartire (subito o dopo un certo intervallo di tempo) per soddisfare un'altra richiesta. Questo aspetto è stato annotato in questo lavoro già con i risultati della "versione 0" (paragrafo 5.1). OR-Tools infatti calcola la soluzione considerando che tutti i veicoli partano allo stesso istante temporale da tutti gli stabilimenti. Questo risultato viene reimmaginato in questo lavoro, portandolo verso un contesto reale, come il numero di tratte percorse per distribuire i prodotti secondo lo schema di distribuzione. Questo è il motivo per cui in output si usa il termine "viaggi" invece che

"veicoli".
Applicare dunque una soluzione che permette il rientro dei veicoli porta a ridurre il numero di veicoli spiegato per ogni soluzione. Per realizzare ciò occorre implementare una sorta di "scheduling" dei veicoli all'interno dell'orizzonte temporale in esame.

Altro limite di OR-Tools è legato all'impossibilità di strutturare una qualunque variante di 2E-CVRP, ossia il CVRP a due livelli o, generalizzando, una qualunque distribuzione a n livelli. Un possibile scenario di distribuzione *"Multi-Echelon"* a n livelli è ad esempio quello che tiene conto di punti specifici da attraversare quando un veicolo è a corto di carburante, rappresentato dall'insieme dei punti di rifornimento e dei centri distribuzione.
Applicando OR-Tools a uno schema del genere, quest'ultimo utilizzerebbe questi punti una sola volta per un solo veicolo, ragionamento errato per uno scenario di distribuzione reale, forzando inoltre il passaggio presso questi punti. La versione 1 realizza una prima versione di 2E-CVRP applicandola al risultato di OR-Tools (e non facendola calcolare a questo), che non forza il vincolo del dover passare necessariamente presso strutture secondarie.
Una variante più complessa di quanto già realizzato potrebbe calcolare un tragitto che include l'insieme di punti di rifornimento e centri distribuzione attraverso cui passare (rendendolo un ME-CVRP), raggiungendo gli obiettivi richiesti e rispettando i vincoli imposti. Questo porterebbe alla realizzazione di tratte che si devono avvicinare il più possibile a quelle ottenute in risultato da OR-Tools, che si considerano ottimali, rispettando gli obiettivi da raggiungere.

Altri limiti incontrati nell'esperimento sono quelli legati alle API Google.
Ad esempio, come fatto già notare nella versione 3, si potrebbe sfruttare il servizio *Advanced Directions* (a pagamento) per calcolare le mappe di distribuzione tramite un modello a richiesta, in modo da averle sempre aggiornate in tempo reale, oltre a permettere la gestione di ulteriori eventi esogeni per la SC quali traffico, congestioni stradali, dissesti naturali e tutti quegli eventi che rendono non percorribili certi tratti stradali. Questo rende gli scenari di distribuzione ulteriormente "reattivi", soprattutto per quando si passa dall'ambiente di simulazione a quello delle consegne effettive nel rispetto della SCN ottimizzata.

Un ulteriore miglioramento allo strumento proposto è dato poi dall'implementazione di ulteriori feed per la gestione di eventi esterni. Si consideri ad

esempio un feed di notizie che tiene traccia di scioperi dei lavoratori o scioperi degli operatori nel settore di trasporti su mare nel caso sia necessario che i veicoli debbano attraversare un tratto di mare su un traghetto.

6.2 Aspetti migliorabili: prospetti futuri

Si passa a questo punto alla parte relativa agli aspetti non trattati nella realizzazione dell'esperimento.
L'aspetto sicuramente più importante riguarda quello legato ai costi di ogni operazione (già accennato nel paragrafo 5.4). Ci si aspetta in uno scenario di distribuzione reale di raggiungere quanto già realizzato con l'ultima versione cercando di minimizzare i costi operazionali. Non è stato possibile realizzare ciò con le versioni proposte in quanto tra i dati forniti dall'azienda del settore Automotive non erano presenti informazioni sui costi operazionali.
Considerando le simulazioni già realizzate, ci si aspetta che percorrendo la minima distanza possibile o effettuando le consegne nel minor tempo possibile si consumi meno carburante, il che vuol dire che i costi complessivi saranno minori. Non è però presente nel sistema proposto una variabile o un dato che tiene conto di tutti i dati sulla costificazione cercando di minimizzarne il costo totale, poiché non ammessa da OR-Tools.

Un aspetto interessante da poter inserire in questo strumento è dunque l'idea di implementare un **processo parallelo** al calcolo della distribuzione ottimale in grado di ottimizzare i processi di costificazione.
Per realizzare questo tipo di processo parallelo si possono sfruttare altri concetti della Ricerca Operativa o dell'Intelligenza Artificiale che non è stato possibile inserire per la simulazione della distribuzione.

Per quanto riguarda la ricerca operativa si potrebbero sfruttare le potenzialità dell'**Ottimizzazione Robusta**, che porta alla risoluzione di quei problemi in cui si vuole soddisfare una certa domanda composta da un set di richieste in cui sono presenti situazioni di incertezza sul loro valore effettivo, con l'obiettivo di raggiungere una certa condizione di "robustezza" del sistema a seguito dell'aleatorietà della domanda. Questo sistema avrà vincoli espressi non in forma esatta, ma tramite distribuzioni di probabilità. In questi casi infatti, non si avrà una domanda pari a x ma una che ha una probabilità P di essere pari a x.
L'idea di applicare questo tipo di ottimizzazione è legato al fatto che in una

situazione reale non si ha mai una misura certa sulla domanda, sebbene se la si possa stimare basandosi su dati raccolti in orizzonti temporali passati. È sempre presente una misura di "incertezza" in questi dati, che li rende delle variabili aleatorie.
L'obiettivo è quindi quello di ridurre l'impatto dell'incertezza su una certa decisione da prendere e minimizzare la probabilità che questa incertezza - intesa come *rischio* - si verifichi, esprimendo il tutto tramite vincoli tipici dell'ottimizzazione convessa o della programmazione lineare (intera o mista-intera).
Inserire il concetto di rischio permette inoltre di portare all'interno della risoluzione di questi problemi altre conoscenze, come quelle legate alla valutazione dei rischi (o in inglese *"Risk assessment"*).
Un esempio di lavoro di Ottimizzazione robusta applicato all'ottimizzazione dei costi e delle risorse è quello proposto nel periodo di Febbraio 2020 dal collettivo composto da L. Gar Goei, C. Chii Yeh e T. Kai Wei per la gestione delle risorse acquifere di Singapore [11].
Applicando un processo del genere in parallelo all'elaborazione della distribuzione dei veicoli si otterrebbe una simulazione robusta che soddisfa un modello di domanda con incertezza, quindi più prossimo a uno scenario reale, che cerca di ottimizzare sia gli aspetti legati alla distribuzione su strada sia quelli legati ai costi e alle risorse.

Per quanto concerne l'IA invece, si possono sfruttare tecnologie come il *Reinforcement Learning* (RL) e le **reti neurali** per realizzare i processi di ottimizzazione dei costi e delle risorse. Nel campo della *Supply Chain Optimization* sono stati proposti molti studi e realizzate molte pubblicazioni che sfruttano queste due tecnologie, e tutt'oggi continua ad essere argomento di ricerca.
In generale il *Reinforcement Learning* riguarda la creazione di "agenti intelligenti" in grado di prendere delle azioni o decisioni su un dato ambiente in un certo stato in modo da massimizzare una certa misura di "ricompensa".

Gli algoritmi di RL si esprimono solitamente mediante il Processo Decisionale di Markov (dall'inglese *Markov Decision Process*, o MDP), considerato la formulazione matematica dei problemi di RL.

Questo processo è definito dal set (S, A, R, P, γ), dove:

- S rappresenta l'insieme di tutti gli stati in cui si trova l'ambiente
- A rappresenta l'insieme di tutte le azioni possibili

- R rappresenta la distribuzione delle ricompense per una coppia (stato, azione)
- P rappresenta la distribuzione delle probabilità per una coppia (stato, azione) di passare ad un altro stato
- γ rappresenta un fattore di *discount*

L'obiettivo del MDP è di trovare una funzione dello stato $\pi(s)$ che specifica l'azione corretta da fare quando ci si trova in un particolare stato, in modo da massimizzare la misura di ricompensa. Questa funzione è nota come *"policy"*.

Applicando la combinazione tra reti neurali e RL nel contesto della *Supply Chain* si può ulteriormente allargare la portata della simulazione realizzata per questa tesi permettendogli ad esempio di gestire aspetti che vanno a monte della distribuzione logistica, fino alla gestione della catena di produzione, risolvendo problemi relativi alla gestione dell'inventario. In questi casi l'obiettivo della rete neurale è quello di trovare la *policy* migliore per gestire l'intera *Supply Chain Network* e soddisfare una richiesta con incertezza minimizzando i costi operazionali, mentre nel frattempo (nel processo parallelo) viene calcolata la distribuzione ottimale per portare i prodotti ai clienti o alla destinazione richiesta (che può essere un centro distribuzione o un altro stabilimento).

Un esempio di gestione dell'inventario in una *Multi-Echelon Supply Chain* è quello realizzato dal collettivo Grid Dynamics nel periodo di Febbraio 2020, che sfrutta una *Deep Q Network* per trovare la *policy* ottimale per il contenimento dei costi [12].
Come ulteriori esempi, anche gli studi [13] e [14] realizzati nel 2018, portano a risultati equivalenti, applicando le stesse conoscenze con soluzioni diverse.

In figura 6.1 è presente un semplice scenario di ottimizzazione della gestione dell'inventario proposto da Grid Dynamics.

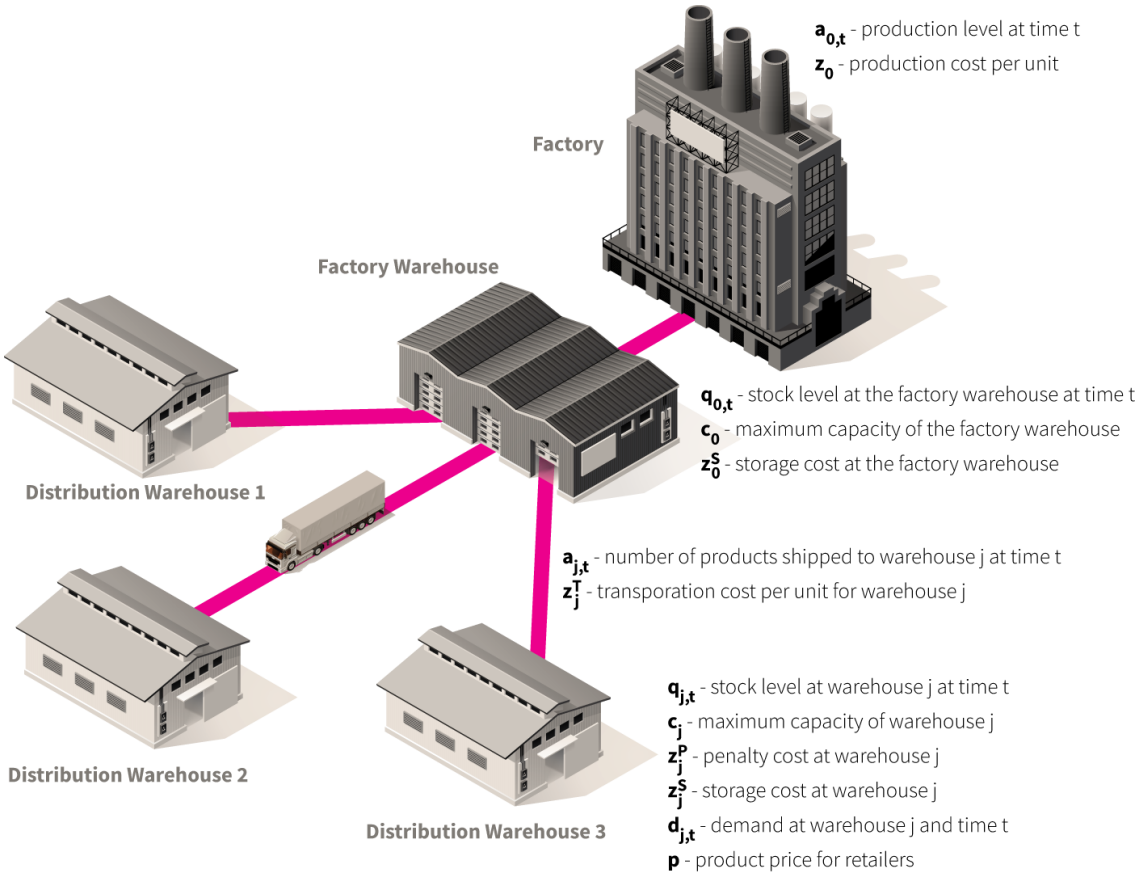

Figura 6.1. Scenario semplice di ottimizzazione dell'inventario.
Fonte: Grid Dynamics [12]

Concludendo, questo argomento risulta evidente essere in continua evoluzione e senza una "soluzione definitiva". Le sfide legate all'adattamento dei campi della Ricerca Operativa e dell'Intelligenza Artificiale alla *Supply Chain Network* continueranno ad essere obiettivo di ricerca negli anni a venire, ma è comunque possibile raggiungere un certo livello di simulazione efficiente, affidabile, proattiva e robusta, andando incontro alle esigenze di qualunque tipo di azienda, a prescindere dalla sua grandezza.

Appendice A

Implementazione in Python del RDBSCAN

```
def recursiveDBSCAN(waypoints, MINRADIUS, MAXRADIUS,
  minClusterSize, maxClusterSize,
     minNumberOfClusters):

  bestClusters = []
  bestAverageClusterSize = 0
  bestRadius = 0
  minRadius = MINRADIUS
  maxRadius = MAXRADIUS
  mPerRadian = 6371008.8 #mean earth radius in
     meters

  while minRadius < maxRadius:
    radius = (minRadius + maxRadius) / 2
    eps = radius / mPerRadian

    #DBSCAN haversine needs data in form [lat, long]
    dbscan = DBSCAN(eps, algorithm ='ball_tree',
      metric = 'haversine',
      min_samples=minClusterSize)
      .fit(np.radians(waypoints))
    labels = dbscan.labels_
    numberOfClusters = len(set(labels))
```

```
    clusters = pd.Series([waypoints[labels == n] for
        n in range(numberOfClusters)])

    if numberOfClusters < minNumberOfClusters:
      maxRadius = radius - 1
    else:
      minRadius = radius + 1

      #evaluate average cluster size
      sumOfElements = 0
      clusterSizes = []
      for cluster in clusters:
        sumOfElements+=len(cluster)
        clusterSizes.append(len(cluster))
      avgClusterSize = sumOfElements / len(clusters)

      if avgClusterSize > bestAverageClusterSize:
        bestAverageClusterSize = avgClusterSize
        bestClusters = clusters
        bestRadius = radius

#here minRadius = maxRadius
if len(bestClusters) == 0:
  print('NO_SOLUTION_FOUND')
  return pd.Series([])

#here bestClusters is not empty
finalClusters = []

for cluster in bestClusters:
  if len(cluster) > maxClusterSize:
    newClusters = recursiveDBSCAN(cluster,
      MINRADIUS,
      MAXRADIUS, minClusterSize,
      maxClusterSize,
      minNumberOfClusters)
    #recursive call
    finalClusters.extend(newClusters)
  else:
```

```
        finalClusters.append(cluster)
    return finalClusters
```

Bibliografia

[1] Ministero dello Sviluppo Economico, *Piano nazionale Industria 4.0 (2017-2020)*, https://www.sviluppoeconomico.gov.it/images/stories/documenti/Piano_Industria_40.pdf

[2] B. Gesing (DHL), S. Peterson (IBM), D. Michelsen (IBM), *Artificial Intelligence in Logistics*, https://www.dhl.com/content/dam/dhl/global/core/documents/pdf/glo-core-trend-report-artificial-intelligence.pdf

[3] Berkeley University, *CS 188 | Introduction to Artificial Intelligence, Fall 2020*, https://inst.eecs.berkeley.edu/~cs188/fa20/

[4] T.M. Mitchell, McGraw Hill, *Machine Learning, seconda edizione*, https://www.cs.cmu.edu/~tom/NewChapters.html

[5] Nvidia (2016), *A visual representation of AI, machine learning, and deep learning*, https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-a

[6] J. K. Lenstra, A. H. G. Rinnooy Kan (1981), *Complexity of vehicle routing and scheduling problems*, https://www.researchgate.net/publication/229563032_Complexity_of_vehicle_routing_and_scheduling_problems

[7] M. Ester, H.P. Kriegel, J. Sander, X. Xu (1996), *A density-based algorithm for discovering clusters in large spatial databases with noise*, https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

[8] K. Bujel, F. Lai, M. Szczecinski, W. So, M. Fernandez (2018), *Solving High Volume Capacitated Vehicle Routing Problem with Time Windows using Recursive-DBSCAN clustering algorithm*, https://arxiv.org/pdf/1812.02300.pdf

[9] *OpenWeather*, https://openweathermap.org/

[10] *OpenWeather: Weather conditions*, https://openweathermap.org/weather-conditions

[11] L. Gar Goei, C. Chii Yeh e T. Kai Wei (2020) *Using Robust Optimisation and Mixed Integer Programming to manage Singapore's water resources*, https://medium.com/dsaid-govtech/using-robust-optimization-and-mixed-integer-programming-to-manage-singapo

[12] Grid Dynamics (2020) *Deep reinforcement learning for supply chain and price optimization*, https://blog.griddynamics.com/deep-reinforcement-learning-for-supply-chain-and-price-optimization/

[13] L. Kemmer, H. von Kleist, D. de Rochebouet, N. Tziortziotis, J. Read (2018) *Reinforcement learning for supply chain optimization*, https://www.researchgate.net/publication/328676423_Reinforcement_learning_for_supply_chain_optimization

[14] A. Oroojlooyjadid, L. V. Snyder, M. Takác (2018) *Applying Deep Learning to the Newsvendor Problem*, https://arxiv.org/pdf/1607.02177.pdf