Presentazione progetto dell’esame di Ricerca Operativa

Laurea Magistrale in Ingegneria Informatica e dell’Automazione

†

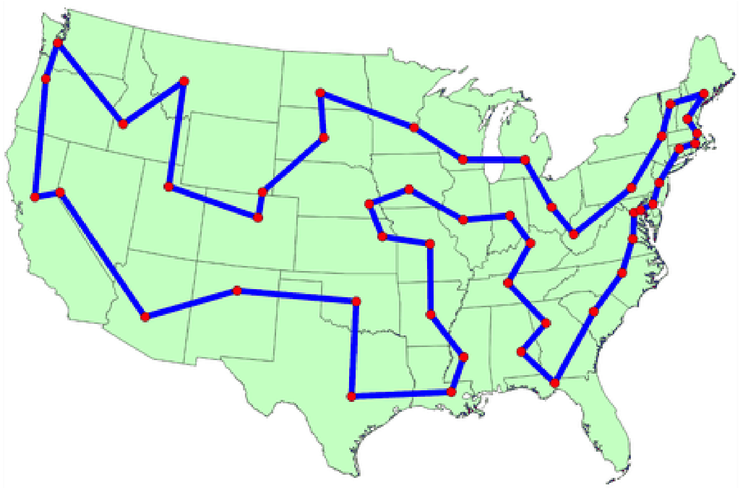
Studente: Luca Padovan

TSP PICKUP AND DELIVERY

A partire dalla base (nodo 0) un corriere deve soddisfare n richieste di prelievo e consegna di documenti (ogni documento è prelevato in un nodo e consegnato in un altro nodo; ogni nodo è riferito a una singola richiesta).

Noto il tempo di percorrenza dei singoli archi, si vuole minimizzare la durata del percorso, con partenza e rientro al deposito.

TRAVELLING SALESMAN PROBLEM



Il *Problema del Commesso Viaggiatore*

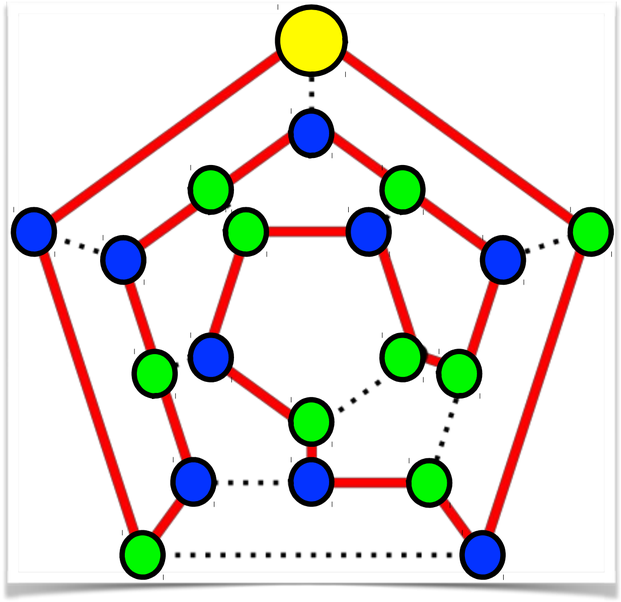
consiste nell’individuare un circuito hamiltoniano di costo minimo in un assegnato grafo orientato G=(V,E)

CASO PARTICOLARE: PICKUP AND DELIVERY

MAGAZZINO (nodo giallo): il corriere deve partire dal nodo e tornarci dopo aver soddisfatto tutte le richieste.

PICKUP (nodo blu):nodo di prelievo dei documenti

DELIVERY(nodo verde): nodo di consegna dei documenti



DEFINIZIONE PROBLEMA

- Si vuole soddisfare le richieste di un insieme di utenti, prelevando e consegnando i documenti. Si tratta quindi di determinare un insieme ottimo di percorsi su grafo rispettando i vincoli dati, in modo da minimizzare la durata del percorso PROBLEMA DI OTTIMIZZAZIONE.

- L’AMMISSIBILITÀ delle possibili SOLUZIONI è data da un insieme di vincoli:
  - 1) Partenza e arrivo al nodo deposito;
  - 2) Precedenze di pickup e delivery.

MODELLO MATEMATICO - NOTAZIONE

- G = (V,E)
- V = insieme dei nodi
- E = insieme degli archi
- X_{ij} definisce se l'arco appartiene al percorso
- $Y_{ij} >= 0$ e $Z_{ij} >= 0$ definiscono il carico prelevato e consegnato

$$\sum_{i \in S} d_i = \sum_{i \in S} p_i = q$$

PRESUPPOSTO: ammontare dei documenti da prelevare uguale all'ammontare dei documenti da consegnare

MODELLO MATEMATICO (1)

$$\min \sum_{(i,j)\in E} C_{ij} X_{ij}$$

Funzione Obiettivo

$$\sum_{(i,j)\in \delta^-(j)} X_{ij} = 1, \;\; j \in V$$

Un arco entrante nel nodo j

$$\sum_{(i,j)\in \delta^+(i)} X_{ij} = 1, \;\; i \in V$$

Un arco uscente dal nodo i

$$\sum_{(i,j)\in \delta^+(S)} X_{ij} \geq 1, \;\; S \subset V : 0 \in S$$

Mi impone la raggiungibilità dal nodo 0, quindi che la soluzione mi fornisca un grafo connesso

MODELLO MATEMATICO (2)

$$\sum_{j \in \delta^+(S)} Y_{ij} - \sum_{k \in \delta^-(S)} Y_{ki} = p_i, i \in S$$

$$\sum_{j \in \delta^+(S)} Z_{ij} - \sum_{k \in \delta^-(S)} Z_{ki} = -d_i, i \in S$$

Flussi che attraversano ciascun nodo modificando il flusso attraverso i requisiti di pickup e di delivery

$$\sum_{j \in \delta^+(S)} Y_{0j} - \sum_{k \in \delta^-(S)} Y_{k0} = -q$$

$$\sum_{j \in \delta^+(S)} Z_{0j} - \sum_{k \in \delta^-(S)} Z_{k0} = q$$

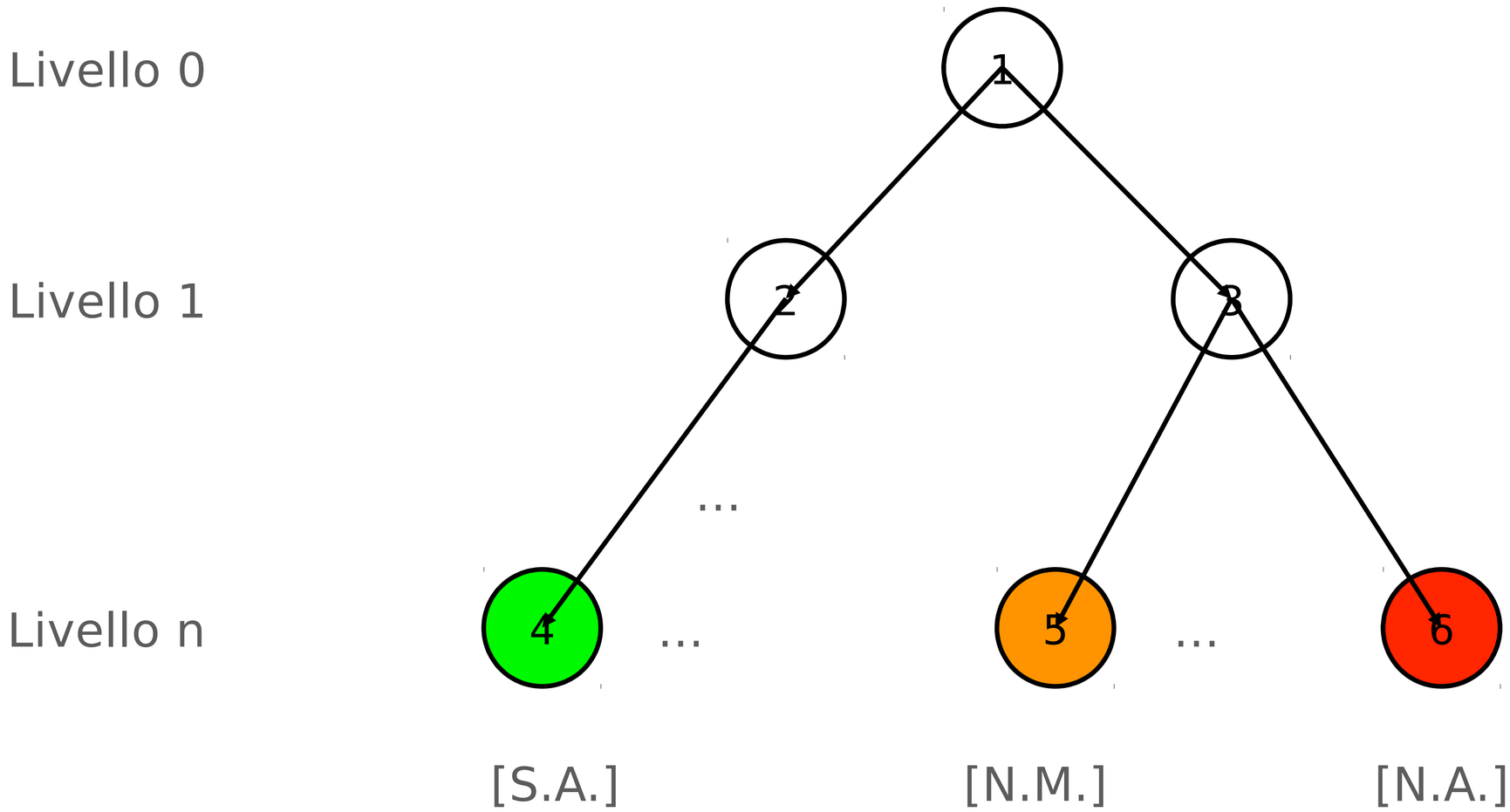
Definiscono il flusso sorgente

Xij ∈ {0,1}, Yij ∈ {0,1}, Zij ∈ {0,1}

PRIMO APPROCCIO...

- Il problema appartiene alla famiglia dei problemi di routing e rilassando i vincoli di precedenza ci si può ricondurre al classico TSP.
- Possiamo applicare lo schema classico del Branch-and-Bound per problemi di programmazione lineare intera in cui si costruisce un albero, basandosi sulla valutazione del lower bound.
- Al nodo radice si considera il problema completo e per ottenere un lower-bound si eliminano i vincoli di precedenza.
- Si risolve il problema risultante come un problema di assegnamento.
- Se la soluzione è ammissibile allora non si sviluppano ulteriori nodi, altrimenti si fa branch generando dei sotto-problemi (nodi figli).

...BRANCH-AND-BOUND...



…OSSERVAZIONI

- Il numero di nodi da valutare, nel caso peggiore, è esponenziale!
- Metodo risolutivo poco efficiente
- Per poter affrontare il problema, NP-Hard, si deve necessariamente ricorrere ad approcci di tipo euristico così da trovare soluzioni ammissibili!

APPLICAZIONE - LINGUAGGIO DI PROGRAMMAZIONE

- Per la soluzione del problema sono stati utilizzati due linguaggi di programmazione:
  - Python per la realizzazione delle euristiche più semplici e per la realizzazione dei grafici grazie alla libreria Matplotlib e Numpy.
  - Java per la realizzazione delle euristiche ispirate a Simulated Annealing e Genetic Algorithm, per via della velocità di esecuzione che un linguaggio compilato consente.

DATI

- Utilizzo di benchmark tratti dal sito https://hhperez.webs.ull.es/

  Il file è strutturato come segue:
  - id, compreso tra 0 e 100 (0=depot)
  - coordinata x
  - coordinata y
  - 6 campi non necessari per il nostro problema
  - campo che indica se il nodo è pickup(1) o delivery(0)

DATI

progetto.py × dati.dat ×

```
 0   50   50    0 9999 9999 9999    0    0 0
 1   21   46    0 9999 9999 9999   10    0 1
 2   24   37    0 9999 9999 9999   10    0 0
 3   29   29    0 9999 9999 9999   10    0 1
 4   37   24    0 9999 9999 9999   10    0 0
 5   46   21    0 9999 9999 9999   10    0 1
 6   55   21    0 9999 9999 9999   10    0 0
 7   64   24    0 9999 9999 9999   10    0 1
 8   71   29    0 9999 9999 9999   10    0 0
 9   77   37    0 9999 9999 9999   10    0 1
10   80   46    0 9999 9999 9999   10    0 0
11   80   55    0 9999 9999 9999   10    0 1
12   77   64    0 9999 9999 9999   10    0 0
13   71   71    0 9999 9999 9999   10    0 1
14   64   77    0 9999 9999 9999   10    0 0
15   55   80    0 9999 9999 9999   10    0 1
16   46   80    0 9999 9999 9999   10    0 0
17   37   77    0 9999 9999 9999   10    0 1
18   29   71    0 9999 9999 9999   10    0 0
19   24   64    0 9999 9999 9999   10    0 1
20   21   55    0 9999 9999 9999   10    0 0
21   23   31    0 9999 9999 9999   10    0 1
22   30   23    0 9999 9999 9999   10    0 0
23   39   18    0 9999 9999 9999   10    0 1
24   50   17    0 9999 9999 9999   10    0 0
25   60   18    0 9999 9999 9999   10    0 1
26   69   23    0 9999 9999 9999   10    0 0
27   77   30    0 9999 9999 9999   10    0 1
28   82   39    0 9999 9999 9999   10    0 0
29   84   50    0 9999 9999 9999   10    0 1
30   83   60    0 9999 9999 9999   10    0 0
```

CONVENZIONE

- Nodi con le caratteristiche precedentemente descritte
- Per misurare la distanza tra due nodi si utilizza la distanza euclidea.

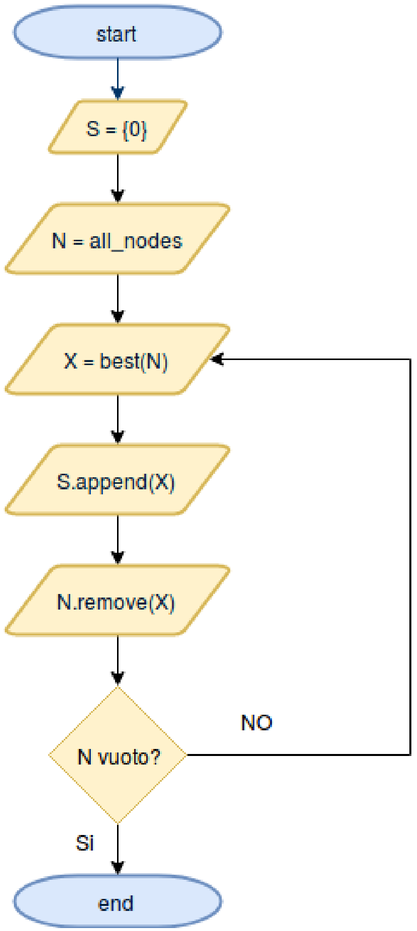
$$d(i,j)=\sqrt{\sum_{k=1}^{n}(x_{ik}-y_{jk})^2}$$

EURISTICHE

- Costruttive:
  - Greedy Best:
    - Pickup First
    - FIFO ordine di richiesta
    - FIFO nearest neighbor
- Neighbor Based:
  - City Swap
  - Tabu Search
  - Simulated Annealing

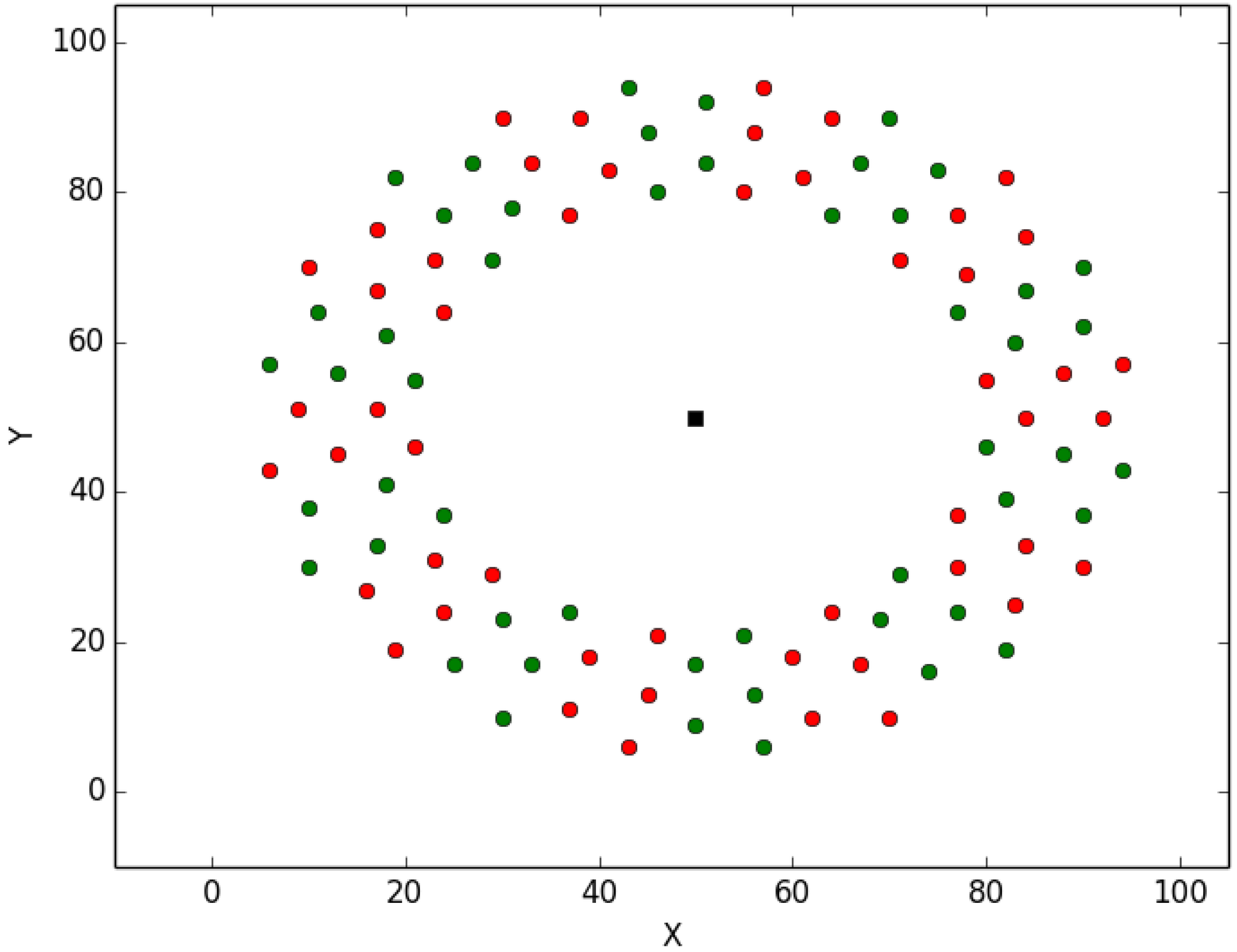
  Population based:
  - Path relinking
  - Genetic Algorithm

ALGORITMO GENERALE GREEDY

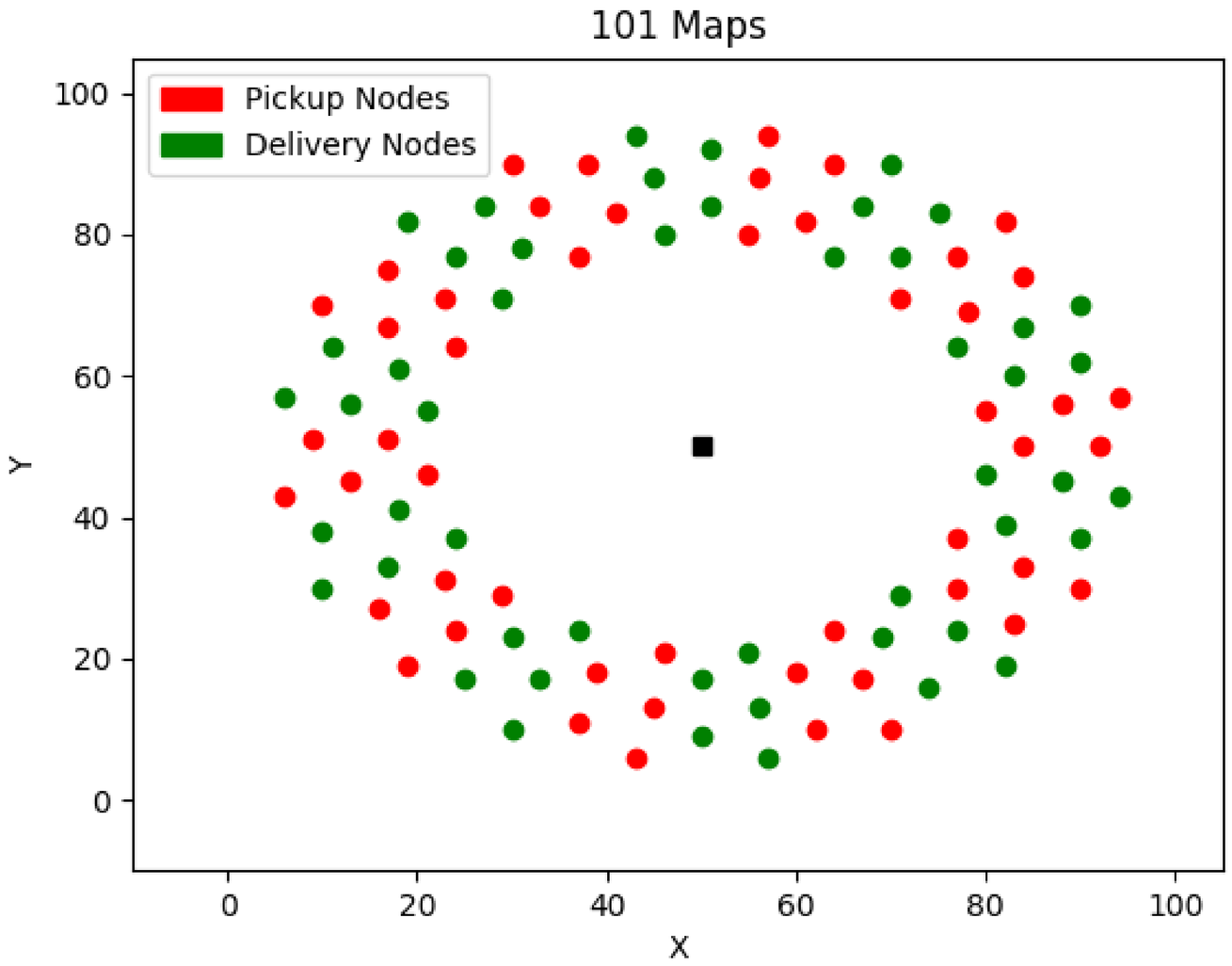


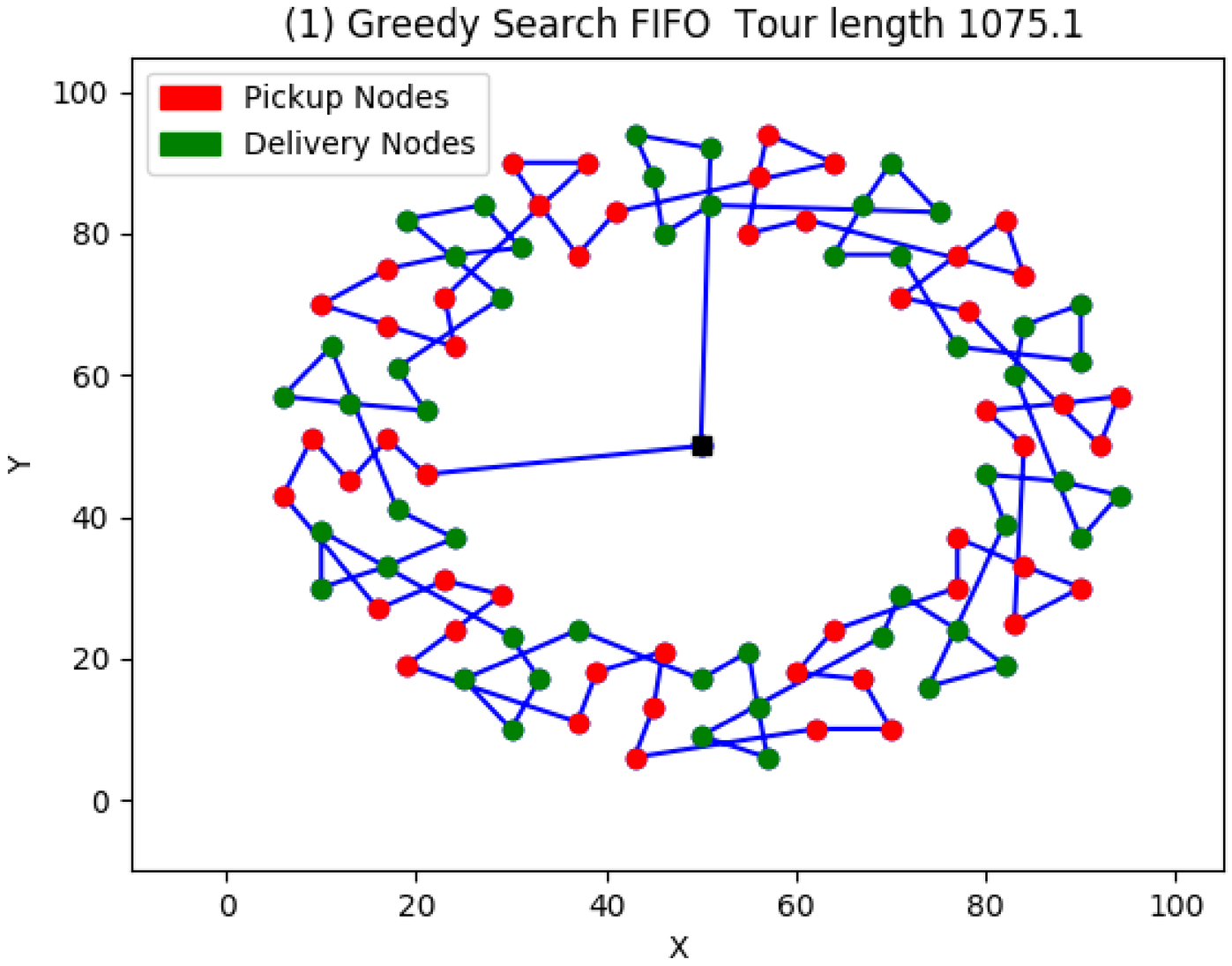
- Facile implementazione e notevole efficienza computazionale.
- Determino la soluzione tramite una sequenza di decisioni parziali (localmente ottime), senza mai modificare le decisioni prese.
- Purtroppo in generale non mi garantiscono l’ottimalità.

Ecco la mappa delle città che il nostro commesso deve visitare. In rosso sono rappresentati i nodi di prelievo, in verde i nodi di consegna.

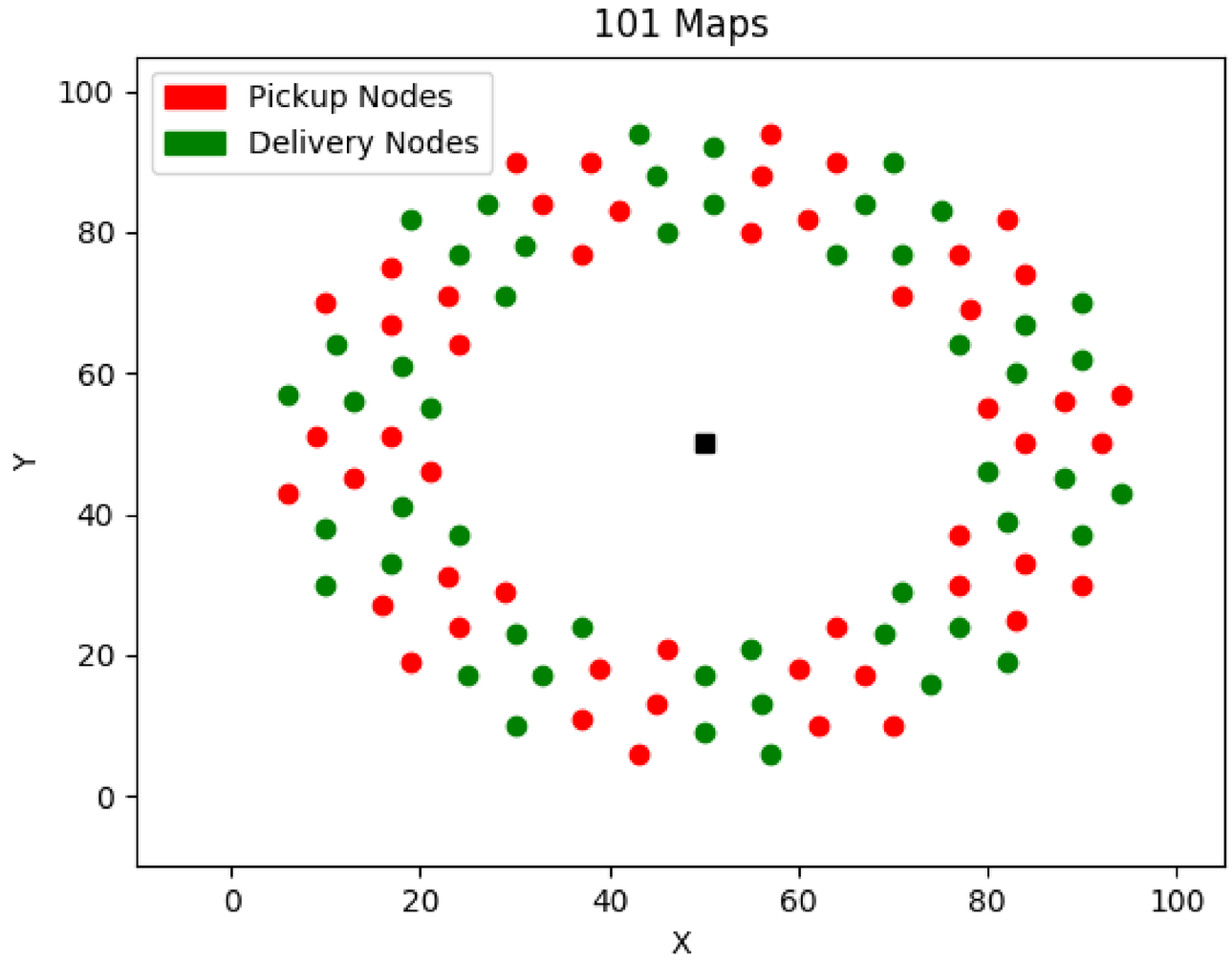


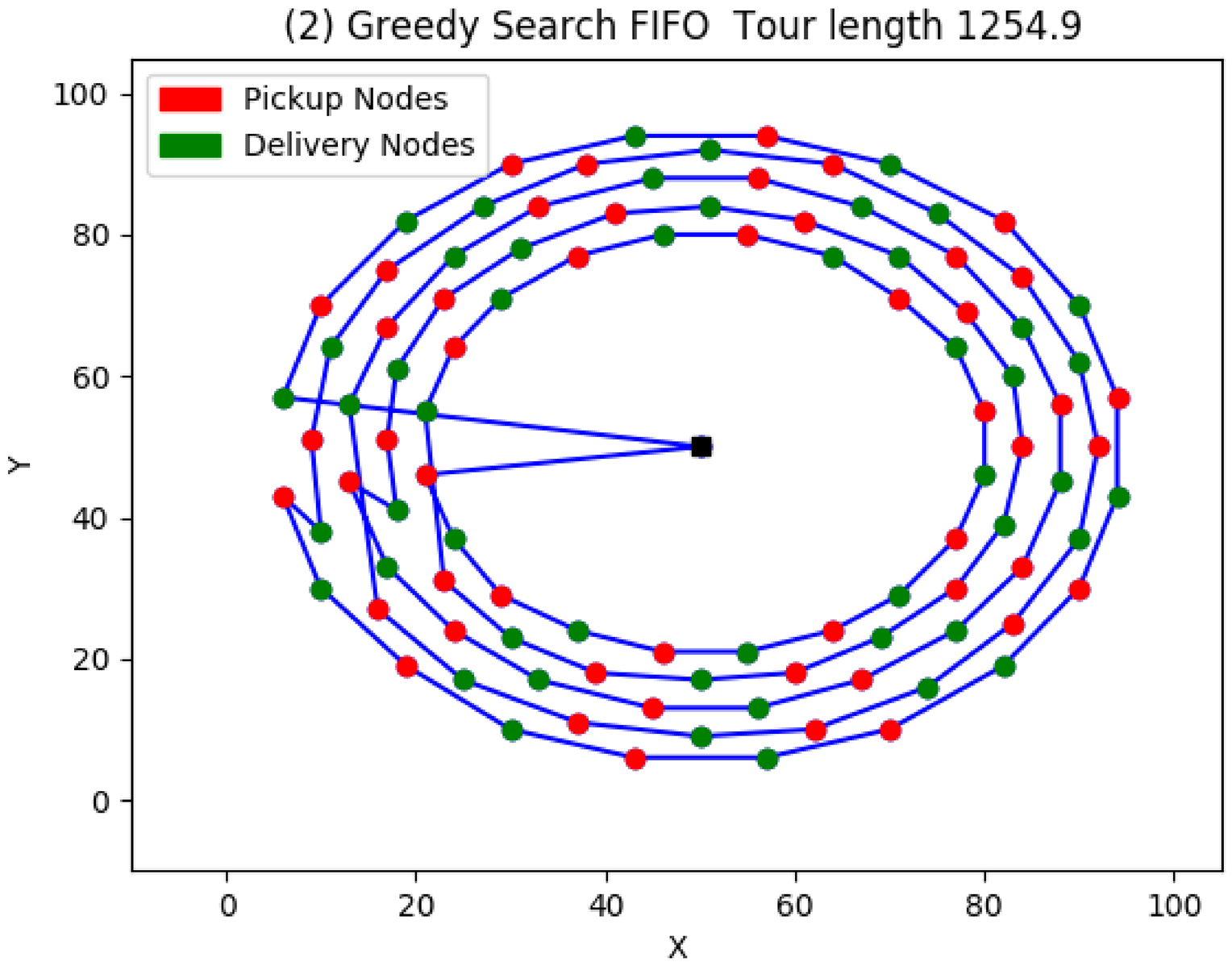
La prima strategia Greedy visita tutti i nodi di prelievo e successivamente tutti i nodi di consegna.



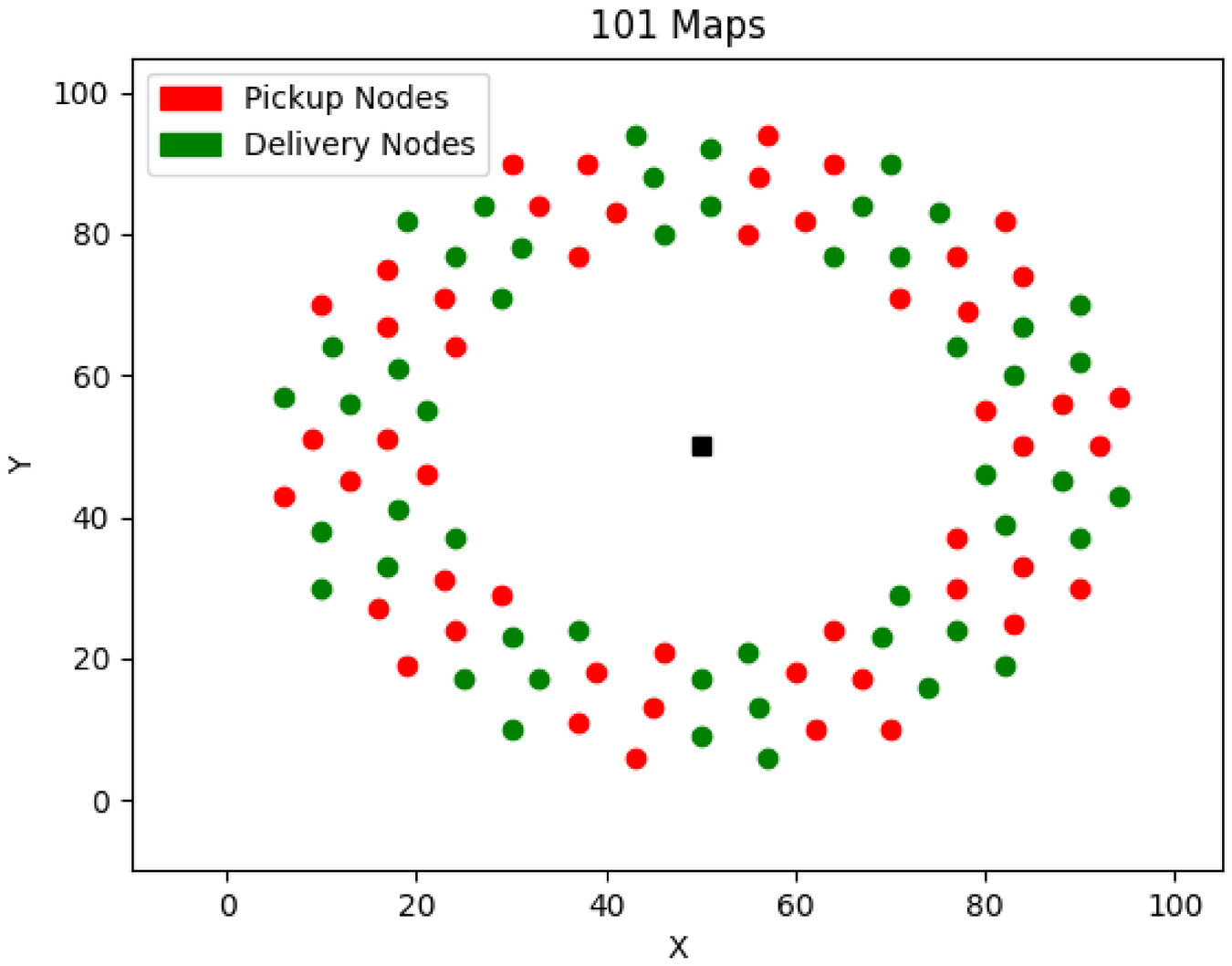

(1) Greedy Search FIFO Tour length 1075.1
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

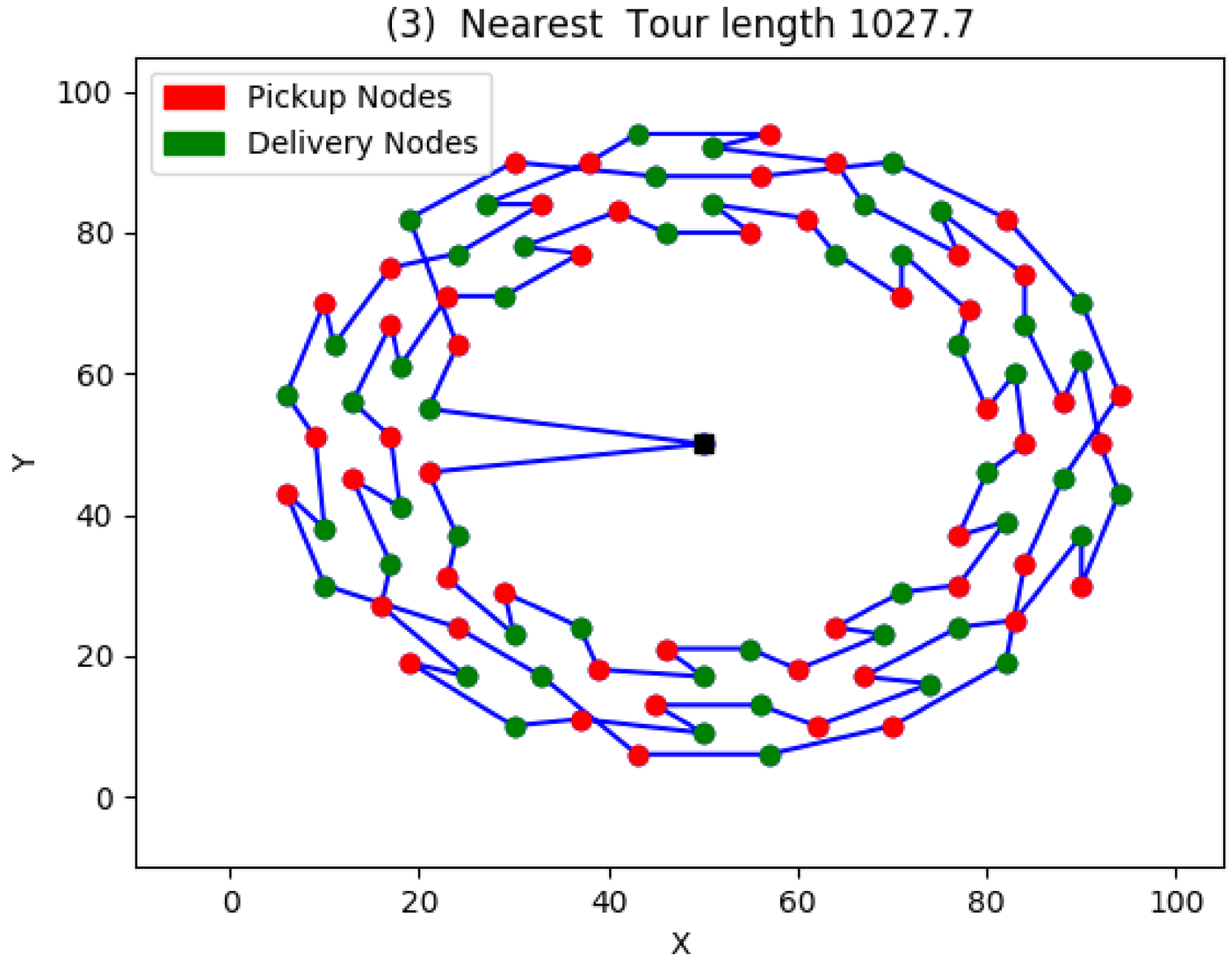
La seconda strategia Greedy visita i nodi nell’ordine in cui sono scritti nel file di input.




(2) Greedy Search FIFO Tour length 1254.9
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

La terza strategia Greedy seleziona il nodo successivo tra tutti i nodi di prelievo disponibili, successivamente si reca al corrispondente nodo di consegna.




(3) Nearest Tour length 1027.7
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

RISULTATI

- Greedy FIFO: 1075.05080059
- Greedy Ordered: 1254.94199253
- Greedy Nearest: 1027.72892051

OSSERVAZIONI

- Non garantisce l’ottimalità della soluzione
- Compie decisioni parziali “localmente ottime” senza mai modificarle.
- Costruisce una soluzione attraverso iterazioni successive, effettuando ad ogni iterazione la scelta più favorevole compatibile con i vincoli del problema.

COME MIGLIORARE?

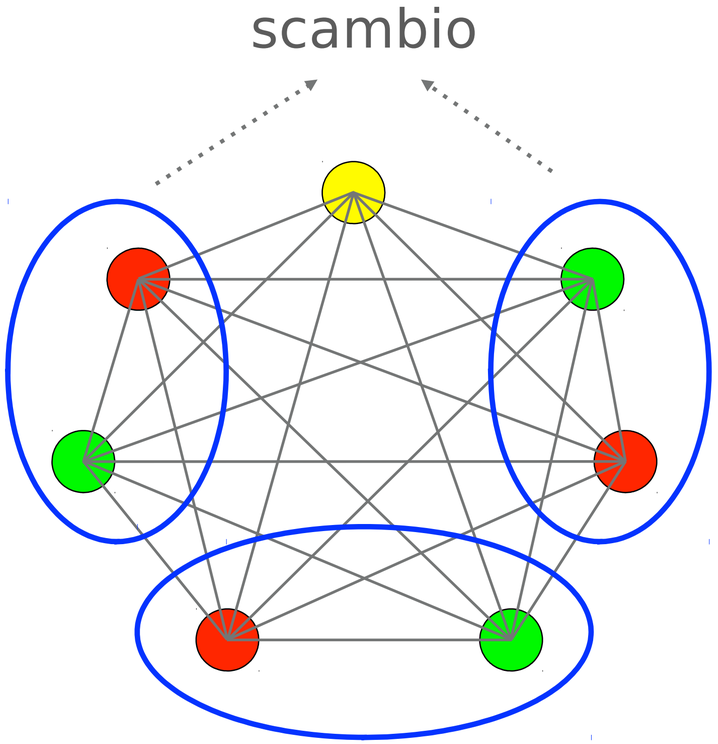
- Proviamo a utilizzare un’euristica di ricerca locale: City Swap

LOCAL SEARCH

- La ricerca locale è l’algoritmo di riferimento per tutte le metaeuristiche basate sul concetto di intorno.
- L’intorno $N(x_k)$ è dato attraverso la MOSSA che descrive operativamente cosa modifico di x_k per generare tutte le soluzioni in $N(x_k)$.
- L’insieme delle soluzioni a lui vicine dove per vicino si intende non troppo diverso, simile, che differisce solo per poche componenti.
- IDEA: data la soluzione corrente x_k ammetto dei cambiamenti parziali (perturbazioni locali) nella soluzione, che riguardano il valore di alcune variabili.
- Se l’intorno gode della proprietà di raggiungibilità, cioè per ogni coppia di soluzioni esiste una successione di soluzioni che le collega tale per cui ogni soluzione appartiene all’intorno della soluzione pretendete, allora la ricerca è in grado di esplorare tutto F.

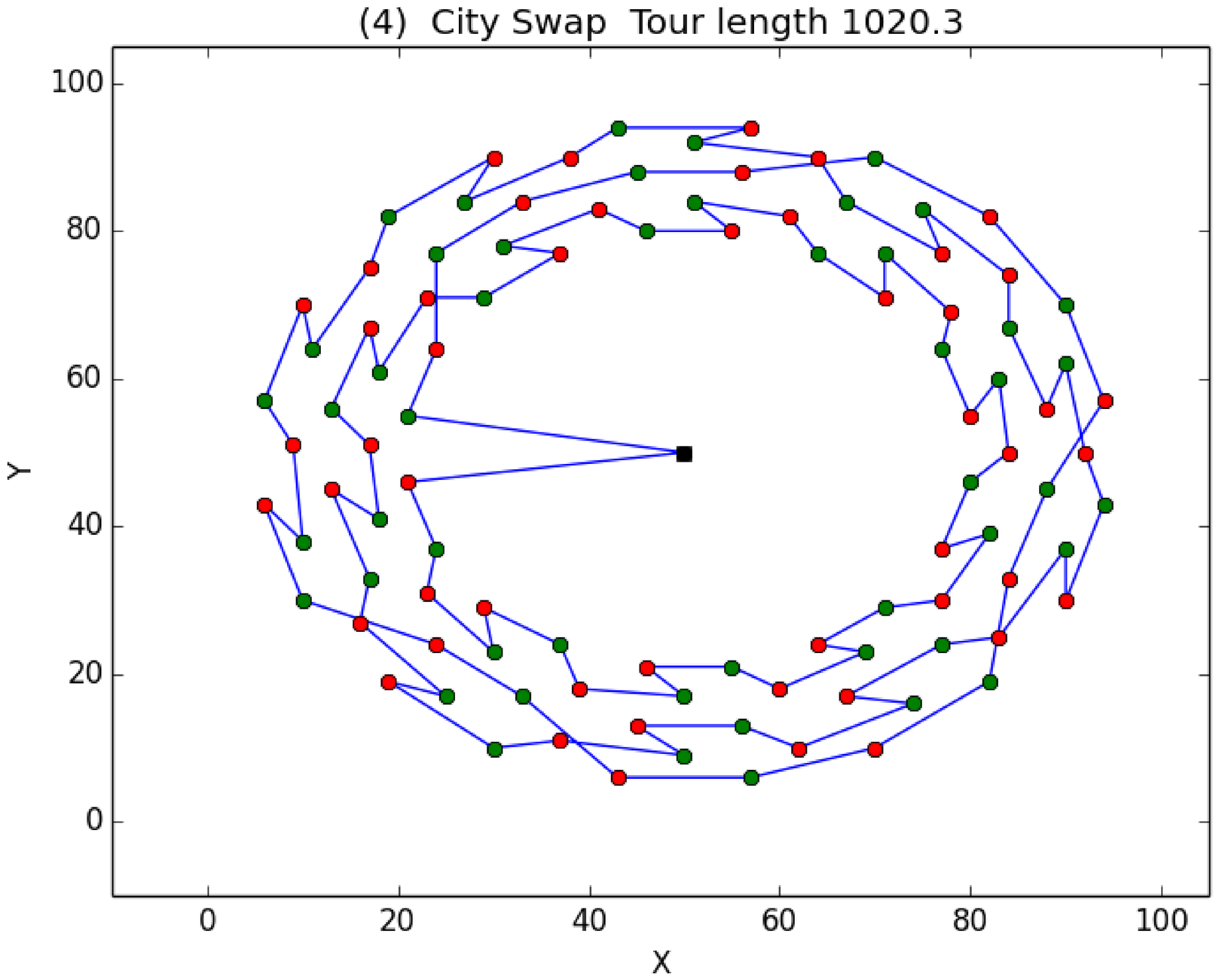
CITY SWAP

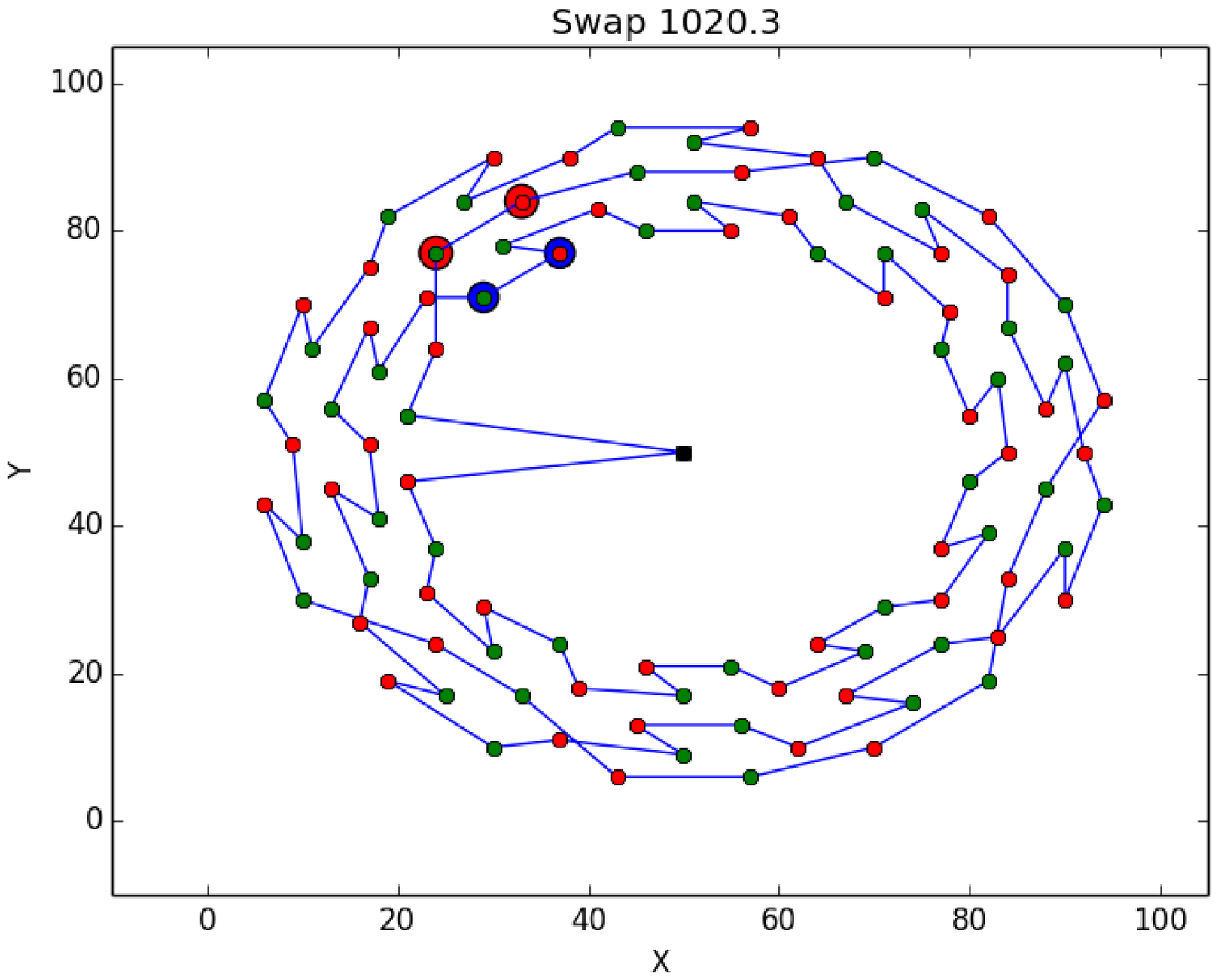
- L’intorno è ottenuto dallo scambio di coppie di nodi
- Coppia formata dal nodo di pickup e dal suo relativo nodo di delivery
- Perché non scambio di due nodi?
- In questo modo mantengo la soluzione sempre ammissibile, poiché non vengono alterate le precedenze di pickup e delivery della coppia
- Come soluzione di partenza viene utilizzata la migliore tra le 3 greedy.

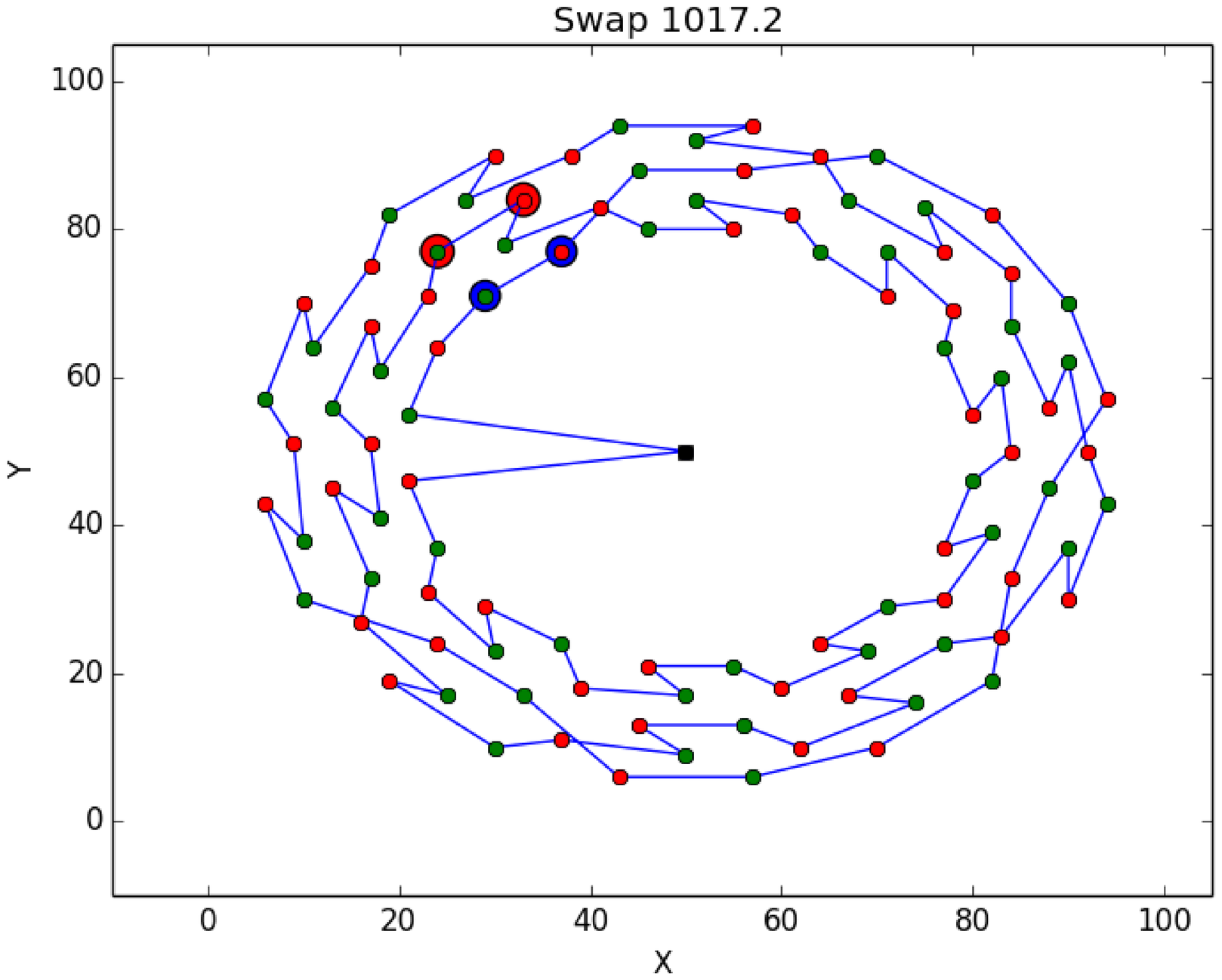


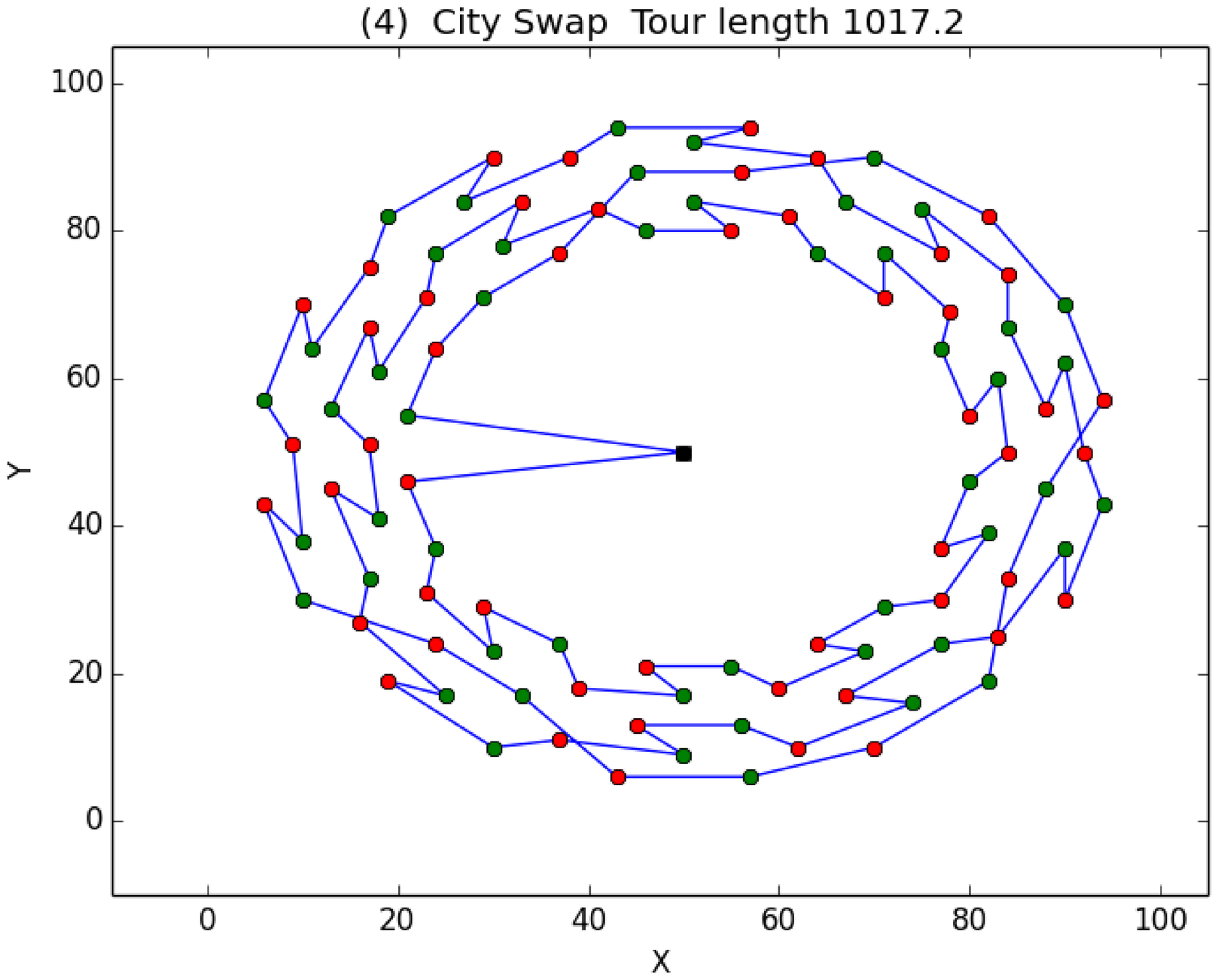
ALGORITMO

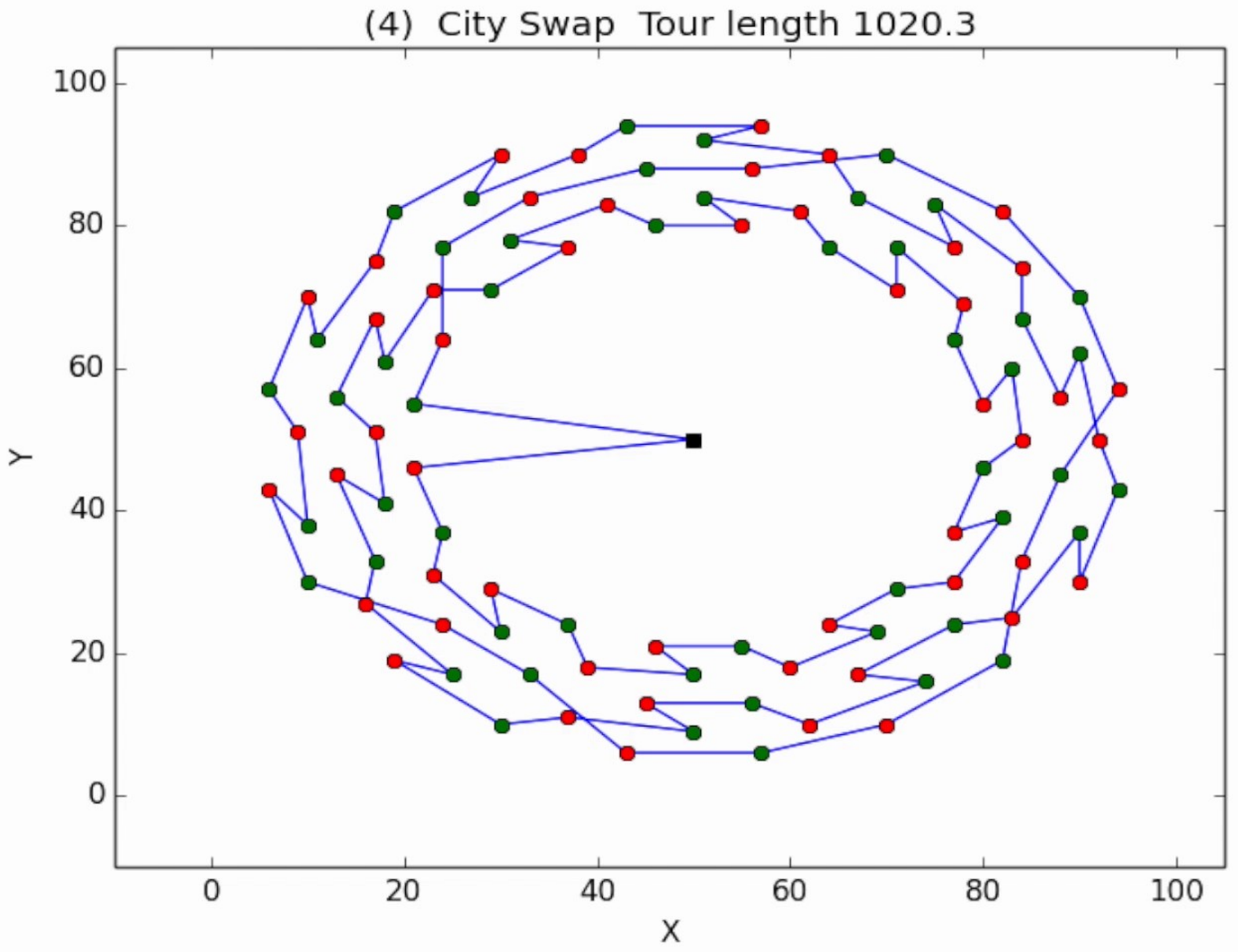
```
So = Best(S_greedy)                    #la soluzione migliore data dalle greedy
while go=True:
      for wi in range(1,n,2):
          for xi in range(1,n,2):
          swap coppie
    Si=scelgo swap migliore
    if Si migliore di So
        So = Si
    else go=False
return So
```


(4) City Swap Tour length 1020.3
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


Swap 1020.3
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


Swap 1017.2
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


(4) City Swap Tour length 1017.2
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


(4) City Swap Tour length 1020.3
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

OSSERVAZIONI

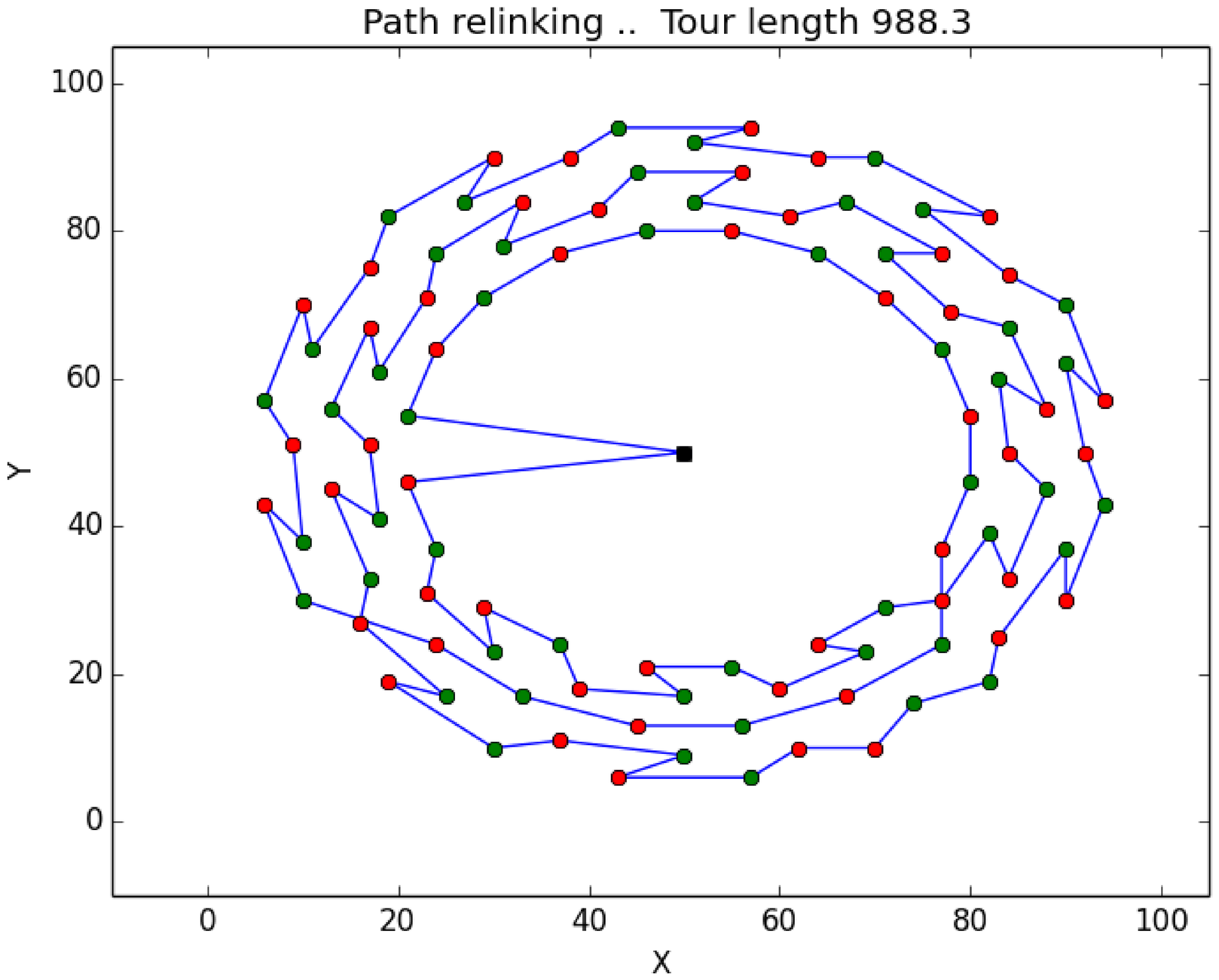
- Porta ad un ottimo locale, no modo esatto
- Ad ogni iterazione si tratta di risolvere un problema di ottimizzazione. Quindi la procedura restituisce una qualsiasi soluzione migliore di x_k.
- Resta intrappolata negli ottimi locali.
- La soluzione prodotta è determinata dalla scelta dell'intorno, dalla strategia di esplorazione e dal punto iniziale, che determina come risultato l'ottimo locale nel cui bacino di attrazione è posizionato il punto iniziale se si segue una strategia best improvement.
- COME PROSEGUIRE? Si può trovare una soluzione a questi problemi utilizzando l'euristica Path Relinking, che esplora le soluzione ottenute operando delle mosse a partire da una popolazione di soluzioni elite, avvicinandosi ad una soluzione target s*.

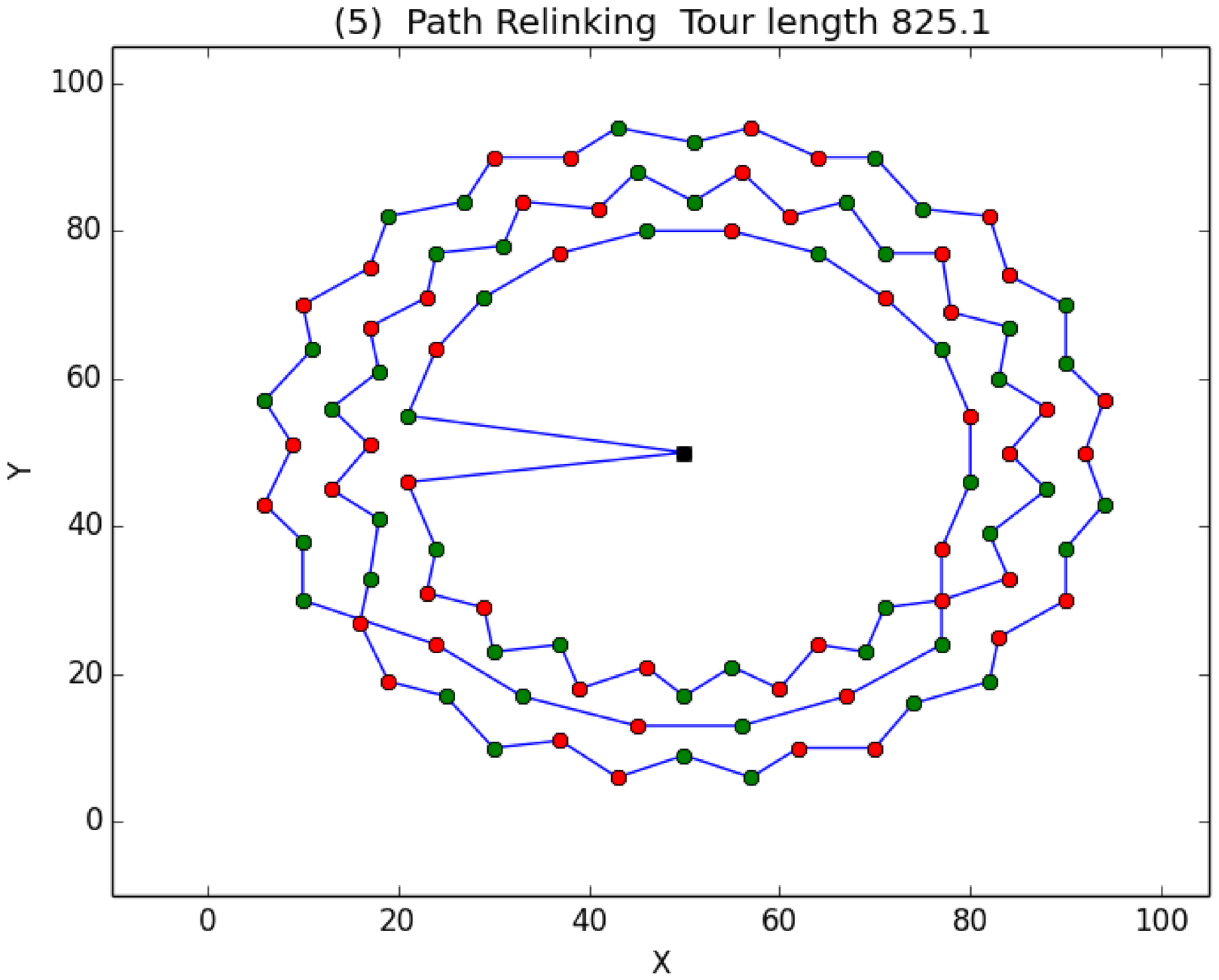
PATH RELINKING

- Si costruisce una popolazione di soluzioni elite {Si} partendo dalla migliore soluzione trovata ed effettuando la mossa swap(scambio di due nodi).
- Si operano quindi in successione delle mosse di scambio di nodi per avvicinarsi alla soluzione target s*.
- Ad ogni passo si valuta ogni mossa potenziale, cioè la modifica di una componente di Si con una della target s*, per ciascuna delle componenti per cui Si e s*differiscono (scambio di due nodi)
- Tra tute le soluzioni trovate, una per ogni mossa possibile, si seleziona quella più conveniente e si itera l’algoritmo.
- Conclusa l’esecuzione si sceglie la migliore soluzione trovata.

ALGORITMO

```
s_best = Best(S_greedy)                    #la soluzione migliore data dalle greedy
while go:
        for i in range(1,n):
            sol = swap(i,s_best)
            if ammissibile(sol) and presente(sol,list_solution)==0:
                list_solution.append(sol)
        s_best = best_solution(list_solution,nodes)
    return s_best
```


Path relinking .. Tour length 988.3
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


(5) Path Relinking Tour length 825.1
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

OSSERVAZIONI

- Soluzione migliorata rispetto le precedenti euristiche
- Una limitazione che rimane nella Path Relinking riguarda il fatto di non avere memoria.
- Per questo lungo percorsi diversi si rischia di ritornare su soluzioni già visitate.
- Per risolvere questo problema è necessario uscire dal bacino di attrazione degli ottimi locali.
- Esistono euristiche che ammettono soluzioni anche peggiorative e non ammissibili durante la propria esecuzione (penalizzandole in caso di violazione di vincoli), ma ovviano a questo problema!

TABU SEARCH

- Si parte da una soluzione generica, scelgo la migliore trovata e si valutano tutte le possibili soluzioni a distanza di una mossa.
- Si seleziona la soluzione migliore dell’intorno valutato e posso accettare anche peggioramenti, così da poter sfuggire al bacino di attrazione dell’ottimo locale in cui mi trovo.
- Mantengo comunque in memoria le soluzioni migliori visitate.
- Per evitare loop, e di ritornare nello stesso bacino di attrazione utilizzo la Tabu List.
- L’algoritmo finisce nel caso siano eseguite N iterazioni globali (per esempio N=50)

TABU LIST

- Per evitare loop, e di ritornare nello stesso bacino di attrazione utilizzo la Tabu List.
- Lunghezza della memoria k (per esempio k=20), quando è piena si cancella la prima mossa e si inserisce la nuova mossa effettuata (non considero il criterio di Aspirazione)
- Nella tabella memorizzo le mosse effettuate.
- Per esempio effettuo lo scambio di due nodi(34 e 37), nella tabella memorizzo la mossa [34,37] e [37,34].
- Prima di scegliere una nuova soluzione dovrò sempre controllare che i due nodi selezionati per lo scambio non siano già presenti nella Tabu List.

ALGORITMO

Step 1.

k=1

Initial solution S1

Sbest = S1

tabu_list = []

list_best_solution = []

list_best_solution.append(sBest)

Step 2.

Genero le possibili soluzioni sol

if ammissibile and non_presente and check_tabu_list:

    list_solution.append(sol)

#seleziono la soluzione migliore per la mossa effettuata

new_s_best = best_solution(list_solution,nodes)

#aggiorno la tabu_list

update tabu_list

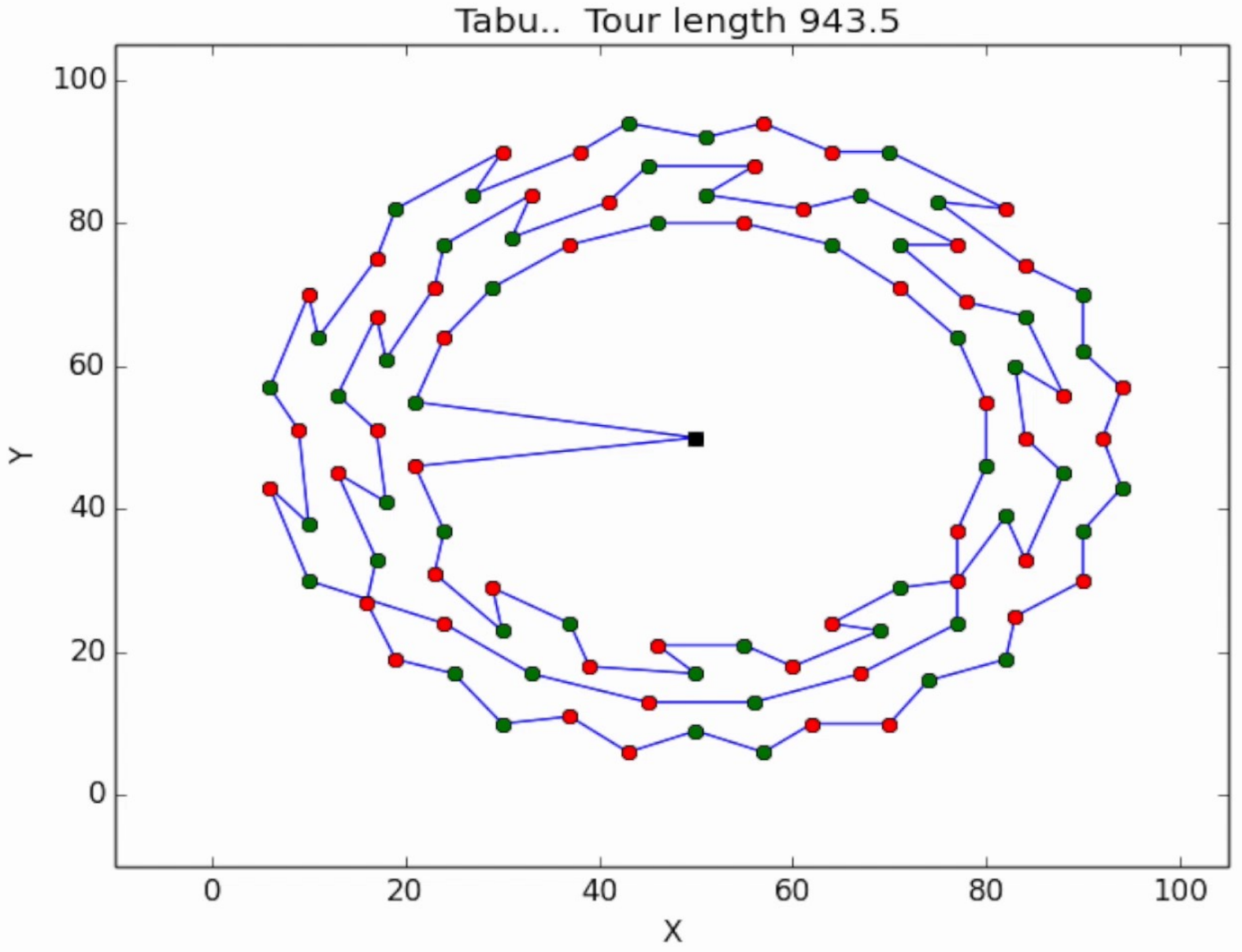
Go to Step 3.

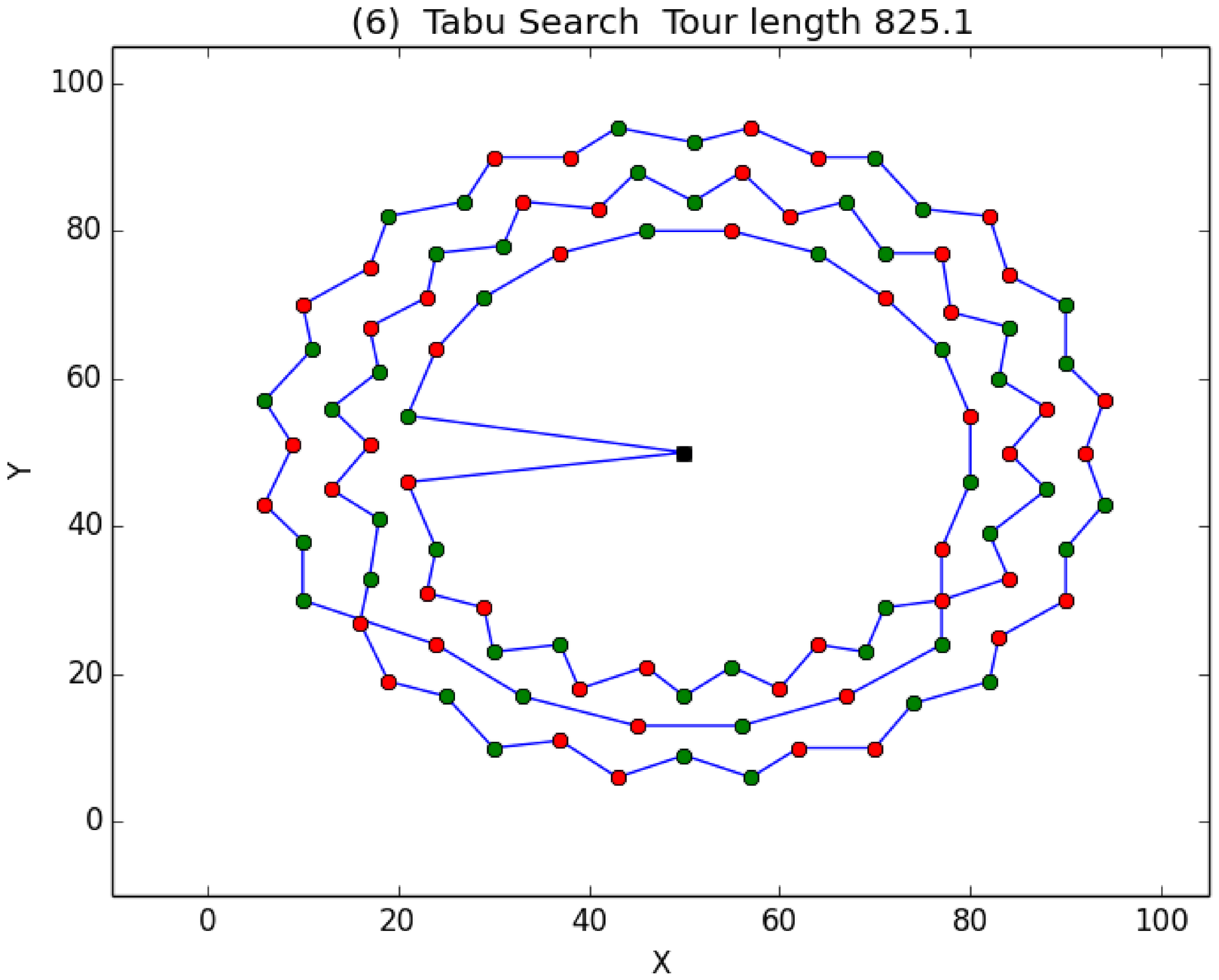
Step 3.

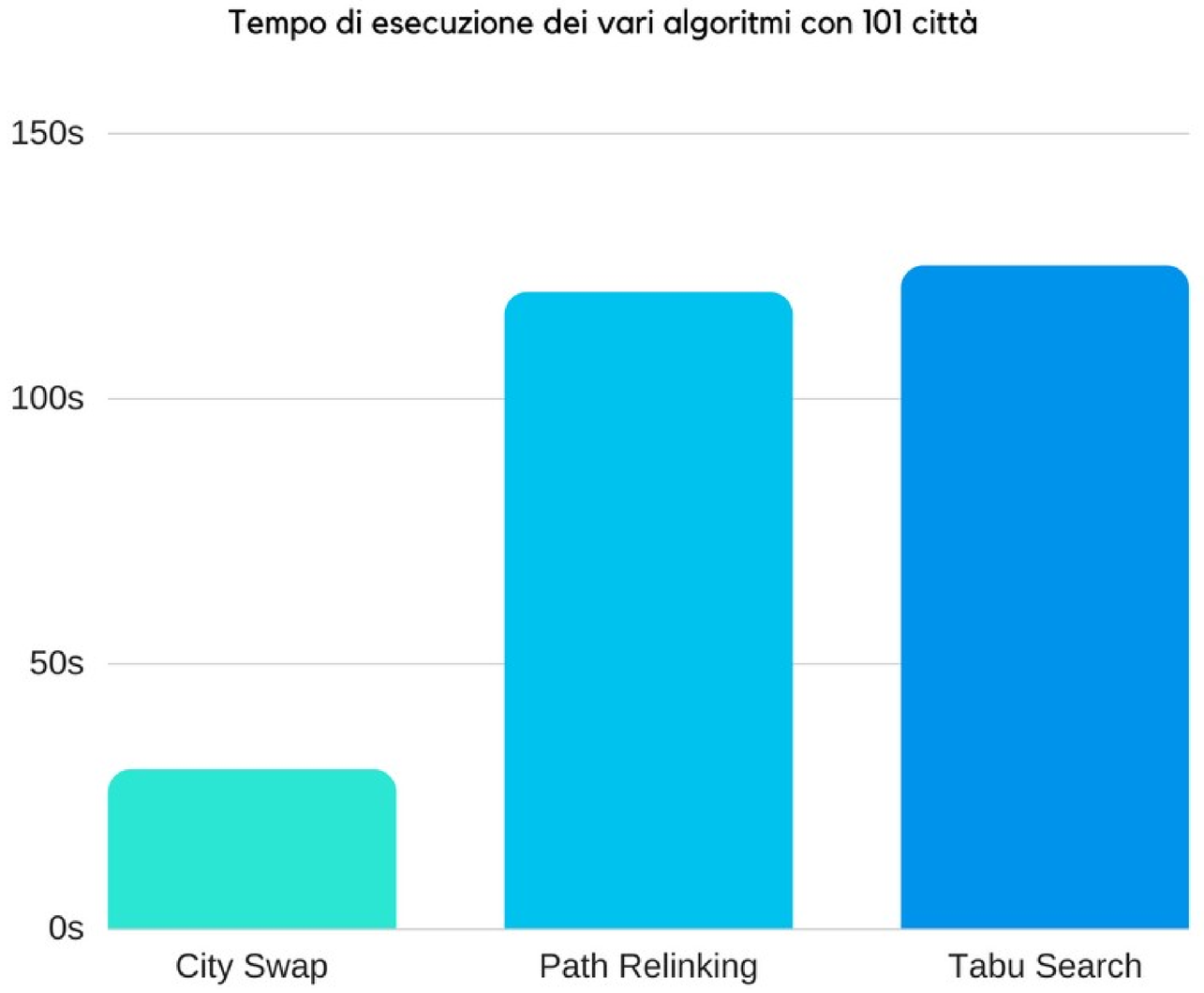
k = k+1 ;

If stopping condition = true *then* STOP, Return(S_{best})

    else go to Step 2


Tabu.. Tour length 943.5
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

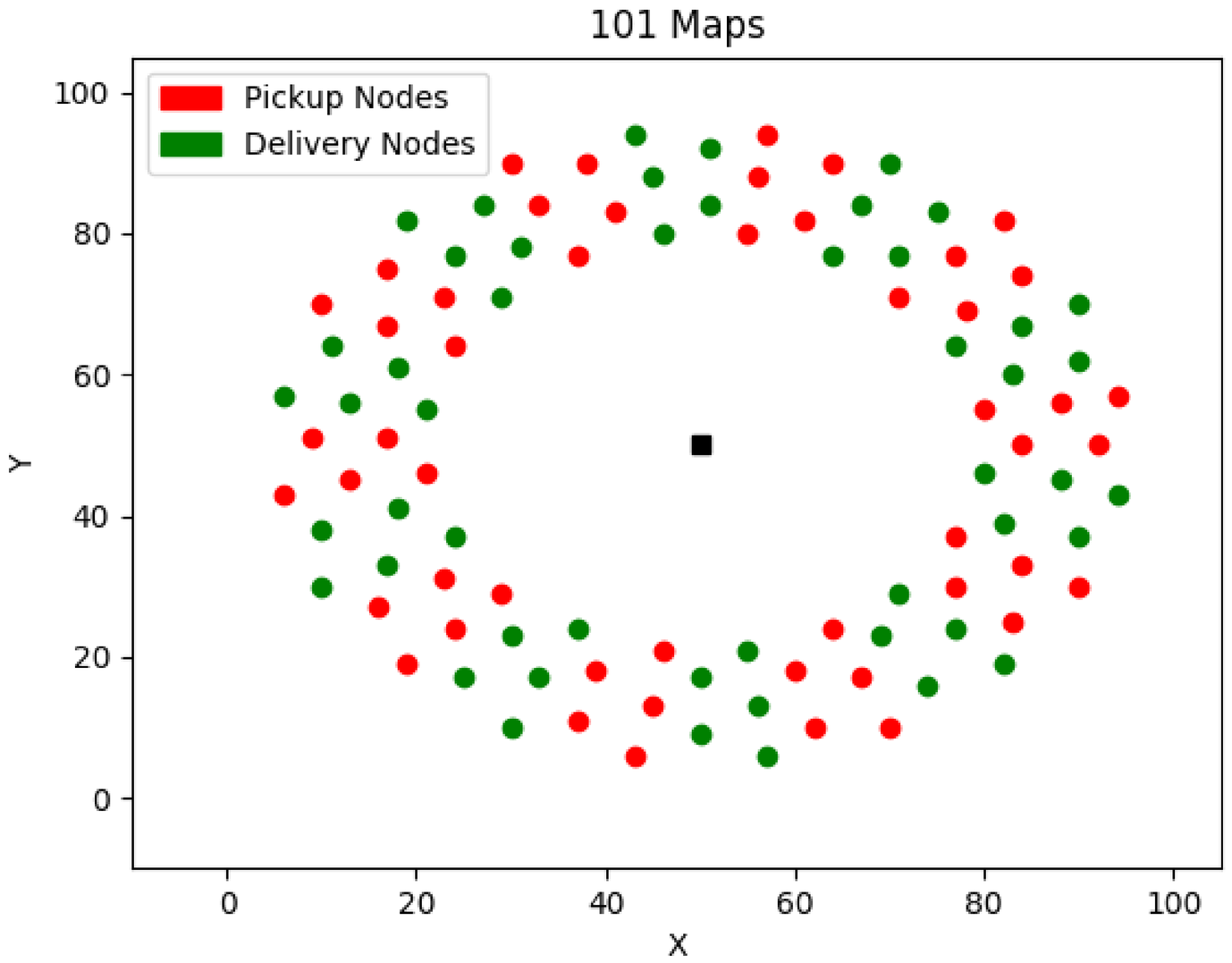

(6) Tabu Search Tour length 825.1
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

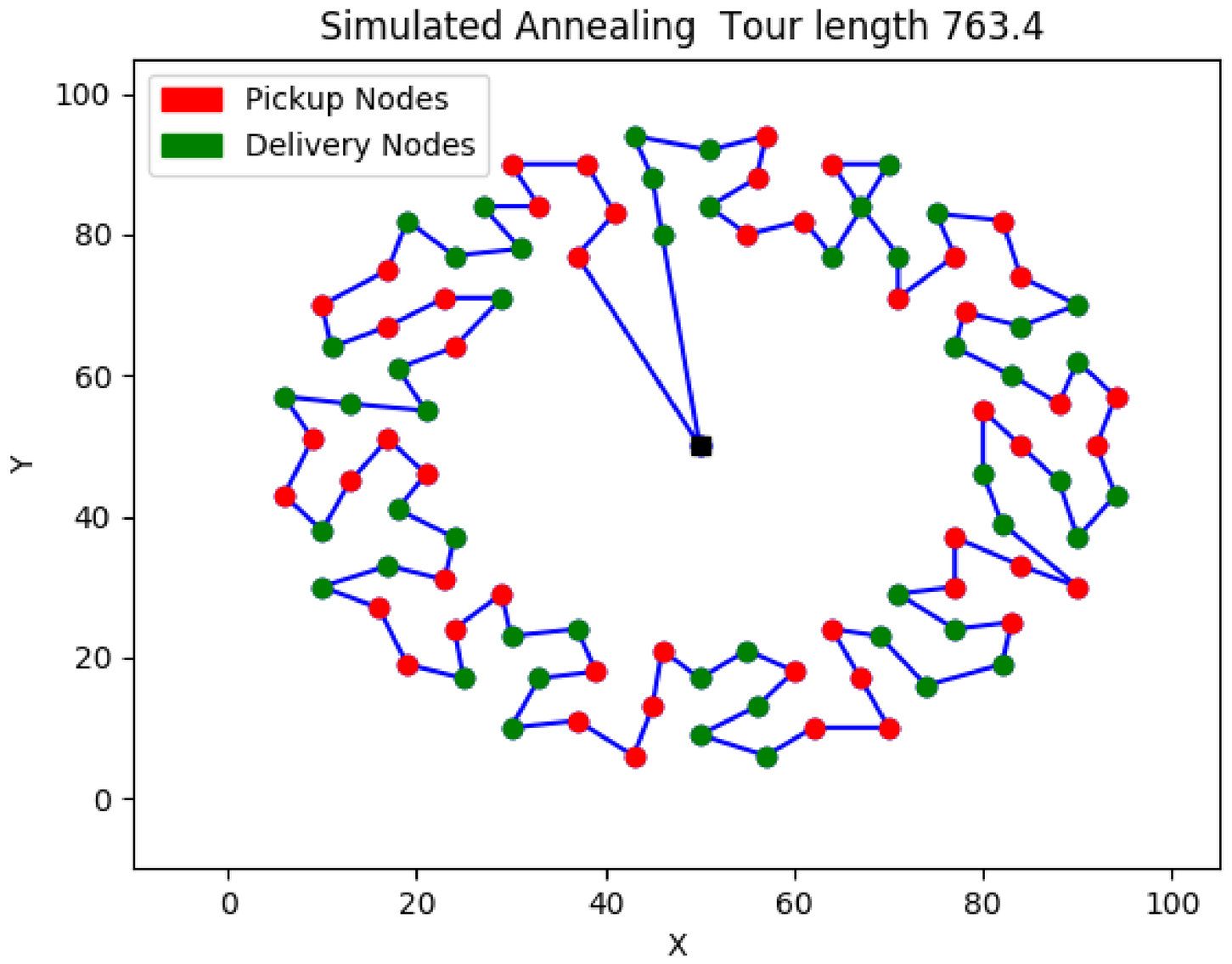

Tempo di esecuzione dei vari algoritmi con 101 città
150s
100s
50s
0s
City Swap
Path Relinking
Tabu Search

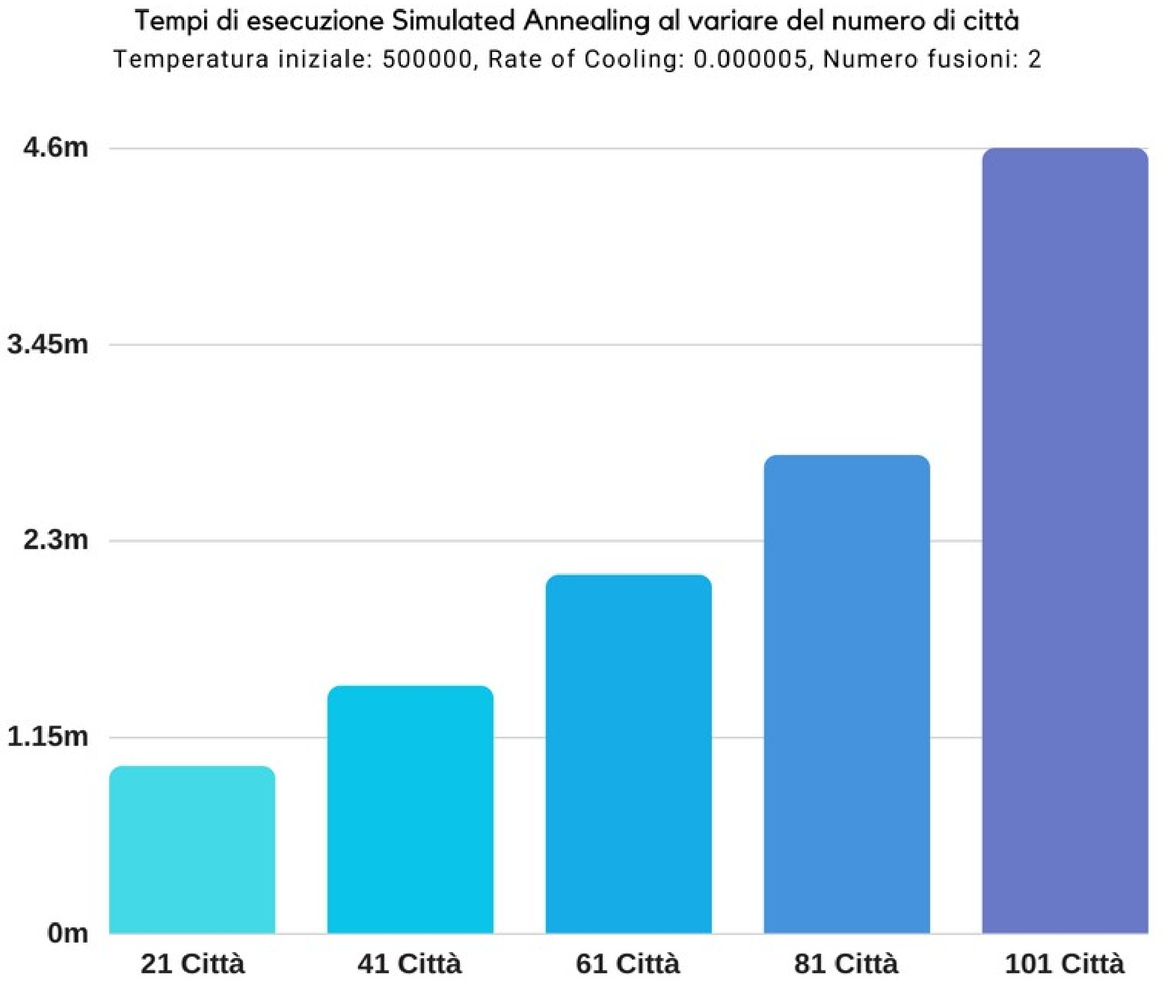
SIMULATED ANNEALING

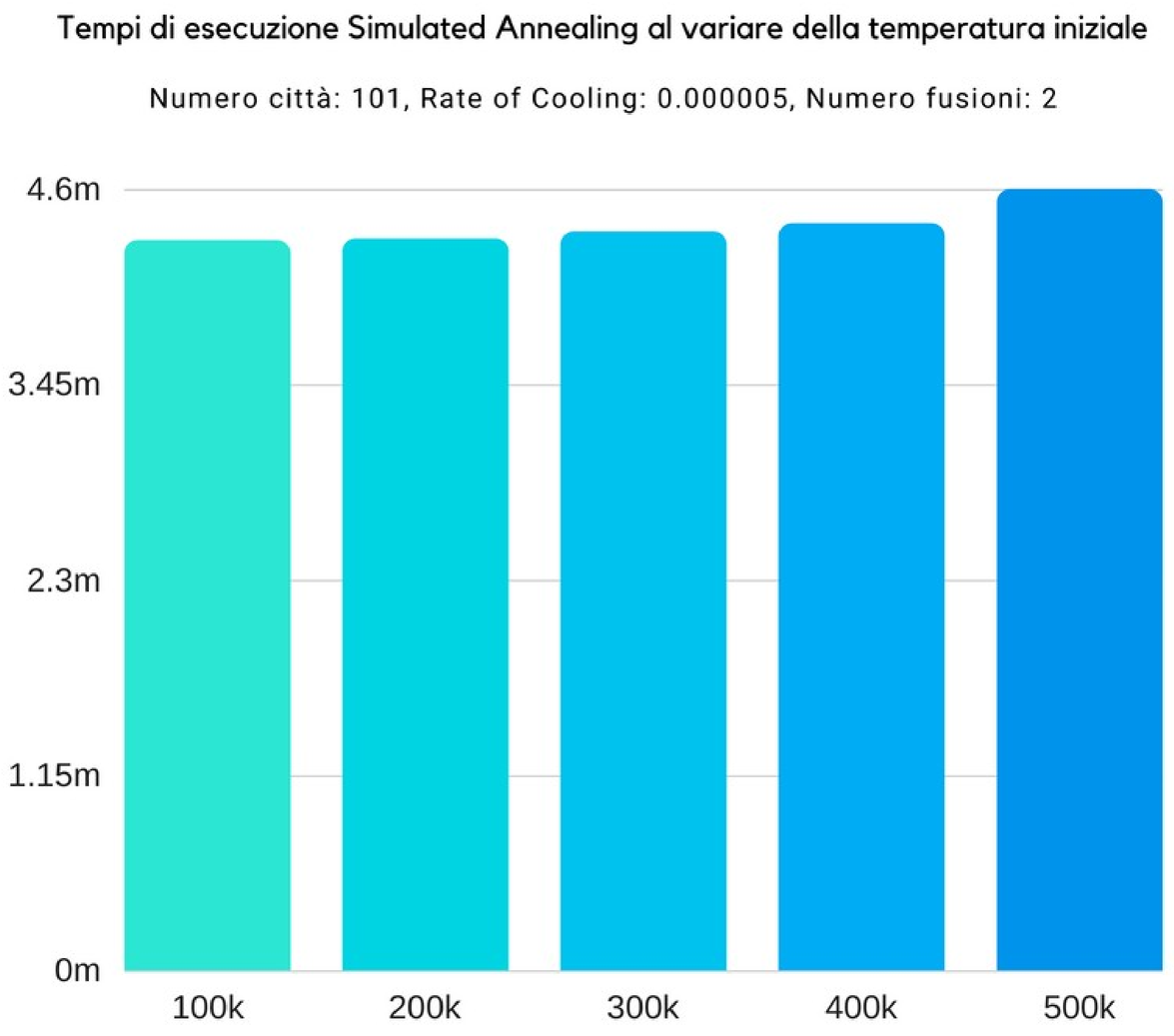
Una Local Search probabilistica in cui l'accettazione di una soluzione ammissibile peggiore viene regolata da una legge probabilistica.

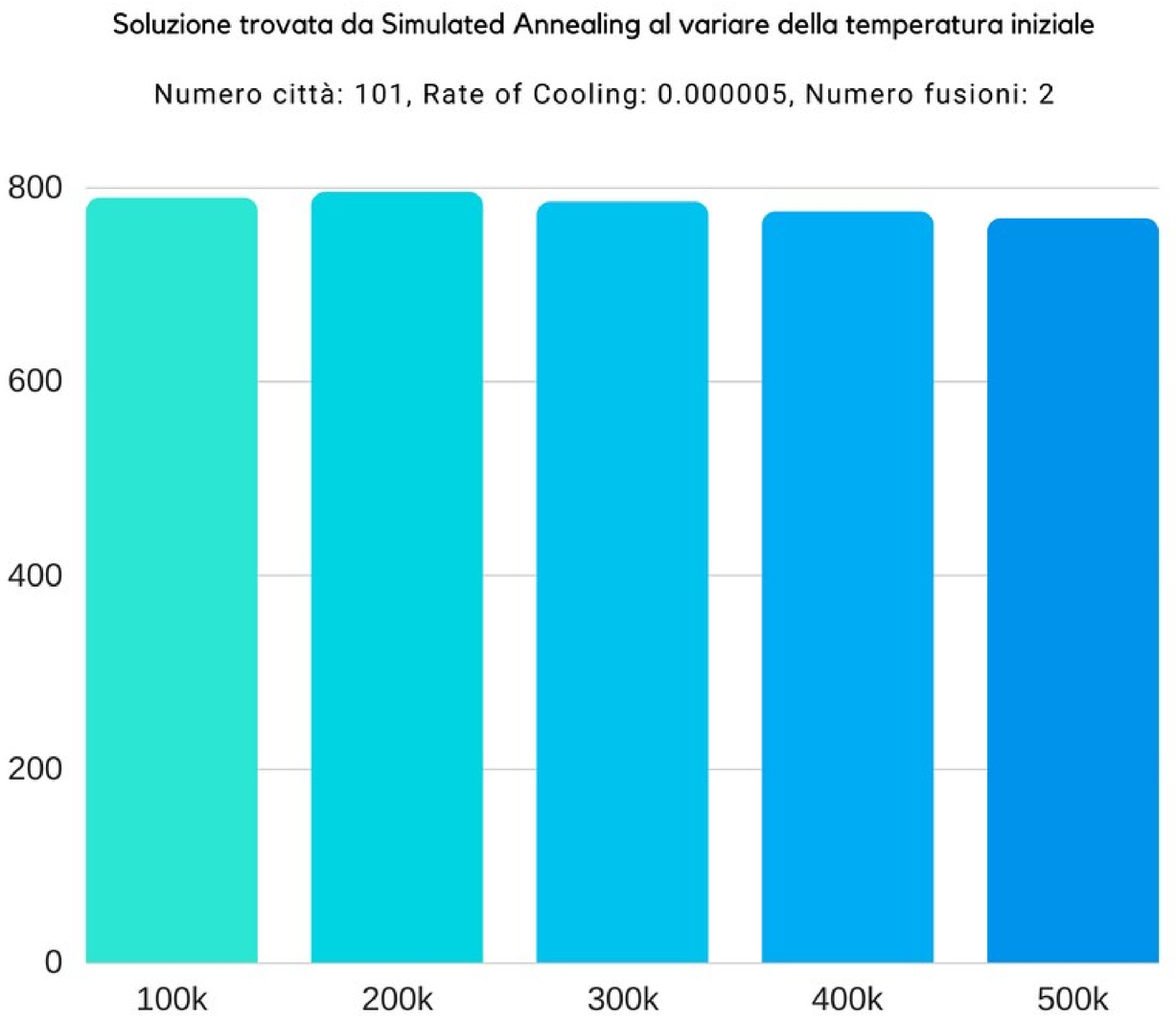
- Parte da una soluzione iniziale ammissibile X_0
- Ad ogni iterazione k, UNA soluzione X_{next} è scelta A CASO nell'intorno della soluzione corrente X_k
- Se $f(X_{next}) < f(X_k)$ ci si sposta sulla nuova soluzione (una mossa di miglioramento viene sempre accettata)
- Altrimenti la decisione se spostarsi su di una soluzione PEGGIORE X_{next} o restare in X_k dipende da una funzione casuale $e^{-\Delta E/T}$ basata su
  - $\Delta E = f(X_{next})-f(X_k)$ la variazione del costo della funzione obiettivo,
  - un parametro T (temperatura) che diminuisce lungo la ricerca secondo un determinato criterio


101 Maps
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


Simulated Annealing Tour length 763.4
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

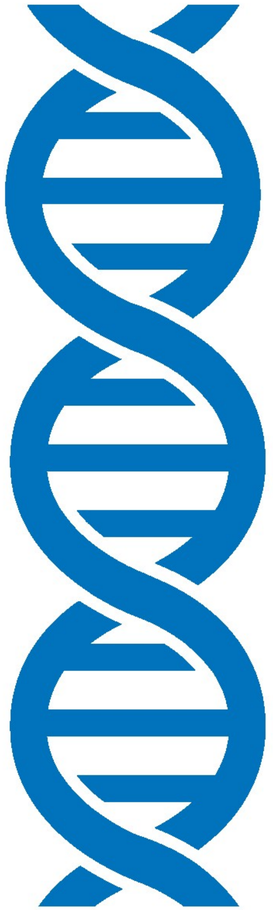

Tempi di esecuzione Simulated Annealing al variare del numero di città
Temperatura iniziale: 500000, Rate of Cooling: 0.000005, Numero fusioni: 2
4.6m
3.45m
2.3m
1.15m
0m
21 Città
41 Città
61 Città
81 Città
101 Città


Tempi di esecuzione Simulated Annealing al variare della temperatura iniziale
Numero città: 101, Rate of Cooling: 0.000005, Numero fusioni: 2
4.6m
3.45m
2.3m
1.15m
0m
100k
200k
300k
400k
500k


Soluzione trovata da Simulated Annealing al variare della temperatura iniziale
Numero città: 101, Rate of Cooling: 0.000005, Numero fusioni: 2
800
600
400
200
0
100k
200k
300k
400k
500k

GENETIC ALGORITHM

- Basati sulla teoria dell’ evoluzione di Darwin
- Si lavora contemporaneamente su un’intera popolazione di individui che devono adattarsi all’ambiente
- Il processo si auto-organizza come nei sistemi biologici
- L’evoluzione è basata sul meccanismo di selezione naturale
- La popolazione di individui si evolve da una generazione a quella successiva.



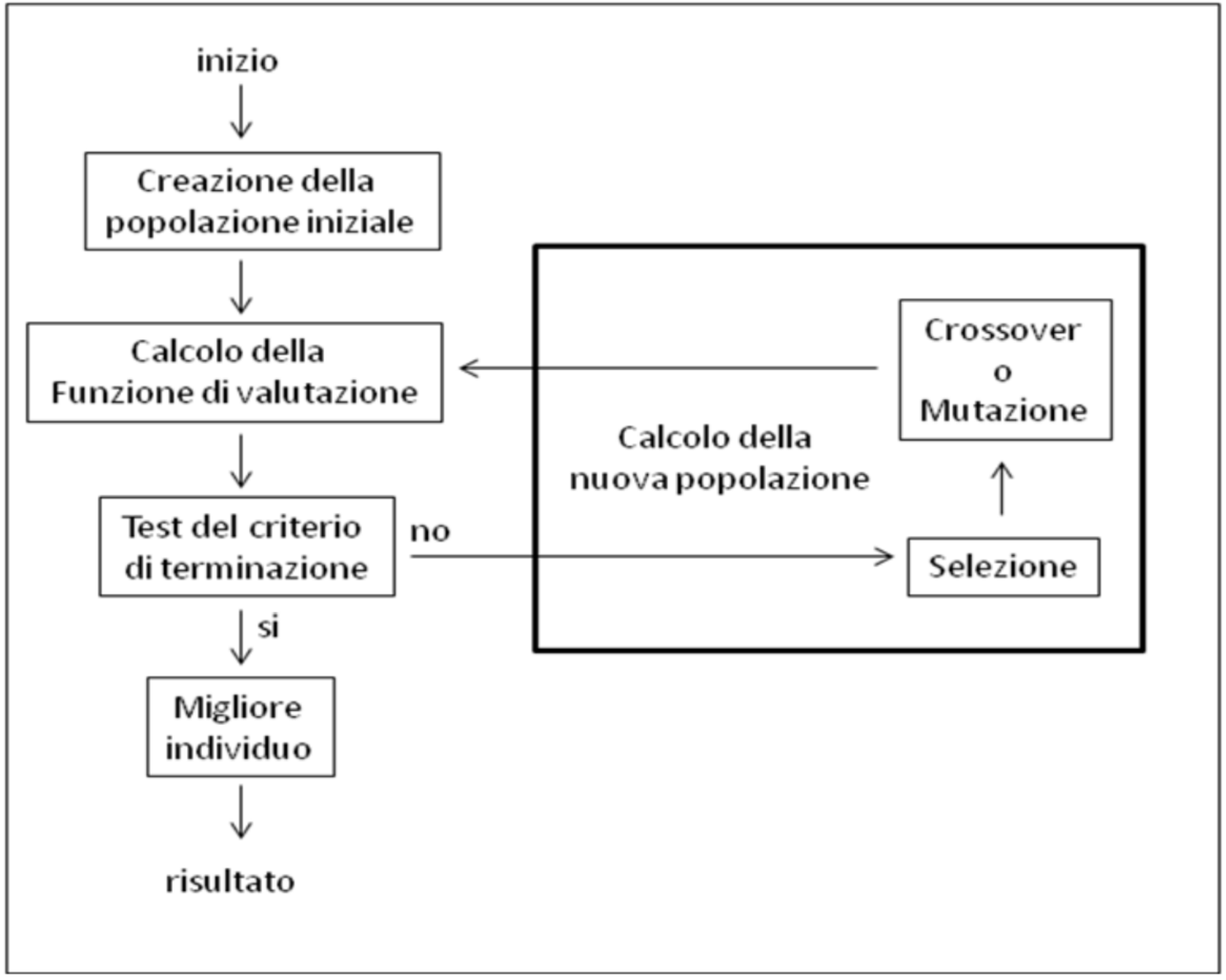
- L’intero processo viene interpretato come un algoritmo di ottimizzazione, dove la fitness (grado di adattamento all’ambiente) diviene la funzione obiettivo in base alla quale si valuta la qualità di una soluzione.
- La riproduzione viene regolata dal seguente meccanismo, che deve mimare la selezione naturale:

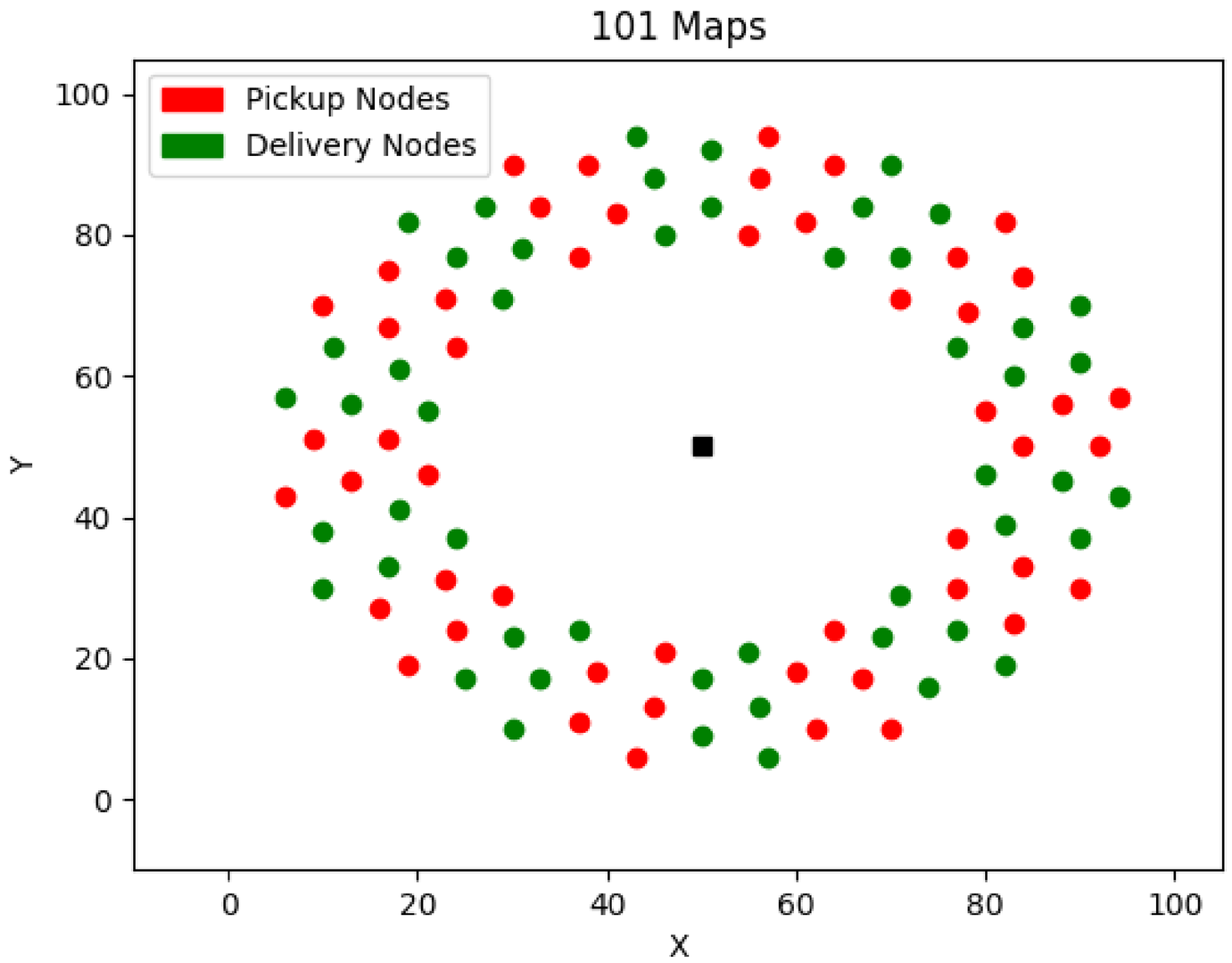
  – I genitori sono scelti secondo un criterio che favorisce i migliori (ad esempio in base alla fitness)

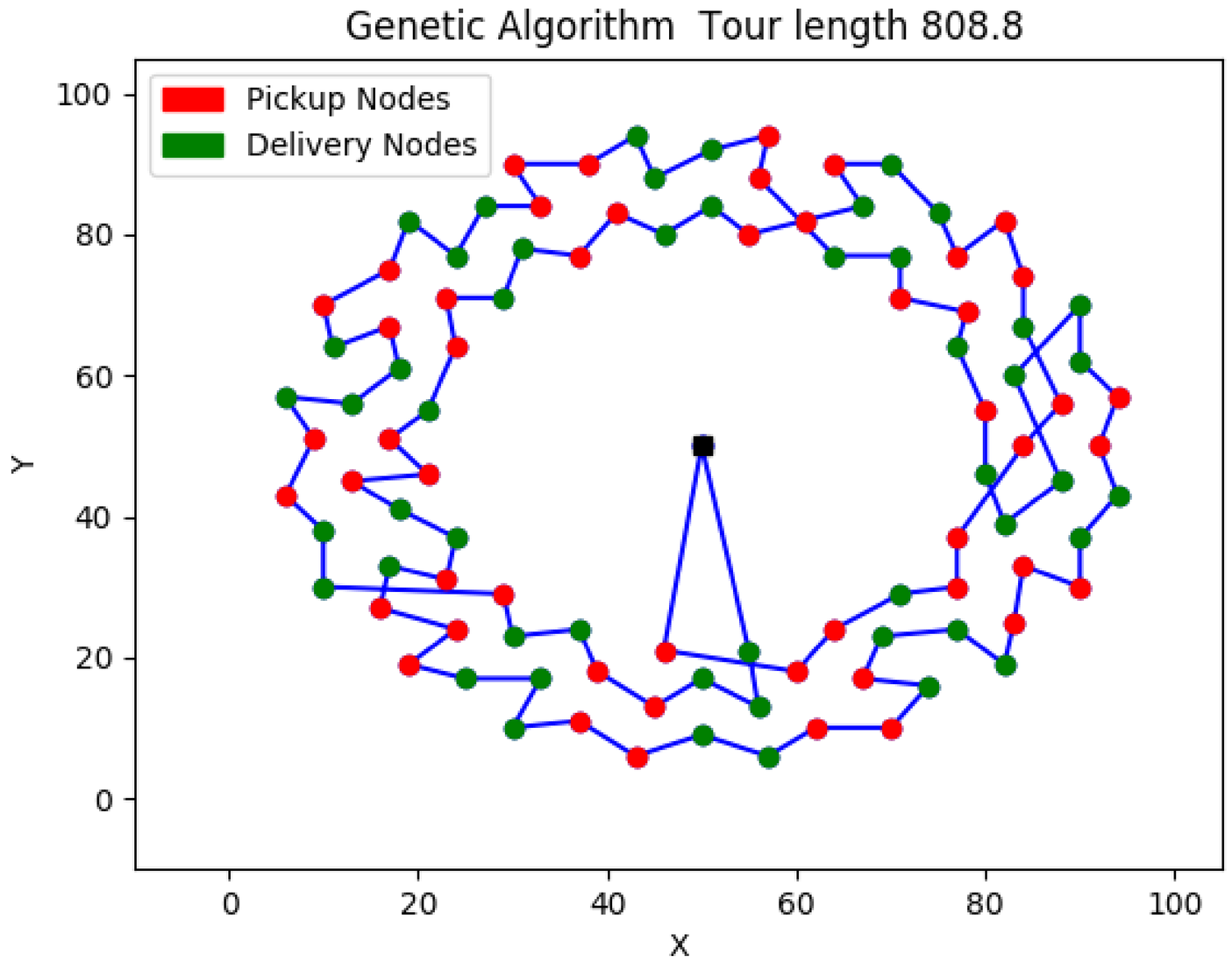
  – La generazione dei figli deve conservare e trasmettere i caratteri “buoni” di entrambe i genitori

  – Elitismo che permette il passaggio dei migliori individui alla generazione successiva (implementa un meccanismo di memoria)

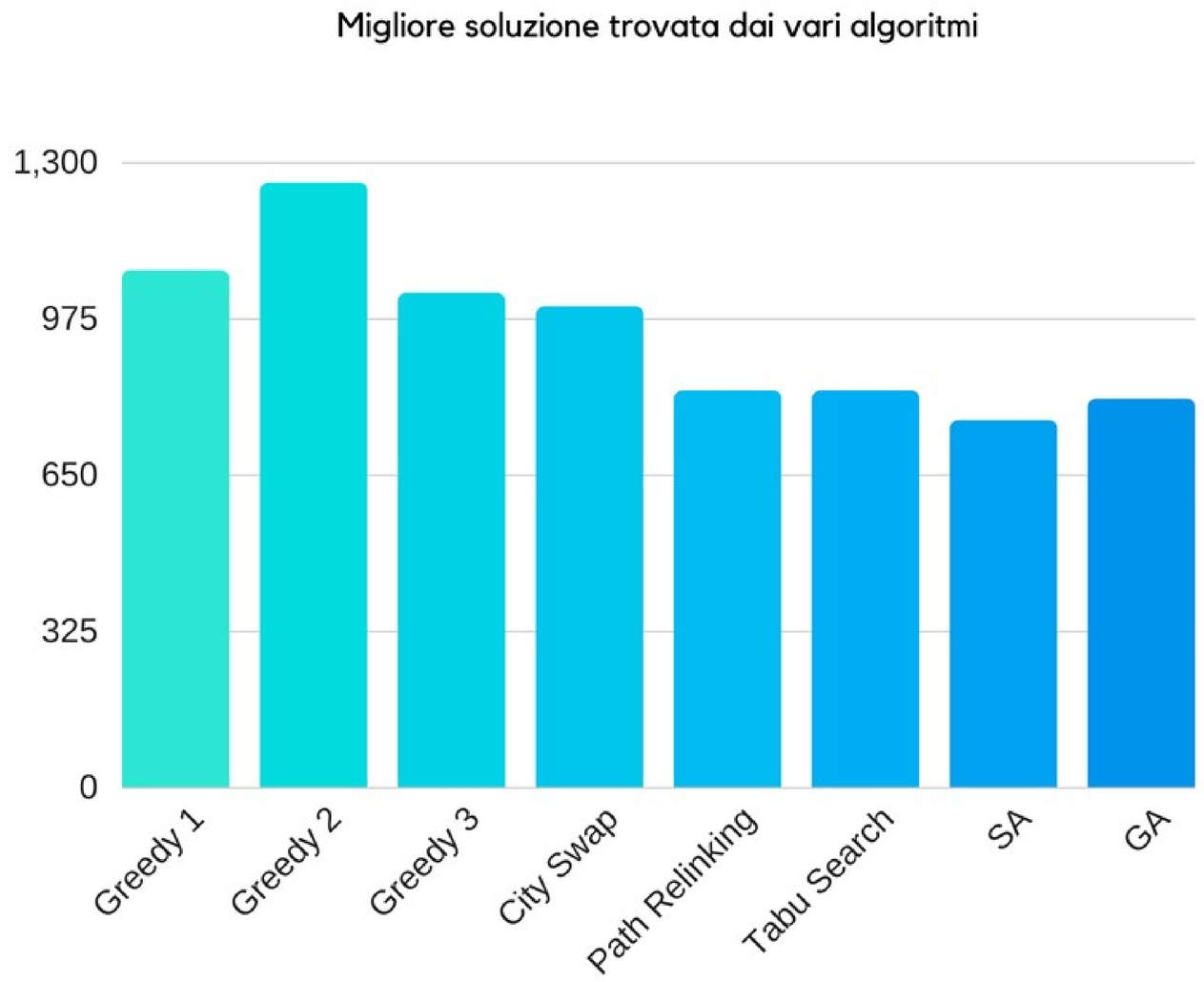
  – Mutazione avviene casualmente su un singolo individuo. Componente necessaria per estendere la ricerca al di la delle potenzialità legate alla popolazione iniziale.


inizio
Creazione della popolazione iniziale
Calcolo della Funzione di valutazione
Test del criterio di terminazione
no
si
Migliore individuo
risultato
Calcolo della nuova popolazione
Crossover o Mutazione
Selezione


101 Maps
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X


Genetic Algorithm Tour length 808.8
Pickup Nodes
Delivery Nodes
100
80
60
40
20
0
Y
0
20
40
60
80
100
X

CONCLUSIONI



CONCLUSIONI

✝

- Gli algoritmi Greedy sono di facile implementazione e computazionalmente leggeri, di contro fornisco soluzioni accettabili ma che nella maggior parte dei casi non sono ottime. Questo problema si risolve utilizzando euristiche di miglioramento, accettando un leggero aumento del tempo di esecuzione.
- Le euristiche come Simulated Annealing e Genetic Algorithm forniscono risultati nettamente migliori rispetto alle soluzioni precedenti, richiedono tuttavia un maggior tempo di esecuzione.
- L’impiego degli uni o degli altri dipende quindi dal tipo di istanza del problema che si vuole risolvere, e dai requisiti (di tempo, di consumo massimo di risorse o di qualità della soluzione) che si devono rispettare.

GRAZIE PER L’ATTENZIONE