Progetto di Algoritmi Avanzati

Capacitated Vehicle Routing Problem
Algoritmi: Costruttivi, a 2 Fasi e Genetici

Giulio Pilotto - Matricola:1140718

Luglio 2019

Abstract

Questo elaborato presenta il progetto di Algoritmi Avanzati sottomesso al prof. Bresolin dell'Università di Padova. Il progetto richiede di implementare almeno 2 algoritmi che risolvono istanze di : Capacitated Vehicle Routing Problem. Nel seguente elaborato oltre ad implementare i 2 algoritmi richiesti , il primo come metodo costruttivo; Clarke and Wright e il secondo come metodo a 2 fasi:ClusterFirst - Route Second per il quale sono state implementate due tecniche di routing: Nearest-Neighbourn e Dijkastra. Sono stati implementati altri 2 algoritmi uno che cade sempre nella classe dei metodi a 2 fasi: RouteFirst - ClusterSecond, e un altro che cade nella categoria degli algoritmi metauristici detti anche algoritmi genetici. Inoltre, è stato implementato un metodo di selezione dei centroidi che tiene conto sia della distanza dal deposito sia della distanza inter-cluster. Tutti gli algoritmi sono stati testati sulle 16 istanze fornite dalla libreria TSPLIB95. I risultati confermano che a fronte di una minore accuratezza di una soluzione rispetto all'ottimo ma una maggiore richiesta di risorse in termini di tempo, possono portare a soluzioni ben approssimate. In particolare scegliere i centrodi con il metodo RadiusRadar permette di guadagnare qualche decimo a fronte di un tempo inore. Le soluzioni posso essere migliorate attrverso gli algoritmi genetici che fungono da esploratori di uno spazion locale.

1 Introduzione

Con la nascita dei nuovi servizi per il cliente, che danno assistenza e portao i prodotti direttamente sulla soglia di casa (es:Amazon Prime), la gestione della logistica predilige gran parte degli investimenti. Questo fatto diventa ancor più importante se diamo un occhiata ai dati forniti dall' Associazione Nazionale Filiera Industria Automobilistica (ANFIA) nel 2017 1 , dove viene sottolineato che i camion movimentano l'80% delle merci su terra [1]. Gli investimenti non riguardano solo le strutture o gli strumenti fisici, ma soprattutto i software per la gestione e l'ottimizzazione della distribuzione dei beni. Grazie all'informatizzazione dei processi da parte del nuovo programma di industrializzazione chiamato "industria 4.0", si può godere di banche dati ricche ed aggiornate, attraverso le quali, possono essere implementate tecniche di ricerca operativa e ottimizzazione.

Il Vehicle Routing Problem **(VRP)** è un classe reale di problemi di soddisfacimento di vincoli, in inglese detto anche Constraint Satisfaction Problem **(COP)**. Alla fine degli anni cinquanta Dantzig and Ramser hanno formalizzato il problema [2] che riguarda la distribuzione di benzina da un deposito principale ad un grande numero di stazioni di servizio. Da quel momento l'interesse nei problemi VRP è evoluto da un piccolo gruppo di matematici ad un grande gruppo di ricercatori da differenti discipline ancora oggi coinvolti.

VRP consiste nell'ottimizzare l'uso di un insieme di veicoli per prelevare della merce da uno o più depositi e consegnarle presso dei clienti che richiedono una certa quantità di merce. Le stazioni , i depositi sono distribuiti in uno spazio rappresentato da delle distanze. I mezzi si spostano percorrendo tali distanze, che possono essere rappresentate in maniera diversa. La maggioranza the problemi reali sono sepsso più complessi del classico VRP. Quindi in pratica VRP è aumentato con dei vincoli, come ad esempio la capacità del veicolo: dove ogni veicolo è caratterizzato da una capacità limitata di carico delle merci, passando quindi da VRP a **Capacitated - VRP**. Gli obiettivi sono molteplici:

- Minimizzare dei costi di trasporto (istanze percorse, consumo di carburante)
- Minimizzare il numero di veicoli
- Rispettare il vincolo di capacità imposto sui veicoli.
- Assegnare un percorso ad ogni singolo veicolo.

VRP è un problema di ottimizzazione combinatoria NP hard,che può essere risolto esattamente trovando l'ottimo solo per piccole istanze del problema. In ogni caso gli approcci euristici che non garantiscono l'ottimalità, forniscono ottimi risultati in pratica. Negli ultimi 30 anni sono stati applicati anche dei metodi meta-euristici

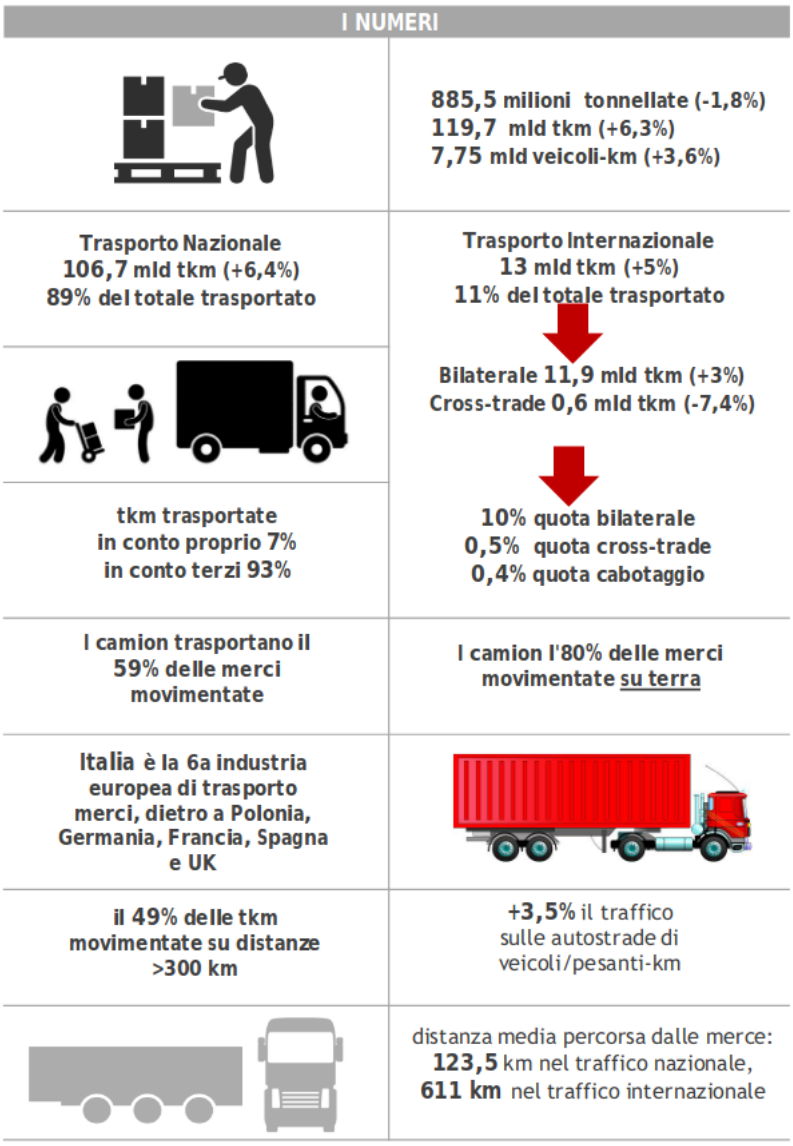

Figure 1: Italia, traffico merci su strada, ANFIA su dati ISTAT 2017

che hanno mostrato una nuova direzione alla ricerca sulla famiglia di problemi VRP. In questo elaborato vengono prese in esame le istanze fornite dalla libreria TSPLIB95[3].

Sono stati implementati i seguenti metodi per risolvere le istanze CVRP:

- Metodo Costruttivo: Clarke and Wright
- Metodo a due fasi:
  - Cluster-First Route-Second: Fisher and Jaikumar
  - Route-First Cluster-Second: Dijkastra
- Meta-euristiche: Algoritmi Genetici

1.1 Rappresentazione del problema

CVRP è la versione più comune dei problemi VRP. Ciò che caratterizza queta tipologia di problemi è il fatto che l servizio è di semplice ocnsgna senza raccolta. Inoltre le richieste dei clienti sononote a pripri e deterministiche e devono essere soddisfatte da un solo veicolo; tuti i veicoli sono identici e basati su un singolo deposito centrale. Gli unici vincoli imposti riguardano le capacità di veicoli. L'obbiettivo è minmizzare il costo totale di servizio che può essere una funzione del numero dei route, della lunghezza complessiva o del tempo di percorrenza. Consideraimo ora la rappresentazione su grafo di questo problema.

- Sia $G = (V,A)$ un grafo completo dove $V = \{0,...,n\}$ è l'insieme dei vertici e A quello degli archi.
- I vertici $i = i,...,n$ corrispondono ai clienti, mentre il vertice 0 corrisponde al deposito.
- Ad ogni arco $(i,j) \in A$ è associato un costo non negativo $c_{i,j}$, che rappresenta il costo di trasferimento dall'indice i all'indice j
- In genere l'uso di loop non è consentito e ciò è imposto definendo $c_{i,j}=+\infty \ \ \forall \ i \in V$.Nella nostra implementazione invece abbiamo definito $c_{i,j}=0 \ \ \forall \ i \in V$.
- Dato un insieme $S \subseteq V$, $d(S)$ denota la richiesta complessiva dei clienti in S: $d(S) = \sum_{s \in S} d_s$.
- Indicando con r una *route*, $d(r)$ denota la richiesta complessiva dei vertici da esso visitati.
- Un insieme K di veicoli è disponibile presso il deposito. Tali veicoli sono tutti identici di capacità C; una semplice condizione di ammissibilità del problema richiede $d_i \leq C \ \forall i \in V \setminus \{0\}$.
- Ogni veicolopuò percorrere al più un *route*.
- Si assume che $K \geq K_{min}$ dove K_{min} è il minimo numero di veicoli necessari per servire tutti i clienti.
- In questa implementazione utilizziamo il numero minimo di veicoli necessari per servire tutti i vertici in S, pari al *lower bound* $K_{min} = d(S)/C$

Gli obbiettivi ed i vincoli già citati nella sezione precedente vengono ora descritti in maniera formale, associando ai vincoli il modello matematico che li esprime.

$$M_{i,j}^k = \begin{cases} 1 : sse(i,j) \in A \wedge M_{i,j} \in r(k) \mid k \in K \wedge r \in route \\ 0 : altrimenti \end{cases}$$

Definiamo inoltre q_i la richiesta associata ad ogni cliente visitato da un circuito e C_k la capacità del veicolo $k \in K$.

La funzione obbiettivo descritta formalmente è la seguente:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{i,j} \cdot x_{i,j}^k \tag{1}$$

al quale sono applicati i seguenti vincoli:

$$\sum_{k \in K} \sum_{j \in V} M_{i,j}^k = 1 \ \forall i \in V \tag{2}$$

$$\sum_{i \in V} q_i \sum_{j \in V} M_{i,j}^k \leq C_k \ \forall k \in K \tag{3}$$

$$\sum_{j \in V} M_{0,j}^k = 1 \ \forall k \in K \tag{4}$$

$$\sum_{i \in V} M_{i,h}^k - \sum_{j \in V} M_{h,j}^k = 0 \ \forall k \in K, \ \forall h \in V \tag{5}$$

$$\sum_{i \in V} M_{i,0}^k = 1 \ \forall k \in K \tag{6}$$

La funzione obbiettivo 1, minimizzare il costo dei km totali percorsi da ogni singolo veicolo. Il vincolo 2 impone che ogni cliente deve essere servito da un solo veicolo. Il vincolo 3 assicura che il limite sulla capacità dei veicoli venga rispettato. I vincoli 4, 5 , 6 sono vincoli che impongono ad ogni veicolo di partire dal deposito il nodo 0, e collegarsi ad un nodo h, unico per ogni route,e di ritornare al deposito, indicato nella nostra implementazione sempre dal nodo 0. Questi vincoli definiscono la struttura della route. **Questa sezione va rivista**

2 Related Works

Quasi tutti i metodi implementati sono euristici perchè nessuno degli algoritmi può garantire di trovare una soluzione ottima per grandi istanze, in un limite di tempo computazionale ragionevole. Un approccio euristico non esplora l'intero spazio di ricerca, piuttosto cerca di trovare una soluzione basandosi sulle informazioni che ha sul problema. Le euristiche che risolvono istanze CVRP sono identificati come metodi costruttivi (**constructive**) e di raggruppamento **()clustering)**. I metodi costruttivi costruiscono una soluzione gradualmente,aggiornando continuamente l'informazione che riguarda il costo, ma potrebbe non contenere nessuna fase di miglioramento o di ottimizzazione della soluzione. Gli algoritmi costruttivi più famosi sono: Clarke and Wright's savings algorithm [4], [5] , [6] , Matching based algorithm e Multi-route improvement heuristic [6]. I metodi di clustering, risolvono il problema in due fasi , ed è per questo che vengono chiamati metodi a 2 fasi. l'approccio route-first cluster second è caratterizzato dalle due fasi seguenti:

- **Fase 1 - Clustering:** Un algoritmo di clustering viene utilizzato per raggruppare i clienti in cluster da dare in pasto alla seconda fase
- **Fase 2- Routing** Nella seconda fase, per ogni cluster creato nella prima fase viene ricercata la strada più corta sfruttando tecniche di ottimizzazione.

Alcuni famosi algoritmi a due fasi sono: Petal algortihm [7], Sweep algorithm [8] e Fisher and Jaikumar [9]. Questo studio investiga e compara le performance dei metodi costruttivi e di raggruppamento per risolvere istanze di CVRP. In particolare, nell'algoritmo di Fisher and Jaikumar, è stato implementato un algoritmo per la selezione dei centroidi chiamato Radar-Radius.Questo massimizza la distanza tra centroidi (inter-centroid distance) e la distanza tra i centroidi e il deposito in modo da garantire la migliore copertura, a seconda della distribuzione del deposito e dei clienti. Questo algoritmo è stato implementato scegliendo un deposito per ogni istanza in TSPLIB95,solitamente il deposito con id identificativo 1. Sono stati usati due metodi diversi per realizzare il routing tra i clienti di uno stesso cluster: Nearest Neighbourn e Dijkastra. I raggruppamenti formati dai metodi di clustering sono successivamente sottomesse a tecniche di ottimizzazione del routing. In particolare è stato implementato un Algoritmo Genetico **(GA)** [10], i quali sono famosi algoritmi utilizzati per risolvere istanze Travelling Salesman Problem (TSP). **Bensi i metodi costruttivi generano già delle soluzioni molto vicino all'ottimo, l'algoritmo Genetico è applicato anche al Savings algorithm per avere una misura di performance.**

3 Materiali e Metodi

3.1 Metodi Costruttivi

Un metodo in questa categoria costruisce le strade per i veicoli mentre cerca di minimizzare la distanza percorsa. L'algoritmo di Savings di Clarke and Wright [4], trova una soluzione utilizzando un'euristica, quindi il risultato non è sempre la soluzione ottima, ma dovrebbe risultare mol+to vicina all'ottimo con un risparmio di tempo computazionale elevato rispetto ad un algoritmo brute force.

3.2 Algoritmo di Savings di Clarke and Wright

L'algoritmo si basa su una coda ordinata di risparmi, detta anche *savings list*. Ogni record della coda è ottenuto collegando due *route* che non hanno nodi in comune eccetto il deposito. Dall'unione si ottiene una singola *route*, come mostrato nella figura 2, dove il nodo 0 rappresenta il deposito.

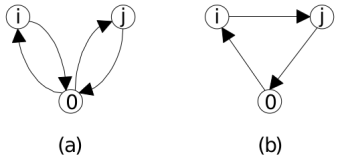

Figure 2: Inizialmente nella figura 1(a) i clienti i e j sono visitati da *route* separate. Un alternativa per visitare i due clienti con la stessa *route* è presentata in 1(b).

I costi di trasporto calcolati in 1(a) sono:

$$D_a = C_{o,i} + C_{i,0} + C_{o,j} + C_{j,0} \tag{7}$$

Mentre i costi di trasporto in 1(b) sono:

$$D_a = C_{o,i} + C_{i,j} + C_{o,j} \tag{8}$$

Ora il costo di trasporto risparmiato detto *saving distance* per visitare i e j nella stessa *route* invece che visitarli con due *route* separate è:

$$D_a = D_a + D_b = +C_{i,0} + C_{0,j} + C_{i,j} \tag{9}$$

3.2.1 Sequenziale e Parallelo

La distanza risparmiata è direttamente proporzionale alla vicinanza dei clienti e inversamente proporzionale alla vicinanza al deposito. Più grande è la distanza risparmiata più vicini si troveranno i clienti, e più distanti essi saranno dal deposito. Le coppie di clienti sono ordiate in ordine decrescente secondo il valore di *savings* che è stato calcolato sulla coppia. La costruzione delle *routes* parte dal primo record della coda. Quando una coppia i e j viene considerata se questa è già presente in una *route* allora si passa alla prossima coppia per l'inserimento. Ci sono due approcci proposti per l'algoritmo di *savings* **sequenziale** e **parallelo**.

- Nell'approccio **sequenziale** se una coppia di cliente non corrisponde con uno dei nodi presenti nella strada, la coppia viene ignorata. Ogni volta che un cliente è inserito in una *route*, l'algoritmo deve cominciare dall'inizio siccome le combinazioni che non erano disponibili fino a prima dell'inserimento potrebbero essere disponibili ora. Questo approccio crea una *route* alla volta, richiedendo di ripassare la coda di *savings* più di una volta.

- Nell'approccio **parallelo** crea più *routes* contemporaneamente. Quando una coppia i clienti è presa in considerazione, viene confrontata con tutte le *routes* che sono già state generate, e non solamente con una singola *route*. Se la coppia viene inserita in una route, tutte le *routes* generate fino ad ora vengono confrontate per trovare possibili *route* da unire ulteriormente con attraverso un *merge*. Altrimenti,se la coppia di clienti non corrisponde con nessuna *route*,allora una nuova *route* viene create, questa nell'approccio sequenziale invece verrebbe scartata. Nella versione parallela la lista dei *savings* viene passata una sola volta per tutta la sua lunghezza.

```
def Clarke&Wright_Sequenziale:
  Capacity = graph.getCapacity()
  Demand:list = graph.getCapacity()
  list savings = calculateSavings (i,j)  ∀ i,j ∈V | i≠j∧i≠0
  savings.sort(descending)
  list routes = []

  while(∑(r)∈R ∑(v)∈V (v ∈ r) ≠ |V| and len(savings)>0):
    pair = savings.pop()
    if(pair not served yet):
      route = Route.create(pair)

      for save in savings:
        if(i or j in save are in route):
          route.join(i or j)
          savings.remove(save)
          savings.updateDistance(i,j)
          restart from top of savings


################## MERGE ########################
if routeA != routeB:
if routeA.capacity() + routeB.capacity() <= Capacity:
Merge(routeA,RouteB)
routes.remove(RouteA or RouteB)
```

Listing 1: Implementazione parallela di Clarke and Wright

```
def Clarke&Wright_Parallelo:
  Capacity = graph.getCapacity()
  list savings = calculateSavings (i,j)  ∀ i,j ∈V | i≠j∧i≠0
  savings.sort(descending)
  list routes = []
  routes.append([0 - i - 0])   ∀ i ∈V | i≠0

  for save in savings:
    i , j = tuple(save)
    routeA = null
    routeB = null
```

```
    routeSelected = null
    for route in routes
      if route.served(i):
        routeA = route
        routeSelected = route
      elif route.served(j)
        routeB = route
        routeSelected = route

  ################## MERGE ########################
  if routeA != routeB:
    if routeA.capacity() + routeB.capacity() <= Capacity:
      Merge(routeA,RouteB)
      routes.remove(RouteA or RouteB)
```

Listing 2: Implementazione parallela di Clarke and Wright

References

[1] Associazione Nazionale Filiera Industria Automobilistica ANFIA. Dossier trasporto merci su strada, 2019.

[2] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science, Vol. 6, No. 1 (Oct., 1959), pp. 80-91*, 2008.

[3] G. Reinelt. Tsplib95. *Institut fur Angewandte Mathematik*, 1990.

[4] J.W Clarke, G. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *perations Research, Vol. 12, 1964, pp. 568-581.*

[5] Jens Lysgaard. Clarke and wright's savings algorithm. *Department of Management of Science and Logistic, the Aarhus School of Business.*

[6] R. Kawtummachai T.Pichpibula. An improved clarke and wright savings algorithm for the capacitated vehicle routing problem. *Department of Management of Science and Logistic, the Aarhus School of Business.*

[7] C. Hjorring D. M. Ryan and F. Glover. Extensions of the petal method for vehicle routing. *Journal of the Operational Research Society,44:289-296.*

[8] Alias S.M. Shamsuddin G.W. Nurchahyo, R.A. and M.. md. Sap. Sweep algorithm in vehicle routing problem for public transport. *Journal Antarabangsa (Teknologi Maklumat).*

[9] M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Journal of the Operational Research Society,44:289-296.*

[10] D.E: Goldberg. Genetic algorithm in search, optimization and machine learning". *New York Addison-Wesley.*