Business analytics - Homework CVRP

Alessandro Bonadeo, Gianluca Cappiello, Diego Morichelli

10 Agosto, 2022

1 Introduzione

Nel seguente elaborato si va a risolvere il CVRP simmetrico utilizzando due note euristiche: lo sweep e il tabu search. La prima è di tipo *costruttivo* in quanto la soluzione viene generata espandendo ogni route seguendo un preciso criterio. Inoltre quest'algoritmo si dice essere *sequenziale* in quanto l'intero processo si può vedere diviso in due parti eseguite una di seguito all'altra. Il tabu search, invece, è un algoritmo *iterativo* in quanto parte da una soluzione iniziale (non ottima) e cerca di migliorarla. Siccome utilizzare una soluzione iniziale generica potrebbe richiedere un eccessivo tempo di calcolo, si è deciso di partire dalla soluzione generata dallo sweep, che si può supporre essere abbastanza vicina all'ottimo.

2 Sweep

Lo sweep è un algoritmo di tipo sequenziale, in particolare segue l'approccio *cluster-first, route-second*: infatti, il primo passo consiste nel raggruppare le destinazioni in cluster che rispettino i vincoli di capacità per ogni veicolo. Ogni cluster corrisponde a una route che viene successivamente ottimizzata risolvendo un problema di TSP. La particolarità di questo metodo sta nel criterio con cui vengono formati i cluster: fissando il deposito come centro del sistema di riferimento cartesiano, si considerano le coordinate polari di ogni destinazione, le quali vengono ordinate rispetto all'angolo. Dunque si forma la prima route considerando le prime n destinazioni tali per cui la capacità del veicolo sia soddisfatta e la $n+1$-esima destinazione sia la prima a violare il vincolo, dopodiché si passa alla route successiva. L'interpretazione geometrica del metodo consiste nell'immaginare un raggio uscente dal deposito che ruotando in senso antiorario "*spazzi*" (da cui il termine *sweep*) una destinazione dopo l'altra rispettando i limiti imposti dal vincolo di capacità. Una evidente limitazione del metodo è che, data la sua natura sequenziale, non è previsto un numero massimo di veicoli utilizzabili. Siccome la consegna dell'elaborato richiede esplicitamente che il numero massimo di veicoli sia fissato, si è deciso di ovviare a questo

problema implementando una strategia di ridistribuzione delle destinazioni, utilizzando un criterio prossimità, nel caso in cui il vincolo di veicoli disponibili non sia soddisfatto dall'algoritmo originale. Ovviamente esistono casi in cui anche utilizzando questa strategia non si giunge a nessuna soluzione ammissibile (come il caso in cui almeno unna delle destinazioni in eccesso abbia domanda superiore a ogni capacità residua) ma si ritiene, in base a esperimenti su diversi dataset [1], che questi siano per lo più casi patologici e si osservano ottime performance dello sweep su problemi di diversa struttura. Il paper da cui si è preso spunto per la realizzazione dell'algoritmo è quello di Gillet e Miller [2].

2.1 Documentazione

Di seguito si descrivono le funzioni MATLAB utilizzate per implementare l'algoritmo:

- [*veichles, optimal_value, dimension, capacity, coordinates, demand*] = ...
  extract_*dataset(dataset)*
  Questa funzione estrae le informazioni utili dai file di testo formattati contenenti i diversi problemi VRP. In particolare prende in input il dataset che si vuole testare e restituisce i seguenti output:
  - *veichles*: il numero di veicoli disponibili;
  - *optimal_value*: il costo della soluzione ottima;
  - *dimension*: la dimensione del problema ovvero il numero di destinazioni incluso il deposito;
  - *capacity*: la capacità di ogni veicolo;
  - *coordinates*: le coordinate di ogni destinazione e del deposito;
  - *demand*: la domanda per ogni destinazione.

  Le operazioni che si eseguono sono strettamente legate a come gli autori dei problemi hanno deciso di formattare il testo.

- $path = \text{tsp\_solve}(x, y)$
  La funzione *tsp_solve* è necessaria per ottimizzare le route una volta formati i diversi cluster. In questo elaborato è stata implementata utilizzando l'esempio fornito da MATLAB [3]: l'algoritmo consiste nel risolvere il TSP risolvendo il problema di programmazione lineare intera corrispondente e in particolare ciò viene fatto utilizzando le funzioni del pacchetto *Optimization Toolbox* di MATLAB. Poiché seguendo questo approccio non è possibile definire i vincoli riguardanti i subtour, questi vengono eliminati seguendo un approccio iterativo.

- $c = \text{get\_cost}(route, coords)$
  Funzione che restituisce il costo di una $route$.

- $c = \text{get\_cost\_solution}(S, coords)$
  Funzione che restituisce il costo totale di una soluzione S.

- plot_solution(S, *coords*)
  Questa funzione restituisce il plot della soluzione S, ovvero le route ottimizzate, sul piano cartesiano.

- $feasible = \text{isfeasible}(S, demand, capacity)$
  Questa funzione verifica l'ammissibilità della soluzione S, cioè controlla se ogni route rispetta il vincolo di capacità. L'output è un valore logico: pari a 1 se la soluzione è ammissibile e 0 altrimenti.

- $solution = \text{sweep}(vehicles, dimension, capacity, coords, demand)$
  In questa funzione è implementato il vero e proprio algoritmo. Prende in input le stesse variabili fornite in output dalla funzione *extract_dataset* e restituisce un cell array contenente la soluzione. Questa è costituita, per ogni route, dagli indici corrispondenti alle destinazioni facenti parte di una determinata route. Nella prima parte si va a centrare le destinazioni rispetto al deposito, ovvero si considera quest'ultimo come origine del sistema cartesiano di riferimento. Dopodiché le coordinate delle destinazioni (salvate nella variabile *locations* vengono trasformate da cartesiane in polari. Per calcolare l'angolo in funzione della posizione nel piano cartesiano si utilizza la funzione built-in di MATLAB *atan2*, mentre per calcolare il raggio si applica semplicemente il teorema di Pitagora. Gli angoli vengono successivamente ordinati insieme alle rispettive domande.
  Il secondo blocco di codice consiste nell'assegnazione di ogni destinazione alla route corrispondente: si parte dalla destinazione avente angolo minore e si aggiunge alla prima route; si prosegue con la seconda verificando che il vincolo di capacità del veicolo sia rispettato. Quando questo avviene si ripete il processo per la route successiva. Dopo aver assegnato l'ultima destinazione, si avrà a disposizione il cell array *routes* contenente gli indici appartenenti a ogni route. Una volta costruita l'ultima route disponibile (ricordando che *vehicles* è il numero massimo di veicoli utilizzabili), si vanno a distribuire le eventuali destinazioni rimanenti alle route appena create. Per fare ciò, si parte dalla prima destinazione in eccesso e la si aggiunge alla route più vicina avente capacità residua sufficiente per soddisfare la rispettiva domanda. In caso si trovi una destinazione avente domanda superiore alla capacità residua di ogni route si ferma l'algoritmo che restituisce il messaggio di soluzione ammissibile non trovata.
  Lo step finale consiste nell'applicare il TSP a ogni route generando così le soluzioni finali che, come detto, saranno salvate nel cell array *solution*.

3 Tabu Search

L'algoritmo Tabu è una meta-euristica iterativa utilizzata per migliorare una soluzione già esistente. Si basa su un'esplorazione dello spazio delle soluzioni vincolata a delle mosse ritenute *tabu*, cioè non permesse. Il processo iterativo consiste nel passare da una soluzione alla successiva cercando quest'ultima nel vicinato della soluzione attuale, talvolta ottenendo soluzioni peggiori: ciò permette di ottenere un'esplorazione più completa dello spazio delle soluzioni ed evita il problema di rimanere intrappolati in un minimo locale.
Per evitare dei cicli le soluzioni visitate vengono inserite in una coda contenente le soluzioni non ammissibili: una determinata soluzione visitata all'iterazione k viene definita *tabu* e non può essere rivisitata fino alla iterazione $k+\theta$, dove θ è detto *tabu duration* e rappresenta la lunghezza della coda.
L'applicazione del tabu search al problema CVRP segue le indicazioni del lavoro [4].

3.1 Documentazione

Si descrivono ora le funzioni MATLAB implementate.

- $S = \text{tabu}(S, coords, demand, capacity, N\_max, alpha\_max, M, P, \ldots$
  $beta\_max, theta\_min, theta\_max)$
  In questa funzione viene implementata la vera e propria ricerca *tabu*: data la soluzione iniziale S, la funzione cerca $alpha\_max$ soluzioni "vicine" usando l'algoritmo TANE ed altrettante usando l'algoritmo TANEC. Per ogni vicino si controlla se esso risulta ammissibile come soluzione, cioè se è presente o meno nella *tabu list*, la quale ha lunghezza $\theta \in [theta\_min, theta\_max]$. Tra le soluzioni ammissibili trovate viene selezionata quella con costo minore e si ripete il processo per (al più) N_max volte.

- $S = \text{tane}(S, coords, demand, capacity, M, P, beta\_max)$
  Questa funzione implementa l'omonimo algoritmo di costruzione di un "vicino" della soluzione S data in input.
  L'algoritmo consiste nella selezione casuale di due routes presenti in S, dalle quali vengono estratti (in modo casuale) al più M vertici dalla prima route ed al più P vertici dalla seconda. Tali sottoinsiemi di vertici vengono scambiati tra loro ed inseriti nella nuova route tramite la procedura di *insertion*, prestando attenzione a non violare il vincolo di capacità. Successivamente, le nuove routes vengono ottimizzate tramite l'algoritmo *2-opt*.

- $S = \text{tanec}(S, coords, demand, capacity, M, P, beta\_max)$
  In questa funzione, in maniera analoga alla precedente, viene implementato l'algoritmo TANEC per la costruzione di un "vicino" della soluzione S data in input.
  La struttura dell'algoritmo non risulta molto diversa dal TANE, ma, a differenza di quest'ultimo, la selezione dei nodi da scambiare tra le routes non viene effettuata in modo casuale, bensì viene data priorità ai vertici che risultano più lontani dal

centroide della loro route e più vicini a quello dell'altra.

- $R = \text{insertion}(R, VP, coords)$
  Questa funzione permette di inserire, in modo sequenziale, un insieme di vertici VP in una specifica route R.
  L'algoritmo tiene conto dell'aumento di costo che ogni singolo inserimento provoca, per cui ogni vertice viene inserito in modo da minimizzare tale costo. In particolare, un qualsiasi vertice $v_k \in VP$ viene inserito tra i nodi $i, j \in R$ tali per cui il valore $c_{ik} + c_{kj} - c_{ij}$ è minimo.

- $R = \text{opt2}(R, coords, beta\_max)$
  Questa funzione applica la strategia di ottimizzazione *2-opt* alla route R al più $beta\_max$ volte.
  In particolare, data la route $R = [v_0, v_1, v_2, \ldots, v_k, v_0]$ l'algoritmo calcola quanto è conveniente scambiare, a livello di costo, il sottoinsieme di archi $X = \{(v_i, v_{i+1}), (v_j, v_{j+1})\}$ con la coppia $Y = \{(v_i, v_j), (v_{i+1}, v_{j+1})\}$. Ciò viene eseguito, per ogni coppia di archi, calcolando l'*improvement* $\delta_x = (c_{i,i+1} + c_{j,j+1}) - (c_{ij} + c_{i+1,j+1})$, cioè di quanto varia il costo effettuando tale scambio. Ad ogni iterazione vengono scambiati gli archi che corrispondono al $\delta_{max} = \max\{\delta_x\}$ se $\delta_{max} > 0$, altrimenti l'algoritmo si arresta.

4 Risultati ottenuti

Per testare l'efficienza degli algoritmi presentati sono stati scelti dei dataset dalla libreria disponibile online CVRPLIB [1], sui quali sono stati effettuati gli esperimenti.
Ogni dataset è composto da un insieme di n nodi, ciascuno caratterizzato da una *label* corrispondente alla posizione nel dataset (il nodo 1 corrisponde al deposito); le coordinate dei nodi ed il valore della domanda per ognuno di essi sono note. Sono inoltre dati la quantità k di veicoli disponibili, la loro capacità ed il costo della soluzione ottima.
Il primo algoritmo applicato è lo Sweep, la cui soluzione è deterministica e non riceve dati di input.
Alla soluzione ottenuta è stato applicato il Tabu Search che, avendo al suo interno delle componenti aleatorie, ad ogni test restituisce una soluzione diversa. Perciò sono stati effettuati molteplici esperimenti ed in questa sezione riportiamo, per ogni dataset, il miglior risultato, in relazione al costo della soluzione, ottenuto da 10 test consecutivi.
I risultati qui riportati sono relativi ai seguenti parametri scelti per bilanciare efficienza/costo computazionale dell'algoritmo:

- $N\_max = 100$;
- $alpha\_max = 100$;
- $M = 2$;

- $P = 2$;
- $beta\_max = 1000$;
- $[theta\_min, theta\_max] = [10, 20]$.

Riportiamo qui le caratteristiche dei dataset usati ed i risultati ottenuti da ciascuno di essi.

1. **B-n39-k5**
   Caratteristiche del dataset:
   - n = 39;
   - k = 5;
   - capacità = 100;
   - costo soluzione ottima = 549.

   tempo medio di esecuzione sweep = 0.624s
   costo soluzione sweep = 613

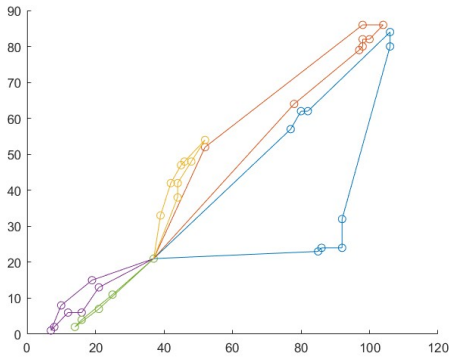
Route	Vertici	Costo	Capacità impiegata
1	1 6 16 12 13 23 17 27 26 2 1	211.66	86%
2	1 33 3 34 22 25 38 31 24 28 1	192.33	94%
3	1 32 37 18 39 36 5 29 4 1	74.19	99%
4	1 15 14 7 19 11 8 9 1	75.55	90%
5	1 30 35 21 20 10 1	59.71	71%

Table 1: B-n39-k5: SWEEP

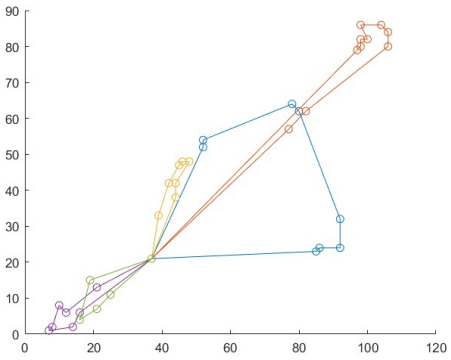
tempo medio di esecuzione tabu = 82.34s
costo miglior soluzione tabu = 555

Route	Vertici	Costo	Capacità impiegata
1	1 28 39 33 26 13 12 16 6 1	162.89	100%
2	1 2 27 23 17 31 24 38 25 22 34 3 1	197.05	95%
3	1 32 37 18 36 5 29 4 1	60.5	84%
4	1 8 20 21 7 19 14 11 9 1	77.21	99%
5	1 30 10 35 15 1	57.48	62%

Table 2: B-n39-k5: TABU SEARCH



(a) B-n39-k5: SWEEP



(b) B-n39-k5: TABU SEARCH

2. **B-n52-k7**
   Caratteristiche del dataset:

   - n = 52;
   - k = 7;
   - capacità = 100;
   - costo soluzione ottima = 747.

   tempo medio di esecuzione sweep = 0.83s
   costo soluzione sweep = 922

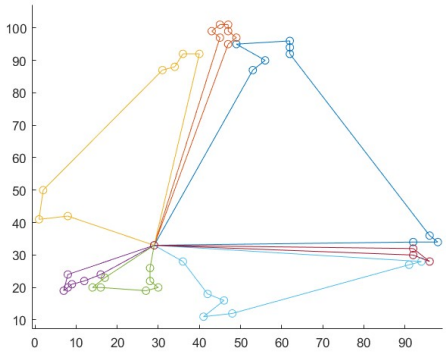
Route	Vertici	Costo	Capacità impiegata
1	1 22 40 15 24 13 51 23 46 5 1	226.33	98%
2	1 27 18 50 16 20 33 35 14 47 1	145.84	100%
3	1 17 10 39 49 3 44 8 34 36 1	157.64	99%
4	1 52 6 9 28 30 38 1	54.22	100%
5	1 48 11 45 7 37 26 42 1	49.97	83%
6	1 41 43 21 2 31 19 4 32 1	152.77	98%
7	1 29 25 12 1	136.21	28%

Table 3: B-n52-k7: SWEEP

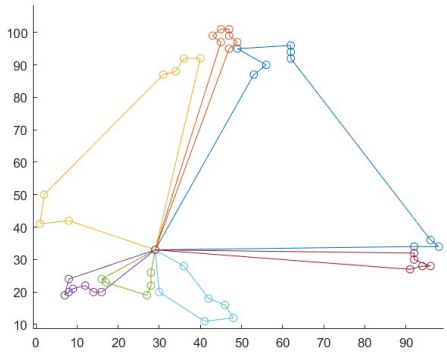
tempo medio di esecuzione tabu = 123.15s
costo miglior soluzione tabu = 829

Route	Vertici	Costo	Capacità impiegata
1	1 22 40 15 24 13 51 23 46 5 1	226.33	98%
2	1 27 18 50 16 20 33 35 14 47 1	145.84	100%
3	1 17 10 39 49 3 44 8 34 36 1	157.64	99%
4	1 45 11 30 28 9 6 52 1	57.15	98%
5	1 38 48 7 26 42 1	42.23	76%
6	1 37 2 31 19 21 43 41 1	63.53	96%
7	1 4 32 25 29 12 1	136.93	39%

Table 4: B-n52-k7: TABU SEARCH



(a) B-n52-k7: SWEEP



(b) B-n52-k7: TABU SEARCH

3. **A-n80-k10**
   Caratteristiche del dataset:

   - n = 80;
   - k = 10;
   - capacità = 100;
   - costo soluzione ottima = 1763.

   tempo medio di esecuzione sweep = 1.397s
   costo soluzione sweep = 2102

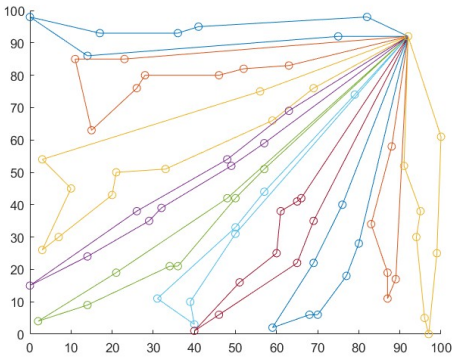
Route	Vertici	Costo	Capacità impiegata
1	1 50 39 59 33 5 46 74 1	191.61	92%
2	1 51 23 55 73 77 71 67 68 37 1	190.95	87%
3	1 54 10 56 70 57 16 34 65 78 43 1	243.8	100%
4	1 52 40 42 66 48 26 47 61 4 1	240.52	100%
5	1 32 20 36 27 76 21 18 75 1	255.15	80%
6	1 14 30 28 58 17 62 60 1	218.39	97%
7	1 13 45 6 31 79 44 69 7 24 1	214.85	96%
8	1 63 25 9 38 3 35 12 1	197.38	86%
9	1 2 64 53 80 29 1	163.66	96%
10	1 8 11 72 49 19 15 22 1	185.92	95%

Table 5: A-n80-k10: SWEEP

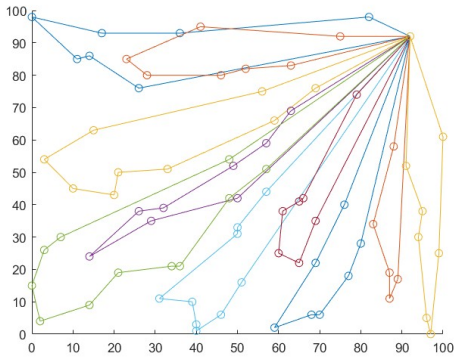
tempo medio di esecuzione tabu search = 127.32s
costo miglior soluzione tabu = 1958

Route	Vertici	Costo	Capacità impiegata
1	1 50 59 33 5 23 46 73 1	198.37	85%
2	1 74 39 51 77 71 67 68 37 1	144.53	92%
3	1 54 55 10 56 16 34 65 78 43 1	210.42	86%
4	1 52 4 61 47 42 48 26 18 1	211.26	97%
5	1 40 57 70 66 36 27 20 76 21 32 75 1	265.07	99%
6	1 79 69 44 17 62 58 60 28 30 1	224.51	99%
7	1 14 13 45 6 31 7 24 1	157.1	94%
8	1 12 35 3 38 9 25 63 1	197.38	86%
9	1 2 64 53 80 29 1	163.66	96%
10	1 8 11 72 49 19 15 22 1	185.92	95%

Table 6: A-n80-k10: TABU SEARCH



(a) A-n80-k10: SWEEP



(b) A-n80-k10: TABU SEARCH

4. **M-n101-k10**
   Caratteristiche del dataset:

   - n = 101;
   - k = 10;
   - capacità = 200;
   - costo soluzione ottima = 820.

   tempo medio di esecuzione sweep = 1.44s
   costo soluzione sweep = 1041

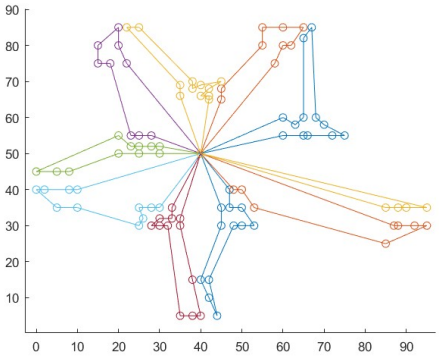
Route	Vertici	Costo	Capacità impiegata
1	1 91 88 87 84 83 85 86 93 95 89 90 92 1	123.7	100%
2	1 99 97 96 94 98 101 100 2 76 1	93.45	95%
3	1 6 4 8 5 3 7 9 10 13 15 12 11 1	97.13	90%
4	1 14 16 17 20 19 18 29 27 24 1	94.88	100%
5	1 22 23 26 28 31 40 37 35 30 25 21 1	84.8	90%
6	1 33 34 38 39 36 32 52 51 53 50 48 1	97.93	100%
7	1 44 43 47 49 46 45 61 59 57 60 41 42 1	105.74	100%
8	1 70 69 58 55 54 56 65 62 73 63 67 68 1	104.78	100%
9	1 66 64 75 81 71 74 78 80 1	124.42	90%
10	1 82 79 77 72 1	114.44	40%

Table 7: M-n101-k10: SWEEP

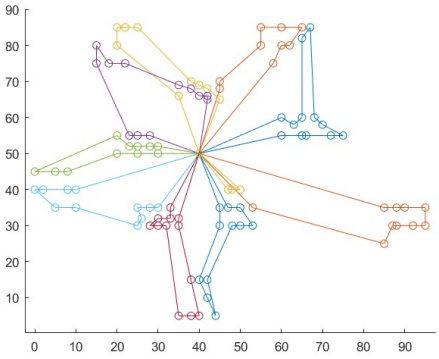
tempo medio di esecuzione tabu search = 144.62s
costo miglior soluzione tabu = 953

Route	Vertici	Costo	Capacità impiegata
1	1 91 88 87 84 83 85 86 93 95 89 90 92 1	123.7	100%
2	1 99 97 96 94 98 101 100 3 2 1	93.85	100%
3	1 11 16 17 15 13 10 7 5 76 1	91.66	90%
4	1 6 4 8 9 12 14 18 20 19 29 27 24 1	92.81	95%
5	1 22 23 26 28 31 40 37 35 30 25 21 1	84.8	90%
6	1 33 34 38 39 36 32 52 51 53 50 48 1	97.93	100%
7	1 44 43 47 49 46 45 61 59 57 60 41 42 1	105.74	100%
8	1 67 63 73 62 65 56 54 55 58 69 70 1	104.13	95%
9	1 75 81 80 78 74 71 72 77 79 82 1	129.2	100%
10	1 68 66 64 1	29.35	35%

Table 8: M-n101-k10: TABU SEARCH



(a) A-n80-k10: SWEEP



(b) A-n80-k10: TABU SEARCH

5. **E-n101-k8**
   Caratteristiche del dataset:

   - n = 101;
   - k = 8;
   - capacità = 200;
   - costo soluzione ottima = 817.

   tempo medio di esecuzione sweep = 1.21s
   costo soluzione sweep = 858

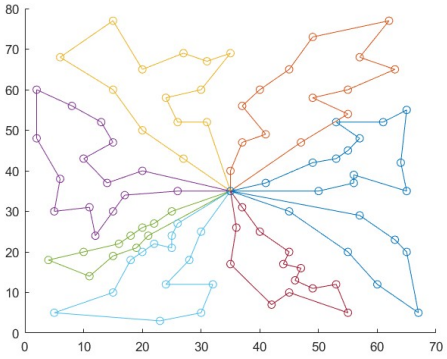
Route	Vertici	Costo	Capacità impiegata
1	1 13 81 69 25 30 35 79 34 80 4 78 77 29 1	98.25	90%
2	1 51 82 52 10 36 72 66 67 21 31 71 2 70 28 1	127.01	100%
3	1 32 89 63 11 33 91 64 12 65 50 20 8 53 1	132.58	94.5%
4	1 19 84 9 83 49 48 37 47 46 18 85 62 6 61 90 1	128.36	99.5%
5	1 7 97 100 94 86 17 87 45 92 99 60 1	76.31	98.5%
6	1 95 96 98 93 38 101 15 39 44 16 58 43 88 14 1	113.01	99%
7	1 59 3 42 23 24 57 76 75 73 74 22 41 54 1	88.49	88.5%
8	1 27 5 40 68 26 56 55 1	94.68	59%

Table 9: E-n101-k8: SWEEP

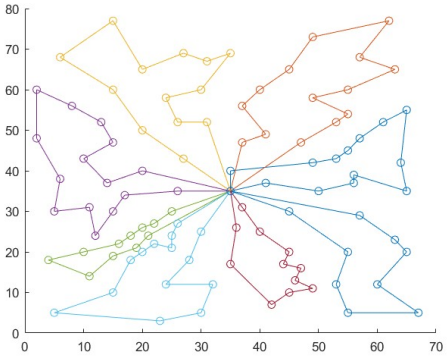
tempo medio di esecuzione tabu search = 151.5s
costo miglior soluzione tabu = 848

Route	Vertici	Costo	Capacità impiegata
1	1 28 77 78 4 80 79 35 30 25 69 81 13 29 1	94.18	92.5%
2	1 51 34 82 52 10 36 72 66 67 21 31 71 2 70 1	126.9	97.5%
3	1 32 89 63 11 33 91 64 12 65 50 20 8 53 1	132.58	94.5%
4	1 19 84 9 83 49 48 37 47 46 18 85 62 6 61 90 1	128.35	99.5%
5	1 7 97 100 94 86 17 87 45 92 99 60 1	76.31	98.5%
6	1 95 96 98 93 38 101 15 39 44 16 58 43 88 14 1	113.09	99%
7	1 59 3 42 23 76 75 73 74 22 41 54 1	70.03	71%
8	1 27 5 57 24 68 40 26 56 55 1	107.08	76.5%

Table 10: E-n101-k8: TABU SEARCH



(a) E-n101-k8: SWEEP



(b) E-n101-k8: TABU SEARCH

6. **M-n151-k12**
   Caratteristiche del dataset:

   - n = 151;
   - k = 12;
   - capacità = 200;
   - costo soluzione ottima = 1053.

   tempo medio di esecuzione sweep = 1.827s
   costo soluzione sweep = 1121

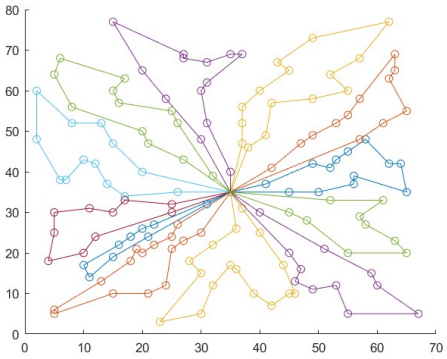
Route	Vertici	Costo	Capacità impiegata
1	1 139 13 151 81 151 69 25 30 122 130 4 78 117 77 29 1	77.68	96.5%
2	1 80 79 35 136 36 137 121 82 34 103 51 112 1	95.68	99%
3	1 133 2 123 52 10 104 72 66 67 129 21 31 71 102 70 1	116.95	97%
4	1 28 32 11 109 132 33 91 127 64 65 12 63 128 1	109.84	95.5%
5	1 147 89 149 124 20 108 50 144 48 8 107 53 1	98.32	90%
6	1 19 83 49 125 37 47 46 126 9 115 84 61 90 1	104.32	91.5%
7	1 148 119 6 85 18 114 87 17 62 7 1	80.68	97%
8	1 97 100 105 94 86 142 45 92 60 113 1	66.15	97.5%
9	1 95 96 93 38 99 101 120 141 39 15 143 43 98 118 14 1	92.37	96%
10	1 59 138 88 145 44 16 58 3 116 146 42 23 134 74 41 54 1	103.27	98%
11	1 22 73 75 76 57 24 68 40 140 111 106 1	97.02	96.5%
12	1 27 150 5 26 56 131 55 135 110 1	79.43	68%

Table 11: M-n151-k12: SWEEP

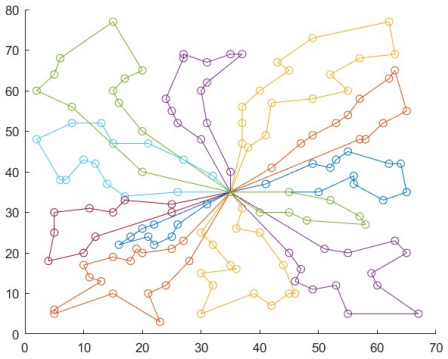
tempo medio di esecuzione tabu = 214.9s
costo miglior soluzione tabu = 1070

Route	Vertici	Costo	Capacità impiegata
1	1 13 69 81 151 135 25 30 122 4 78 117 77 29 1	75.76	94%
2	1 80 130 79 35 36 136 121 82 34 103 51 112 1	88.22	99%
3	1 133 2 123 52 10 104 72 137 66 67 129 21 31 71 102 70 1	120.8	100%
4	1 28 32 11 109 132 33 91 64 127 63 149 89 128 1	85.46	98%
5	1 8 124 20 108 12 65 50 144 37 48 19 1	117.52	81.5%
6	1 147 53 107 83 49 125 47 46 126 9 115 84 61 90 1	89.73	97.5%
7	1 148 119 6 85 18 114 87 17 62 7 1	80.68	97%
8	1 97 100 105 94 86 60 93 96 95 113 1	49.62	99%
9	1 88 43 143 44 15 39 141 120 45 142 92 101 99 38 98 118 1	110.79	99.5%
10	1 54 59 41 74 134 23 42 146 16 58 145 116 3 138 14 1	88.7	93%
11	1 22 73 75 76 57 24 68 40 140 26 56 5 111 1	111.08	99.5%
12	1 106 27 150 131 55 110 139 1	51.65	59.5%

Table 12: M-n151-k12: TABU SEARCH



(a) M-n151-k12: SWEEP



(b) M-n151-k12: TABU SEARCH

7. **M-n200-k17**
   Caratteristiche del dataset:

   - n = 200;
   - k = 17;
   - capacità = 200;
   - costo soluzione ottima = 1373.

   tempo medio di esecuzione sweep = 3.37s
   costo soluzione sweep = 1418

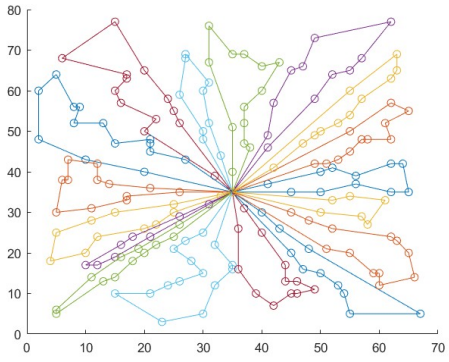
Route	Vertici	Costo	Capacità impiegata
1	1 155 139 13 81 151 164 25 30 122 69 117 185 29 1	68.95	99%
2	1 77 197 78 159 4 80 130 170 79 35 165 186 1	78.95	92.5%
3	1 112 51 103 158 34 82 121 136 36 137 10 1	89.54	94.5%
4	1 177 52 104 162 72 66 67 189 21 123 2 1	105.68	96%
5	1 133 70 102 71 31 129 161 132 33 182 91 163 28 1	93.39	92.5%
6	1 168 32 190 11 109 64 127 160 191 128 1	71.66	98%
7	1 89 149 63 12 65 50 176 108 20 124 183 8 147 1	110.92	100%
8	1 53 154 107 195 83 49 125 169 48 144 37 47 9 19 1	103.46	96%
9	1 167 84 200 115 175 126 46 18 85 119 61 90 1	77.97	97.5 %
10	1 148 6 174 114 87 17 62 100 105 97 7 157 1	75.59	99%
11	1 184 94 86 142 192 92 60 113 1	62.11	95%
12	1 95 99 194 45 141 39 120 193 101 38 93 152 96 1	85.61	90.5%
13	1 14 118 98 88 145 173 43 143 15 44 16 58 179 3 138 1	92.14	98.5%
14	1 153 59 116 146 42 23 134 76 75 172 74 41 54 1	68.14	98%
15	1 22 73 198 57 187 24 68 199 181 106 1	93.22	95%
16	1 111 156 5 140 188 40 171 26 56 166 180 150 27 1	82.67	95%
17	1 196 55 131 135 178 110 1	58.19	56%

Table 13: M-n200-k17: SWEEP

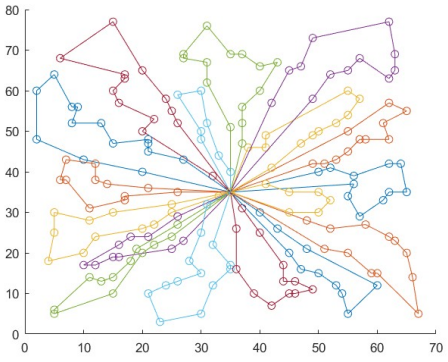
tempo medio di esecuzione tabu = 226.8s
costo miglior soluzione tabu = 1379

Route	Vertici	Costo	Capacità impiegata
1	1 151 81 178 55 135 164 25 30 122 69 117 185 1	78.83	89.5%
2	1 77 197 78 159 4 80 130 170 79 35 165 186 1	78.95	92.5%
3	1 112 51 103 158 34 82 121 10 2 177 133 1	70.21	99%
4	1 52 104 162 72 136 36 137 66 67 189 21 123 1	116.52	97.5%
5	1 163 109 91 127 64 182 33 132 161 129 31 71 102 70 1	97.14	98.5%
6	1 128 191 160 11 190 32 168 28 1	55.68	86%
7	1 89 149 63 12 65 50 176 108 20 124 183 8 147 1	110.92	100%
8	1 53 154 107 195 83 49 125 169 48 144 37 47 9 19 1	103.46	96%
9	1 167 84 200 115 175 46 126 85 119 61 90 1	71.98	96.5%
10	1 148 6 174 18 114 87 17 62 100 105 97 7 157 1	80.21	100%
11	1 118 98 194 92 192 142 86 94 60 184 1	64.6	97%
12	1 95 99 101 193 120 45 141 39 15 38 93 152 96 1	88.56	91.5%
13	1 113 14 88 145 173 43 143 44 16 58 179 3 138 1	80.28	95.5%
14	1 153 59 116 146 42 23 134 76 75 172 74 41 54 1	68.14	98%
15	1 22 73 198 57 187 24 40 199 181 106 1	79.98	98%
16	1 111 156 5 140 188 68 171 26 56 166 131 180 150 1	95.47	93%
17	1 27 196 110 13 139 155 29 1	38.41	64.5%

Table 14: M-n200-k17: TABU SEARCH



(a) M-n200-k17: SWEEP



(b) M-n200-k17: TABU SEARCH

4.1 Sweep vs Tabu

Una prima analisi dei due metodi utilizzati può essere fatta osservando i tempi di esecuzione, nonostante essa dipenda anche dall'hardware utilizzato per effettuare gli esperimenti.
Nella tabella sottostante riportiamo i tempi medi registrati dai due algoritmi su 100 test.

Dataset	Tempo Sweep (s)	Tempo Tabu (s)
B-n39-k5	0.624	82.34
B-n52-k7	0.83	123.15
A-n80-k10	1.397	127.32
M-n101-k10	1.44	144.62
E-n101-k8	1.21	151.5
M-n151-k12	1.827	214.9
M-n200-k17	3.37	226.8

Table 15: Sweep vs Tabu: time

Indubbiamente lo Sweep risulta essere molto più veloce, vista la sua natura costruttiva; tuttavia il Tabu è un algoritmo estremamente sensibile ai valori dei parametri utilizzati, come ad esempio il numero di iterazioni o il numero di vicini, per cui l'elevato costo computazionale può essere giustificato dalla scelta dei parametri oltre al costo intrinseco dell'algoritmo stesso. Osserviamo inoltre che, in generale, il tempo di esecuzione cresce con la dimensione del problema.

Un paragone più significativo sulle performance degli algoritmi riguarda il costo della soluzione trovata dai due metodi.
Riportiamo in tabella il costo della soluzione ottima, i costi delle soluzioni di Sweep e Tabu Search e la percentuale di miglioramento da parte del Tabu della soluzione iniziale data dallo Sweep.

Dataset	Costo ottimo	Costo Sweep	Costo Tabu	Miglioramento
B-n39-k5	549	613	555	9.46%
B-n52-k7	747	922	829	10.09%
A-n80-k10	1763	2102	1958	6.85%
M-n101-k10	820	1041	953	8.45%
E-n101-k8	817	858	848	1.17%
M-n151-k12	1053	1121	1070	4.55%
M-n200-k17	1373	1418	1379	2.75%

Table 16: Sweep vs Tabu: cost

Osserviamo che, in generale, gli algoritmi si comportano molto bene sui problemi affrontati: a livello di costo non si notano particolari cambiamenti di performance al variare della dimensione del problema, mentre si registrano sempre dei miglioramenti da parte del Tabu Search rispetto alla soluzione dello Sweep.
Vediamo nel dettaglio il comportamento di ogni metodo.
Lo **Sweep** riesce sempre a trovare una buona soluzione, creando routes ben distinte tra loro, nei limiti delle potenzialità del metodo stesso. I risultati ottenuti non sono condizionati dalla dimensione del problema affrontato (vedi M-n200-k17), bensì dalla sua struttura: disposizione dei clienti nello spazio rispetto al deposito e loro rispettiva domanda (vedi A-n80-k10).
Come già detto, il **Tabu Search** riesce sempre a migliorare la soluzione data dallo Sweep, il ché non è scontato viste le buone performance del metodo costruttivo. Tuttavia ciò era prevedibile poiché lo Sweep effettua un vero e proprio *clustering* dei nodi dove ogni cluster coincide con una route, mentre il Tabu permette di "sciogliere" questi clusters creando routes più complesse con traiettorie che si intersecano tra loro.
Osserviamo che si riesce ad arrivare ad un costo relativamente vicino a quello ottimo: ciò è dovuto alla potente tecnica di esplorazione dell'algoritmo e non risente della dimensione elevata del problema (vedi M-n200-k17), bensì della bontà della soluzione iniziale considerata. Il metodo risulta essere molto sensibile alla scelta dei parametri, per cui i risultati ottenuti non sono i migliori ottenibili da questo algoritmo. Effettuando un adeguato tuning dei parametri di input è possibile ottenere soluzioni migliori, spesso aumentando il costo computazionale.

References

[1] "Cvrplib." http://vrp.atd-lab.inf.puc-rio.br/index.php/en/.

[2] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations Research*, vol. 22, no. 2, pp. 340–349, 1974.

[3] "Tsp solver." https://it.mathworks.com/help/optim/ug/traveling-salesman-problem-based.html.

[4] G. Barbarosoglu and D. Ozgur, "A tabu search algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 26, no. 3, pp. 255–270, 1999.