Routing

Cenni di TSP e VRP

11/07/2001 7.01

Contenuto e scopo presentazione

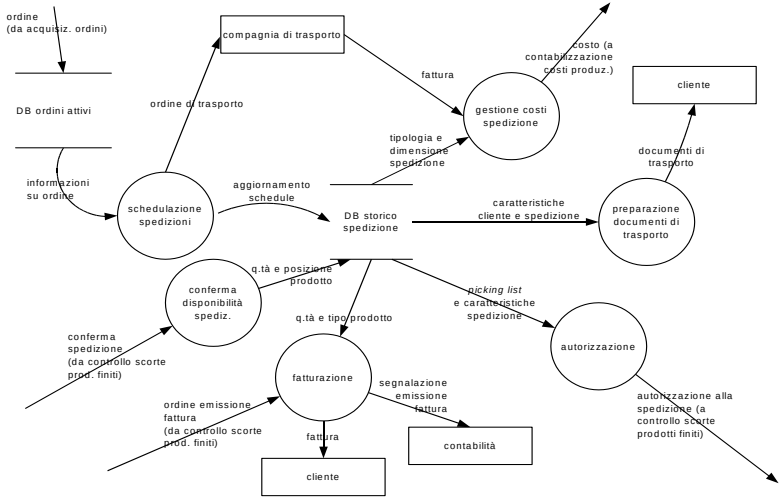
Contenuto

- vengono introdotti modelli e metodi per problemi di Commesso Viaggiatore: *Traveling Salesman Problem* (TSP)

Scopo

- fornire strumenti di supporto alle decisioni operativo in ambito logistico
- evidenziare pregi e limiti dei modelli matematici quando sono applicati al di fuori dei contesti per cui sono stati sviluppati
- fornire delle metodologie base che possono essere generalizzate a molti problemi logistici

Gestione spedizioni



Reti

Rete: $G=(N,A)$

- insieme N di nodi $i = 1, 2, 3,..., n$
- insieme A contenuto in $(N \times N)$ di archi $(i,j) \in N \times N$, a ogni arco (i,j) è associata un costo (distanza) c_{ij}

In logistica capita spesso di lavorare su reti per problemi di distribuzione (ma anche per altre situazioni),

i nodi rappresentano: magazzini, centri di distribuzione, clienti, task;

gli archi rappresentano: strade, set-up

Problemi su reti

- definizione di strutture distributive
  - alberi minimi
- trasporto elementare
  - percorsi minimi
  - trasporto
  - transshipment
  - vehicle scheduling problem
  - ...
- distribuzione/raccolta complessi
  - commesso viaggiatore
  - ...

Il problema del commesso viaggiatore

Problema:

Un commesso viaggiatore deve visitare i suoi clienti in un certo numero di città.

Vuole partire da casa e farvi ritorno senza passare più di una volta in ciascun'altra città e facendo meno strada possibile.

Traveling Salesman Problem — TSP

Applicazioni

pianificare le consegne ai clienti

prelievi in un magazzino automatico

sequenziamento di lavori su una macchina con tempi (o costi) di attrezzaggio dipendenti dalla sequenza

tagliare la carta da parati

lay out circuiti integrati

Formulazione su una rete

Data una rete: $G=(N,A)$

- circuito: una successione di archi che si chiude su stessa

$$(i,j), (j,k), (k,l), \dots , (z,i)$$

- costo di un circuito:

$$c_{ij} + c_{jk} + c_{kl} + \dots + c_{zi}$$

- Circuito Hamiltoniano: un circuito che tocca tutti i nodi, ciascuno una sola volta

Il problema del commesso viaggiatore consiste nel determinare un circuito hamiltoniano di costo minimo.

TSP

Traveling Salesman Problem *TSP(G,c)*

Istanza: una rete connessa $G=(N,A)$, un costo (distanza) c_{ij} associato ad ogni arco $(i,j) \in A$.

Soluzione: un circuito hamiltoniano, i.e., un circuito che tocchi una sola volta tutti i nodi della rete.

Obiettivo: minimizzare il costo (la lunghezza) del circuito.

Il problema della carta da parati

- Si voglio tagliare *n* fogli di carta da parati da un rotolo.
- Ogni rotolo ha dei disegni che si ripetono identici e sono lunghi *1*
- L'inizio del rotolo ha posizione *0*, nel foglio *i* il disegno deve cominciare a *s(i)* e finire a *f(i)*

Il problema della carta da parati

attaccando il foglio *j* di seguito al foglio *i* si spreca una lunghezza

$$c_{ij} = s(j) - f(i) \quad \text{se } f(i) \le s(j)$$
$$c_{ij} = s(j) - f(i) + 1 \quad \text{se } f(i) > s(j)$$

aggiungendo un foglio $n+1$ *con*

$$s(n+1) = f(n+1) = 0$$

si ottiene è un *TSP* con $n+1$ città

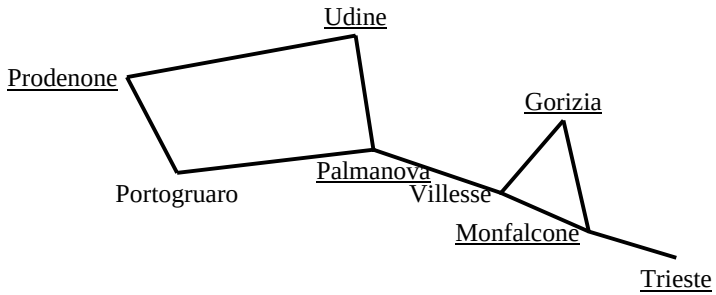
Problemi equivalenti

- percorsi hamiltoniani di lunghezza minima
  - fra due nodi qualunque:
    si aggiunge un nodo a distanza *0* dagli altri
  - fra due nodi dati:
    si fondono assieme i due estremi
- *m* commessi viaggiatori (*m–TSP*)
  - considerare un problema di percorso hamiltoniano minimo in cui vi sono *m* copie della città di partenza, con le identiche distanze dagli altri nodi ed a distanza infinita l'una dall'altra
- visitare ogni città almeno una volta
  - si considera la rete dei percorsi minimi e non quella stradale. I c_{ij} sono pari alla distanza minima tra *i* e *j*

Esempio

Problema: visitare le città principali del Friuli - Venezia Giulia a partire da Trieste e tornando a Trieste.

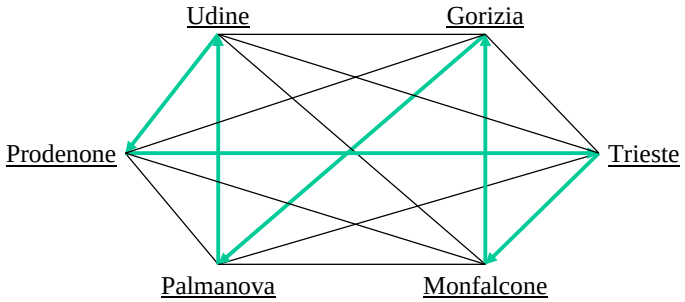
La rete stradale del Friuli-Venezia Giulia non ammette circuito hamiltoniano. Le città sottolineate sono quelle da visitare.



Esempio

Rete dei percorsi minimi

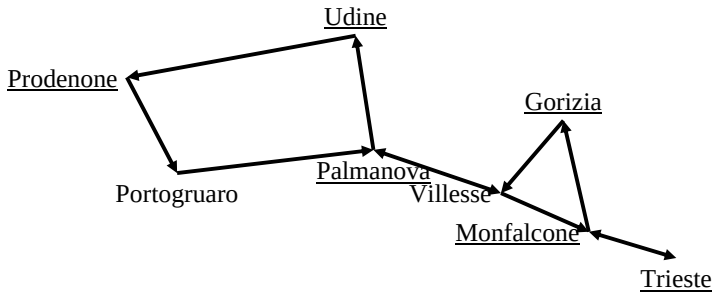
Si noti, ad esempio, l'arco diretto Pordenone – Trieste che non passa da Palmanova e Monfalcone



circuito hamiltoniano minimo

Esempio

Nel percorso di ritorno non ci si ferma a Palmanova e Monfalcone in quanto già visitate



Esempio

Commenti:

- Definire i costi dei percorsi tra le varie città può non essere banale.

  Ad esempio, tra Pordenone e Palmanova ci sono strade più brevi rispetto all'autostrada che passa per Portogruaro. Il loro tempo di percorrenza è però maggiore.

  Quanta benzina in più si è disposti a sacrificare per guadagnare tempo?
- Calcolare le distanze da ogni città di interesse a tutte le altre città di interesse a partire dalla rete stradale può essere abbastanza oneroso. La rete stradale infatti considera centinaia di nodi e l'algoritmo di Dijkstra nel cercare la distanza tra due nodi, *A* e *B*, visita tutti i nodi che sono più vicini a *A* di *B*.
- Calcolare il circuito hamiltoniano minimo può essere oneroso.

Complessità del *TSP*

il *TSP* è un problema difficile: *NP-hard:*
non sono noti *algoritmi efficaci* ed *efficienti*

esistono algoritmi efficienti ma non efficaci (euristiche)
gli algoritmi efficaci non sono polinomiali
il TSP rimane difficile anche se vale la disuguaglianza triangolare

Disuguaglianza triangolare

- **disuguaglianza triangolare**:
  il percorso diretto tra due nodi non è più costoso di qualunque indiretto:

$$c_{ij} \leq c_{ik} + c_{kj}$$

  Per i problemi logistici tale ipotesi è pressoché sempre verificata.

Nel seguito si assume sempre che
- tutti i costi siano non negativi
- valga la disuguaglianza triangolare (che i TSP siano *metrici*)
- le distanze siano simmetriche

Euristiche vs. Algoritmi esatti

Quando utilizzare euristiche invece che algoritmi esatti

- quando le dimensioni del problema sono eccessive rispetto a quelle risolvibili attraverso algoritmi esatti
- quando i problemi, anche se limitate dimensioni, devono essere risolti in tempi estremamente brevi
- quando i dati del problema sono approssimati e quindi non vale la pena cercare la soluzione esatta
- quando si risolvono problemi simili, ma non identici a quelli affrontati dagli algoritmi esatti. Questi ultimi sono molto meno generalizzabili delle euristiche
- quando si devono trattare problemi dinamici

Tipi di euristiche

costruttive
- costruiscono un circuito hamiltoniano

di miglioramento
- partono da un circuito hamiltoniano e ne producono uno migliore

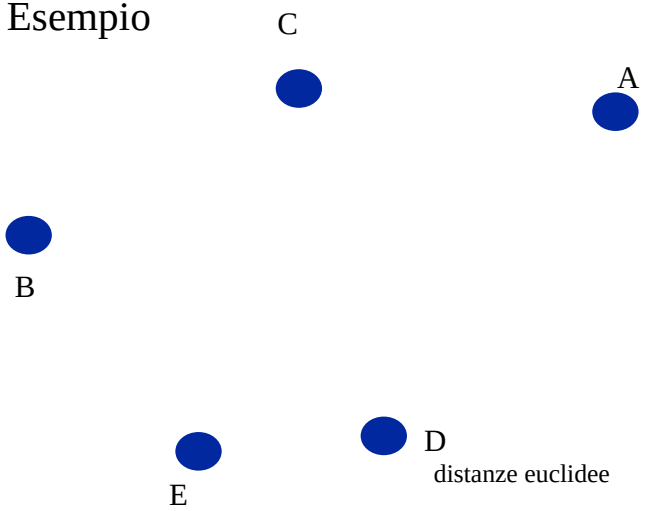
Nodo più prossimo
Nearest Neighbor

Algoritmo base

- parte da un nodo di riferimento (tipicamente il magazzino centrale oppure un estremo del arco meno costoso)
- passa al nodo più vicino

Variante: a doppia crescita

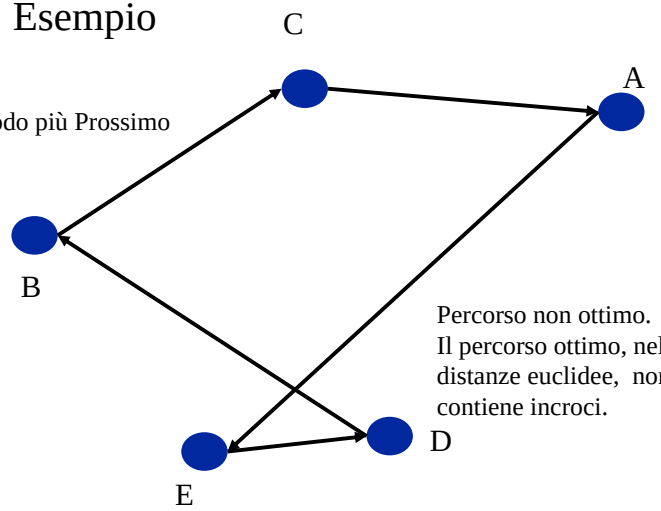
Esempio



distanze euclidee

Esempio

Euristica Nodo più Prossimo



Percorso non ottimo.
Il percorso ottimo, nel caso di distanze euclidee, non contiene incroci.

Nodo più prossimo
Nearest Neighbor

Commenti:

- non funziona molto bene:
  - i primi archi sono brevi, gli ultimi molto lunghi
- però gli archi molto lunghi sono pochi:
  - può essere una buona base da migliorare
- semplice da implementare
- errore di approssimazione $< \frac{1}{2}\, ceil(\log(n)) + \frac{1}{2}$
- soluzioni mediamente del 25% superiori al minimo effettivo
- complessità computazionale $O(n^2)$

Inserimento

Algoritmo base

si parte da un circuito parziale

si inseriscono nuovi nodi

Regole di inserimento (errore medio, errore massimo):

- nodo più vicino (20%, 200%)
- nodo più lontano (10%, *ceil(log(n))+1*)
- inserimento più economico (17%, 200%)
- inserimento casuale (11%, *ceil(log(n))+1*)

Esempio

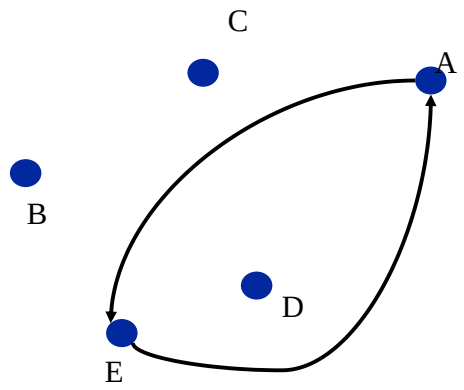
Matrice distanze

	A	B	C	D	E
A	0	85	47	57	87
B	85	0	43	52	38
C	47	43	0	48	58
D	57	52	48	0	32
E	87	38	58	32	0

Si noti la simmetria della matrice che implica che le distanze tra i nodi non dipendono dai versi di percorrenza.

Esempio

Euristica Inserimento
Nodo più Lontano



Passo 1)
I nodi a distanza massima sono A ed E. AE = 87.
Si forma un circuito parziale A-E-A da A ad E e ritorno da E a A.
A-E-A è lungo 87 + 87 = 174.

Esempio

Passo 2)

Si cerca il nodo più distante da entrambe i nodi A ed E, cioè il nodo per cui sia massima la massima distanza da A ed E.

Si confrontano quindi le distanze dei seguenti nodi:

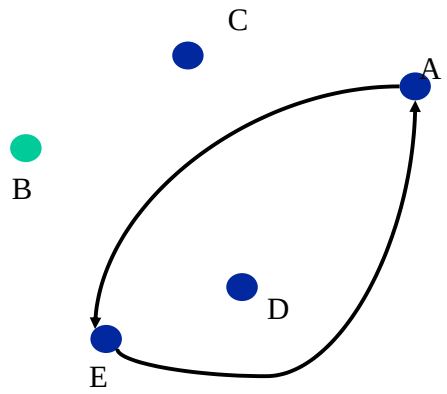
- B: BA = 85, BE = 38, distanza massima BA = 85
- C: CA = 47, CE = 58, distanza massima CA = 58
- D: DA = 57, DE = 32, distanza massima DE = 57

Il nodo più distante è B. Infatti, BA è il valore massimo tra 85, 57 e 57.

(continua)

Esempio

Euristica Inserimento
Nodo più Lontano



Esempio

(continuazione)

Si inserisce il nodo B nel circuito parziale.

Data la simmetria delle distanze e' indifferente dove avviene l'inserzione.

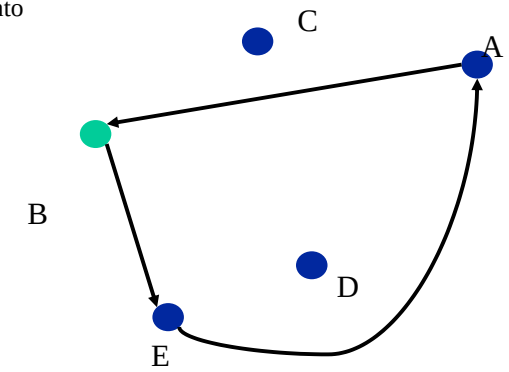
Si supponga di inserire B nel percorso di andata da A ad E.

In questo caso si risparmiano i 87 Km da A ad E, ma si pagano i 85 Km da A a B e gli 38 da B ad E. L'aumento complessivo e' di 85 + 38 - 87 = 36 Km.

Il nuovo percorso e' A-B-E-A lungo 85 + 38 + 87 = 210.

Esempio

Euristica Inserimento
Nodo più Lontano



Esempio

Passo 3)

Si cerca il nodo più distante dai nodi del circuito parziale corrente, cioè dai nodi A, B ed E.

In altre parole, si cerca il nodo *r* cui sia massima la massima distanza da A, da B e da E.

Si confrontano quindi le distanze dei seguenti nodi:

- C: CA = 47, CB = 43, CE = 58, distanza massima CB = 58
- D: DA = 57, DB = 52, DE = 32, distanza massima DE = 57

Il nodo più distante è C. CB è il valore massimo tra 58 e 57.

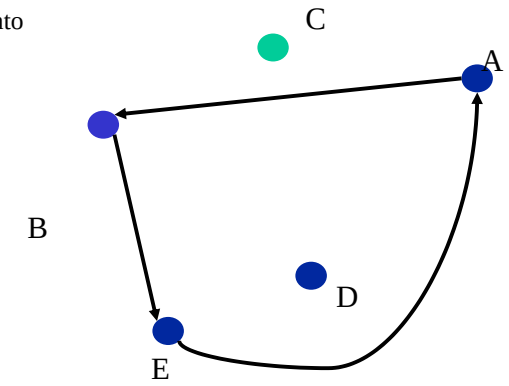
Si inserisce il nodo C nel circuito parziale.

Bisogna decidere se conviene inserire C tra A e B, oppure tra B ed E, oppure infine tra E e A.

(continua)

Esempio

Euristica Inserimento
Nodo più Lontano



Esempio

(continuazione)

Si fanno tutti i tentativi:

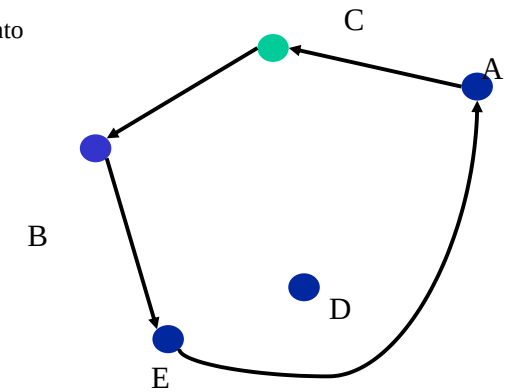
- C tra A e B: si risparmiano i 85 Km da A a B, ma si pagano i 47 Km da A a C e i 43 da C ad B. Incremento totale 47 + 43 - 85 = 5;
- C tra B e E: si risparmiano i 38 Km da B ad E, ma si pagano i 43 Km da B a C e i 57 da C ad E. Incremento totale 43 + 58 - 38 = 63;
- C tra E e A: si risparmiano i 87 Km da E a A, ma si pagano i 58 Km da E a C e i 47 da C ad A. Incremento totale 58 + 47 - 87 = 18.

Conviene quindi inserire C tra A e B in modo da incrementare al minimo la lunghezza del percorso parziale.

Il nuovo percorso parziale è A-C-B-E-A lungo 47 + 43 + 38 + 87 = 215.

Esempio

Euristica Inserimento
Nodo più Lontano



Esempio

Passo 4)

Non rimane che inserire D nel circuito parziale.

Bisogna decidere se conviene inserire D tra A e C, oppure tra C e B, oppure tra B ed E, oppure infine tra E ed A.

Si fanno tutti i tentativi:

- D tra A e C: si risparmiano i 47 Km da A a C, ma si pagano i 57 Km da A a D e gli 48 da D ad C.

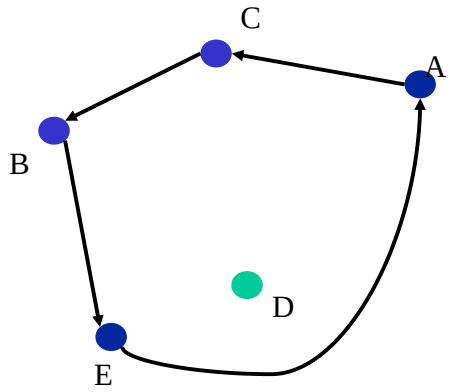
  Incremento totale 57 + 48 - 47 = 58;
- D tra C e B: si risparmiano i 43 Km da C a B, ma si pagano i 48 Km da C a D e gli 52 da D ad B.

  Incremento totale 48 + 52 - 54 = 46;

(continua)

Esempio

Euristica Inserimento
Nodo più Lontano



Esempio

(continuazione)

- D tra B e E: si risparmiano i 38 Km da B ad E, ma si pagano i 52 Km da B a D e i 32 da D ad E.

  Incremento totale 52 + 32 - 38 = 58;
- D tra E e A: si risparmiano i 87 Km da E a A, ma si pagano i 32 Km da E a D e i 57 da D ad A.

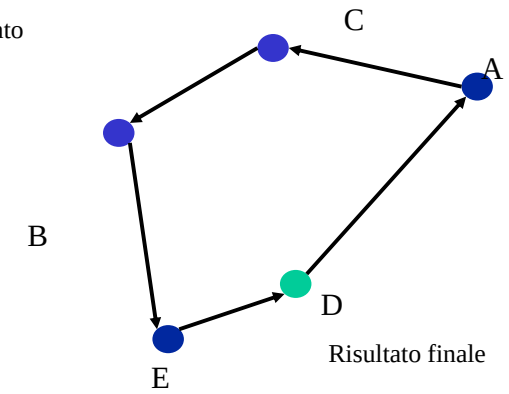
  Incremento totale 32 + 57 - 87 = 2.

  Conviene quindi inserire D tra E e A in modo da incrementare al minimo la lunghezza del percorso parziale.

  Il percorso finale e' A-C-B-E-D-A lungo 47 + 43 + 38 + 32 + 57 = 217.

Esempio

Euristica Inserimento
Nodo più Lontano



Risultato finale

Doppio albero ricoprente

Concetti base:

- **albero**

  una rete:

  con numero di rami pari al numero di nodi meno *1*

  senza circuiti

  connesso

  un **albero ricoprente minimo** è l'albero di lunghezza minima che tocca tutti i nodi. Sia *c(MST)* il costo di tale albero.

- ogni **percorso hamiltoniano** è un albero il cui costo è certamente non inferiore al costo dell'albero ricoprente minimo

Doppio albero ricoprente

- ogni **circuito hamiltoniano** è composto da un percorso hamiltoniano a cui è stato aggiunto un arco. Il costo del **circuito hamiltoniano minimo**, *c(TSP)* è certamente non inferiore al costo dell'albero ricoprente minimo: $c(MST) \leq c(TSP)$
- da un albero ricoprente minimo si può generare un circuito che visiti, eventualmente più volte, tutti i nodi. Basta percorrere tutti gli archi nei due sensi ritornano indietro solo quando non ci sono alternative. Un circuito così ottenuto ha un costo $2c(MST)$, il cui valore è certamente non inferiore a quello del circuito hamiltoniano minimo: $c(MST) \leq c(TSP) \leq 2c(MST)$.
- se vale la **diseguaglianza triangolare**, è possibile ridurre il circuito ottenuto al passo precedente ad un circuito hamiltoniano di costo non superiore $c(CH) \leq 2c(MST)$, basta seguire il circuito, saltando i nodi già visitati. Si ottiene un circuito per cui vale
  $c(MST) \leq c(TSP) \leq c(CH) \leq 2c(MST)$.

Doppio albero ricoprente

Algoritmo:

si parte da un albero ricoprente minimo,

si raddoppia gli archi per avere un giro completo,

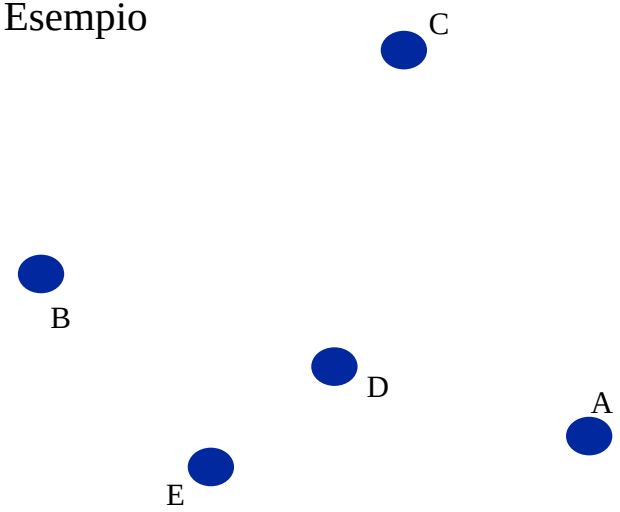
si ricava un circuito hamiltoniano.

Efficacia:

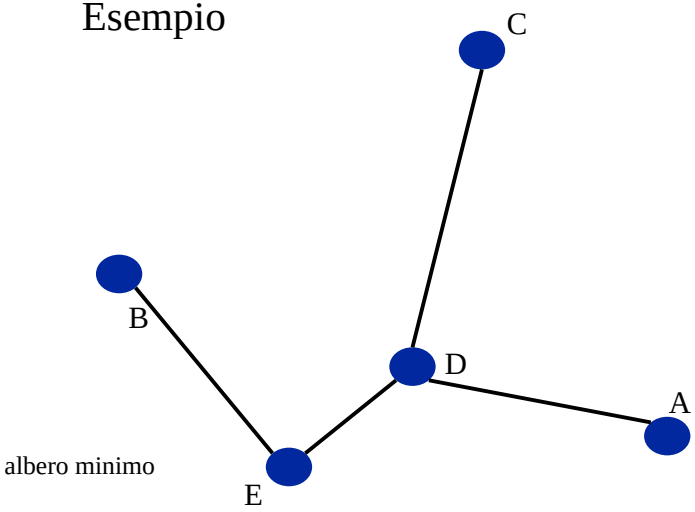
- al peggio 100% più del minimo

  infatti un albero ricoprente minimo non è più lungo del minimo circuito hamiltoniano,
- 38% in media.

Esempio



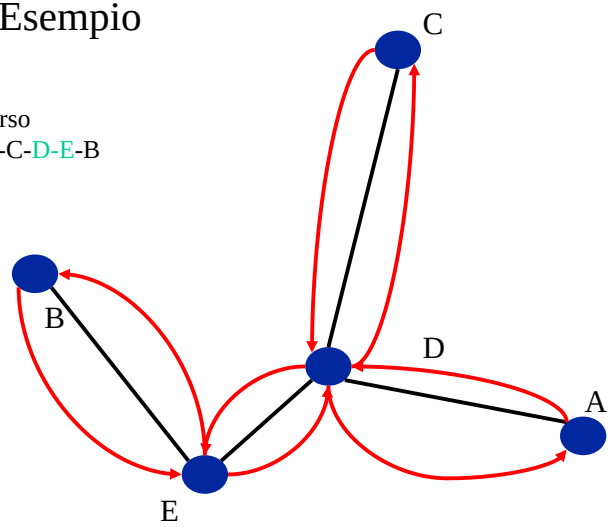
Esempio



albero minimo

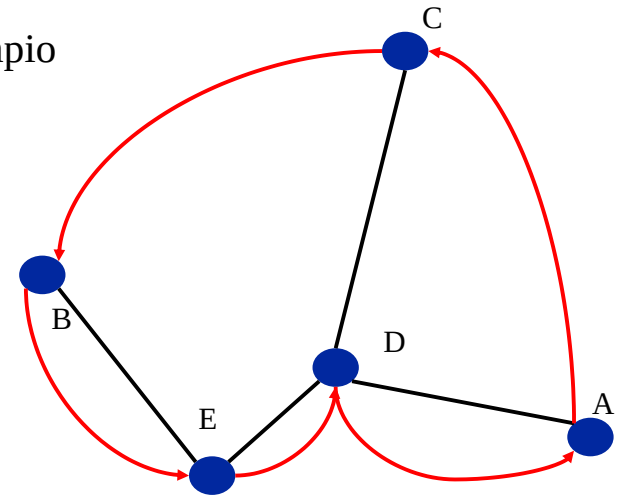
Esempio

primo percorso
B-E-D-A-D-C-D-E-B



Esempio

taglio dei nodi
già visitati,
percorso
B-E-D-A-C-B



NB: si giungeva ad un risultato peggiore partendo dal percorso B-E-D-C-D-A-D-E-B

Christofides

Concetti base:

- **ordine di un nodo**: numero degli archi incidenti.

Algoritmo:

si parte da un albero ricoprente minimo,

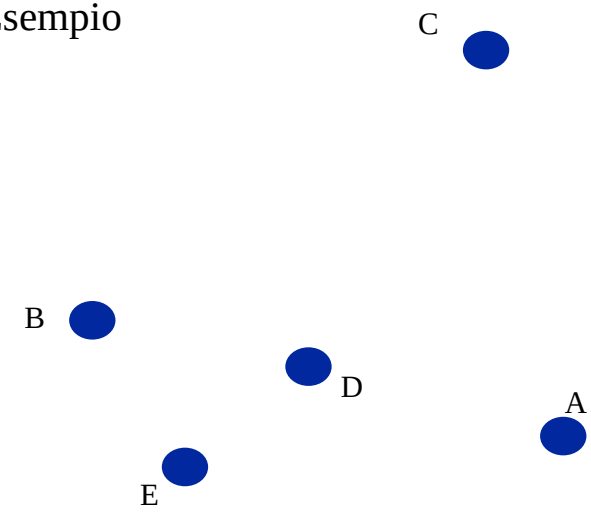
si connettono al meglio i nodi di ordine dispari (matching),

si ricava un circuito hamiltoniano.

Efficacia:

- 50% più del minimo nel peggiore dei casi. Questa è la migliore approssimazione per TSP metrici non euclidei.
- 10 – 20% in media.

Esempio

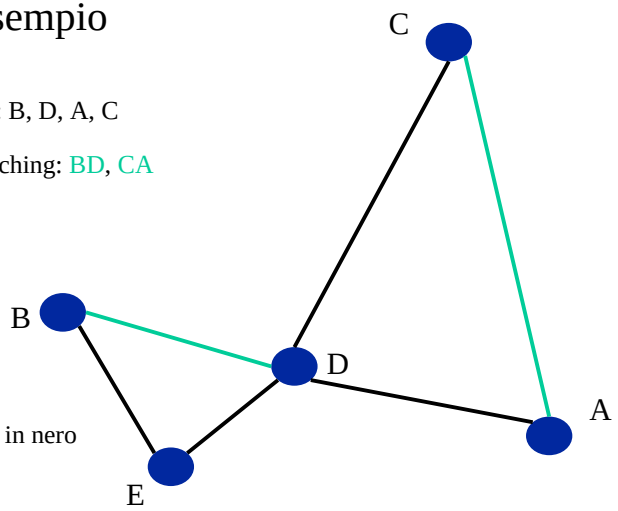


Esempio

nodi dispari: B, D, A, C

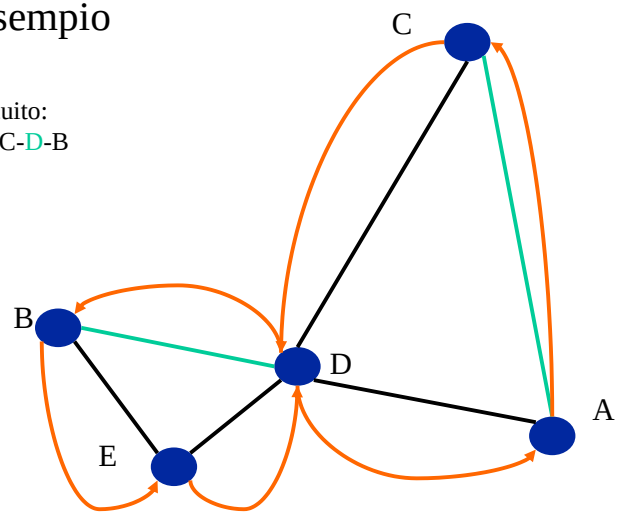
archi di matching: BD, CA

albero minimo in nero



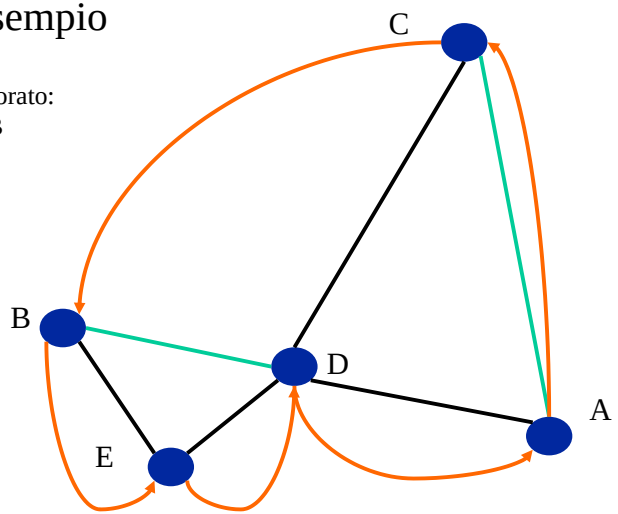
Esempio

primo circuito:
B-E-D-A-C-D-B



Esempio

circuito migliorato:
B-E-D-A-C-B



Risparmi

Algoritmo:

si parte con $n - 1$ circuiti parziali da un nodo arbitrario,

per ogni coppia di circuiti parziali si calcola quanto si risparmierebbe fondendoli assieme,

si fondono i due circuiti parziali più convenienti.

Efficacia: 10% più del minimo in media, ordine *log(n)* nel caso peggiore.

Tipi di euristiche

costruttive
- costruiscono un circuito hamiltoniano

di miglioramento
- partono da un circuito hamiltoniano e ne producono uno migliore

2 – OPT

Concetto base:

scambio 2–opt tra due rami di un circuito:
- si eliminano due rami,
- si ricongiungono i due semicircuiti.

Algoritmo:

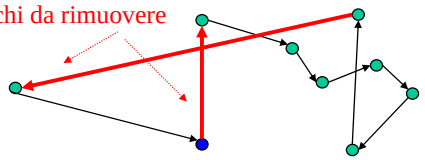
si parte da un circuito hamiltoniano,

si eseguono tutti gli scambi 2–opt (fra tutte le coppie di rami del circuito) che riducono la lunghezza.

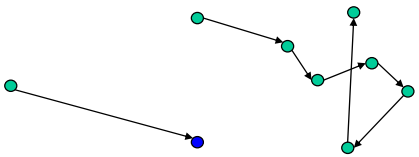
Efficacia: 8% più del minimo in media

2-opt

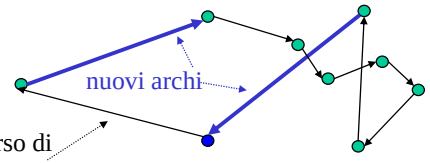
Circuito iniziale



Semicircuiti

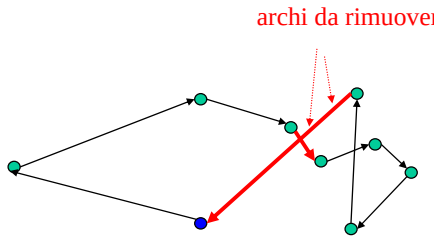


Circuito passo 1

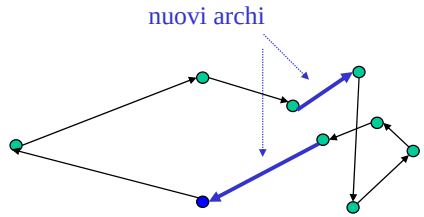


notare l'inversione del verso di percorrenza di un semicircuito

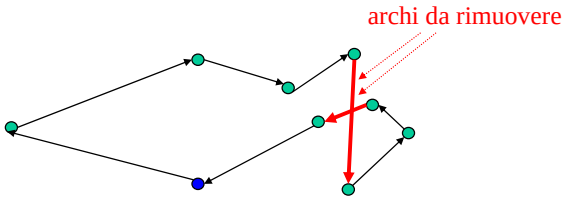
Circuito passo 1



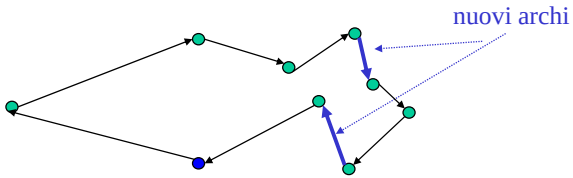
Circuito passo 2



Circuito passo 2



Circuito passo 3



ecc..

3 – OPT

Concetto base:

scambio 3–opt tra tre rami di un circuito:

- si eliminano tre rami,
- si ricongiungono i tre sottocircuiti.

Algoritmo:

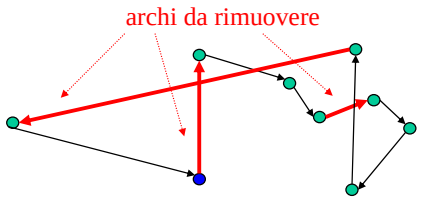
si parte da un circuito hamiltoniano,

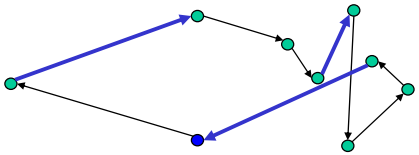
si eseguono tutti gli scambi 3–opt (fra tutte le terne di rami del circuito) che riducono la lunghezza.

Efficacia: 4% più del minimo in media

3 – OPT

Circuito iniziale





Nuovo circuito

Lin – Kernighan

Concetto base:

- *k* – opt con *k* variabile.

Algoritmo:

si effettua *k* – opt,

si valuta se effettuare *(k+1)* – opt.

Efficacia: 1% in più dell'ottimo in media.

Algoritmi esatti

- Basati su modelli MIP risolti con il branch and cut
- Difficile esprimere in modo efficiente che deve esserci un unico grande circuito (*tour*). Vengono utilizzati dei vincoli che sarebbero esponenziali in numero e che quindi vengono aggiunti dinamicamente alla luce delle soluzioni ottenute.

Formulazione matematica

Formulazione del problema.

- Le variabili:
  variabili binarie

$$x_{ij} = \begin{cases} 1 & \text{se arco } (i, j) \text{ appartiene al tour} \\ 0 & \text{altrimenti} \end{cases}$$

- La funzione obiettivo:
  il costo atteso

$$\Sigma_{(i,j)\in E} c_{ij}\, x_{ij}$$

Formulazione matematica

- I vincoli:
  - tutti i nodi devono avere due archi incidenti

$$\Sigma_{(i,j)\in\delta(i)} x_{ij}=2, \qquad \forall i$$

  - non devono esserci *subtour*

$$\Sigma_{(i,j)\in\delta(W)} x_{ij} \geq 2, \qquad \forall W\subset N : 3 \leq |W| \leq |N|/2$$

  oppure

$$\Sigma_{(i,j):\, i,j\in W} x_{ij} \leq |W| - 1, \qquad \forall W\subset N: \quad 2\leq|W|\leq|V|-2,$$

  - un arco appartiene o non appartiene al tour

$$0 \leq x_{ij} \leq 1 \quad x_{ij} \in \{0,1\}, \quad \forall (i,j)\in E$$

Formulazione matematica

Commenti

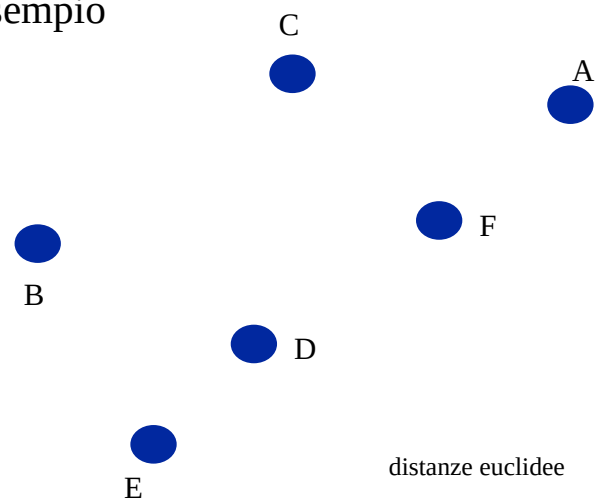
- i tagli di eliminazione dei *subtour* sono in numero esponenziale. Vengono quindi aggiunti in modo dinamico
- il primo tipo di taglio di eliminazione dei *subtour* impone che da ogni subset di nodi escano almeno due archi. $\delta(W)$ è l'insieme degli archi uscenti dal set W
- il secondo tipo di taglio di eliminazione dei *subtour* impone che gli archi selezionati contenuti in ogni subset di nodi formino un percorso. Su ogni nodo devono infatti incidere due di essi e sono in numero tale da potere formare solo un albero
- con tali formulazioni si risolvono problemi fino a 100 nodi in pochi minuti di CPU (2001)
- per problemi basati su distanze euclidee conviene utilizzare algoritmi ad hoc

Formulazione matematica

Commenti

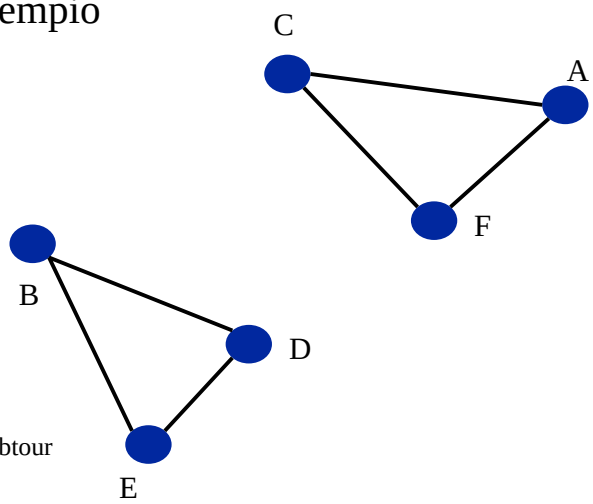
- problemi di dimensione maggiore vengono risolti aggiungendo dei tagli che descrivono delle faccette del poliedro del *TSP* e si stimano i lower bound con rilassamenti meno elementari del rilassamento continuo.
- il miglior sw per il TSP è forse *Concorde*. Risolve TSP con centinaia di nodi in pochi minuti o meno. Risolve TSP con un migliaio di nodi in un ora o poco più (2001).
- i problemi di massima dimensione risolti in modo esatto sono dell'ordine di 10-15.000 nodi.

Esempio



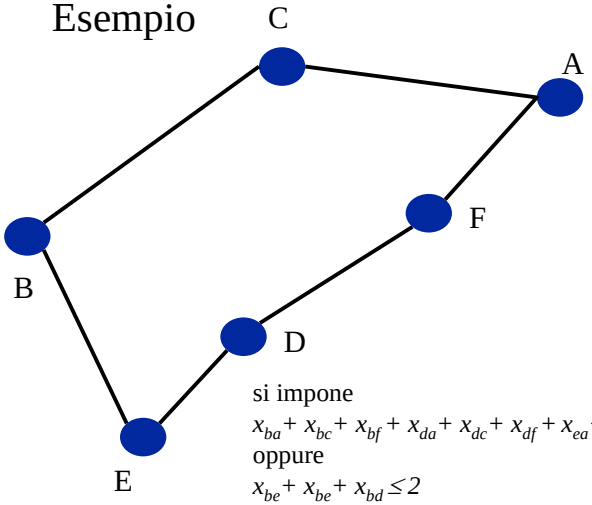
distanze euclidee

Esempio



nessun vincolo di eliminazione di subtour

Esempio



si impone

$x_{ba} + x_{bc} + x_{bf} + x_{da} + x_{dc} + x_{df} + x_{ea} + x_{ec} + x_{ef} \geq 2$

oppure

$x_{be} + x_{be} + x_{bd} \leq 2$

Problemi di piccola dimensione

Piccole istanze di TSP: fino a 100 nodi.

- si genera un circuito con un metodo euristico, si applicano ripetutamente le procedure k-opt fino ad ottenere upper bound, probabilmente molto buono
- Tale upper bound è poi utilizzato nelle procedure di enumerazione per tagliare ad alto livello le soluzioni rilassate non promettenti

TSP asimmetrico

- Non esistono euristiche a prestazione garantita limitata da una costante
- L'euristica dei risparmi è quella che sembra funzionare meglio
- La formulazione matematica esatta è simile al caso simmetrico, ma per ogni nodo si deve imporre che ci sia almeno un arco entrante e un arco uscente. Come lower bound si può usare quelli TSP simmetrico nel caso di distanze quasi simmetriche, mentre conviene usare il problema di assegnazione per distanze molto asimmetriche. Questo ultimo metodo da lower bound scadenti (c.a. 60%) nel caso di distanze simmetriche.

Pianificazione dei percorsi

Problema:

pianificare l'uso efficiente di una ''flotta'' di veicoli con capacità limitata che devono visitare un certo numero di clienti per prelevare o consegnare della merce

(Capacitated) Vehicle Routing Problem – VRP

Si tratta di decidere quali clienti deve visitare ciascun veicolo, ed in che ordine, in modo da ridurre al minimo i costi.

VRP

Vehicle Routing Problem *VRP(G,c,K,I,a,b)*

Istanza: una rete connessa $G=(N,A)$. Un costo (distanza) c_{ij} associato ad ogni arco $(i,j)\in A$. Una flotta di K veicoli identici di capacità b. Un insieme V di clienti, disposti su un sottoinsieme di nodi, caratterizzati da una domanda a_v. Un deposito localizzato sul nodo 0 della rete.

Soluzione: un insieme di K circuiti, contenenti il nodo 0, che coprano tutti i clienti e tali che la somma delle domande associate ad ogni circuito non ecceda b. Ciascun cliente deve essere assegnato ad un unico giro

Obiettivo: minimizzare il costo (la lunghezza) dei k circuiti.

Il problema di VRP ha caratteristiche che lo legano a

- m-TSP
- bin packing

Algoritmi esatti vs. Euristiche

- Gli algoritmi esatti per il VRP si basano su modelli che generalizzano quelli del TSP
- Vengono risolte con algoritmi esatti solo istanze piccole (100 clienti e 5 veicoli)
- In pratica si utilizzano quasi sempre euristiche.

- Studiare gli algoritmi esatti è però sensato perché:
  - si cerca di sviluppare algoritmi che possano affrontare istanze di maggiori dimensioni
  - si acquisisce conoscenza sulla natura del problema. Questa conoscenza può poi essere applicata per sviluppare euristiche più efficaci
  - si possono valutare esattamente gli errori delle euristiche e dei rilassamenti, almeno per problemi di piccola dimensione

Euristiche

Euristiche
- costruttive
  - sequenziali: si costruisce un viaggio alla volta
  - parallele: si costruiscono più viaggi in parallelo
- di miglioramento
- metodi a due fasi
  - *cluster first – route second*
  - *route first – cluster second*

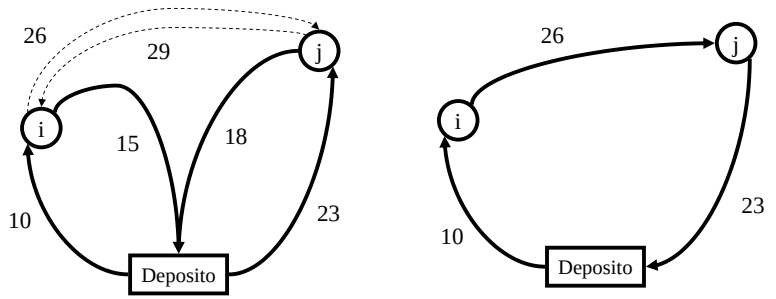
Risparmi o di Clark-Wright per il *VRP*

Algoritmo RisparmiSequenziale *(G)*

1. Inizializzazione
   for ogni nodo *i* genera un circuito parziale tra *0* e *i*
   definisci circuito corrente il circuito che unisce *0* con *1*
   calcola i risparmi ottenuti unendo tutti i possibili circuiti e ordinali
2. Iterazione
   if esiste un risparmio maggiore di zero a partire del maggiore
   then unisci i due circuiti ed itera passo 2
   else if esiste circuito parziale non ancora considerato
   then seleziona un nuovo circuito come circuito corrente ed itera passo 2
   else stop

Ad ogni iterazione bisogna verificare se è possibile unire i due circuiti rispettando i vincoli di capacità e di percorrenza dei mezzi.

Risparmi o di Clark-Wright per il *VRP*



$s_{ij}=c_{0i}+c_{i0}+c_{0j}+c_{j0} - (c_{0i}+c_{ij}+c_{j0}) = c_{i0}+c_{0j} – c_{ij} = 12$
data l'asimmetria si deve valutare anche $s_{ji}.= -1$
Poiché $s_{ij} > s_{ji}$ e $s_{ij} > 0$ conviene fondere i circuiti come in figura

Risparmi o di Clark-Wright per il *VRP*

Algoritmo RisparmiParallelo

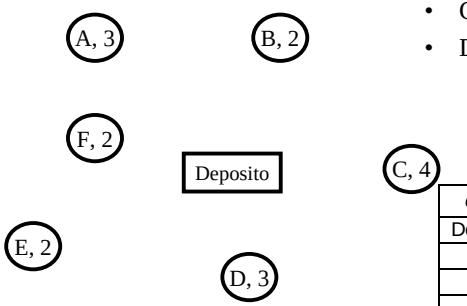
1. Inizializzazione
   for ogni nodo *i* genera un circuito parziale tra *0* e *i*
   calcola i risparmi ottenuti unendo tutti i possibili circuiti e ordinali
2. Iterazione
   if esiste un risparmio maggiore di zero a partire del maggiore
   then unisci i due circuiti ed itera
   else stop

Ad ogni iterazione bisogna verificare se è possibile unire i due circuiti rispettando i vincoli di capacità e di percorrenza dei mezzi.
L'algoritmo parallelo è il più usato.

Risparmi o di Clark-Wright per il *VRP*

- L'euristica dei risparmi non permette il controllo sul numero dei circuiti generati. Possono venire generati più circuiti dei mezzi disponibili.
- E' buona norma risolvere prima un problema di binpacking per verificare a priori se il problema di *VRP* ammette soluzioni dal punto di vista delle capacità dei veicoli.
- L'euristica dei risparmi può venire parametrizzata introducendo un valore α con cui moltiplicare l'addendo c_{ij} considerato in s_{ij}. In questo modo al variare di α si generano soluzioni differenti. L'utente finale sceglierà poi la migliore tra queste.

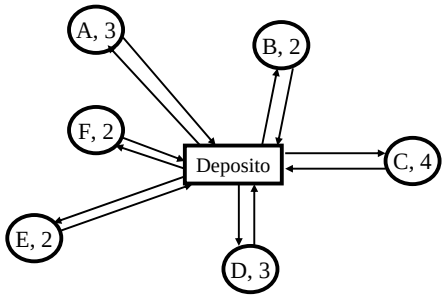
Esempio



- Distanze euclidee e simmetriche
- Capacità veicoli *K=8*
- Domanda clienti nei nodi dei clienti

c_{ij}	Dep	A	B	C	D	E	F
Dep	0	64	58	54	41	58	41
A	64	0	70	95	103	81	40
B	58	70	0	36	92	113	81
C	54	95	36	0	72	112	91
D	41	103	92	72	0	61	71
E	58	81	113	112	61	0	41
F	41	40	81	91	71	41	0

Esempio

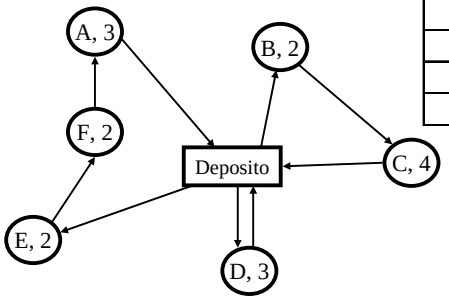


Distanze
- C(0-A-0) = 128
- C(0-B-0) = 116
- C(0-C-0) = 108
- C(0-D-0) = 82
- C(0-E-0) = 116
- C(0-F-0) = 82

Distanza totale = 632
Numero veicoli = 6

Esempio

s_{ij}	A	B	C	D	E	F
A	0	52	23	2	41	65
B	52	0	76	7	3	18
C	23	76	0	23	0	4
D	2	7	23	0	38	11
E	41	3	0	38	0	58
F	65	18	4	11	58	82

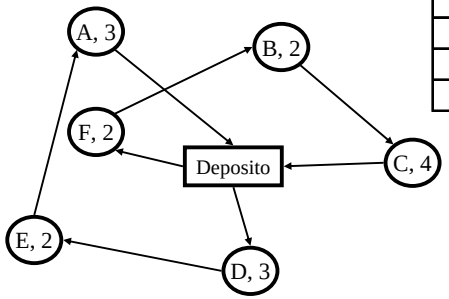


Ulteriori fusioni di circuiti non sono possibili senza violare i vincoli di capacità.

Distanza totale = 433

Esempio

s_{ij}	A	B	C	D	E	F
A	0	52	23	2	41	65
B	52	0	76	7	3	18
C	23	76	0	23	0	4
D	2	7	23	0	38	11
E	41	3	0	38	0	58
F	65	18	4	11	58	82



L'algoritmo non fornisce però una soluzione che utilizza solo due veicoli

Distanza totale = 459

Numero veicoli = 2

Cluster first, route second

Algoritmo ClusterFirstRouteSecond

1. Clustering
   risolvi un problema di assegnamento generalizzato assegnando ogni cliente ad un veicolo
2. Routing
   determina il percorso ottimo di ogni veicolo

Formulazione matematica - clustering

Formulazione del problema.

- Le variabili:
  variabili binarie

$$x_{ik} = \begin{cases} 1 & \textit{se cliente } i \textit{ assegnato veicolo } k \\ 0 & \textit{altrimenti} \end{cases}$$

- La funzione obiettivo:
  il costo atteso

$$\Sigma_k \Sigma_i f(x_{ik})$$

dove $f(x_{ik})$ il costo di inserimento del cliente i nel viaggio k

Formulazione matematica - clustering

- I vincoli:
  - ogni cliente deve essere assegnato

$$\sum_k x_{ik}=1, \quad \forall i$$

  - non deve essere violata la capacità dei veicoli

$$\sum_i a_i x_{ik} \le b \ , \ \forall k$$

  - un cliente è o non è assegnato ad una dato veicolo

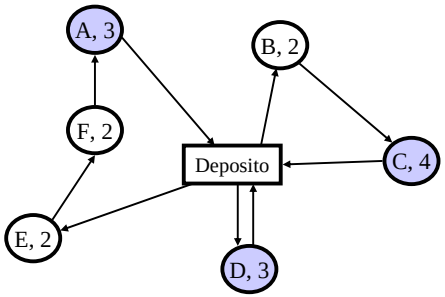
$$0 \le x_{ik} \le 1 \quad x_{ik} \in \{0,1\}, \quad \forall i,k$$

Formulazione matematica - clustering

Commenti:

- è difficile definire la funzione $f(x_{ik})$, quindi ...
  - si generano k percorsi o k cluster con altri algoritmi
  - per ogni cluster k si sceglie il cliente $j(k)$ più distante (*seed*), eventualmente in modo pesato per $a_{j(k)}$
  - approssima $f(x_{ik})$ con $c_{ij(k)} x_{ik}$ dove $c_{ij(k)}$ è il costo di inserzione di i nel percorso *0-j(k)-0*
- le prestazioni di questa euristica decadono se vi sono vincoli diversi dalla capacità. Questi altri vincoli, e.g., sul percorso massimo da eseguire potrebbero essere anche difficili da esprimere nella fase di clustering

Esempio



Tre possibili *seed*

$c_{ij(k)}$	A	C	D
A	0	105	126
B	64	40	109
C	85	0	85
D	80	59	0
E	75	116	78
F	17	78	71

In questo caso la assegnazione generalizzata produce gli stessi tre cluster

Altre forme di clusterizzazione

Sweep

Modified Region Partitioning Scheme

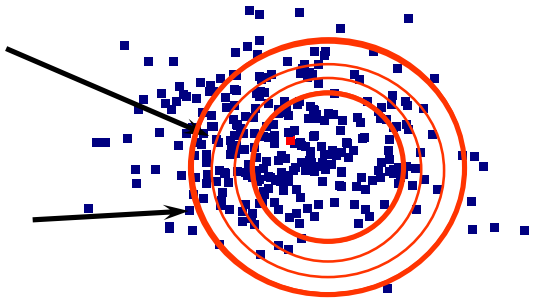
Sweep

Algoritmo Sweep - Metodo dell'orologio :

- si immaginano i clienti disposti sul quadrante di un orologio
- prima i clienti vengono assegnati ai veicoli in base all'ordine in cui li incontrerebbero le lancette dell'orologio
- poi si determinano i percorsi di ciascun veicolo risolvendo un *TSP*

Modified Region Partitioning Scheme *(MRPS)*

1. tante corone circolari quanti sono i mezzi in base alla capacità, più grandi più lontano (nel disegno solo alcune)

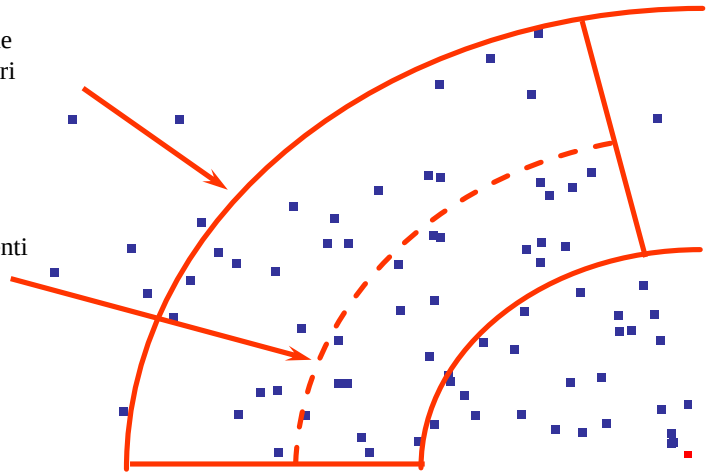


2. unione delle corone circolari con lo stesso numero di clienti. Si assume che tutti i clienti abbiano la stessa domanda

MRPS

3. partizione corone circolari in settori

4. nuovi tagli in insiemi di *q* clienti



Teorema 1 sul TSP

Se vale la disuguaglianza triangolare

costo circuito hamiltoniano ottimo per una regione più magazzino

costo circuito hamiltoniano ottimo clienti di una regione

$$Z \geq \max\{ L^*(N_j), \frac{2}{|N_j|}\sum_{i \in N_j} d_i \}$$

$$Z \leq \min_{i \in N_j} 2d_i + L^*(N_j) \leq \frac{2}{|N_j|}\sum_{i \in N_j} d_i + L^*(N_j)$$

distanza da magazzino cliente i-mo

Teorema 2 sul TSP

Se l'insieme dei clienti è partizionato in *R* regioni contenenti *q* clienti ciascuna (al più tranne la prima che ne può contenere meno)

somma circuiti

$$Z^{RP} \geq \frac{2}{|q|}\sum_{i \in N} d_i$$

compensa prima regione

$$Z^{RP} \leq \frac{2}{|q|}\sum_{i \in N} d_i + 2d_{\max} + \sum_{j \in R} L^*(N_j)$$

Teorema 3 sul TSP

Se le distanze sono euclidee, qualunque sia la politica di partizione

$$\sum_{j \in R} L^*(N_j) \leq L^*(N) + 1.5P^{RP}$$

la lunghezza dei perimetri delle regioni

Teorema 4 sul TSP

Se le distanze sono euclidee e i clienti sono distribuiti i.i.d. su una regione compatta del piano

costante funzione della distribuzione

$$\lim_{|N| \to \infty} \frac{L^*(N)}{\sqrt{|N|}} = \beta(a.s.)$$

Corollario

Nel caso di distanze euclidee, per ogni politica di partizione t.c.

$$\lim_{|N| \to \infty} \frac{P^{RP}}{N} = 0$$

allora

distanza media clienti

$$\lim_{|N| \to \infty} Z^{RP} = \frac{2|N|}{q}\bar{d} \quad (a.s.)$$

regioni

Commenti

MRPS

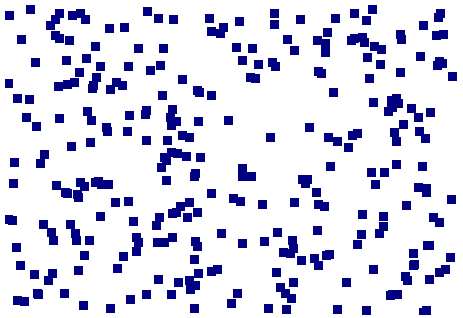
- soddisfa le condizioni su P^{RP}
- mantiene caratteristiche di non decrescenza e concavità del costo medio
- Anily e Federgruen provano l'asintotica ottimalità

Risultati

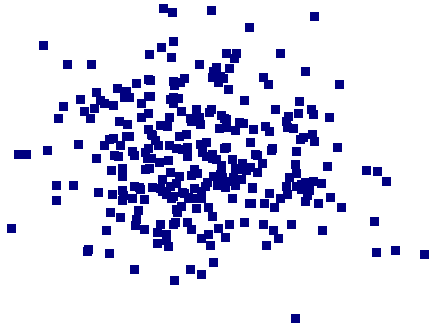
- In media errore minore del
  - 14% per 100 clienti distribuiti uniformemente
  - 5% nel caso di 500 clienti
- Politica di facile comprensione e implementazione (piace ai manager)

Commenti

- su una regione limitata, per $N \to \infty$ e ogni partizione "ragionevole" il corollario è ovvio :
  - i clienti tendono a sovrapporsi
  - la distanza reciproca dei q clienti di ogni regione tende a zero
  - il costo del circuito hamiltoniano tende ad essere quello del raggiungimento del primo cliente
- le prestazioni si degradano significativamente per
  - distribuzioni dei clienti non uniformi
  - numero di clienti limitato



distribuzione uniforme 300 clienti. OK!!



distribuzione normale 300 clienti. Mah??

Che fare?

L'approccio ha dei limiti però...

suggerisce delle **regole pratiche** (alcune ovvie)

Regola 1:

Quando gli altri fattori lo consentono conviene utilizzare pochi grandi automezzi e far visitare da ognuno di essi il maggiore numero possibile di clienti.

Regole pratiche

Regola 2

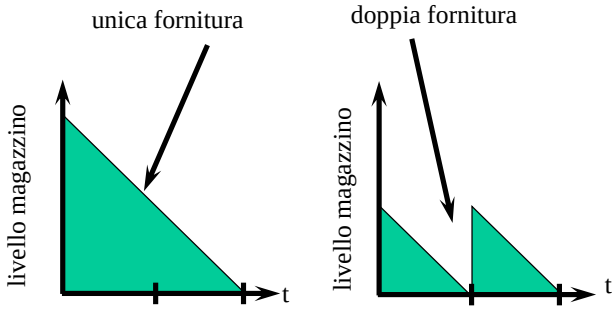
- l'approccio MRPS funziona bene quando i clienti sono numerosi (~100+) e distribuiti su regioni in cui siano facili i collegamenti diretti tra cliente e cliente
  - zone geografiche vaste
  - zone con caratteristiche uniformi (città o regioni pianeggianti)
- ma a posteriori:
  - bisogna verificare il rispetto di eventuali vincoli aggiuntivi
  - introdurre correzioni nel caso di evidenti errori di stima dei clienti da assegnare ad ogni mezzo q_i (il valore iniziale è dettato dalla capacità del mezzo)
- eventualmente iterare

Regole pratiche

Regola 3:

Quando si hanno clienti con notevoli differenze nel tasso di domanda è meglio cercare di assegnare i clienti maggiori a più di un giro di consegna, sfasare quanto più possibile i tempi di consegna e conseguentemente stabilire la merce da consegnare in modo da ridurre i livelli di scorta dei clienti.

Costi conservazione



i costi di conservazione sono quadratici

Generalizzazioni

- Clienti con tasso di domanda diversi
  - i clienti sono suddivisi in sottoclienti uguali
  - approccio con forti limiti
- Clienti con costi di conservazione diversi
  - i clienti vengono partizionati in base al rapporto tra costi e distanze, si ottengono quindi dei sottoproblemi indipendenti
- Costi di conservazione anche per magazzino centrale
  - la politica MRPS è subottima asintoticamente (6%)

Route first, cluster second

Algoritmo RouteFirstClusterSecond

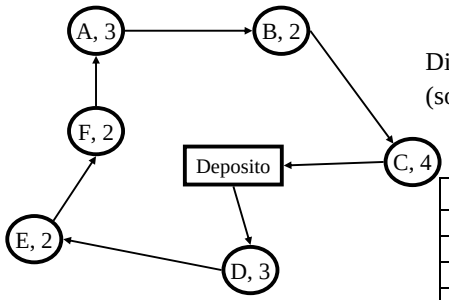
1. Routing.
   risolvi un *TSP* tra tutti i clienti trascurando le capacità
2. Clustering
   ripartisci il circuito tra i veicoli considerandone le capacità

Commenti:

- se la domanda dei clienti può essere suddivisa tra diversi veicoli l'algoritmo sfrutta al meglio la capacità dei veicoli stessi.

Esempio

tour iniziale

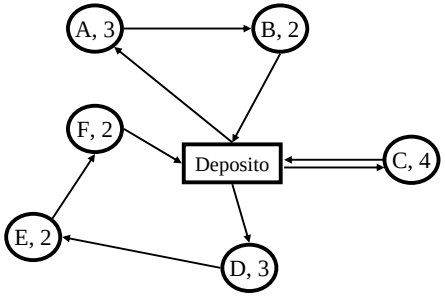


- Distanze euclidee e simmetriche
- Capacità veicoli *K=8*
- Domanda clienti nei nodi dei clienti

Distanza totale = 343
(soluzione inammissibile)

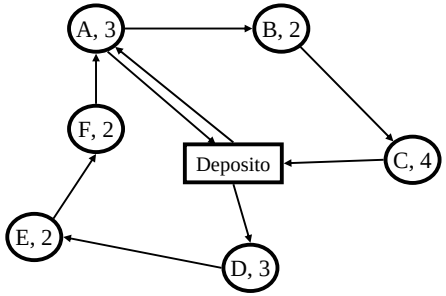
c_{ij}	Dep	A	B	C	D	E	F
Dep	0	64	58	54	41	58	41
A	64	0	70	95	103	81	40
B	58	70	0	36	92	113	81
C	54	95	36	0	72	112	91
D	41	103	92	72	0	61	71
E	58	81	113	112	61	0	41
F	41	40	81	91	71	41	0

Esempio



Domanda non frazionabile
Distanza toltale = 484
Numero veicoli = 3

Esempio



Domanda frazionabile
Distanza toltale = 525
Numero veicoli = 2
Clienti serviti da più veicoli = 1 (A)

Esempio applicativo

- il problema degli alberghi di Trieste

Euristiche di miglioramento

Algoritmi:

intrapercorso
(migliorano un percorso, mantenendo fissati i clienti appartenenti ad esso)
k-opt
OR-opt

interpercorso
(scambiano i clienti tra percorsi)

OR – OPT

Concetto base:

- si sposta un blocco di clienti all'interno della sequenza delle visite
- caso particolare del 3-opt

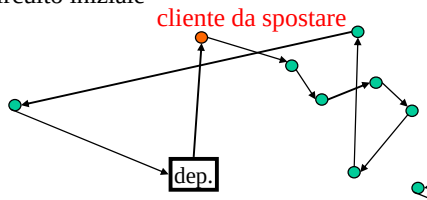
Algoritmo:

si parte da un circuito hamiltoniano e si individua un cliente o una sequenza di clienti

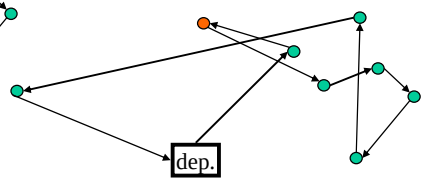
si prova ad anticipare o posticipare la visita dei clienti della sequenza selezionata in tutte le posizioni del circuito

OR – OPT

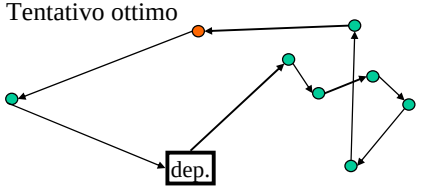
Circuito iniziale



Tentativo intermedio



Tentativo ottimo



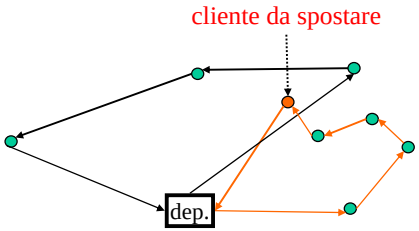
Euristiche interpercorso

Concetti base alternativi:

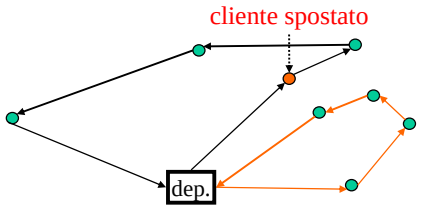
- si sposta un cliente da un circuito all'altro
- si scambiano coppie o due gruppi di clienti tra due circuiti

Euristiche interpercorso

Circuiti iniziali



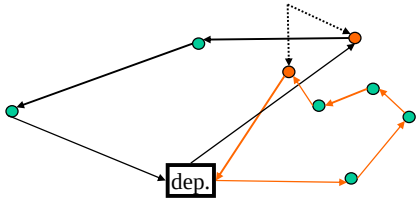
Spostamento di un cliente



Euristiche interpercorso

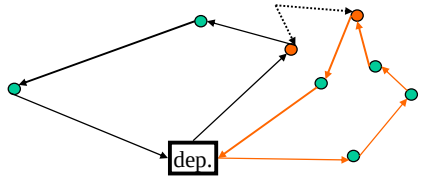
cliente da scambiare

Circuiti iniziali



cliente scambiati

Spostamento di un cliente



Altri problemi in letteratura

- **pick up and delivery**: in alcuni punti la merce viene scaricata in altri viene caricata
- **dial a ride**: versione dinamica del pick up and delivery
- **park and loop**: i percorsi vengono svolti con due mezzi. Il maggiore viene periodicamente parcheggiato e da tale punto partono i giri del minore

Routing su archi

- Chinese Postman Problem
- Rural Postman Problem
- Windy Postman Problem
- Capacitated Postman Problem

Reti euleriane

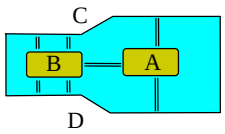
Rete Euleriana *EC(G,c)*

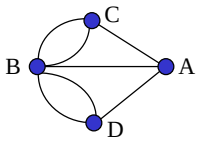
Istanza: una rete connessa *G =(N,A)*

Soluzione: un circuito euleriano, i.e., un circuito che passi una sola volta attraverso ogni arco in *A*.

Il problema dei ponti di Königsberg (Eulero, 1736)

La città vecchia di Königsberg ha sette ponti come indicato in figura, è possibile partire da un punto, attraversarli tutti una sola volta ritornando al punto di partenza? Quale è il percorso chiuso più breve che attraversa tutti i ponti?





Reti euleriane

- Condizioni necessarie e sufficienti di esistenza di un circuito euleriano:
  - rete non orientata: connessione e parità, ogni nodo deve avere un numero pari di archi incidenti;
  - rete orientata: forte connessione e simmetria, ogni nodo deve avere un numero uguale di entranti e uscenti;
  - rete mista: forte connessione, parità e bilanciamento*, ogni nodo deve avere un numero pari di archi orientati e non incidenti, la differenza tra il numero degli archi orientati che attraversano un qualunque taglio non deve eccedere il numero di archi non orientati.

* una rete pari e simmetrica è bilanciata, non è vero il contrario.

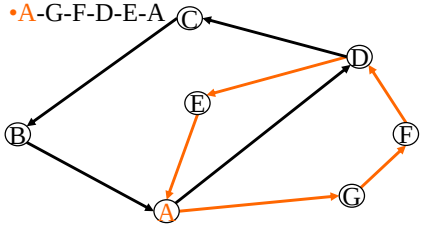
Reti euleriane

Algoritmo *End-Pairing*:

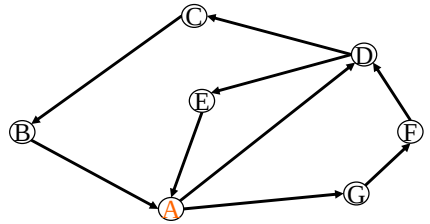
copri gli archi della rete con un insieme di cicli C che non hanno archi in comune

finché $|C| > 1$ cerca coppia di circuiti che abbia un nodo in comune e fondili

circuiti
•A-D-C-B-A
•A-G-F-D-E-A



circuiti fusi nel nodo A
•A-D-C-B-G-F-D-E-A



Reti euleriane

Algoritmo *Fleury*:

seleziona un arco

scegli casualmente un arco adiacente all'ultimo selezionato a condizione che non sia un *bridge*, i.e., la cui eliminazione non disconnetta la rete composta dai soli archi non ancora selezionati

Commento:

sia l'algoritmo *End-Pairing* che quello di *Fleury* si generalizzano banalmente al caso di rete orientata

Chinese Postman Problem

Chinese Postman Problem *CPP(G,c)*

Istanza: una rete connessa $G=(N,A)$, un costo (distanza) c_{ij} associato ad ogni arco $(i,j) \in A$.

Soluzione: un circuito che passa attraverso ogni arco in A almeno una volta.

Obiettivo: minimizzare il costo (la lunghezza) del circuito.

Chinese Postman Problem

Algoritmo (rete euleriana)

- determina un circuito euleriano con l'algoritmo *End-Paring* o di *Flaury*
- costo complessivo è $\Sigma_{(i,j)\in A} c_{ij}$

Chinese Postman Problem

Algoritmo (rete non euleriana)

- rendi la rete euleriana sdoppiando un opportuno sottoinsieme degli archi
- determina un circuito euleriano con l'algoritmo *End-Paring* o di *Flaury*
- costo complessivo è $\Sigma_{(i,j)\in A} c_{ij}$ + *costo archi sdoppiati*

Rendere la euleriana la rete

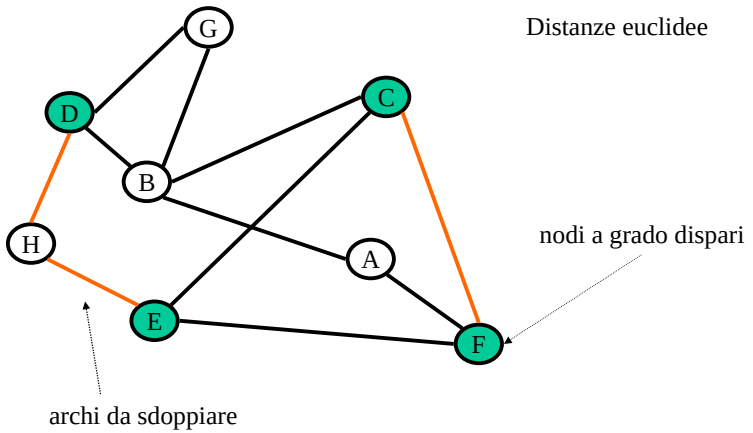
- è facile se la rete è diretta o indiretta,
- è NP-hard se rete mista

Chinese Postman Problem

Algoritmo *RenedereReteEulerianaReteIndiretta*

- considera l'insieme *D* di tutti i nodi di grado dispari
- per ogni coppia di nodi *i,j* calcola il percorso minimimo da *i* a *j*, sia w_{ij} il costo
- risolvi un problema di *matching* tra i nodi di *D* minimizzando i costi w_{ij}, per ogni coppia *i,j* scelta del *matching* si raddoppiano gli archi del percorso minimo

Chinese Postman Problem

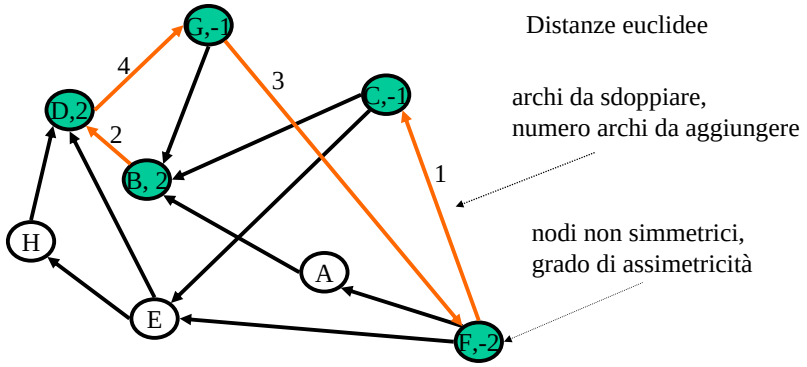


Chinese Postman Problem

Algoritmo *RenedereReteEulerianaReteDiretta*

- sia δ_i la differenza tra il numero degli archi entranti e il numero dei archi uscenti in *i*
- considera l'insieme D^+ (D^-) di tutti i nodi in cui $\delta_i > 0$ ($\delta_i < 0$)
- per ogni coppia di nodi $i \in D^+$, $j \in D^-$ calcola il percorso minimimo da *i* a *j*, sia w_{ij} il costo
- risolvi un problema di *trasporto* tra i nodi di D^+ e D^- minimizzando i costi w_{ij}, dove la capacità/domanda di ogni nodo è δ_i. Per ogni coppia *i,j* scelta del *trasporto* lungo il percorso minimo si aggiungono tanti archi quanto è il flusso calcolato da *i* a *j*

Chinese Postman Problem



Distanze euclidee

archi da sdoppiare, numero archi da aggiungere

nodi non simmetrici, grado di assimetricità

Chinese Postman Problem

- $D^+ = \{B, D\}$, $\delta^+_B = 2$, $\delta^+_D = 2$
- $D^- = \{C, F, G\}$, $\delta^+_C = -1$, $\delta^+_F = -2$, $\delta^+_G = -1$

- percorsi
  - B-D-G-F-C con costo $w_{BC} = 3+5+16+9=33$
  - B-D-G-F con costo $w_{BF} = 3+5+17=24$
  - B-D-G con costo $w_{BG} = 3+5=8$
  - D-G-F-C con costo $w_{DC} = 5+16+9=30$
  - D-G-F con costo $w_{DF} = 5+17=21$
  - D-G con costo $w_{DG} = 5$

Chinese Postman Problem

Problema di trasporto

$\min z = 33x_{BC} + 24x_{BF} + 8x_{BG} + 30x_{DC} + 21x_{DF} + 5x_{DG}$

$x_{BC} + x_{BF} + x_{BG} = 2$

$x_{DC} + x_{DF} + x_{DG} = 2$

$x_{BC} + x_{DC} = 1$

$x_{BF} + x_{DF} = 2$

$x_{BG} + x_{DG} = 1$

$x_{ij} \geq 0$

una soluzione ottima

$z^* = 83 \quad x_{BC} = 1, x_{BG} = 1, x_{DF} = 2 \quad x_{BF} = x_{DC} = x_{DG} = 0$

Chinese Postman Problem

Reti miste:

- per reti miste con tutti i nodi pari esiste una algoritmo esatto detto di *Minieka*
- non esiste algoritmo polinomiale esatto per reti miste con alcuni nodi dispari.
  - il problema su reti miste rimane NP-hard anche per reti planari
  - si può rendere la rete pari duplicando alcuni archi e quindi applicare l'algoritmo di Minieka, eventualmente applicando poi una procedura correttiva se tale algoritmo ha fatto perdere la parità alla rete. Tale approccio, detto *Mixed1*, può non essere ottimo, infatti gli archi duplicati per rendere il grafo pari possono non essere quelli giusti se poi si devono duplicare altri archi per rendere il grafo simmetrico
  - l'approccio simmetrico, *Mixed2*, applica prima Minieka e quindi rende pari la rete risolvendo un problema di matching e/o di trasporto
  - gli algoritmi euristici *Mixed1* e *Mixed2* determinano una soluzione con un errore massimo del 200%. Se si sceglie la migliore soluzione tra i due l'errore scende al 166%

Rural Postman Problem

Rural Postman Problem *RPP(G,c)*

Istanza: una rete connessa $G=(N,A)$, un costo (distanza) c_{ij} associato ad ogni arco $(i,j)\in A$. Un sottoinsieme $R\subset A$ di archi che devono essere visitato

Soluzione: un circuito che passa attraverso ogni arco di R almeno una volta.

Obiettivo: minimizzare il costo (la lunghezza) del circuito.

Note:

- in seguito si indica con G_R la rete indotta dagli archi R

Rural Postman Problem

Caso semplice: rete non orientata, G_R connesso

Algoritmo esatto

- determina i nodi dispari in G_R e risolvi un problema di *matching* tra essi sfruttando i cammini minimi che li congiungono sulla rete completa G
- risolvi un *CPP* su la rete G_R aumentata dagli archi dei cammini minimi scelti dall'algoritmo di *matching*

Rural Postman Problem

Caso NP-hard: rete non orientata, G_R non connesso

Algoritmo euristico *Balance-and-Connect*

- costruisci estendi G_R fino ad ottenere una rete pari risolvendo un problema di *matching*
- connetti le componenti di G_R. In particolare, risolvi un problema di albero minimo che unisce le componenti connesse di G_R utilizzando una rete ausiliaria completa T che ha per nodi le componenti connesse di G_R. Per ogni arco (r,s) di T e per ogni coppia di nodi $i\in r$, $j\in s$, il costo dell'arco (r,s) è uguale al minimo su i e j della somma dei costi dei percorsi minimi da i a j e da j a i su G
- risolvi il RPP in cui gli archi da visitare formano una rete connessa con l'algoritmo precedente

Rural Postman Problem

Commenti:

- la difficoltà dei problemi rimane la stessa nel caso di rete orienta
- gli algoritmi si generalizzano banalmente:
  - se G_R connesso, al primo passo dell'algoritmo si rende la rete simmetrica risolvendo un problema di *trasporto*
  - se G_R non connesso, al primo passo dell'algoritmo si rende la rete connessa risolvendo un problema di *minimum spanning arborescence*

Capacitated Arc Routing Problem

Capacitated Arc Routing Problem *(G,c,K,I,a,b)*

Istanza: una rete connessa $G=(N,A)$. Un costo (distanza) c_{ij} associato ad ogni arco $(i,j) \in A$ e una domanda d_{ij} associato ad ogni arco $(i,j) \in R \subseteq A$. Una flotta di K veicoli identici di capacità b. Un deposito localizzato sul nodo 0 della rete.

Soluzione: un insieme di K circuiti, contenenti il nodo 0, che coprano tutti gli archi e tali che la somma delle domande associate ad ogni circuito non ecceda b. Ciascun arco deve essere assegnato ad un unico giro

Obiettivo: minimizzare il costo (la lunghezza) dei k circuiti

Note:

- il problema è NP-hard anche per reti non orientate o orientate.

Euristiche

Euristiche

- costruttive
- di miglioramento
- metodi a due fasi
  - *cluster first – route second*
  - *route first – cluster second*

Path Scanning

Algoritmo *PathScanning*

- forma un ciclo C aggiungendo successivamente un arco in RE (insieme degli archi R non ancora visitati). Quando, giunto nel nodo i, la capacità del veicolo è sfruttata completamente torna al deposito secondo lungo il percorso più breve $SP(i,0)$
- rimuovi C da RE. Se RE è vuoto stop, altrimenti ritorna al passo precedente

Path Scanning

Gli archi in *RE* vengono selezionati in base al fatto che rispettano la capacità del veicolo e che (alternativamente):

- minimizzano l'incremento del costo
- massimizzano la capacità residua
- minimizzano o massimizzano la distanza dal nodo deposito
- massimizzano la distanza dal nodo deposito se il veicolo non è ancora mezzo pieno, la minimizzano se metà della capacità è già stata utilizzata
- casuale

Augment-merge

Algoritmo *Augment-Merge*

- costruisci un ciclo separato per ogni arco in *R*
- *augment*: a partire dal ciclo più lungo estendi il servizio agli archi in *R* che appartengono a tale ciclo fino a quando la capacità del veicolo lo permette
- *merge*: in ordine decrescente dei risparmi fondi due cicli fino a quando la capacità del veicolo lo permette e fino a quando il risparmio è positivo

Augment-insert

Algoritmo *Augment-InsertVersionI*

inizializzazione

costruisci la lista *L*, i.e., ordina gli archi *(i,j)* di *R* in ordine decrecente rispetto alla somma *SP(0,i) + SP(j,0)*

interazione

augment: considera il primo arco in *L* (il più lontano) estendi il servizio agli archi in *RE* che appartengono a tale circuito corrente fino a quando ci sono archi sul circuito in *RE* e la capacità del veicolo lo permette

insert: inserisci nel circuito corrente gli archi che inducono una deviazione inferiore ad una data soglia *CLIM* e tali per cui la capacità del veicolo non è violata.

Augment-insert

commenti:

- la versione II differisce dalla prima perché, invece di iniziare dagli archi più lontani, parte da quelli con domanda minima
- il parametro *CLIM* è un dato di input. Se assume un valore grande permette ampli *detour*

Construct-strike

Algoritmo *Construct-Strike*

iterazione

- *construct*: aggiungendo un arco alla volta, forma un circuito *C* ammissibile dal punto di vista della capacità tale che *RE-C* sia connesso
- *strike*: poni *RE= RE-C* e ripeti passo *construct* fino a quando *RE* vuoto o fino a quando non si possono fare più circuiti
- rimuovi eventuali archi aggiunti in iterazioni precedenti
- risolvi un problema di *matching* sui nodi di grado dispari nella rete in cui siano stati sottratti gli archi già inseriti in un circuito e due copie del deposito in modo da ottenere una rete euleriana con grado pari positivo per il deposito. Itera fino a quando *RE* non è vuoto

Construct-strike

commento:

- vi sono vari criteri per scegliere in che ordine selezionare gli archi necessari a formare il circuito nella fase *costruct*
- il criterio che sembra funzionare meglio è quello di selezionare l'arco *(i,j)* tale che sia massima la domanda che si intercetta lungo il percorso di domanda minima che va da *j* a *0*

Parallel-insert

Algoritmo *Parallel-Insert*

inizializzazione

- crea un percorso che contenga l'arco in *R* più distante dal deposito

iterazione

- *insert I*: determina l'arco in *RE* più lontano dai circuiti esistenti ed inseriscilo al meglio in uno dei circuiti esistenti che abbiano la capacità di assorbire la domanda. Se tale circuito non esiste e il numero dei circuiti è inferiore a *NRL*, crea un nuovo circuito.
- *insert II*: considera il circuito con carico minimo. Determina l'arco in *RE* che possa essere inserito a costo minimo nel circuito e la cui domanda sia ammissibile data la capacità residua del circuito.
- *exchange*: considera scambi di archi che possano diminuire la lunghezza dei circuiti. Torna ad *insert I*.
- *relax*: rilassa vincoli di capacità e di lunghezza di percorsi di una data percentuale e torna ad *exchange*.

Parallel-insert

commento:

- l'algoritmo produce più soluzioni. Ovviamente si tiene la migliore. A volte data una soluzione, si elimina il circuito meno carico e si reinseriscono gli archi in *RE* e si riparte dal passo *exchange.*

Euristiche costruttive

- Le euristiche che sembrano funzionare meglio sono:
  - le due versioni dell'*Augment-Insert*
  - *Construct-Strike* con la regola di selezione degli archi indicata nei commenti
  - *Path Scanning* con inserzione random
- Non vi sono però euristiche dominanti in ogni istanza

Altre euristiche

In letteratura esistono altre euristiche anche basate su approcci a due fasi.

Nel seguito si presenta l'euristica di *route first-cluster second* da utilizzare quando è possibile decidere anche sul tipo (e quindi la capacità) dei veicoli da utilizzare.

Ulusoy

Algoritmo *Ulusoy*

- *route*: determina un *gigant tour*
- *cluster*: costruisci una rete orientata ausiliaria $T = (V,E)$:
  - ogni nodo in V corrisponde ad un arco in R, esiste poi un nodo finale f
  - ogni arco $(i,j) \in E$ unisce due nodi corrispondenti a due archi in R tali che i sia visitato prima di j ed esista un veicolo con capacità tale da potere soddisfare tutta la domanda degli archi da i a $j-1$.

    Un arco $(i,f) \in E$ implica il soddisfacimento della domanda di tutti gli archi da i compreso in poi
  - il costo associato ad ogni arco $(i,j) \in E$ è uguale al costo fisso di utilizzo del veicolo che può servire tale arco a cui vanno aggiunti i costi di trasferimento da deposito all'arco i, dall'arco i all'arco $j-1$, dall'arco $j-1$ al deposito
  - determina il percorso a costo minimo su T dal nodo corrispondente al primo nodo del *gigant tour* al nodo f. Ognuno degli archi di tale percorso corrisponde ad un circuito di un veicolo.

Altri problemi di routing

Problema generale

- **General Routing Problem**

  data una rete $G=(N,A)$, con costi $c(i,j)$ per ogni arco $(i,j) \in A$, determinare un circuito a costo minimo che includa il sottoinsieme $Q \subset V$ di nodi e il sottoinsieme $R \subset A$ di archi.

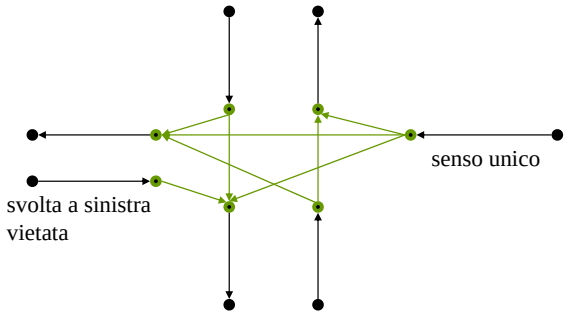
Problemi reali

flotta eterogenea
capacità multiple (peso, volume)
tipi diversi di merci
un veicolo può fare più di un giro
consegna + prelievo
limiti temporali (‘‘finestre’’)
molti depositi

Problemi reali

precedenze tra i clienti
numero di veicoli da decidere
qualche cliente può essere saltato
consegne periodiche
rifornimento di magazzini
struttura dei costi

Rappresentazione delle strade



Rappresentazione di un incrocio stradale sotto forma di rete.
Ad ogni arco di attraversamento dell’incrocio può essere assegnato un costo diverso a seconda della difficoltà della manovra

Bibliografia

Fleet management and logistics, eds Cranic T.G., Laporte G., Kluwer, Boston, Mass (1998).

Handbooks in Operations Research and Management Science, Vol. 8: Network. Routing, eds M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, Elsevier, Amsterdam, NL (1995).

Scienze delle decisioni per i trasporti, eds S. Pallottino, A. Sciomachen, Franco Angeli, Milano, I (1999).

Esercizi

1. Si consideri la fase di assegnazione nell'algoritmo ClusterFirstRouteSecond per il VRP. Si formuli un problema di MIP in cui non si fissa a priori quali siano i *seed*. In particolare, potenzialmente tutti i nodi devono potere essere *seed* ma la soluzione del problema di MIP ne scelga solo *k* in modo che siano minimizzati i costi di inserimento complessivi.