# Studio di Algoritmi per il CVRP

Capacitated Vehicle Routing Problem

Alessia Cinelli 0350265

Andrea Tozzi 0350270

# Il Vehicle Routing Problem

Un problema di **ottimizzazione** che mira a determinare i percorsi ottimali per una flotta di veicoli che devono consegnare merci ai clienti, **minimizzando i costi** totali di trasporto.

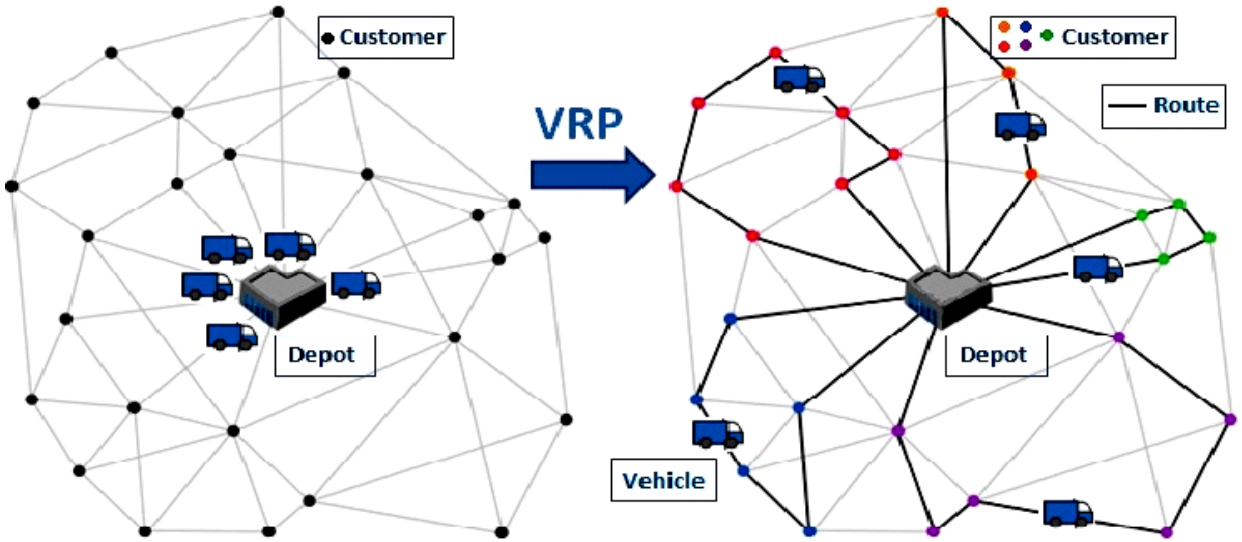
Input :

- **Clienti**
- **Domande**
- **Numero di veicoli**
- **Capacità dei veicoli**
- **Costi percorrenza**

VRP →

Output :

- **Rotte**
- **Costo soluzione**

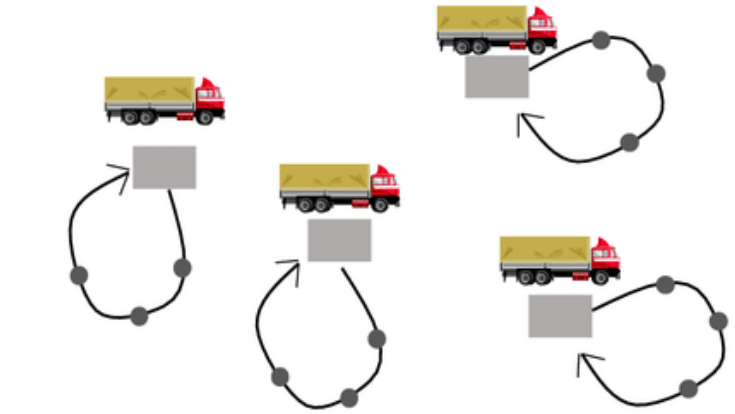


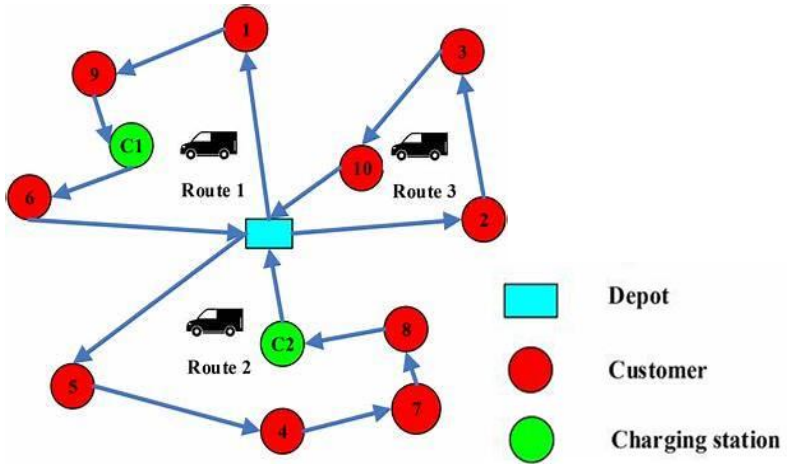

Il VRP è un problema ampiamente studiato nell'ambito della logistica e della pianificazione dei trasporti, essenziale per ottimizzare costi e tempi di consegna.

# Le varianti del VRP

- **Capacitated VRP (CVRP)** → **Oggetto del nostro studio**
  - Unico deposito
  - I veicoli hanno capacità limitata
  - I veicoli tornano deposito
- Asimmetric CVRP (ACVRP)
- Multi-Depot VRP (MDVRP)
- Green VRP (GVRP)
- VRP with Time Windows (VRPTW)
- VRP with Service Times (VRPST)
- VRP with Driving Time Limits (VRPDTL)
- VRP with Backhauls (VRPB)

Multi-Depot VRP





# VRP - Un Problema NP Hard

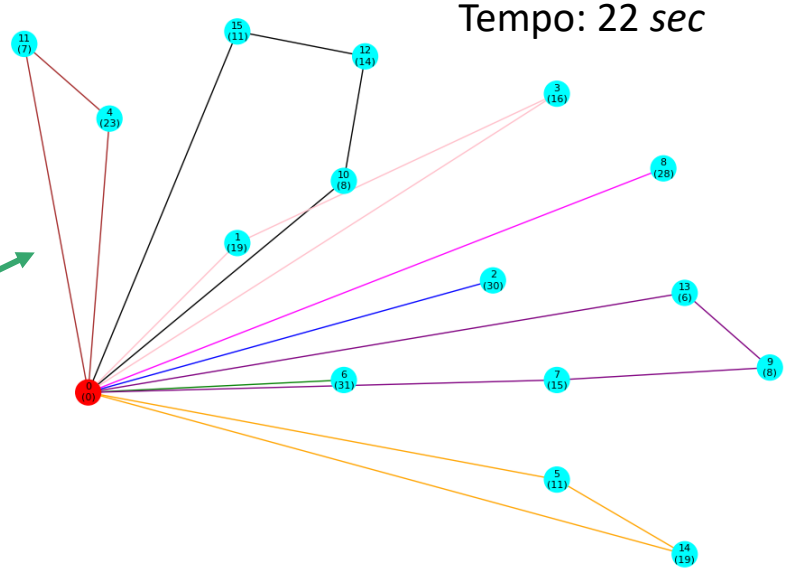
- I problemi NP-Hard sono **almeno tanto difficili** quanto i problemi più difficili in NP (Nondeterministic Polynomial time)

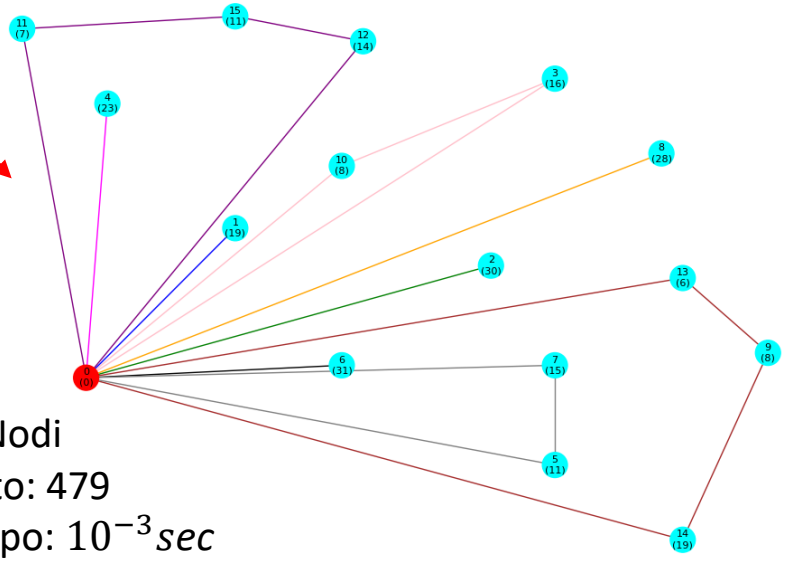
- **Soluzioni Esatte:** Impraticabili per istanze di grandi dimensioni a causa del **tempo di calcolo esponenziale.**

- **Soluzioni Approssimate:** Si utilizzano algoritmi euristici e per trovare soluzioni **quasi ottimali in tempi ragionevoli.**

**Non esiste un algoritmo noto** che risolva VRP in tempo polinomiale poiché:
Il numero di percorsi possibili aumenta esponenzialmente con l'aumento del numero di clienti e veicoli.

16 Nodi
Costo: 452
Tempo: 22 *sec*





16 Nodi
Costo: 479
Tempo: $10^{-3} sec$

# Euristiche e Random

- **Le euristiche** sono metodi di risoluzione dei problemi che utilizzano **un approccio pratico** per trovare soluzioni **sufficientemente buone** in un tempo ragionevole, senza garantire l'ottimalità.

Euristiche per CVRP:

- Nearest Neighbor
- **Clarke-Wright**
- **Sweep**

- **Algoritmi random** per il VRP generano soluzioni casuali con **basso costo** computazionale che **reiterato numerose volte**, aumenta la **possibilità** di trovare buone approssimazioni.

# Obiettivo

*Comprendere **se e quando** conviene utilizzare euristiche e algoritmi random al posto dei modelli esatti (algoritmi deterministici).*

*Principali domande:*

- Come varia **la qualità** delle soluzioni **al crescere** del problema?
- **Differenza tra tempi** di esecuzione **tra algoritmi non esatti** e modello esatto?
- C'è **un algoritmo** che è **preferibile utilizzare** per problemi di un determinato tipo?

# Istanze e Dati

Le istanze, circa 300, sono state reperite dalla libreria VRPLIB di Python, ognuna definita da:

- Un file **«Nome_istanza.vrp»** , in uno dei due formati:
  1. Euclideo: Sono fornite le coordinate (x, y) di ciascun nodo, le cui distanze euclidee rappresentano il costo di percorrenza.
  2. Esplicito: è fornita una matrice dei costi.

- Un file **«Nome_istanza.sol»** contenente le rotte ottimali e il relativo costo.

| Size | Dimensione | Numero Istanze |
|---|---|---|
| Small | $\leq$ 50 nodi | 46 |
| Mid small | $\leq$ 100 nodi | 47 |
| Mid | $\leq$ 250 nodi | 63 |
| Mid large | $\leq$ 500 nodi | 58 |
| Large | $\leq$ 1350 nodi | 51 |
| XLarge | $\leq$ 30000 nodi | 10 |

# Istanze – Problema del costo ottimo

Le soluzioni della libreria VRPLIB, presentavano delle rotte ottime con un *costo effettivo* **diverso** dal *costo ottimo fornito*, ciò a causa di **arrotondamenti nei costi di percorrenza,** questo portava in alcuni casi ad avere soluzioni delle euristiche **inferiori della soluzione ottima.**

Abbiamo sostituito tali costi con i relativi associati alle rotte ottime, per **valutare** con **precisione** le performance degli algoritmi euristici e random proposti.

Il problema presentato però (soluzione calcolata con le euristiche minore della soluzione ottima) persiste. Ciò è dovuto al fatto che **le euristiche non tengono conto del numero massimo di veicoli a disposizione.**

# GitHub Action

1

## Esecuzione dei test

Abbiamo riscontrato alcune difficoltà legate ai tempi di esecuzione e al surriscaldamento dei nostri dispositivi.

2

## Adozione delle GitHub Actions

Sono state utilizzate le GitHub Actions per automatizzare l’esecuzione degli algoritmi in un ambiente isolato e affidabile.

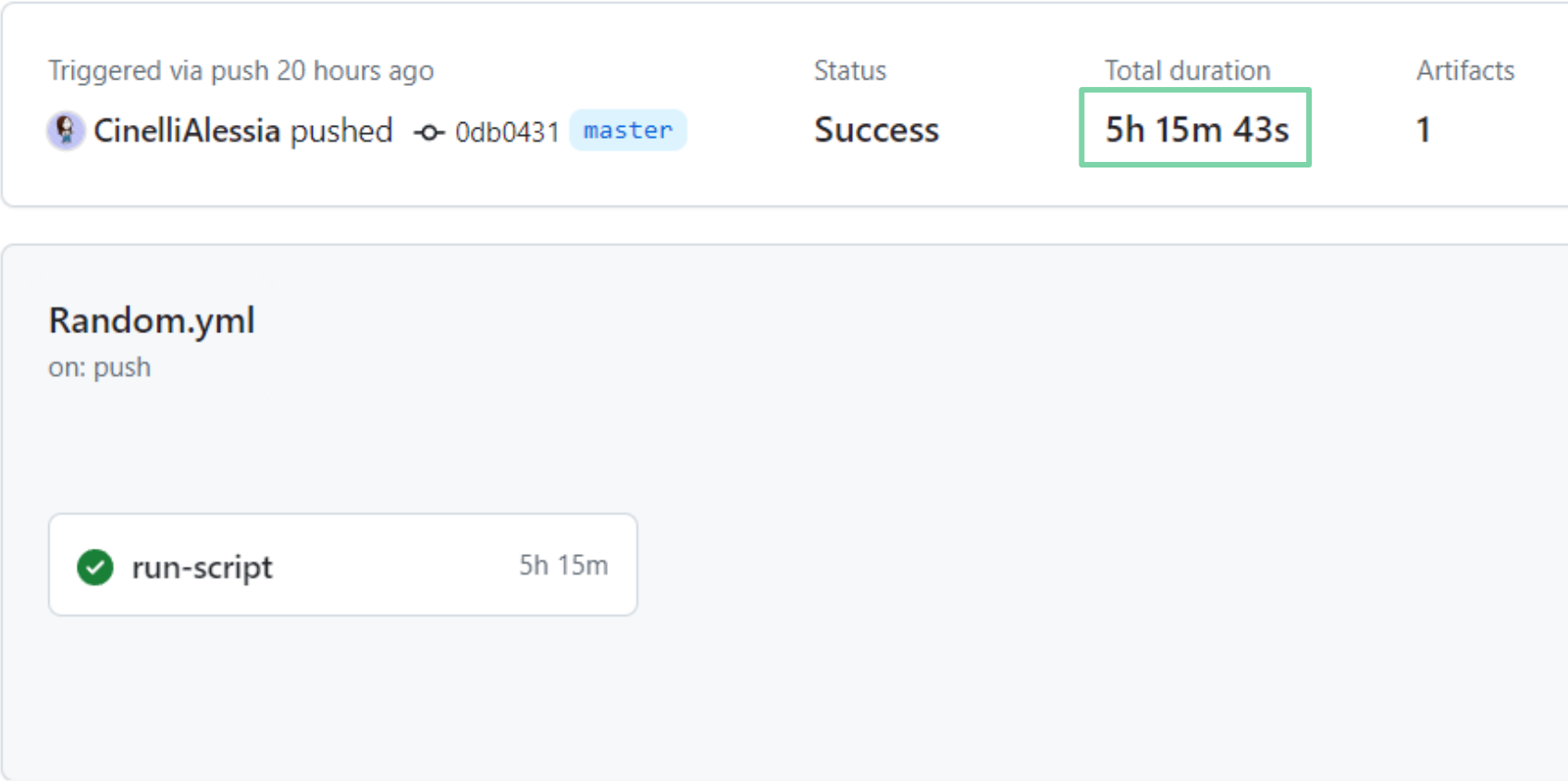
3

## Benefici delle GitHub Actions

Grazie a questo strumento, siamo riusciti ad eseguire le nostre soluzioni in un ambiente scalabile, garantendo una qualità dei risultati omogenea e indipendente dai dispositivi.

# GitHub Action

Questo ci ha permesso di risparmiare molto tempo durante i run degli algoritmi, ma purtroppo ciò non vale per il modello MIP.

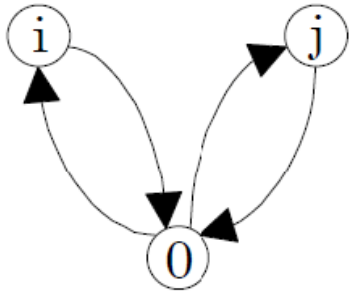


# Clarke & Wright

1

**Inizializzazione**

**Ogni cliente** viene servito da **un veicolo** che torna successivamente al deposito.



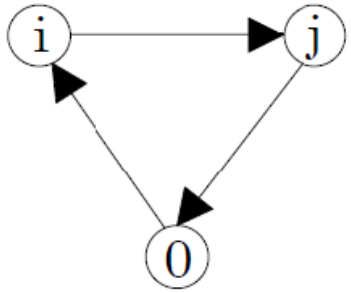

$$c_1 = 2 \cdot c_{0i} + 2 \cdot c_{0j}$$

2

**Calcolo dei risparmi per ogni coppia di nodi**

$$S_{ij} = c_{0i} + c_{0j} - c_{ij}$$

Successivamente ordinati in ordine decrescente




$$c_2 = c_{0i} + c_{ij} + c_{j0}$$

3

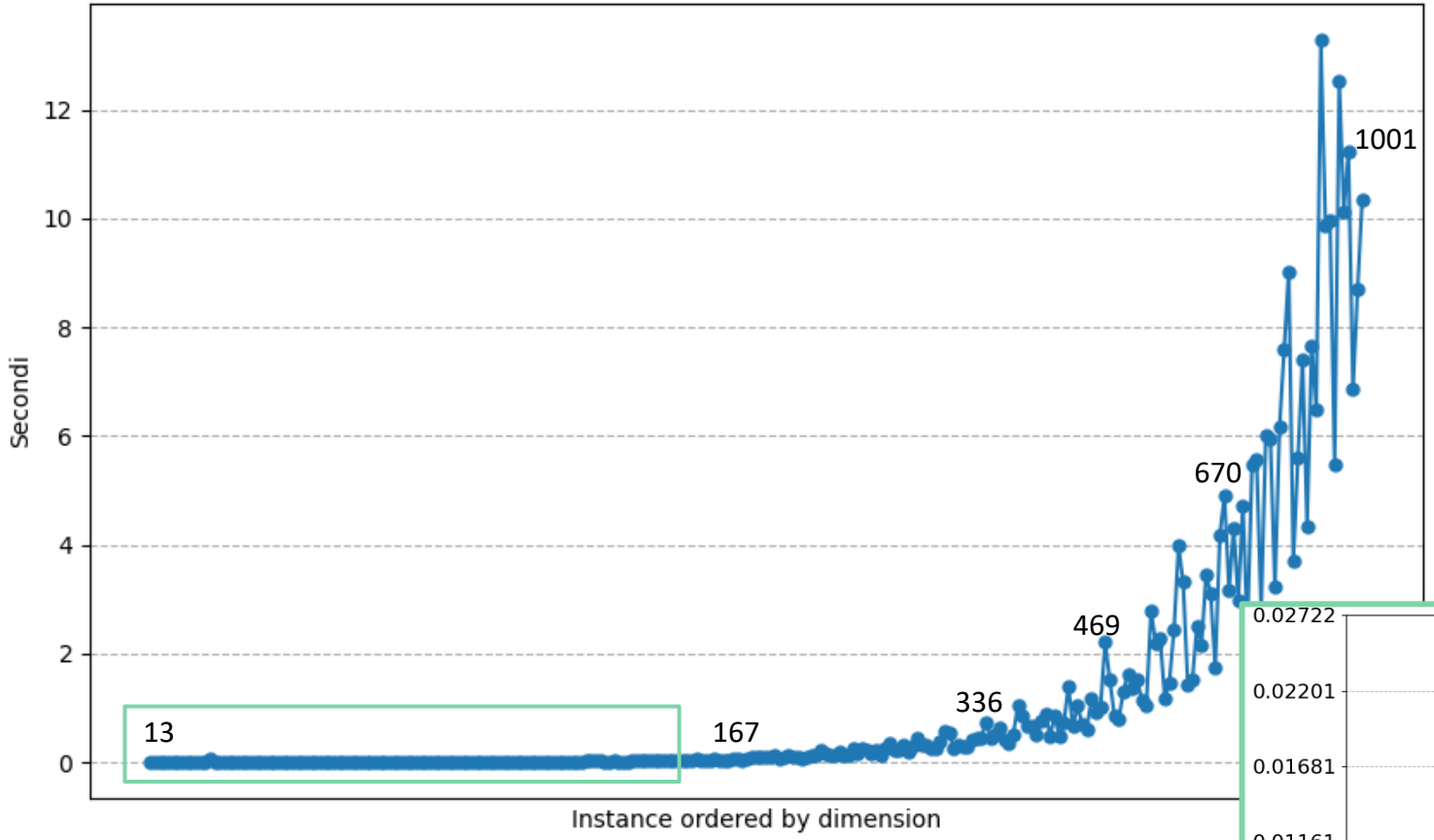
**Unione delle rotte**

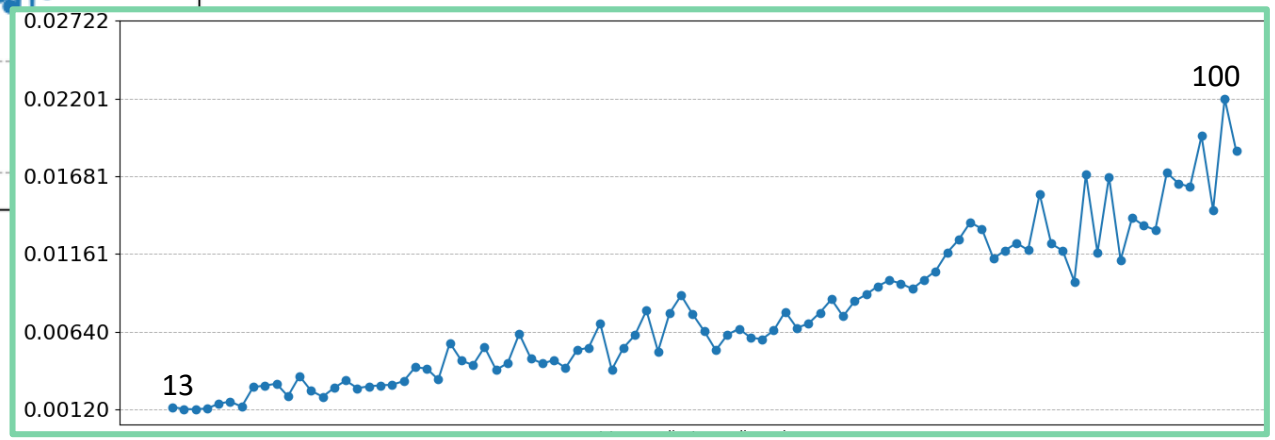
Le rotte di i e j vengono **unite solo se**:

- **i** è in prima posizione e **j** in ultima, o viceversa.
- $S_{ij} > 0$
- La capacità del veicolo è maggiore della domanda totale sulle due rotte.

# Clarke & Wright -Tempi

Tempo di esecuzione all'aumentare della dimensione del problema




Per dimensioni comprese tra **13 e 1250 nodi**

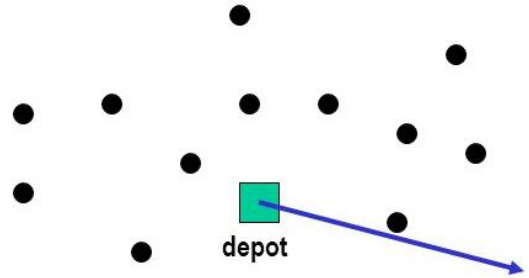
I tempi di esecuzione sono molto bassi:
Da Millisecondi fino ad un massimo di 14 Secondi

# Sweep

**1**

**Calcolo dell'angolo**

Il primo passo consiste nel calcolo dell'angolo tra la **posizione di ogni cliente e il deposito**.




**2**

**Formazione delle rotte**

I clienti vengono aggiunti alla rotta **seguendo l'ordine crescente** degli angoli.

**3**

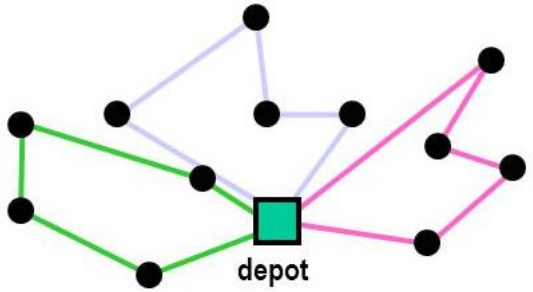
**Verificare la capacità del veicolo**

**Se** la **capacità** attuale del veicolo è **sufficiente** al soddisfacimento della domanda del cliente, esso viene inserito nella rotta. **In caso contrario**, la rotta viene terminata e si inizia **una nuova rotta** con un nuovo veicolo.

**4**

**Ripetizione del Processo**

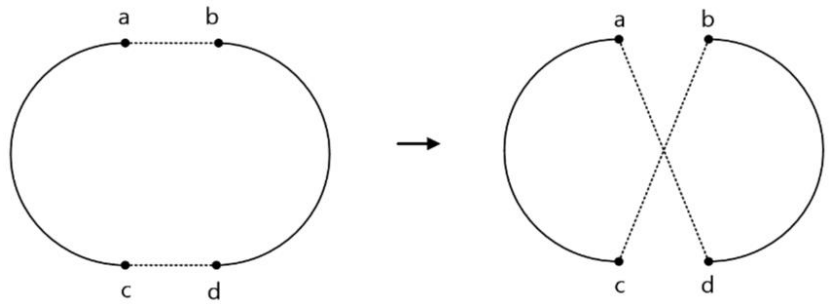
Si ripete il processo finché **tutti i clienti** non sono stati **serviti.**



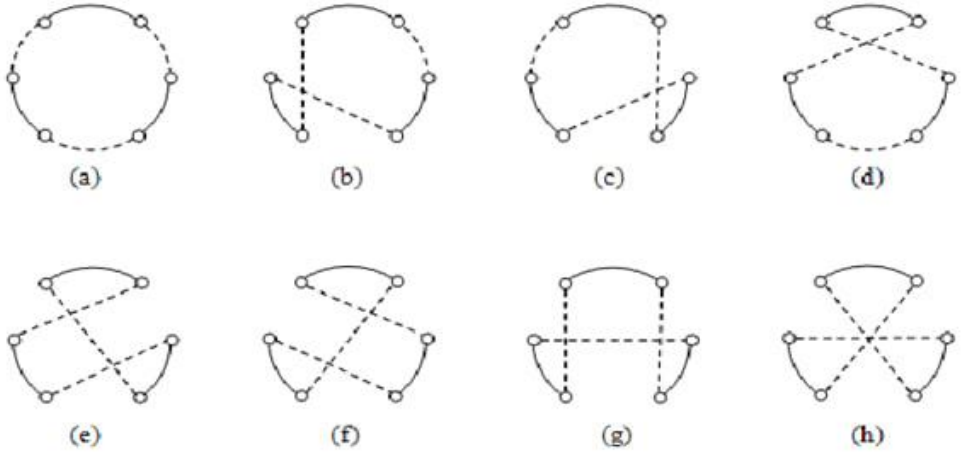
# 2-Opt e 3-Opt

2-Opt e 3-Opt sono **metodi di ottimizzazione** di una soluzione esistente per il problema **del TSP**, che può essere **estesa al VRP**, reiterando **per ogni rotta**.

- 2-Opt seleziona **due archi non adiacenti** e li rimuove, lasciando 4 nodi scollegati. Tali nodi **vengono ricollegati in modo alternato**, formando un nuovo percorso. Se il costo è minore del precedente, esso viene confermato.

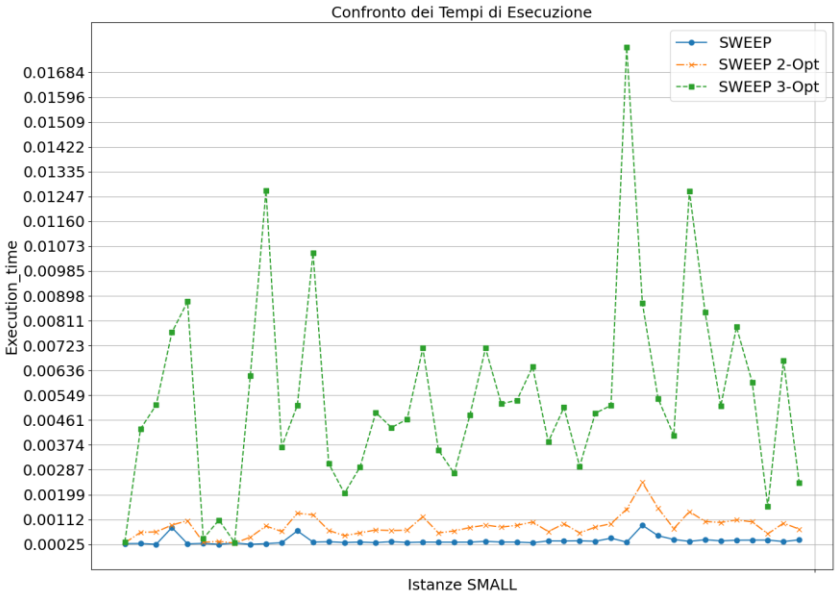


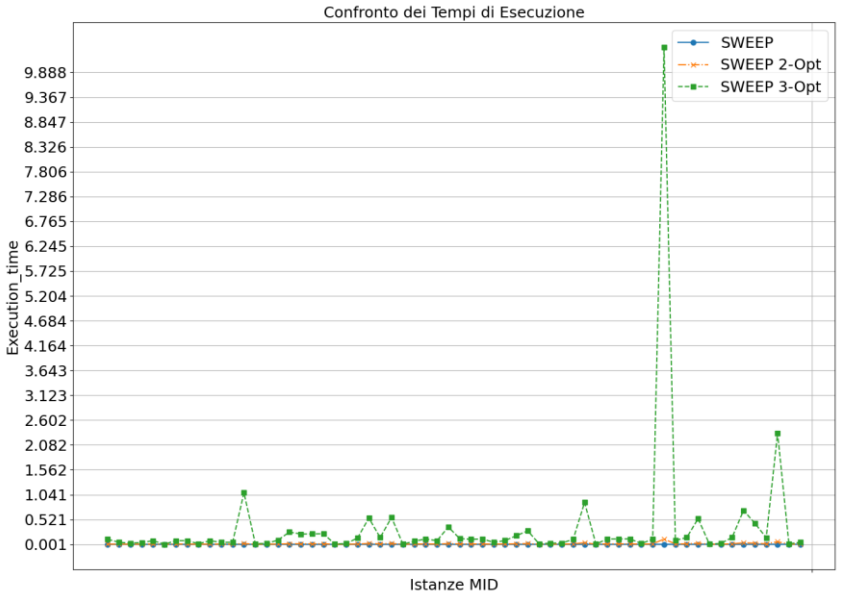

- 3-Opt estende il concetto 2-Opt selezionando **tre archi non adiacenti**. Si valutano i costi degli **otto possibili nuovi percorsi.** Verrà selezionato il percorso con il costo totale più basso.

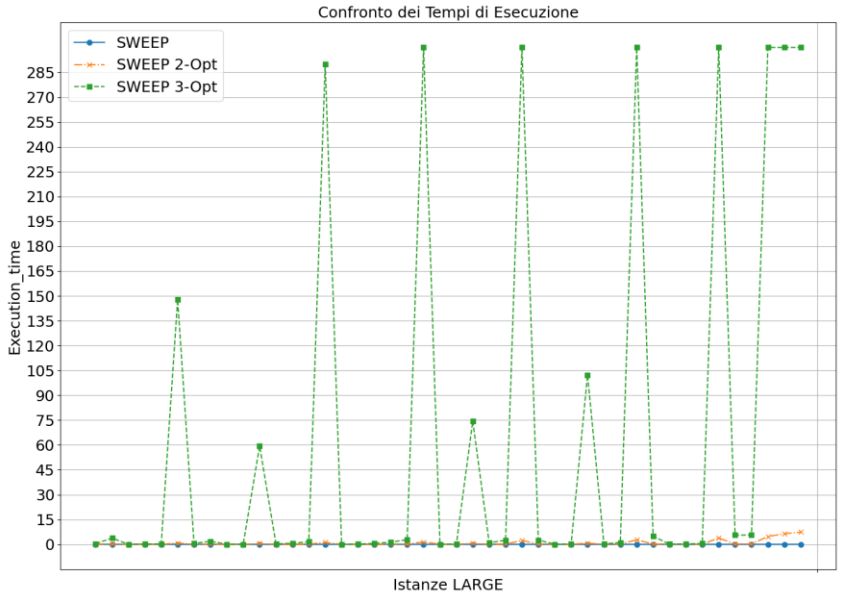



Questi processi vengono ripetuti finché non si trova più alcun miglioramento.

# Sweep vs 2-Opt vs 3-Opt - Tempi

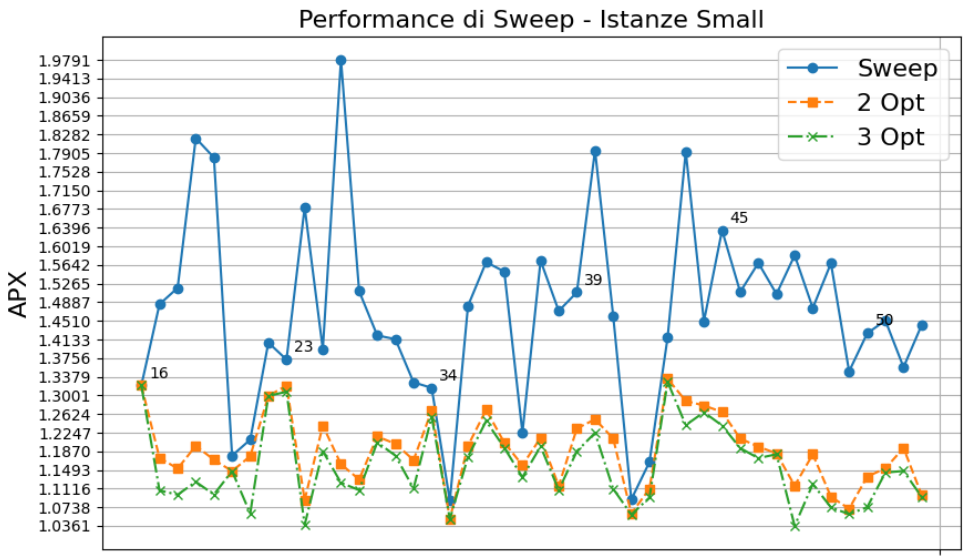


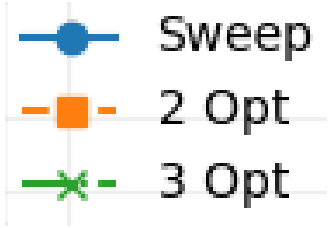



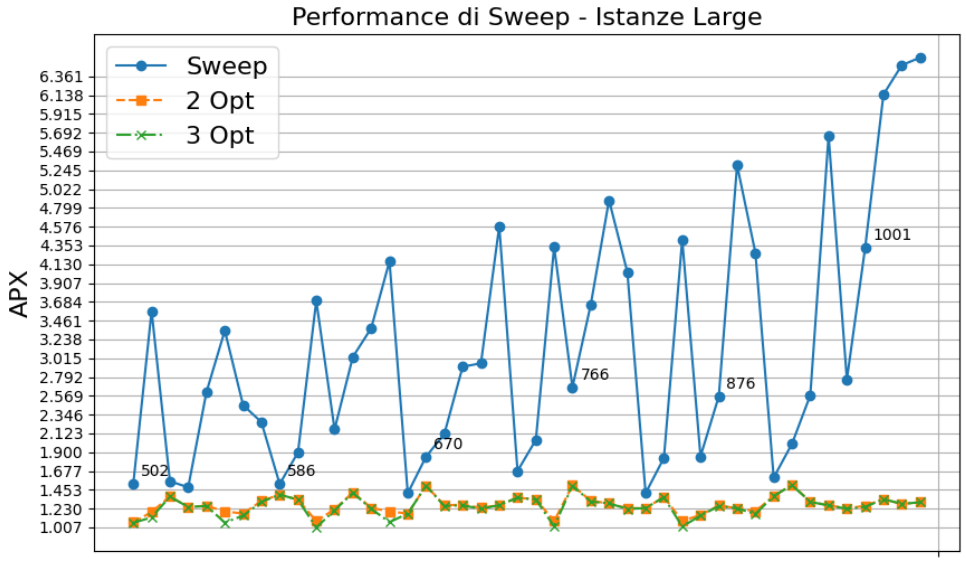



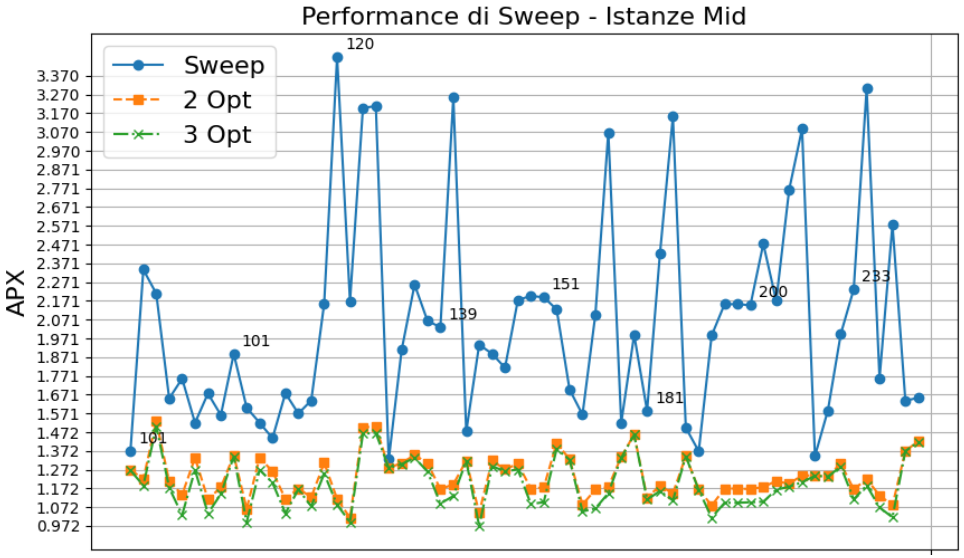

Notiamo uno strano comportamento **in alcune istanze large**, **l'APX e i tempi di esecuzione coincidono o sono molto simili**, questo avviene poiché, in queste istanze particolari, **la soluzione trovata con 2-Opt prevede moltissime rotte**, **ma con pochi clienti in ciascuna**. In questo caso, **3-Opt effettua pochi scambi** (o non li effettua affatto).

# Sweep vs 2-Opt vs 3-Opt - APX



# Algoritmo Random

**Approccio semplice**

Ordinamento casuale dei clienti.
Ogni cliente viene inserito nella prima rotta con capacità rimanente superiore alla sua domanda.

**Nuovo cluster**

Se il cliente non può essere inserito in alcun cluster esistente, ne crea uno nuovo.

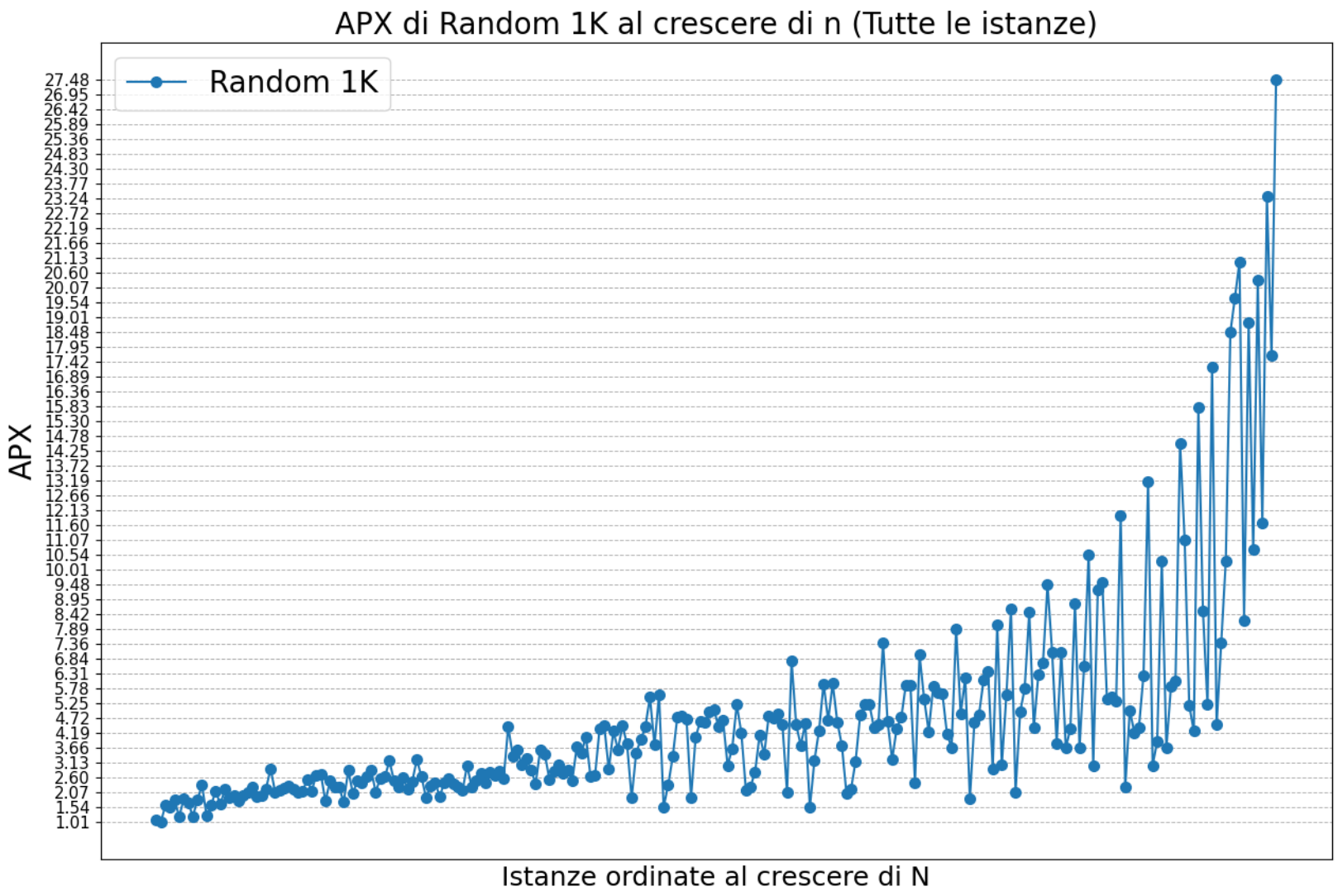
**Vantaggio**

Il punto di forza di un algoritmo random risiede nella sua capacità di essere iterato molte volte grazie alla sua bassa complessità. Viene mantenuta la soluzione migliore tra tutte le esplorate.

**Svantaggio**

La qualità delle soluzioni è altamente variabile poiché fortemente dipendente dal caso.

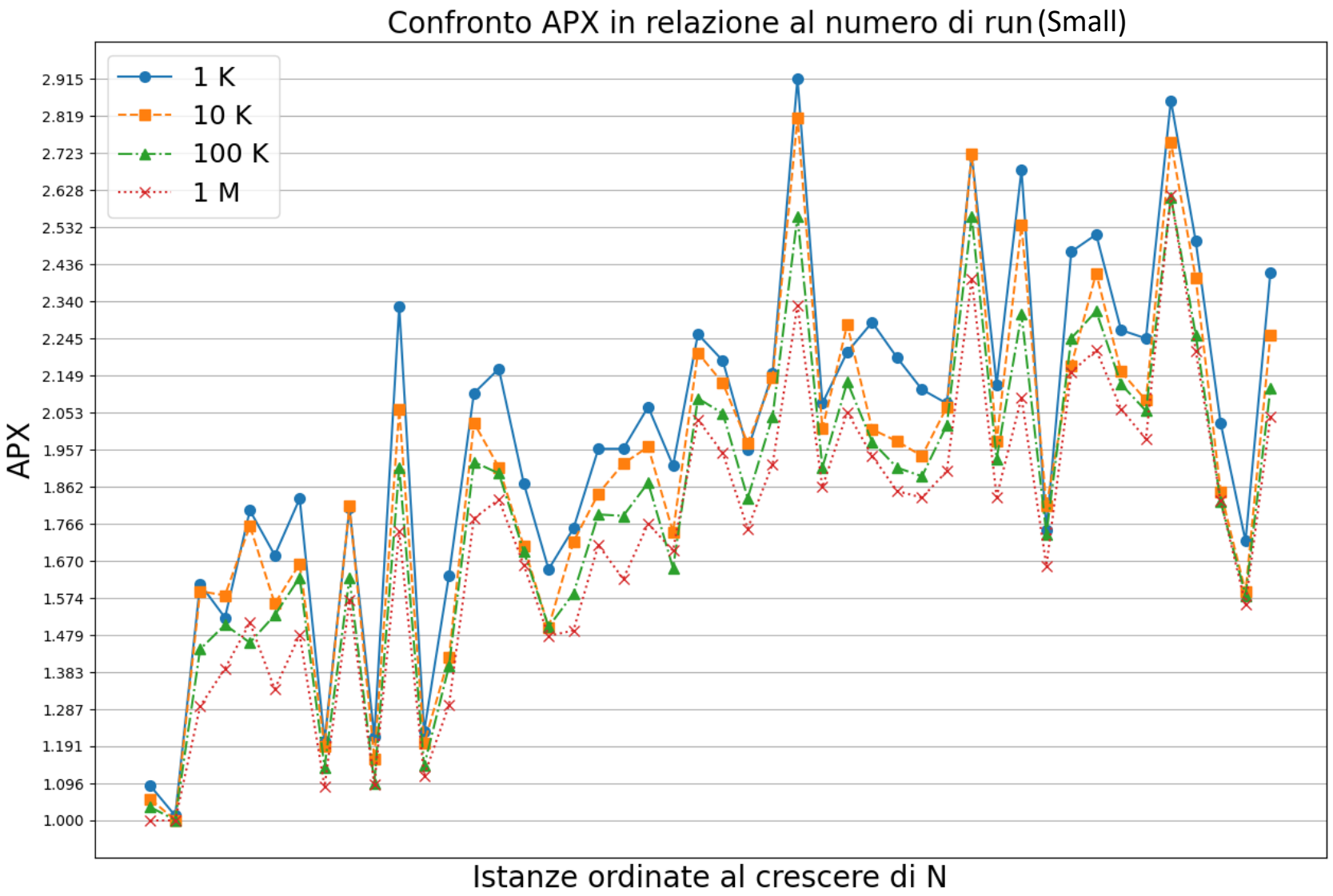
# Algoritmo Random – APX (1K)




Al crescere della dimensione delle istanze, **non ha più senso utilizzare l'algoritmo random** con un numero così **basso di iterazioni (mille)**.

Solo per **4** istanze (**su 273**) il valore dell'APX è **inferiore a 1.5**

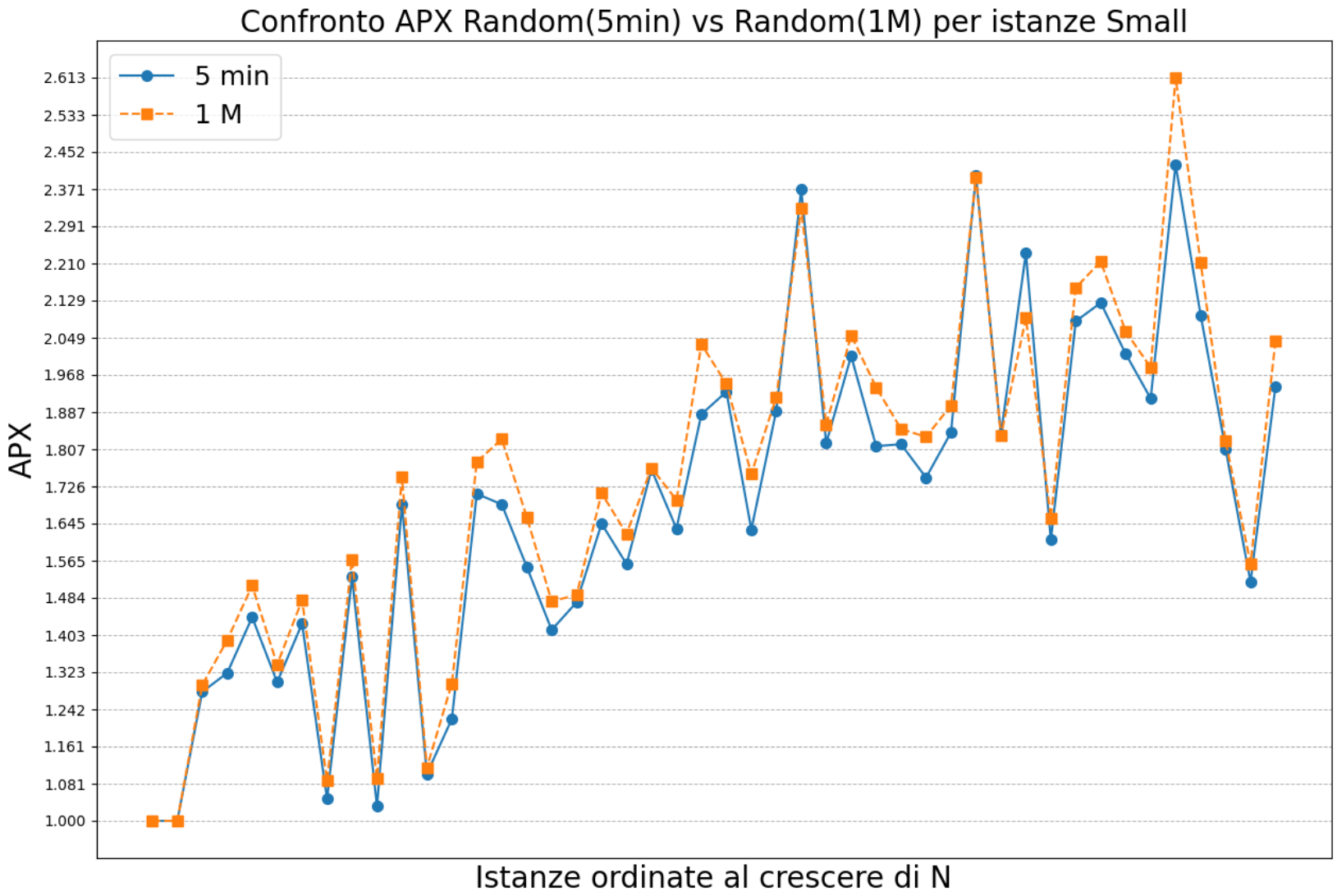
# Algoritmo Random – Numero di Run




Ha senso quindi considerare l'algoritmo per un numero di esecuzioni maggiore.

Con 1 Milione di Run, per le **sole istanze Small**, otteniamo **12** (su 46) istanze con APX inferiore ad 1.5, con un massimo di 60 secondi per istanza.

# Algoritmo Random – 5 minuti

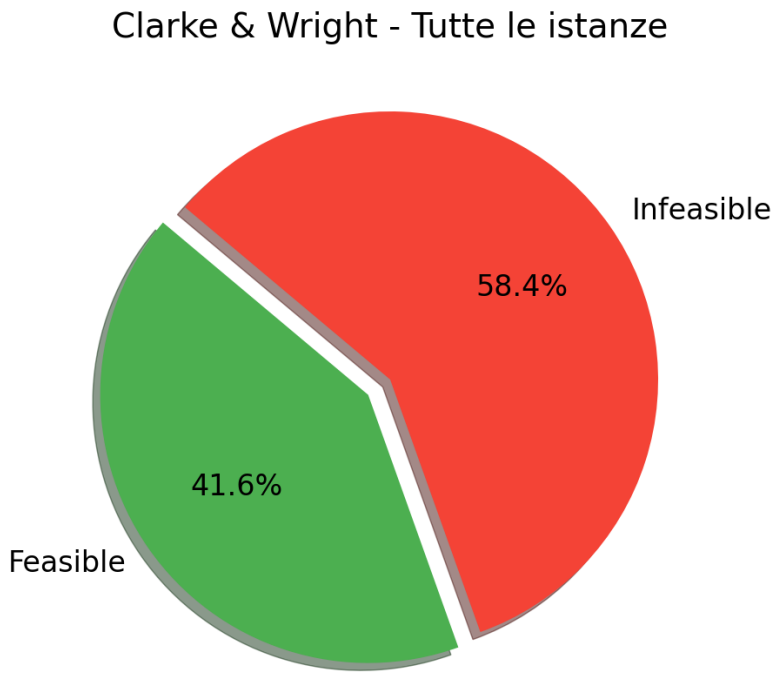


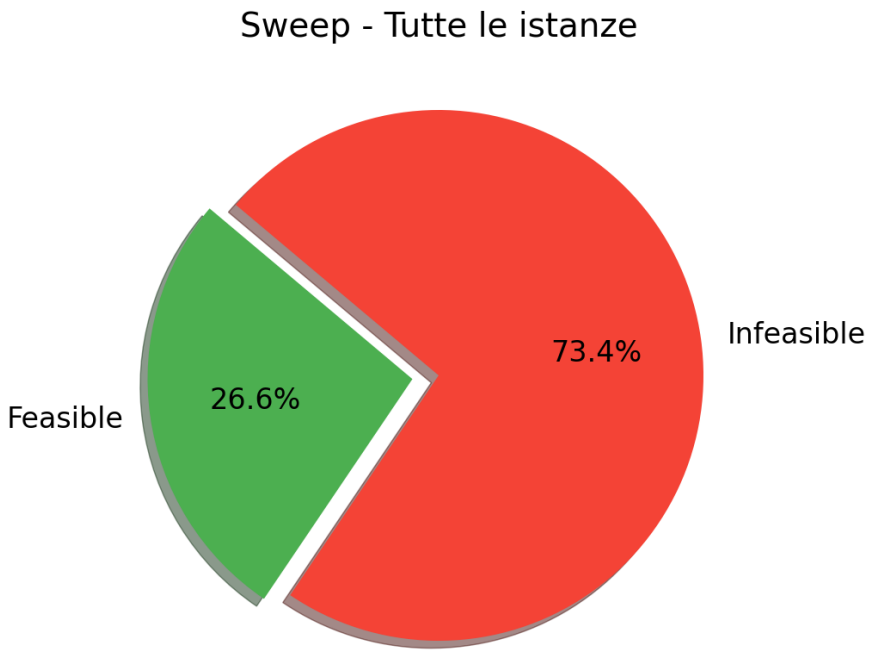

**Idea:** Continuare l'esecuzione dell'algoritmo random per 5 minuti

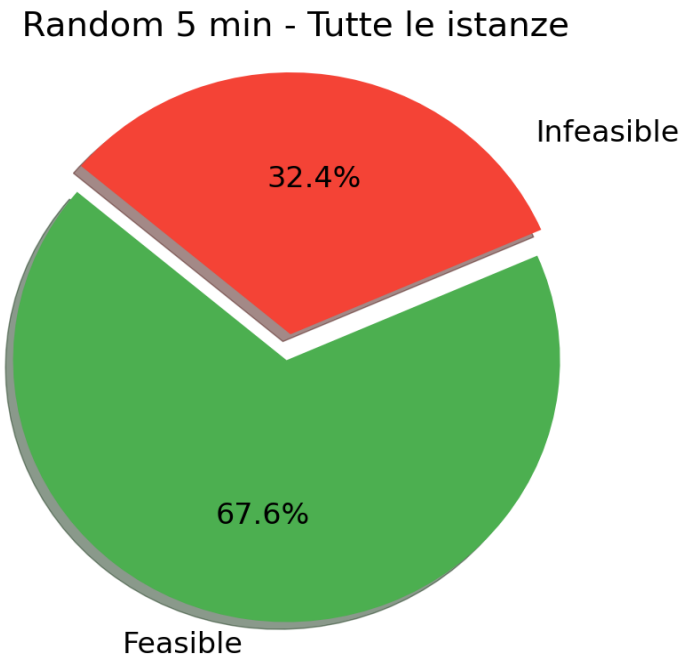
Il miglioramento **non è significativo**, ma è sensato valutare l'algoritmo in questo modo contro i suoi avversari.

# Il problema dei Truck

È importante osservare che gli algoritmi analizzati, per costruzione, **non pongono attenzione al numero di veicoli** resi **disponibili** dall'istanza.
Questo porta le **soluzioni** degli algoritmi studiati a **non** essere **ammissibili.**



# Formulazione MIP (Modello 3 Indici)

## Variabili

- $x_{i,j,h}$: variabile binaria che è 1 se il veicolo $h$ viaggia direttamente dal cliente $i$ al cliente $j$
- $y_{i,h}$: variabile binaria che è 1 se il cliente $i$ è servito dal veicolo $h$, altrimenti 0.
- $u_{i,h}$: variabile continua ausiliaria per l'eliminazione dei sottotour.

## Indici e Parametri

- $i, j \in V$ (insieme dei clienti)
- $h \in K$ (insieme dei veicoli)
- $n$ = numero totale di clienti (escluso il deposito)
- $c_{i,j}$ = costo associato al viaggio dal cliente $i$ al cliente $j$
- $d_i$ = domanda del cliente $i$
- $C$ = capacità del veicolo

**Funzione Obiettivo**

Minimizzare il costo totale di percorrenza

$$\min \sum_{i \in V} \sum_{j \in V} \sum_{h \in K} c_{i,j} \cdot x_{i,j,h}$$

# Formulazione MIP (Vincoli)

1. Ogni Cliente Deve Essere Visitato Esattamente Una Volta

$$\sum_{j \in V, j \neq i} \sum_{h \in K} x_{i,j,h} = y_{i,h} \quad \forall i \in V,\ h \in K$$

$$\sum_{i \in V, i \neq j} \sum_{h \in K} x_{i,j,h} = y_{j,h} \quad \forall j \in V,\ h \in K$$

2. I Veicoli Devono Partire e Tornare al Deposito

$$\sum_{j \in V, j \neq 0} x_{0,j,h} = 1 \quad \forall h \in K$$

$$\sum_{j \in V, j \neq 0} x_{j,0,h} = 1 \quad \forall h \in K$$

3. Ogni Cliente Deve Essere Servito da Esattamente un Veicolo

$$\sum_{h \in K} y_{i,h} = 1 \quad \forall i \in V$$

4. Vincoli di Capacità

$$\sum_{i \in V} d_i \cdot y_{i,h} \leq C \quad \forall h \in K$$

5. Vincoli di Eliminazione dei Sottotour (MTZ)

$$2 \leq u_{ih} \leq n \quad \forall i \in V \setminus \{0\}, \forall h \in K$$

$$u_{i,h} - u_{j,h} + n \cdot x_{i,j,h} \leq n - 1 \quad \forall i \in V,\ j \in V,\ i \neq j,\ h \in K$$

6. **Vincoli di Interezza**

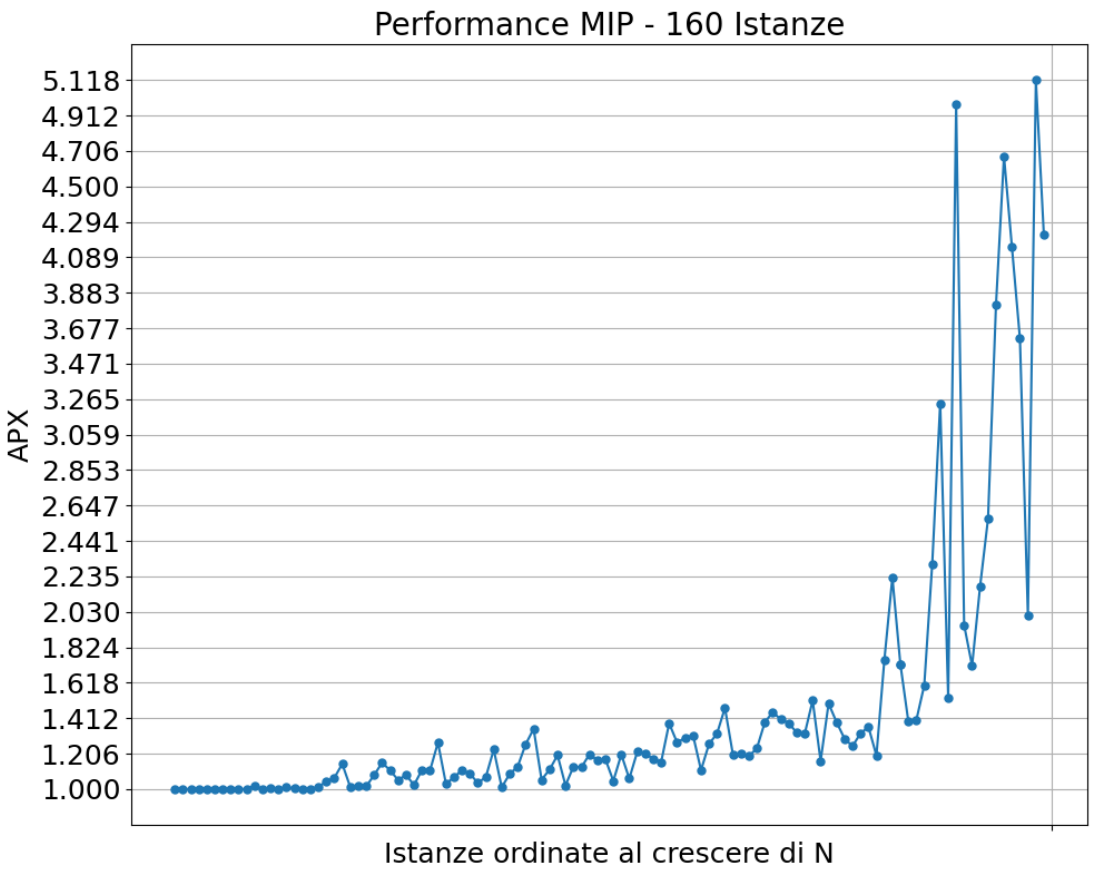
$$x_{ijh} \in \{0,1\} \quad \forall i \in V, \forall j \in V, \forall h \in K$$

$$y_{ih} \in \{0,1\} \quad \forall i \in V, \forall h \in K$$

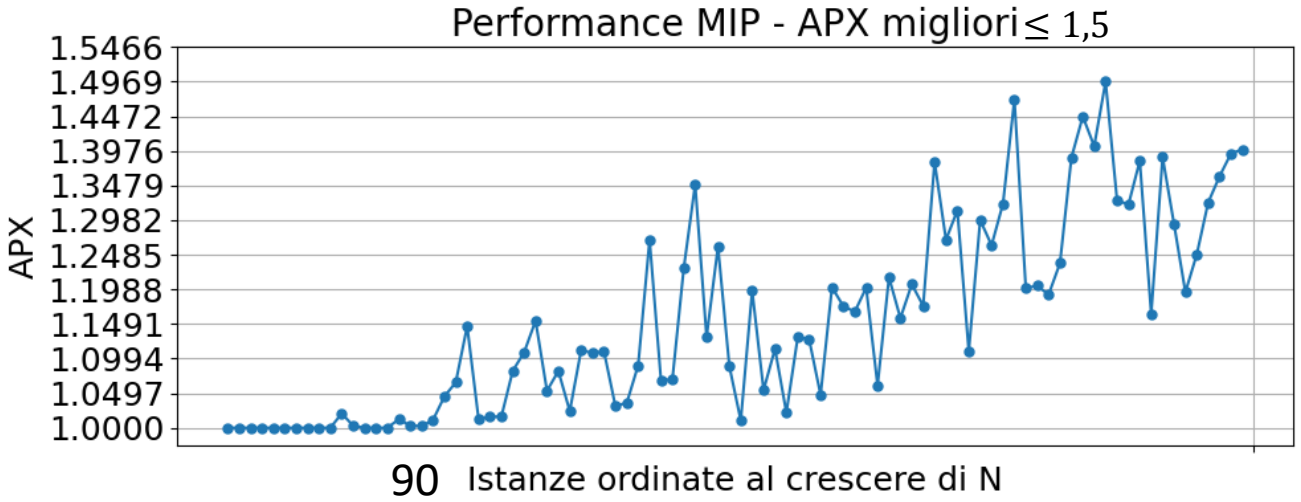
$$u_{ih} \in \mathbb{R} \quad \forall i \in V \setminus \{0\}, \forall h \in K$$

# Formulazione MIP (Gurobi e AMPL)

Abbiamo eseguito il modello MIP sulle prime 160 istanze (13 ore e 20 minuti)
Non è stato eseguito su ulteriori istanze a nostra disposizione, poiché oltre i 200 nodi il modello richiedeva più di 5 minuti nella fase di pre-solve.



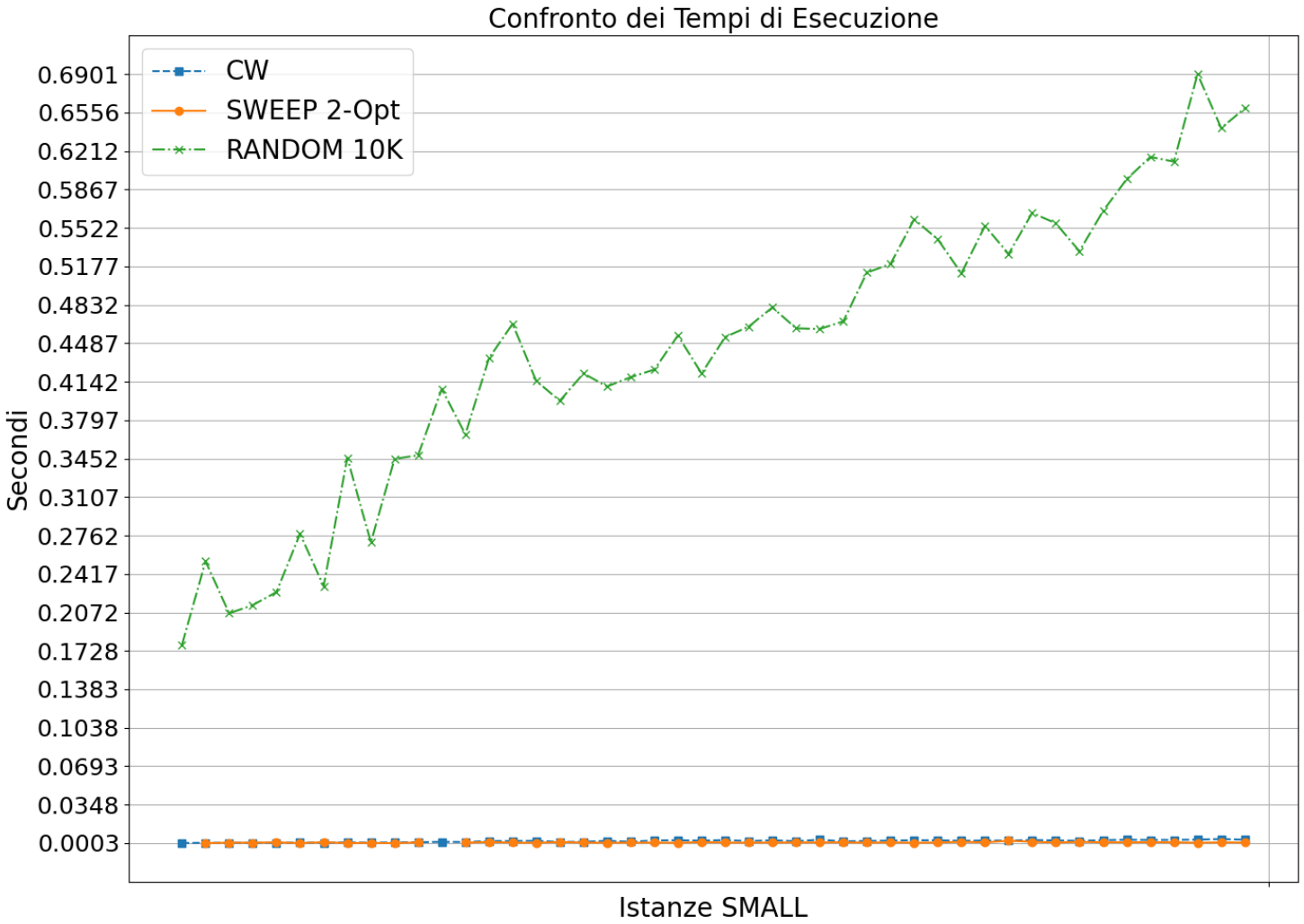

| Istanze Valutate | 160 |
|---|---|
| Soluzioni Ammissibili | 112 |
| Soluzioni Ottime | 10 |
| K non definito | 47 |




90

# Confronti

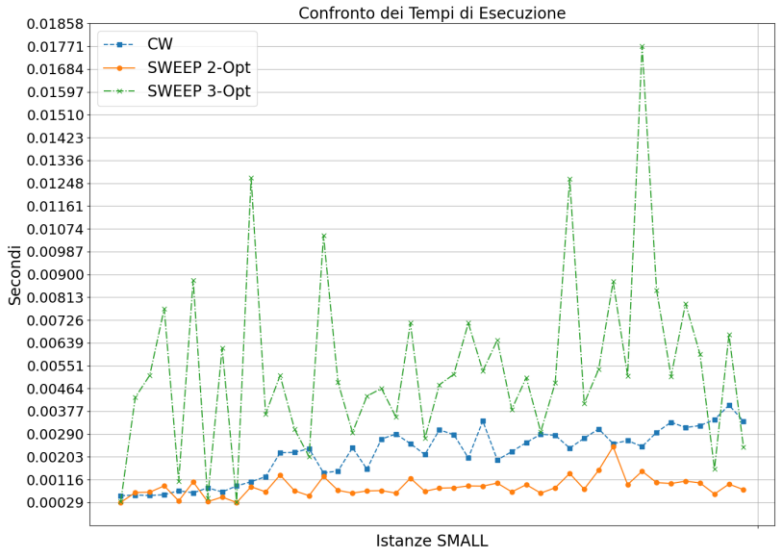
# Tempi

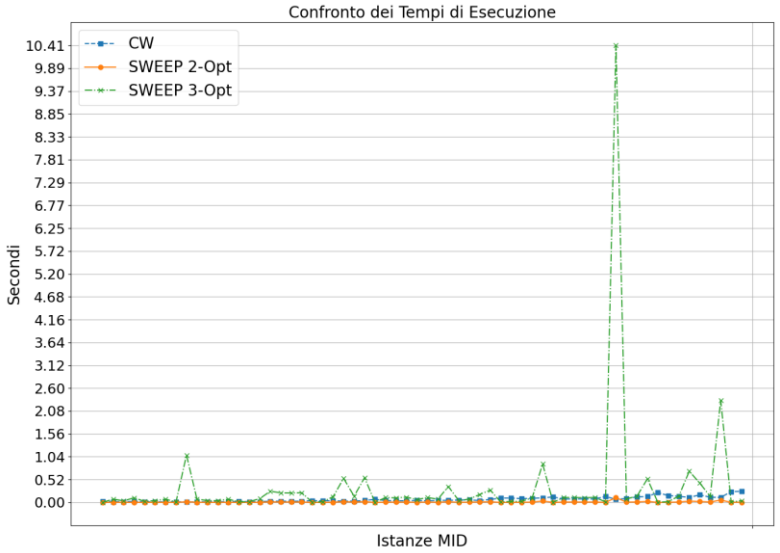


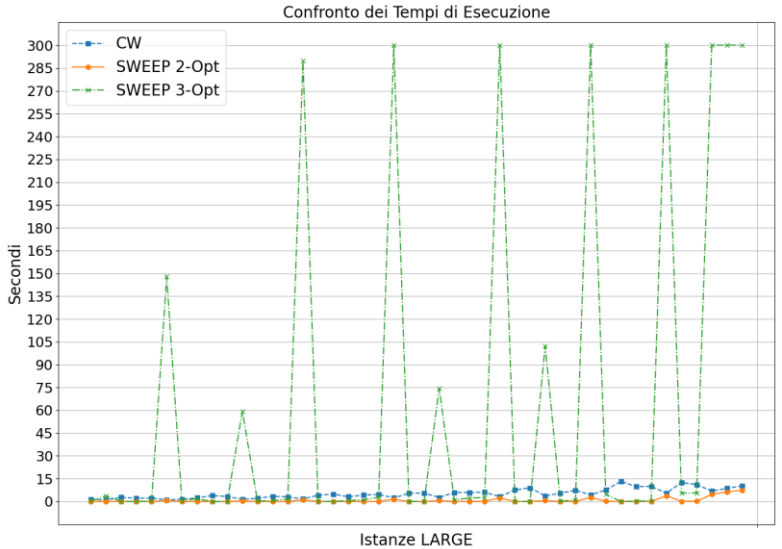

Non vengono considerate le tempistiche del modello MIP con **Gurobi** poiché per quasi tutte le istanze SMALL sono impiegati i 5 minuti massimi concessi.

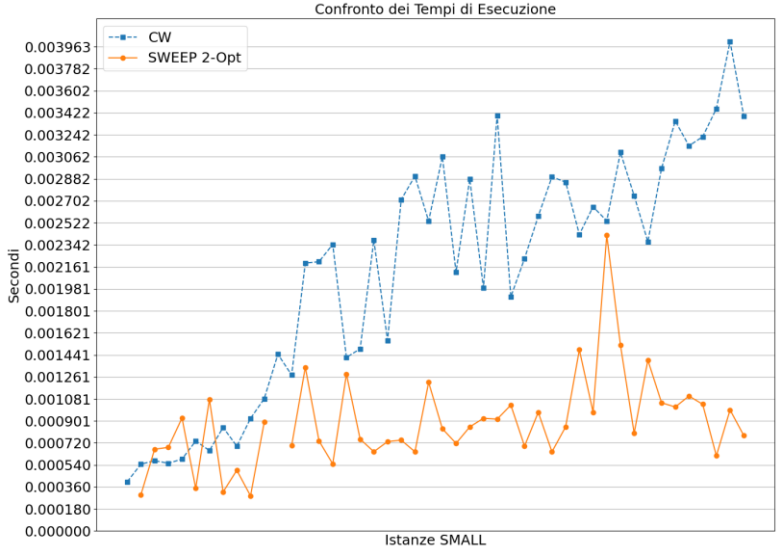
Nonostante i tempi più elevati dell'algoritmo Random, esso è l'unico che può essere parallelizzato su più macchine per ridurre il costo della miglior soluzione.

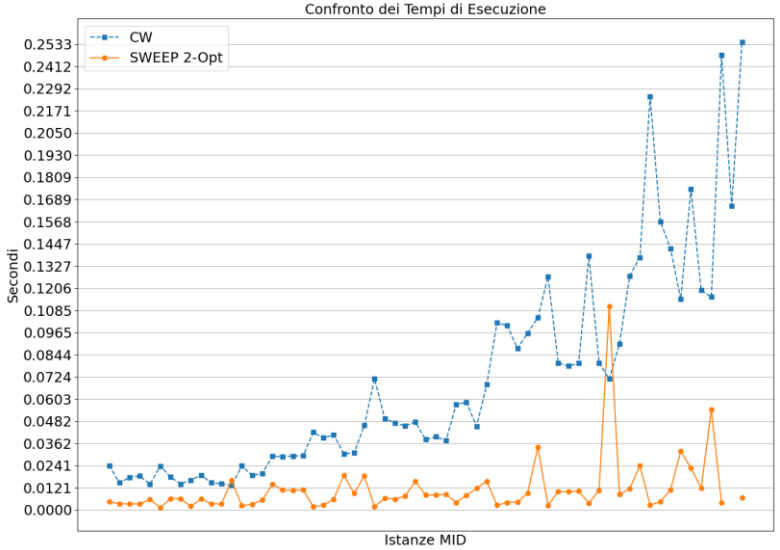
# C&W vs Sweep - Tempi

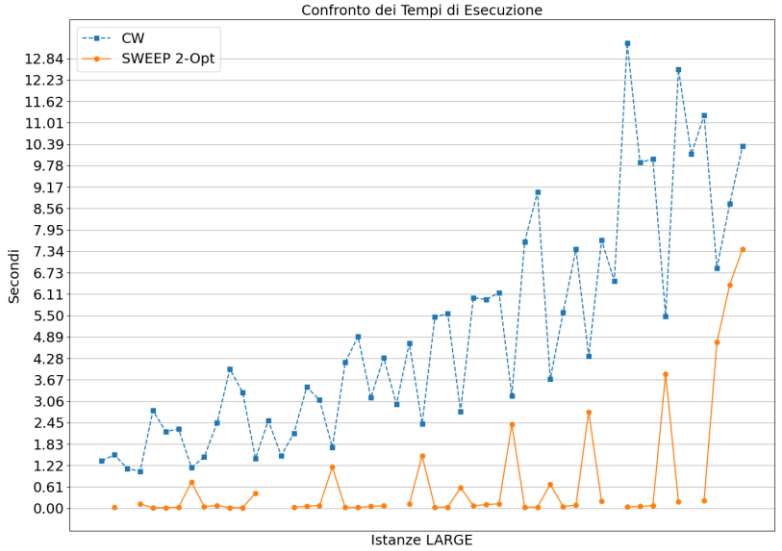


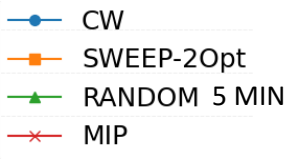



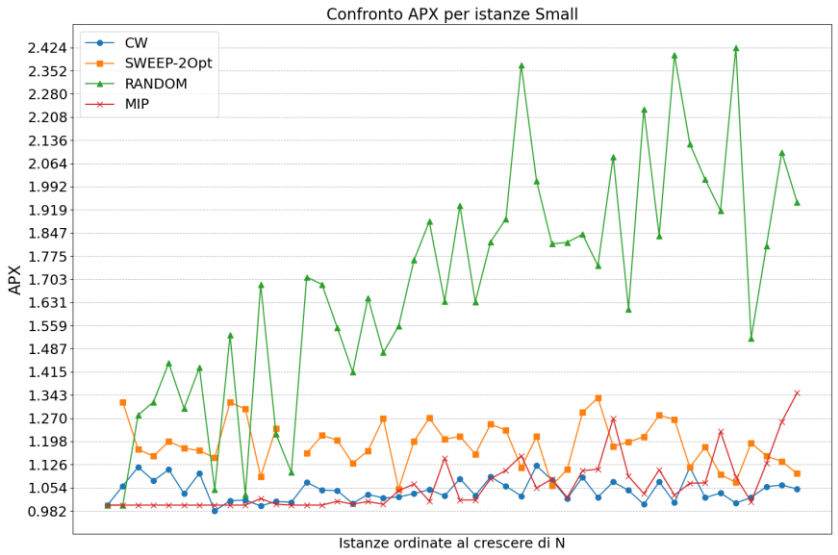



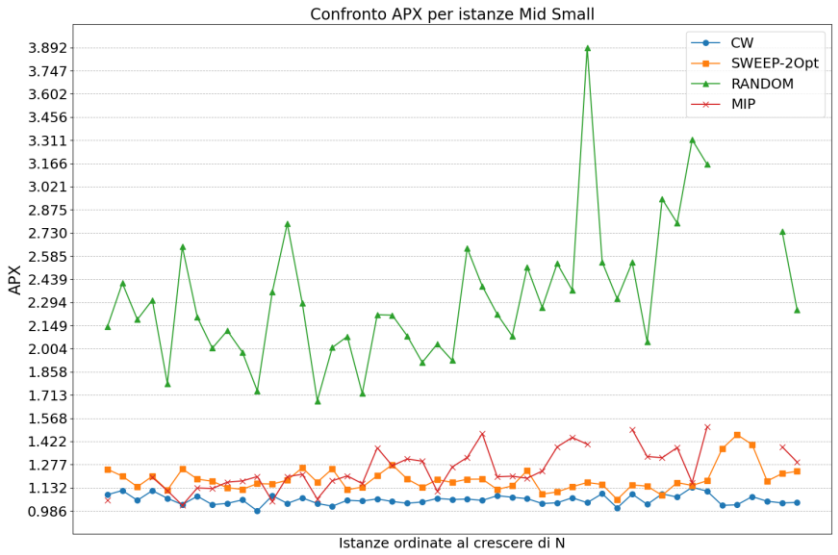



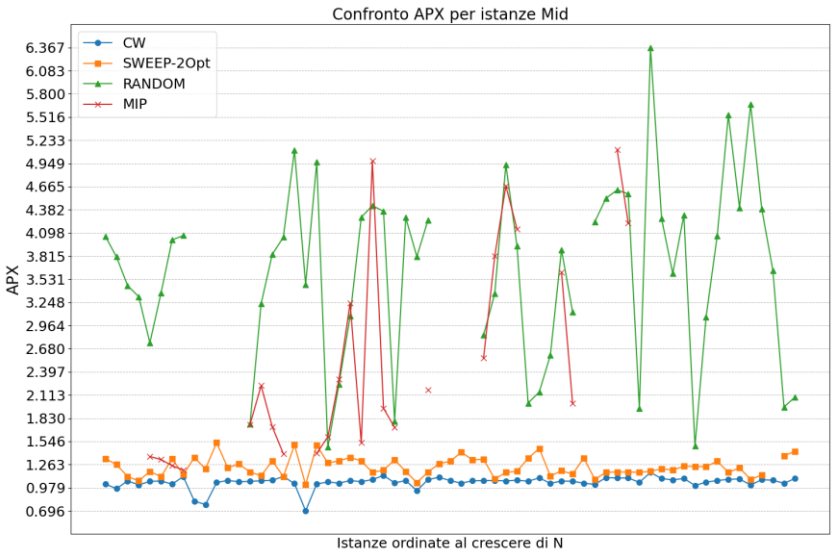



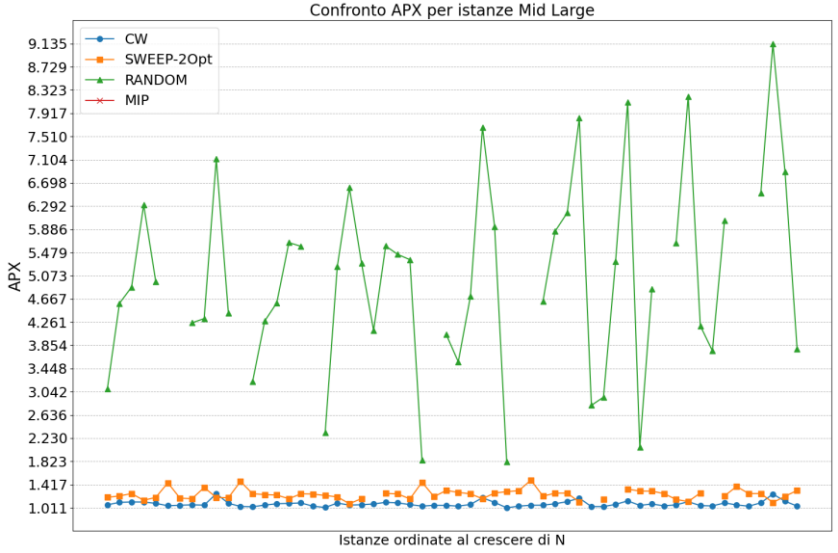
# Qualità delle Soluzioni

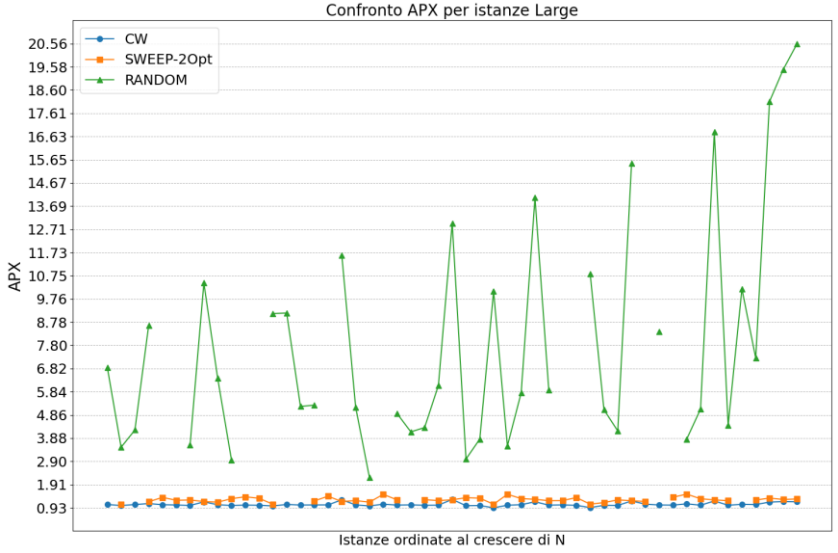


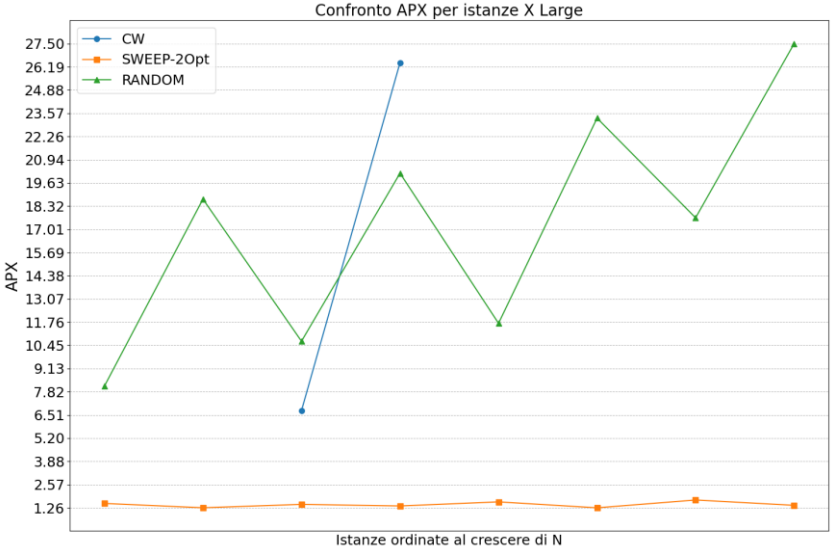



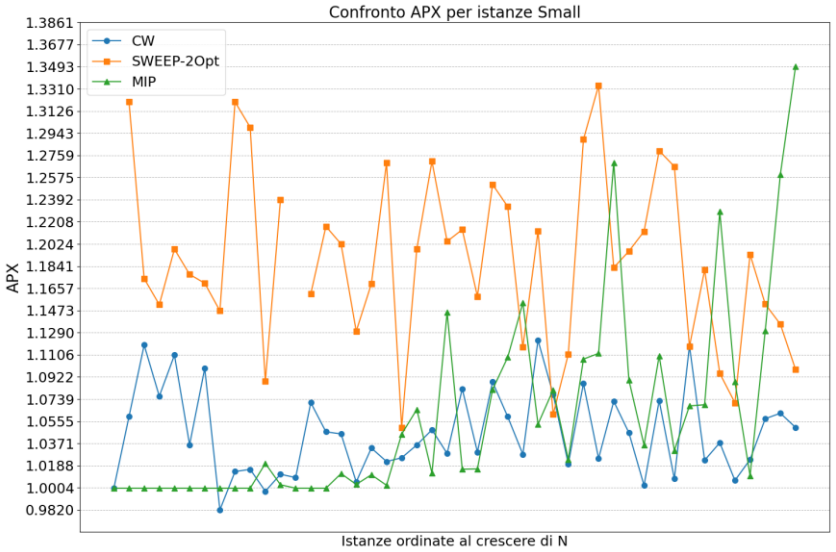



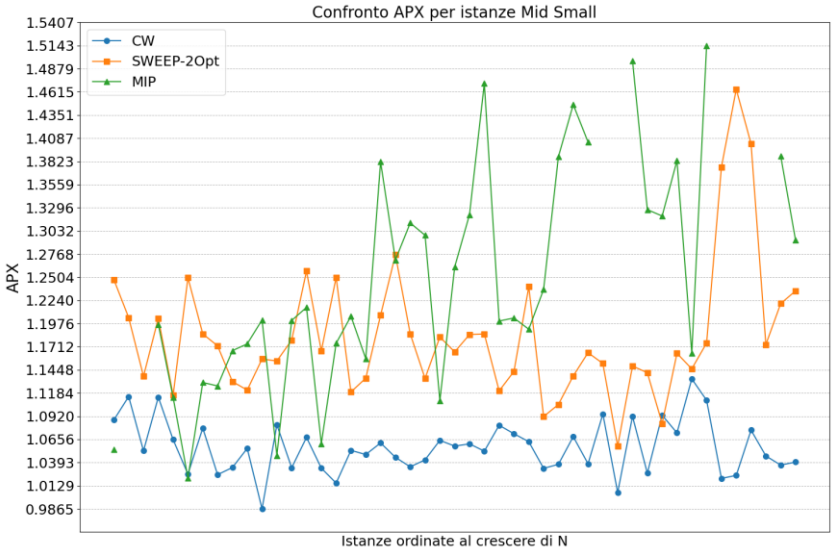



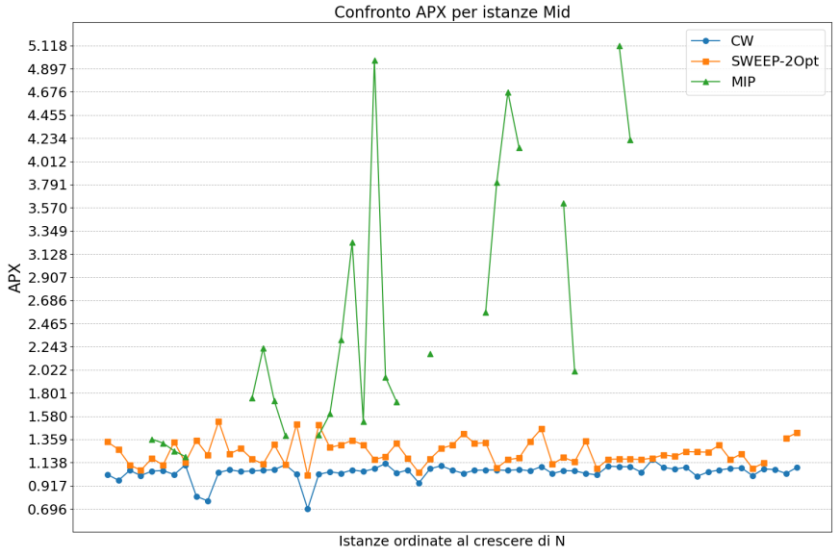



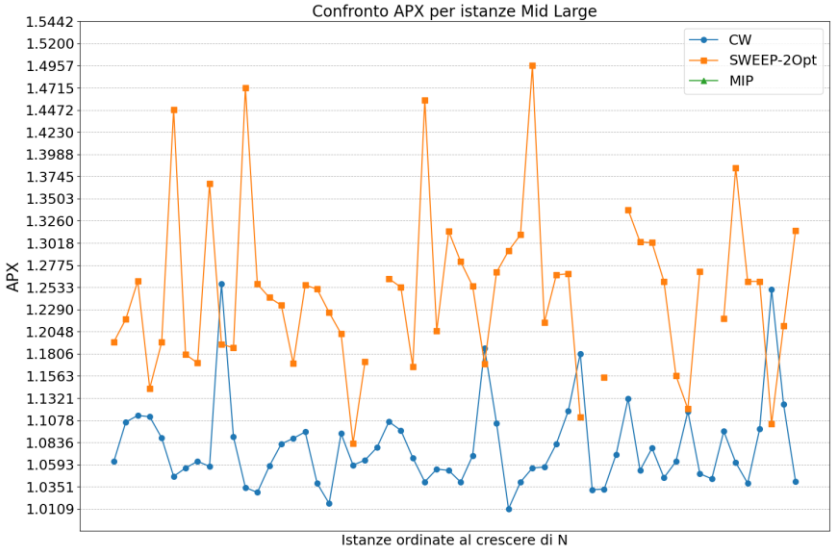



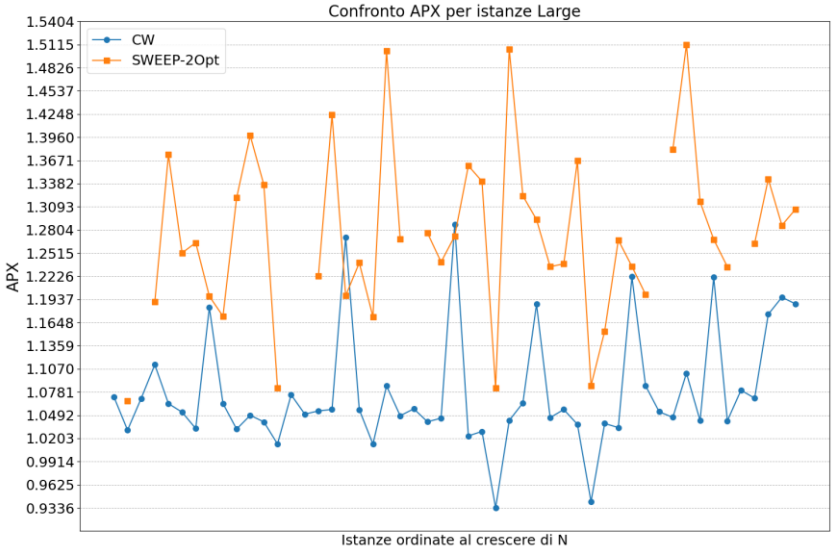
# Qualità delle Soluzioni

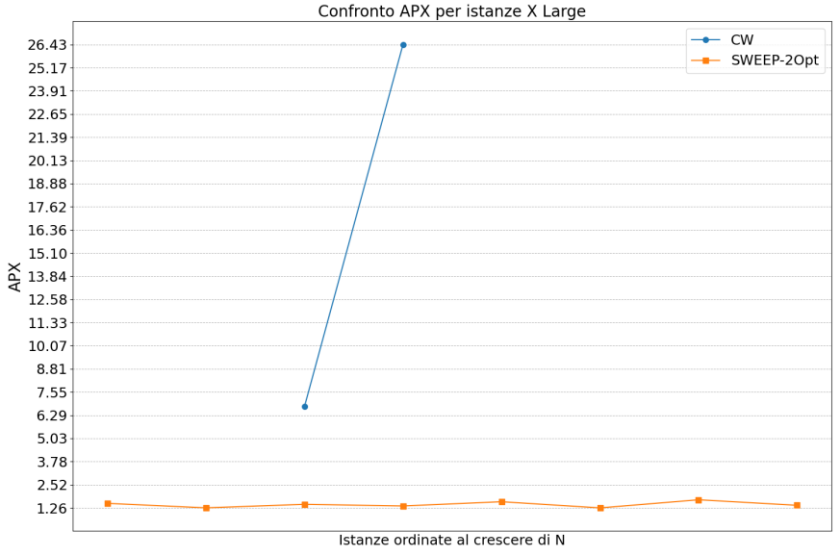


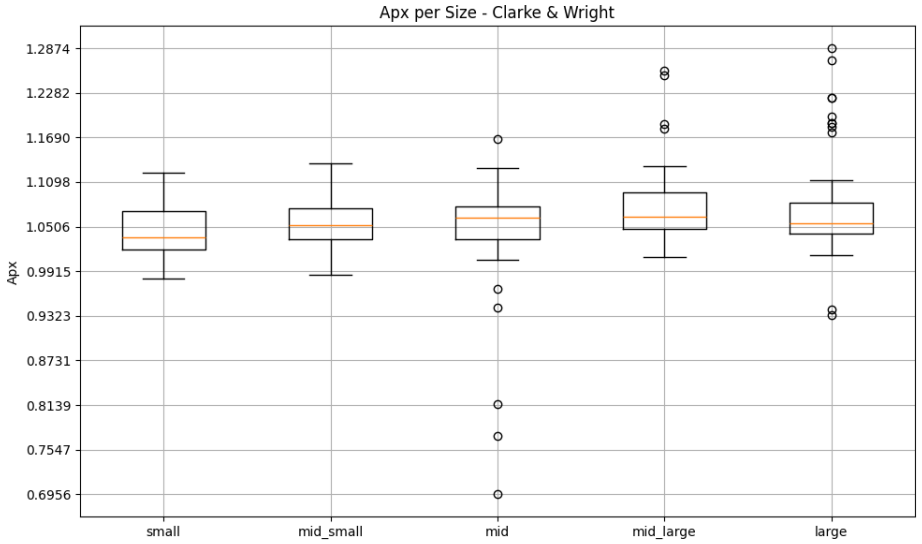



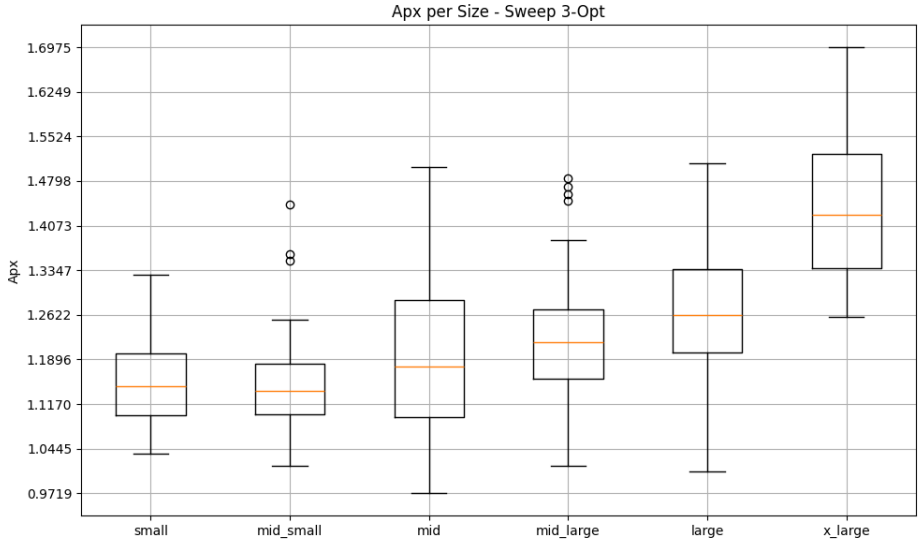



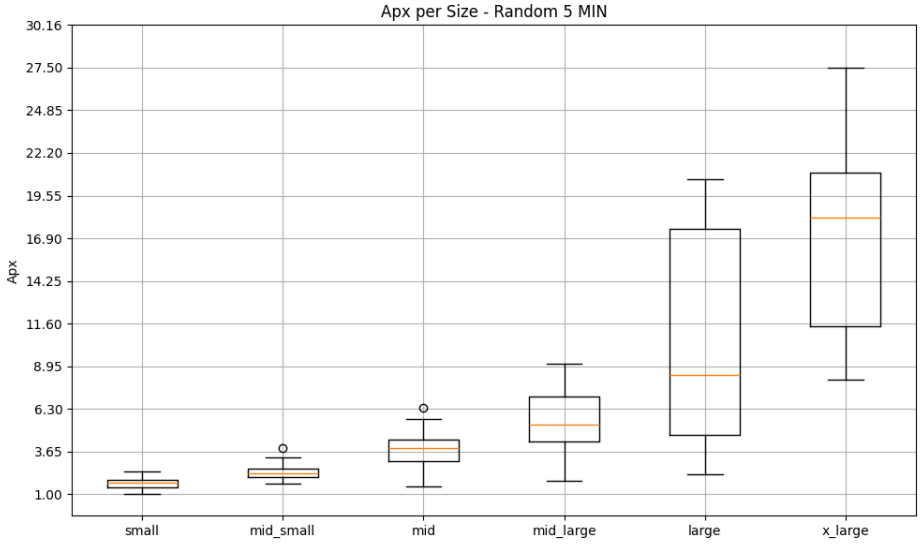



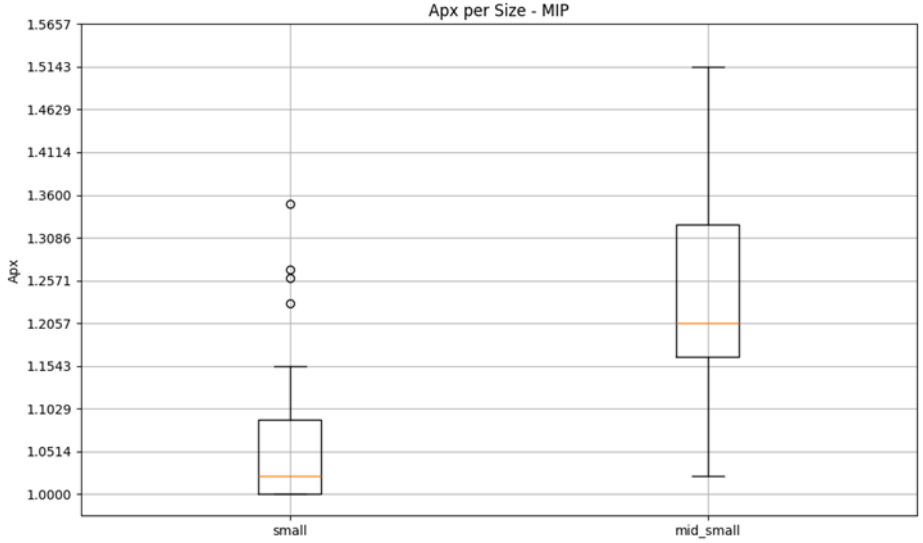
# Qualità delle Soluzioni



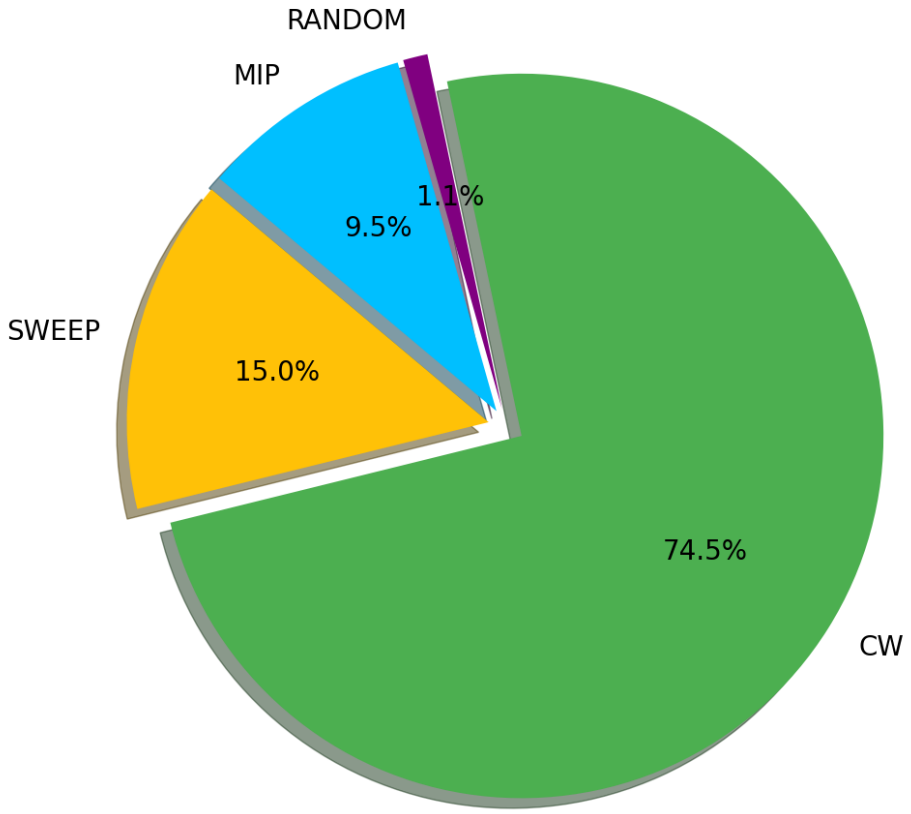

Quanto varia l'APX di ciascun algoritmo al variare della size.



# Algoritmo vincente




Questo grafico rappresenta la percentuale del numero di istanze per cui ogni algoritmo ha ottenuto il **minor costo**, considerando solo le soluzioni ammissibili.

Sweep è stato valutato sulle istanze in cui sono fornite le **coordinate**.
Solo 16 istanze non valutate su 275.

| Numero di soluzioni ottime trovate | |
|---|---|
| Clarke e Wright | 8 |
| Sweep 3-Opt | 1 (Large) |
| Random 5 min | 2 (Small) |
| MIP | 16 (Small) |

# Conclusioni

**Per istanze con dimensioni molto ridotte, sotto i 40 nodi**

- MIP può trovare la soluzione ottima in tempi accettabili, in caso contrario fornisce una soluzione ammissibile accompagnata da un Lower Bound.
- Anche l'euristica Clarke e Wright può fornire buone soluzioni ammissibili.

**Per istanze da mid-small a large**

- L'algoritmo di Clarke e Wright si è distinto per le performance migliori nella maggioranza dei casi, con bassi tempi di esecuzione.

**Per istanze molto grandi, sopra i 2000 nodi**

- L'algoritmo Sweep con 2-Opt si è dimostrato l'unico da prendere in considerazione, per la qualità delle soluzioni e tempi di esecuzione.