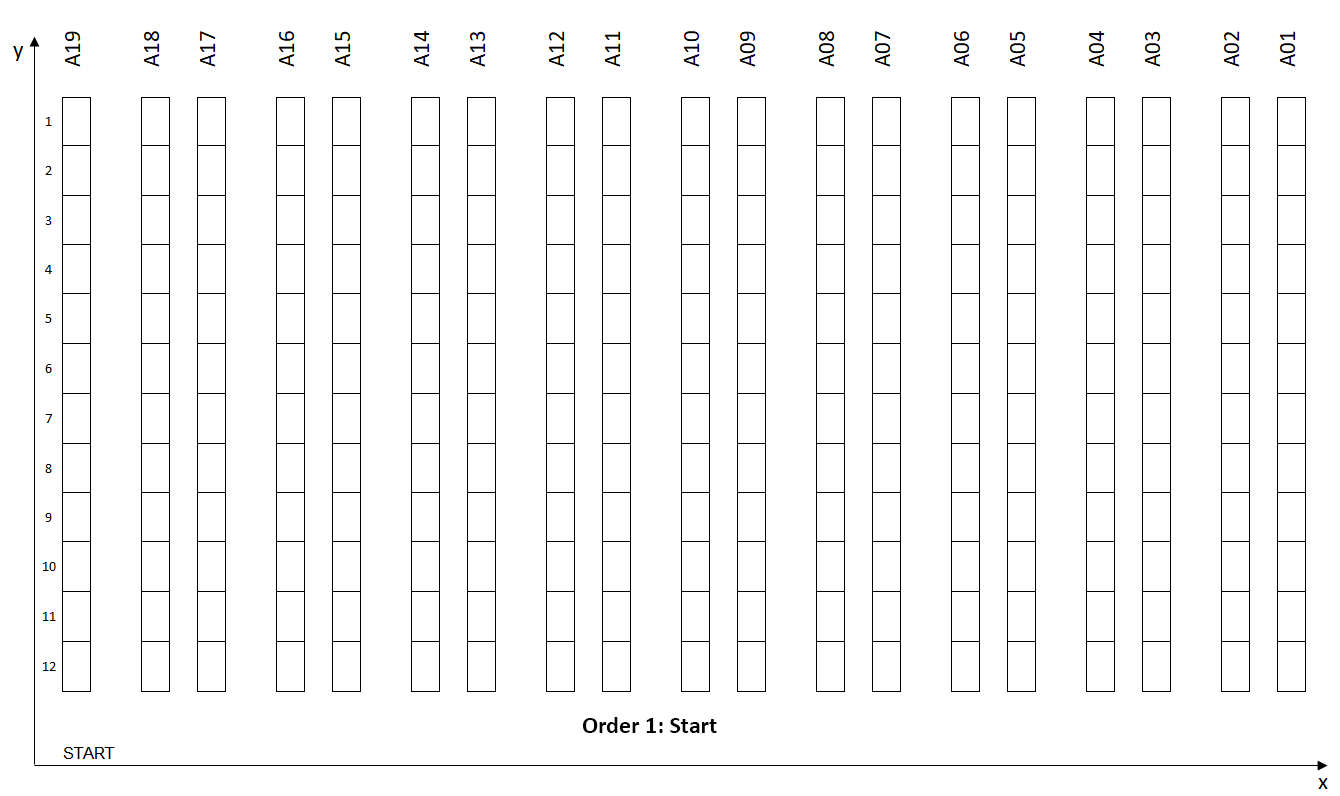
**Improve Werahouse Productivy: using Order Batching (Parte 1)**

*Progettare un modello di simulazione per stimare l'impatto di diverse strategie di Single Picker Routing Problem nella Picking Productivity.*

In un centro di distribuzione (DC), il tempo di percorrenza da un luogo all'altro durante il percorso di prelievo può rappresentare dal 60% al 70% dell'orario di lavoro dell'operatore.

Ridurre questo tempo di percorrenza è il modo più efficace per aumentare la produttività complessiva del DC.



*Figura 1: SCENARIO 1 - Picking routes with 1 order picked per wave*

1. ***Obiettivo***

Questo lavoro ha l’obiettivo di determinare come progettare un modello per simulare l'impatto di diversi processi di prelievo e metodi di instradamento per l’ottimale order picking utilizzando il **Single Picker Routing Problem (SPRP)** per un modello di magazzino bidimensionale (asse-x, asse-y).

SPRP è un'applicazione specifica del problema generale del **Commesso Viaggiatore (TSP)**.

TSP risponde alla seguente domanda:

*"Dato un elenco di storage locations e le distanze tra ciascuna coppia di locations, qual è il percorso più breve possibile per visitare ciascuna posizione di stoccaggio e tornare al deposito?"*

SPRP viene utilizzato per determinare il percorso minimo nel processo di prelievo per preparare uno o più ordini.

1. ***Che cos’è il ‘Wave Picking’?***

Per questo studio, utilizzeremo l'esempio di un E-Commerce tipo DC in cui gli articoli sono immagazzinati in scaffali a 4 livelli. Questi scaffali sono organizzati in più righe (Riga#: 1 … n) e corridoi (Corridoio#: A1 … A\_n).

**Premesse**:

Immagine che contiene carretto, trasporto

Descrizione generata automaticamente

*Figura 2: Werahouse Picking Carts*

1. **Dimensione degli articoli**: Articoli di dimensioni piccole e leggere;
2. **Carrello di prelievo:** Carrello di prelievo leggero con una capacità di 10 ordini;
3. **Percorso di prelievo**: Il percorso di prelievo inizia e finisce nella stessa posizione.

Lo Scenario 1 (*Figura 1*), il peggiore in termini di produttività, può essere facilmente ottimizzato grazie alle seguenti:

* Locations: Gli ordini n. 1 e n. 2 hanno posizioni di prelievo comuni;
* Zone: Gli ordini hanno punti di prelievo in una zona comune;
* Ordini a riga singola: L'efficienza items\_picked/walking\_distance è molto bassa.

Immagine che contiene testo, armadietto

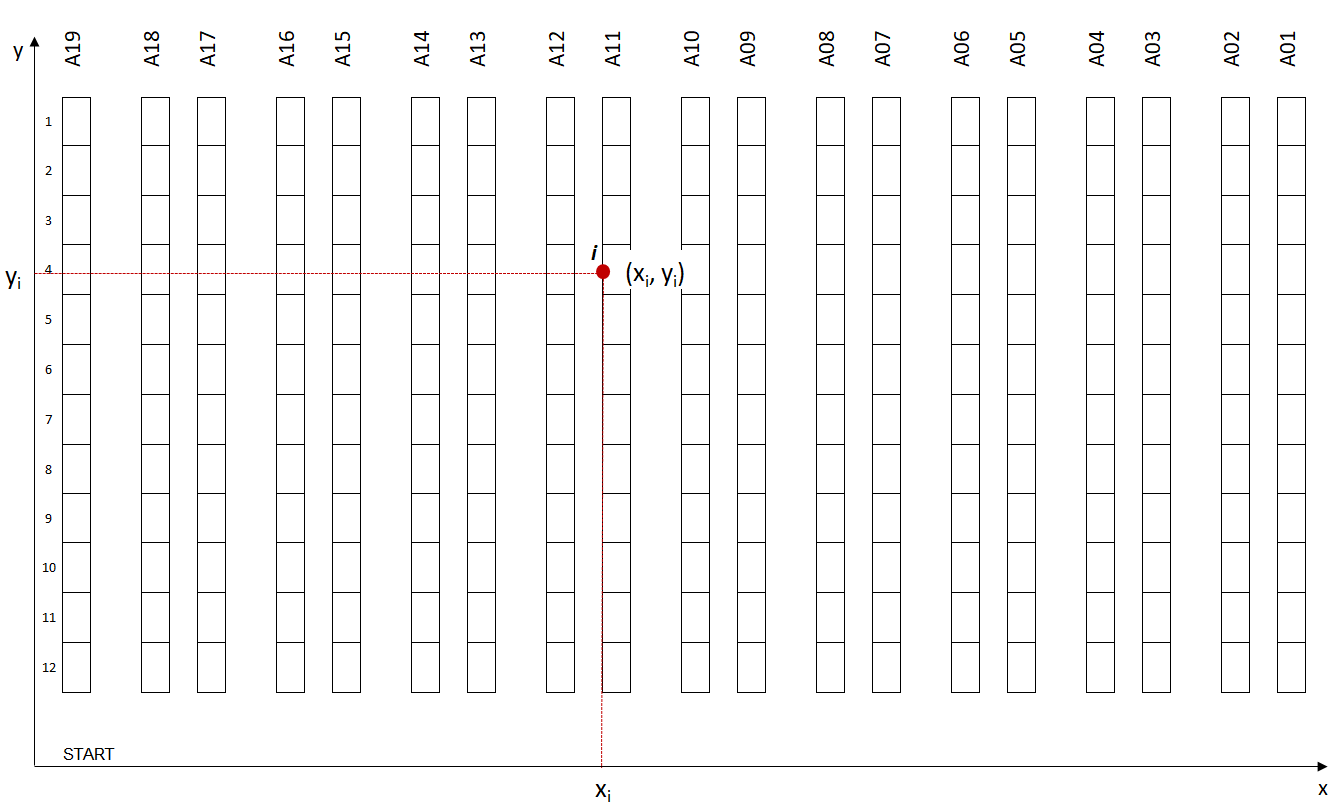
Descrizione generata automaticamente

*Figura 3: SCENARIO 2 – Wave Picking applicata allo Scenario 1*

Il primo modo intuitivo per ottimizzare questo processo è combinare questi tre ordini in un unico percorso di prelievo: questa strategia è comunemente chiamata **Wave Picking**.

Quindi, come detto in precedenza, quello che si andrà a fare è di costruire un modello per simulare l'impatto di diverse strategie di Wave Picking sulla distanza totale percorsa per uno specifico insieme di ordini da preparare.

1. ***Test di diversi Algoritmi di Ottimizzazione***
   1. ***Layout di magazzino con mappatura delle posizioni di stoccaggio***

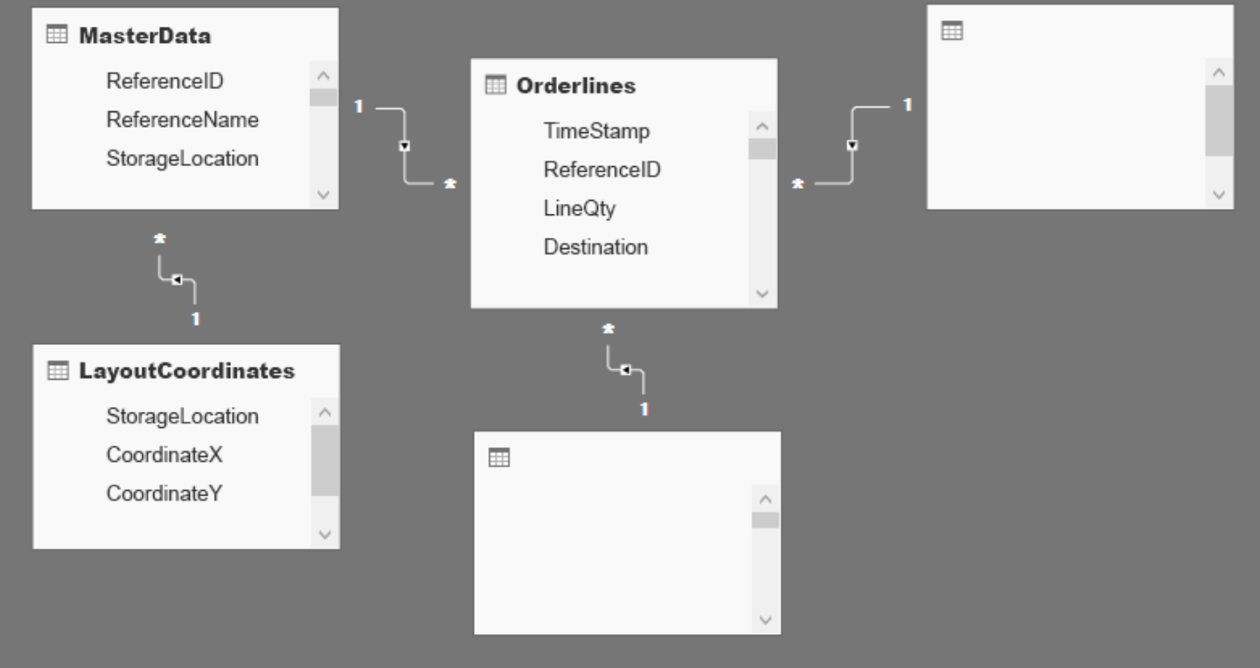


*Figura 4: Warehouse Layout con coordinate 2D*

In base al layout effettivo del magazzino, le posizioni di stoccaggio vengono mappate con coordinate 2-D (x, y) che verranno utilizzate per misurare la distanza percorsa.

Ogni posizione di stoccaggio è collegata a un riferimento utilizzando i dati anagrafici. (Ad esempio, il riferimento #123129 si trova nella coordinata (xi, yi)). È quindi possibile collegare ogni riga dell'ordine a una posizione geografica per il prelievo.

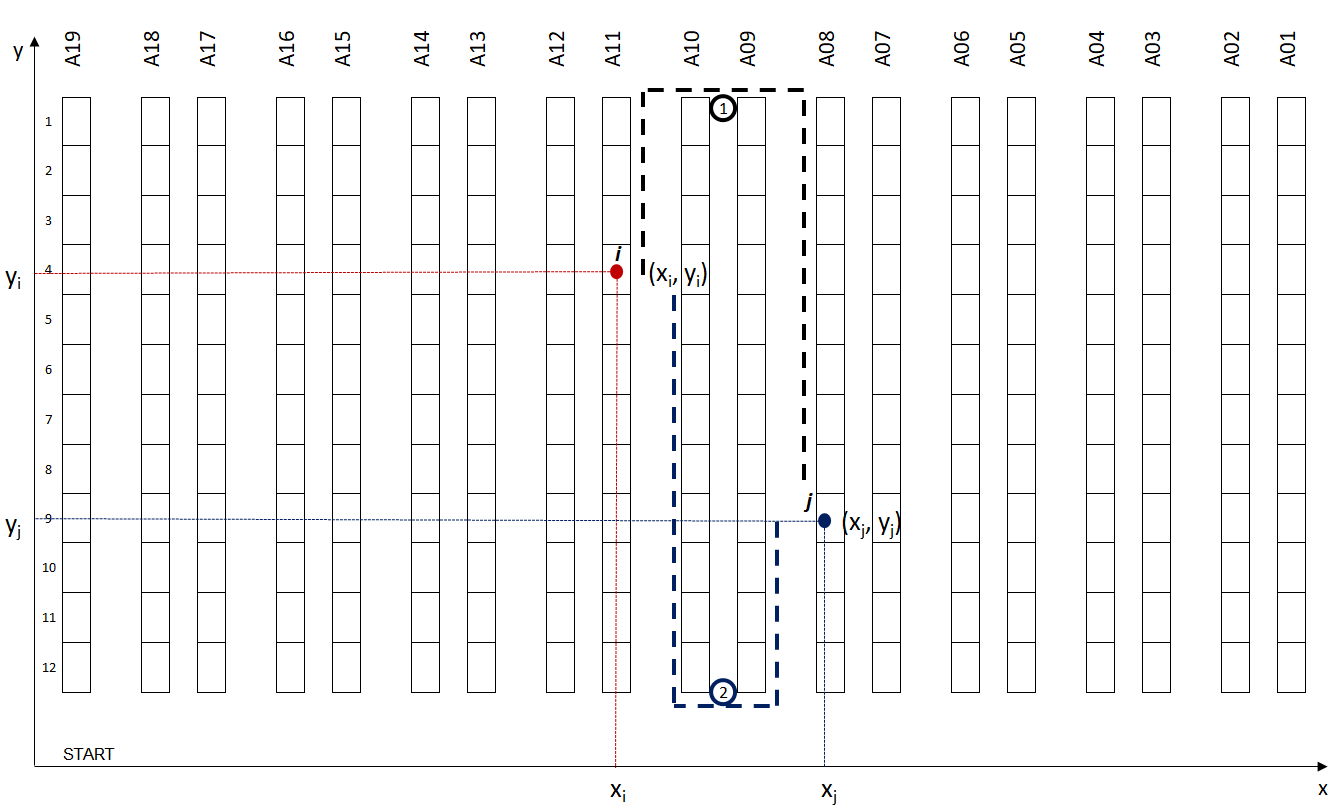
* 1. ***Righe d'ordine dal tuo Warehouse Management System (WMS)***



*Figura 5: Database Schema*

Le righe di OrderLines possono essere estratte dal database WMS e questa tabella deve essere unita (join) alla tabella MasterData per collegare ogni riga ordine a un'ubicazione di stoccaggio e alle relative coordinate (x, y) nel magazzino.

È possibile aggiungere ulteriori tabelle per includere più parametri nel modello come (Destinazione, Tempi di consegna, Imballaggio speciale, etc.).

* 1. ***Funzioni per il calcolo della distanza della Picking Route***
* **Funzione 1:** calcola la distanza tra due picking locations.

*Figura 6: Route differenti fra 2 storage locations nel magazzino*

La funzione 1 verrà utilizzata per calcolare la walking distance da un punto i(xi, yi) a un punto

j(xj, yj).

**Obiettivo**: restituire la walking distance più breve tra i due potenziali routes dal pnt i al pnt j.

**Parametri**:

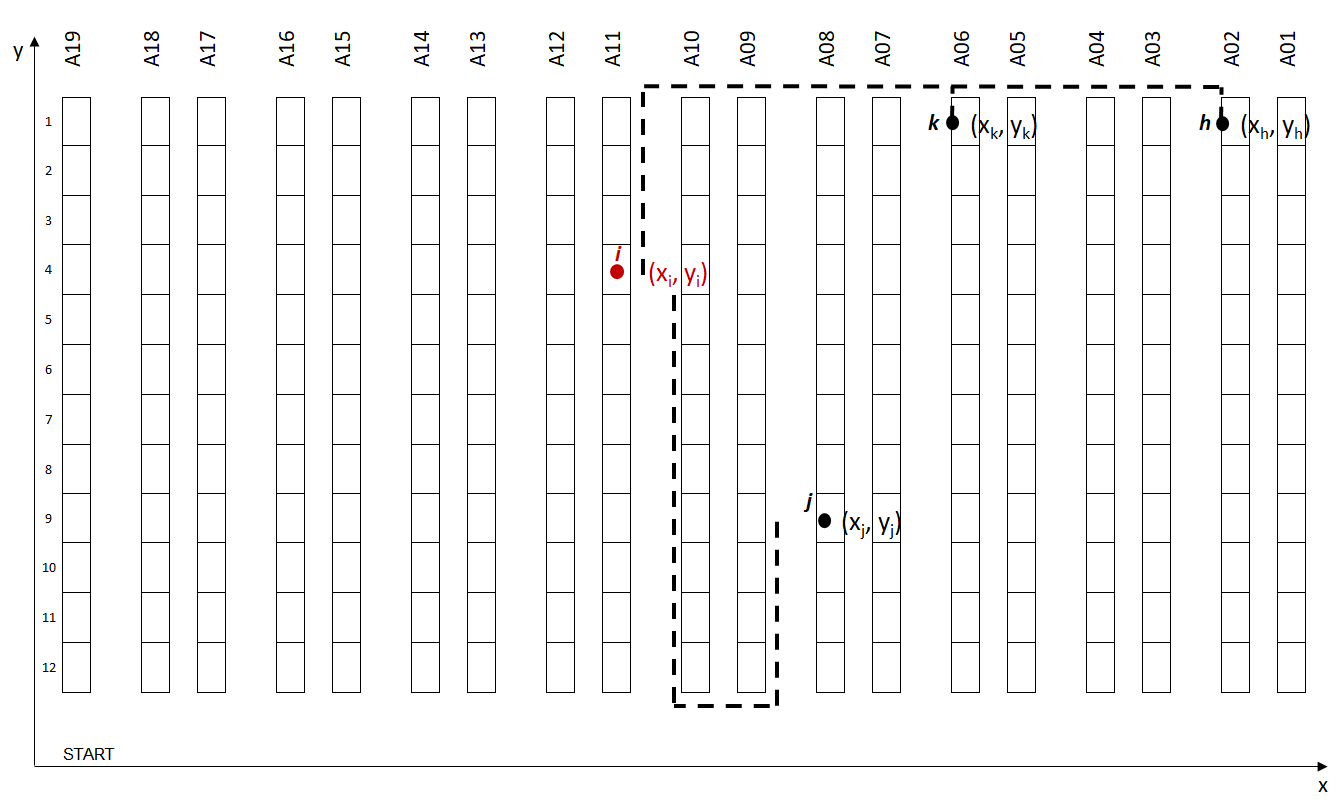
* + y\_low – il punto più basso dell’alley (Asse y)
  + y\_high – il punto più alto dell’alley (Asse y)

**Codice Python:**

1. import numpy as np
2. import pandas as pd
3. import ast
4. from ast import literal\_eval

7. def distance\_picking(Loc1, Loc2, y\_low, y\_high):
8. '''Calcolare Picker Route Distance fra 2 locations'''
9. # Punto di Partenza
10. x1, y1 = Loc1[0], Loc1[1]
11. # Punto di Destinazione
12. x2, y2 = Loc2[0], Loc2[1]
13. # Distanza asse x
14. distance\_x = abs(x2 - x1)
15. # Distanza asse y
16. **if** x1 == x2:
17. distance\_y1 = abs(y2 - y1)
18. distance\_y2 = distance\_y1
19. **else**:
20. distance\_y1 = (y\_high - y1) + (y\_high - y2)
21. distance\_y2 = (y1 - y\_low) + (y2 - y\_low)
22. # Distanza minina sull'asse y
23. distance\_y = min(distance\_y1, distance\_y2)
24. # Distanza Totale
25. distance = distance\_x + distance\_y
26. **return** **int**(distance)

* **Funzione 2:** la prossima location più vicina.



*Figura 7: Lo scenario della successiva Storage Location*

Questa funzione verrà utilizzata per scegliere la location successiva tra diversi candidati per continuare il picking route.

**Obiettivo**: restituire la posizione più vicina come miglior candidato.

**Codice Python:**

1. **def** next\_location(start\_loc, list\_locs, y\_low, y\_high):
2. '''''Trovare la closest next location'''
3. # Distanza per ogni prossimo punto candidato
4. list\_dist = [distance\_picking(start\_loc, i, y\_low, y\_high) **for** i **in** list\_locs]
5. # Distanza Minima
6. distance\_next = min(list\_dist)
7. # Location della Distanza Minima
8. index\_min = list\_dist.index(min(list\_dist))
9. next\_loc = list\_locs[index\_min]
10. list\_locs.remove(next\_loc)
11. **return** list\_locs, start\_loc, next\_loc, distance\_next

* **Funzione 3**: crea il picking route e calcola la walking distance totale.

Questa funzione viene usata per creare il picking route da una serie di ordini da preparare.

* + *Input*: un elenco di (x, y) locations in base agli elementi da prelevare per questo percorso.
  + *Output*: una sequenza ordinata di locations coperti e la walking distance totale.

**Codice Python:**

1. **import** pandas as pd
2. **import** numpy as np
3. **import** itertools
4. **from** ast **import** literal\_eval
5. **from** utils.routing.distances **import** \*
7. **def** create\_picking\_route(origin\_loc, list\_locs, y\_low, y\_high):
8. '''''Calcolare la distanza totale per coprire per una lista di locations'''
10. # Variabile per la distanza totale
11. wave\_distance = 0
12. # Location corrente variabile
13. start\_loc = origin\_loc
14. # Memorizzare i routes
15. list\_chemin = []
16. list\_chemin.append(start\_loc)
18. **while** len(list\_locs) > 0: # Looping fino a quando tutte le locations vengono scelte
19. # Verso la prossima location
20. list\_locs, start\_loc, next\_loc, distance\_next = next\_location(start\_loc, list\_locs, y\_low, y\_high)
21. # Update start\_loc
22. start\_loc = next\_loc
23. list\_chemin.append(start\_loc)
24. # Update distanza
25. wave\_distance = wave\_distance + distance\_next
27. # Distanza finale dall'ultima storage location dall'origine
28. wave\_distance = wave\_distance + distance\_picking(start\_loc, origin\_loc, y\_low, y\_high)
29. list\_chemin.append(origin\_loc)
31. **return** wave\_distance, list\_chemin
    1. ***Funzioni per creare orders waves***

* **Funzione 1:** crea batches di n ordini da raccogliere contemporaneamente.
  + *Input*: order lines dataframe (df\_orderlines), numero di orders per wave(orders\_wave).
  + *Output*: dataframe mappato con il numero di wave (Column : WaveID), il numero totale di waves (waves\_number).

**Codice Python:**

1. **def** orderlines\_mapping(df\_orderlines, orders\_number):
2. '''''Mapping orders con il numero di wave '''
3. df\_orderlines.sort\_values(by='DATE', ascending = True, inplace = True)
4. # Lista dei Numeri univoci degli order
5. list\_orders = df\_orderlines.OrderNumber.unique()
6. dict\_map = dict(zip(list\_orders, [i **for** i **in** range(1, len(list\_orders))]))
7. # Order ID mapping
8. df\_orderlines['OrderID'] = df\_orderlines['OrderNumber'].map(dict\_map)
9. # Raggruppo per Orders by Wave del orders\_number
10. df\_orderlines['WaveID'] = (df\_orderlines.OrderID%orders\_number == 0).shift(1).fillna(0).cumsum()
11. # Conto il numero di Waves
12. waves\_number = df\_orderlines.WaveID.max() + 1
13. **return** df\_orderlines, waves\_number

* **Funzione 2:** listing delle wave\_ID picking route.
  + *Input*: order lines dataframe (df\_orderlines) e numero di wave ( WaveID ).
  + *Output*: lista delle locations i(xi,yi) incluse nel picking route.

**Codice Python:**

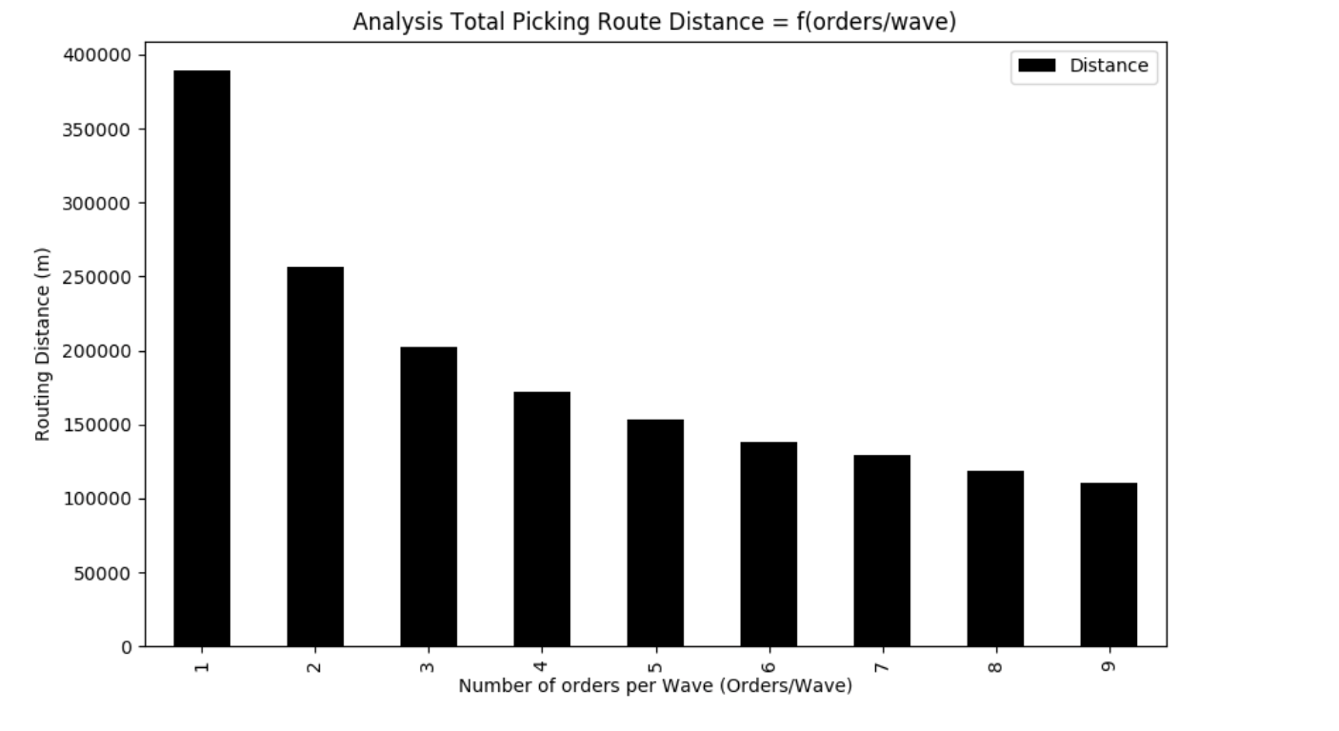
1. def locations\_listing(df\_orderlines, wave\_id):
2. '''Get le storage locations per coprire una wave di orders'''
3. df = df\_orderlines[df\_orderlines.WaveID == wave\_id]
4. # Creo un elenco di coordinate
5. list\_locs = list(df['Coord'].apply(lambda t: literal\_eval(t)).values)
6. list\_locs.sort()
7. # Lista delle coordinate univoche
8. list\_locs = list(k **for** k,\_ in itertools.groupby(list\_locs))
9. n\_locs = len(list\_locs)
10. **return** list\_locs, n\_locs
11. ***Risultati e Prossimi Steps***

Dopo aver impostato tutte le funzioni necessarie per misurare la picking distance, ora si puó testare la strategia del picking route con le picking order lines.

* **Orders** **Waves**: gli ordini sono raggruppati per ordine cronologico del tempo di ricezione da OMS (TimeStamp);
* **Picking** **Route**: la strategia del picking route segue la logica della location più vicina successiva.

Per stimare l'impatto della strategia di wave picking sulla produttività, eseguiremo diverse simulazioni con un numero graduale di ordini per wave:

1. **Misura totale della Walking distance**: quanta distanza percorribile viene ridotta all'aumentare del numero di ordini per percorso?
2. **Record Picking Route per Wave**: registrazione della sequenza di locations per route per ulteriori analisi.



*Figura 8: Risultati di 5000 order lines con un rapporto da 1 a 9 per route*

Questa soluzione è ben lungi dall'essere la più ottimale, i prossimi step saranno:

1. Gli ordini possono essere raggruppati per geographical clusters di picking locations per ridurre la walking distance dei picker.
2. La strategia next closest Location ha i suoi limiti che possono essere facilmente evidenziati dai picking route records.