"Sapienza" Università di Roma - Dipartimento di Ingegneria Informatica, Automatica e Gestionale

Problemi di Instradamento di Veicoli

Renato Bruni
bruni@dis.uniroma1.it

Il materiale presentato è derivato da quello dei proff. A. Sassano e C. Mannino

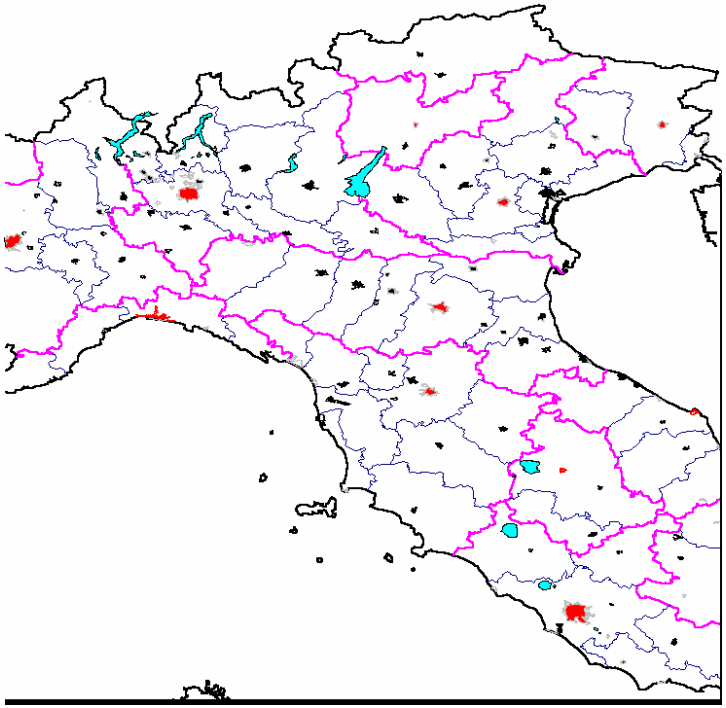
Tornando al Problema di Distribuzione

PROBLEMA GENERALE:

- Trasportare beni da una o più *origini* a una o più *destinazioni*
- Minimizzando dei costi di trasporto
- Rispettando una serie di vincoli:
  - vincoli di capacità dei veicoli
  - vincoli di raggiungibilità delle destinazioni
  - …

Abbiamo deciso dove posizionare le origini (genericamente i depositi) risolvendo la localizzazione di impianti

Ora dobbiamo decidere il resto

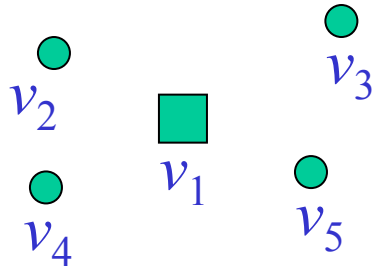


Dati del Problema

Dopo aver risolto la localizzazione, per ogni area si conosce:

1. Origine (o deposito) v_1
2. Destinazioni (o clienti) che v_1 deve servire $\{v_2, v_3, \ldots, v_n\}$

3. Grafo *orientato* $G(V,E)$ deposito-clienti



Nodi $V=\{v_1,\ldots,v_n\}$

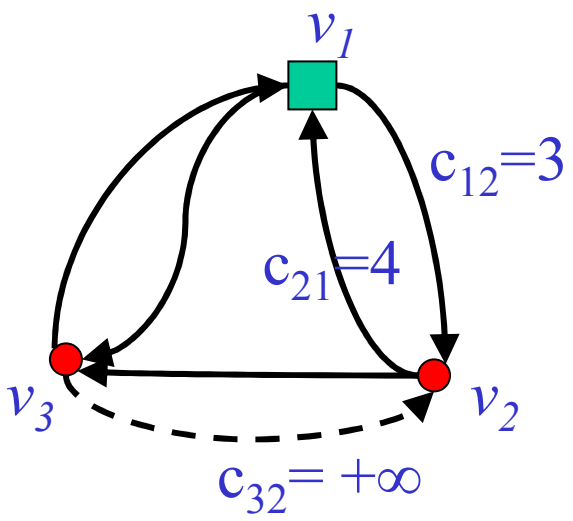
Archi 

se b è *accessibile* da a

4. *Costo dell'arco* da v_a a v_b: c_{ab} è il valore dell'aspetto che si vuole minimizzare, ad es. spesa, distanza, preferibilità, tempo, etc. (è importante scegliere correttamente)

Il Grafo dei Collegamenti

• Vogliamo un grafo orientato che sia completo $G(V,E)$ quindi aggiungiamo anche gli archi relativi a collegamenti che non ci sono mettendo pesi opportunamente alti

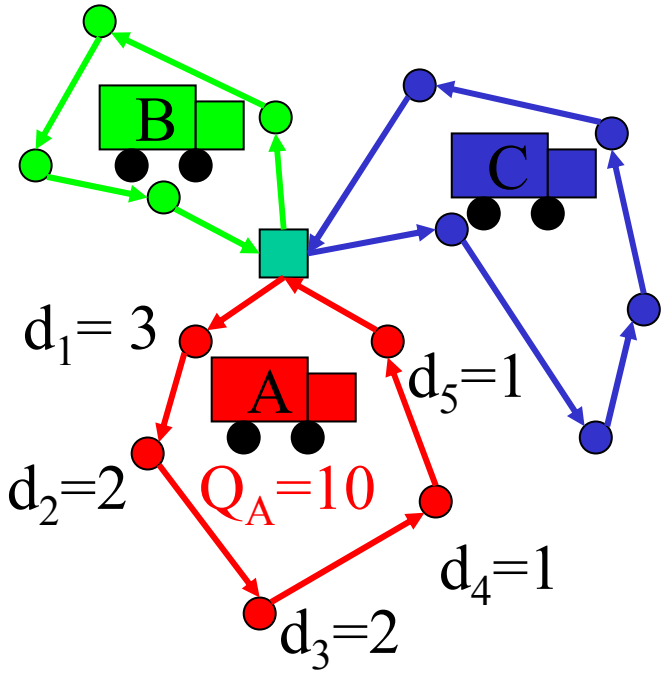


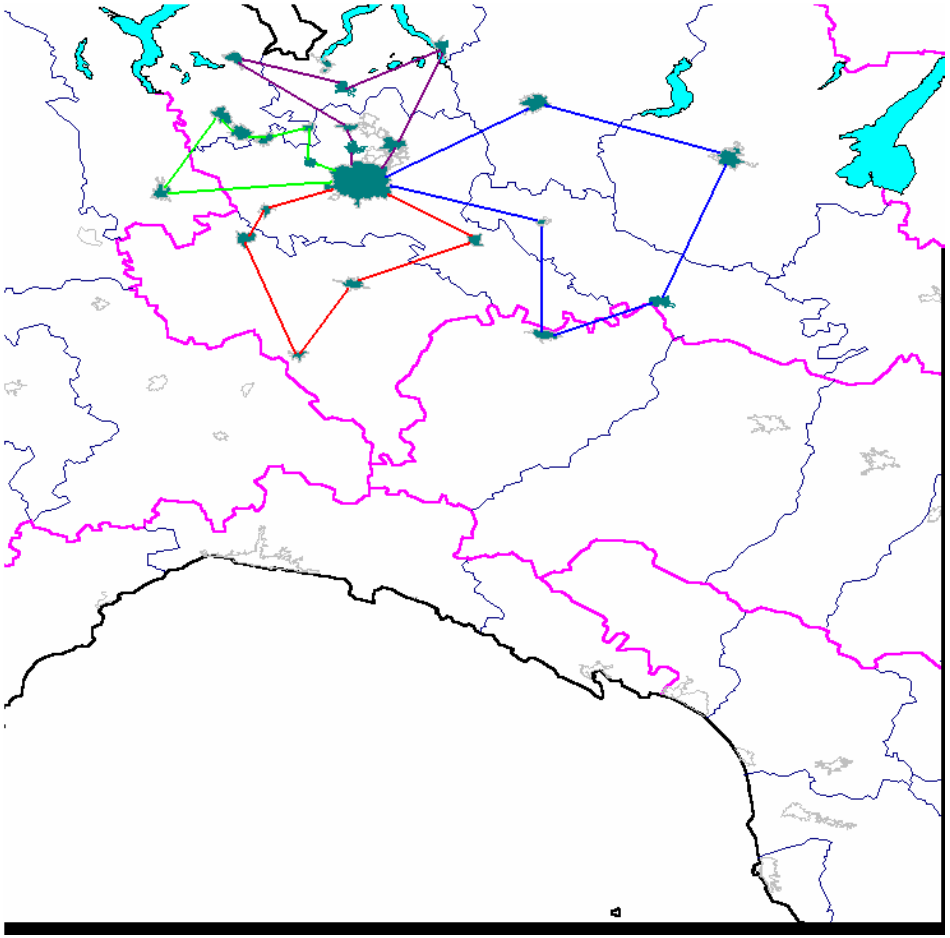
→ poniamo $c_{32} = \infty$ (o meglio un numero >> tutti gli altri costi) se 2 non è accessibile da 3

La Flotta di Veicoli

- Per servire i vari clienti abbiamo una flotta di veicoli
- Ciascun veicolo visita un insieme di clienti e torna al deposito

- *Numero e caratteristiche dei singoli veicoli* devono essere noti





Dati su Domanda e Veicoli

1. d_i: *domanda del cliente* v_i
2. m: *numero di veicoli disponibili*
3. $J=\{1, 2, \ldots, m\}$: *insieme dei veicoli*
4. Di solito Q_j: *capacità del veicolo* $j \in \{1, 2, \ldots, m\}$
5. A volte per ogni cliente bisogna distinguere se è consegna o prelievo (*Pickup or Delivery*)

Eventuali aspetti temporali: orari di apertura limitati, tempi di trasporto, di carico e scarico (tempi di transito)

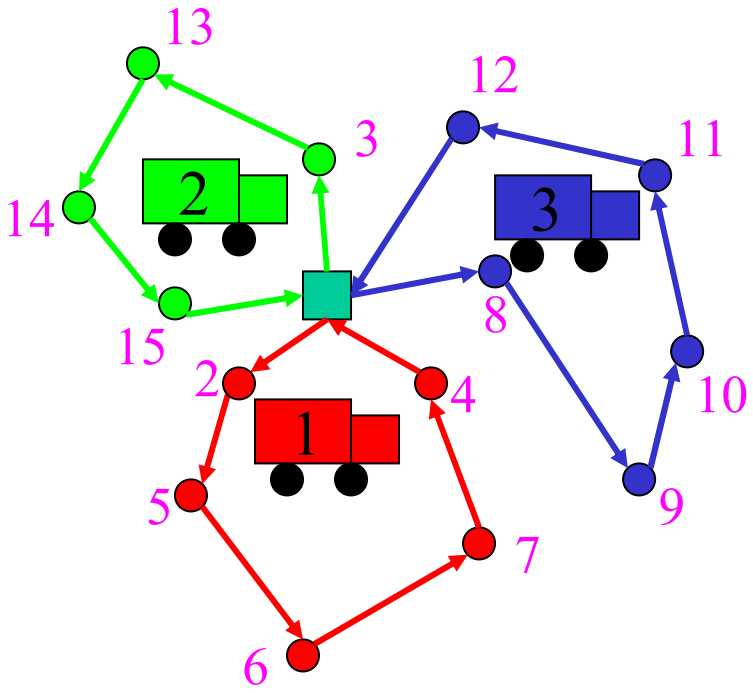
6. *Tempo di transito* da a a b: t_{ab}
7. *Tempo di servizio* di a: t_a
8. T_j: *tempo massimo di rientro del veicolo* $j \in \{1, 2, \ldots, m\}$
9. *Finestre temporali* associate ad ogni nodo v_i: $[inizio_i, fine_i]$

Struttura delle Soluzioni

- Famiglia dei “cluster” clienti/veicolo

$\boldsymbol{C} = \{C_1, C_2, \ldots, C_m\}$

- $C_k \subseteq V \quad k \in \{1, 2, \ldots, m\}$
- $v_1 \in C_k \quad k \in \{1, 2, \ldots, m\}$
- $C_k \cap C_h = \{v_1\} \quad k,h \in \{1, 2, \ldots, m\}$
- $\cup \; C_k = V$
- $\sum_{h \in C_k} d_h \leq Q_k$



$C_1 = \{1,2, 4, 5,6,7\}$

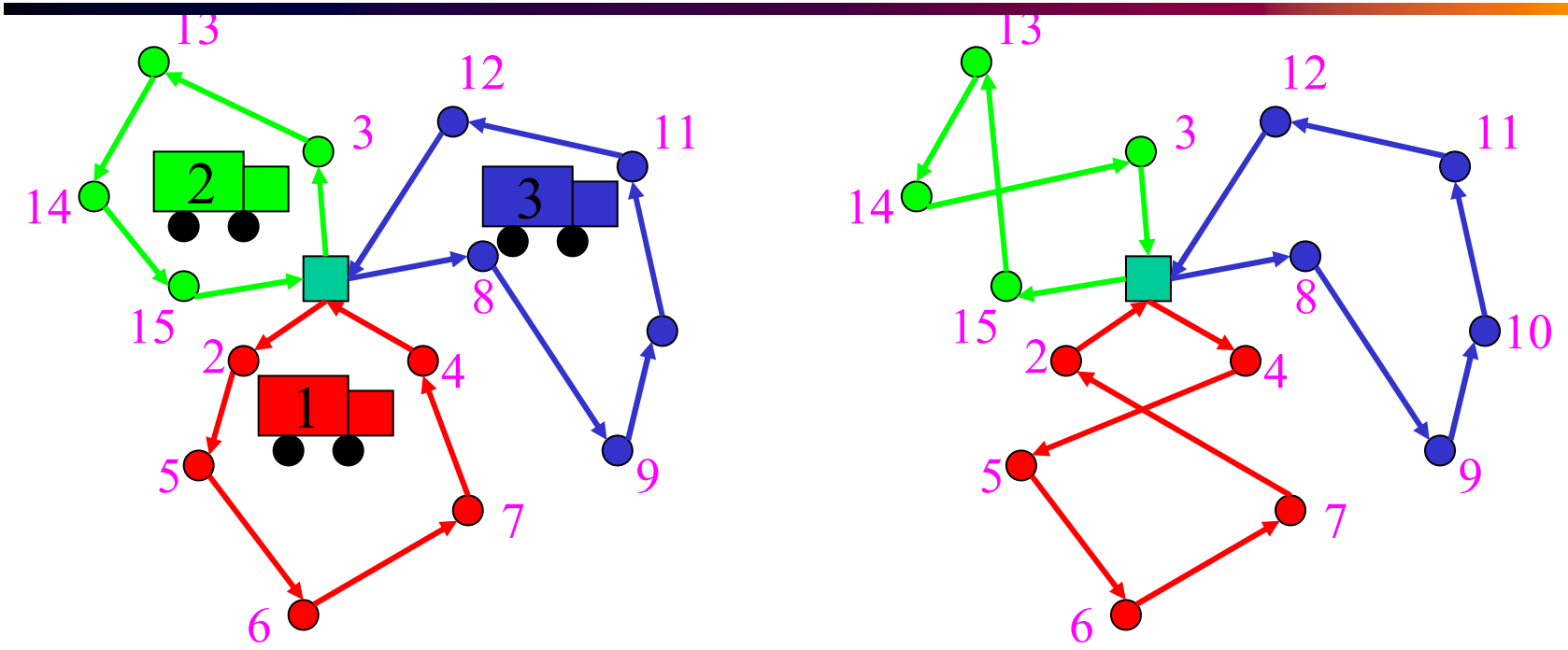
$C_2 = \{1, 3, 13,14,15\}$

$C_3 = \{1,8, 9, 10,11,12\}$

$\boldsymbol{C}$ caratterizza completamente la soluzione ?

No! Ad un “cluster” posso associare diverse sequenze di visita

Clustering e Routing



Esempio di stessi "cluster" ma sequenze ("routing") diverse

Una sequenza è un ciclo T_k orientato che tocca tutti i nodi del "cluster" C_k (***Ciclo Hamiltoniano,*** già visto come trovarli)

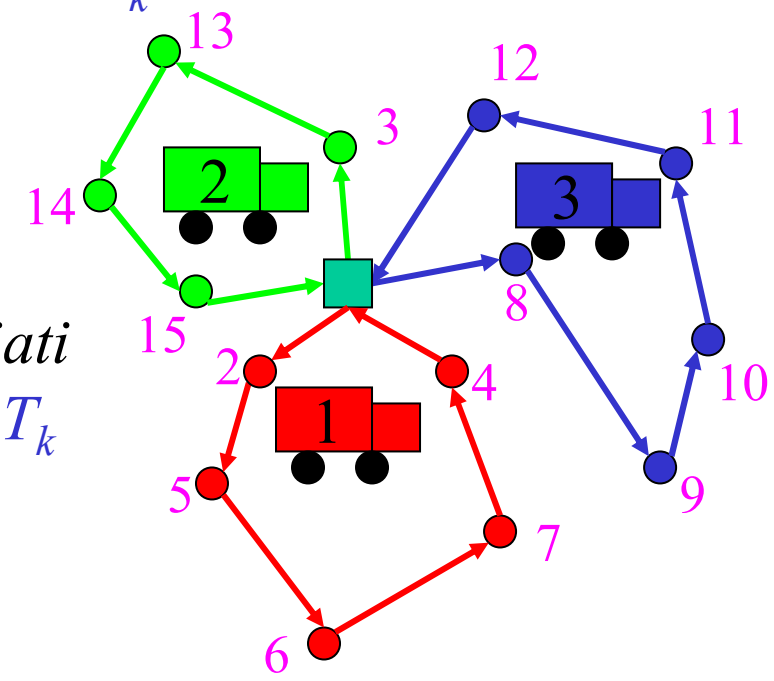
Conclusione: *Una soluzione ammissibile è data dai* ***cicli hamiltoniani*** $\{T_1, T_2, \ldots, T_m\}$ *definiti sui "cluster"* $\{C_1, C_2, \ldots, C_m\}$

Problema Base di Distribuzione

- *Soluzione ammissibile*: $\boldsymbol{T} = \{T_1, T_2, \ldots, T_m\}$
- Un ciclo hamiltoniano T_k per ogni "cluster" C_k
- *Insieme delle soluzioni ammissibili*:

$$\mathcal{T} = \{\boldsymbol{T}_1, \boldsymbol{T}_2, \ldots, \boldsymbol{T}_q\}$$

- $L(T_k)$ *somma dei costi di transito associati agli archi del ciclo hamiltoniano* T_k
- $Z(\boldsymbol{T}) = \sum_{k \in J} L(T_k)$ *costo di transito*



Problema Base di Distribuzione

$$\min\ Z(\boldsymbol{T}) : \ \boldsymbol{T} \in \mathcal{T}$$

Minimizzare il costo di transito
- *servendo la domanda di tutti i clienti*
- *rispettando la capacità dei veicoli*

Formulazione del Problema Base

Proviamo a usare variabili binarie y_{hk} *con:*

$y_{hk}=1$ *se* $v_h \in C_k$

$y_{hk}=0$ *se* $v_h \notin C_k$

Avremmo un modello di questo tipo, con y_k *vettore di incidenza di* C_k

$$\min \sum_{k \in J} TSP(y_k)$$

$$\sum_{k \in J} y_{1k} = m$$

$$\sum_{k \in J} y_{hk} = 1 \quad h=2 \ldots n$$

$$\sum_{h \in V - \{v_1\}} d_h y_{hk} \leq Q_k$$

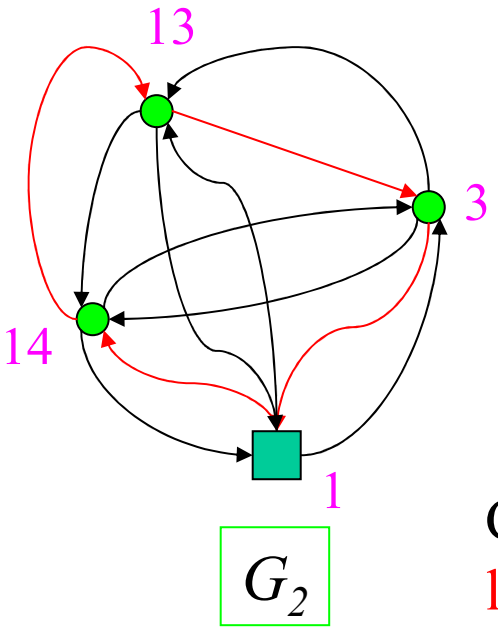
Ma come scriviamo la funzione obiettivo?

Per ogni cluster, sarebbe il costo del ciclo hamiltoniano su quel cluster
Ma noi vogliamo una funzione lineare, non possiamo scriverla come un problema di TSP per ogni cluster

Nuove variabili binarie

Allora dobbiamo usare delle variabili diverse, e purtroppo con 3 indici, quindi la dimensione del modello cresce notevolmente…

Evento elementare: *il veicolo k usa l'arco $v_i v_j$ con* $\{v_i, v_j\} \subseteq C_k$



G_2

Variabile: x_{ij}^k

1 se veicolo k usa $v_i v_j$

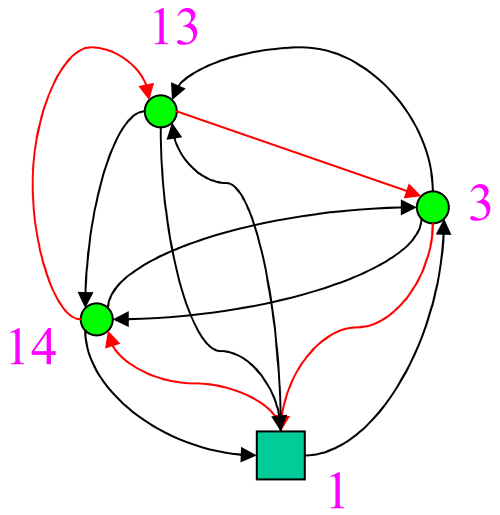
0 altrimenti

Esempio: $x_{13,3}^2 = 1;\ x_{1,13}^2 = 0$

Così potrò esprimere la funzione obiettivo in modo lineare come somma dei pesi degli archi usati

Condizioni di Ammissibilità 1/2

Quando è che un sottoinsieme di archi T di G_k costituisce un ciclo Hamiltoniano di G_k ?



G_2

È intanto necessario che:

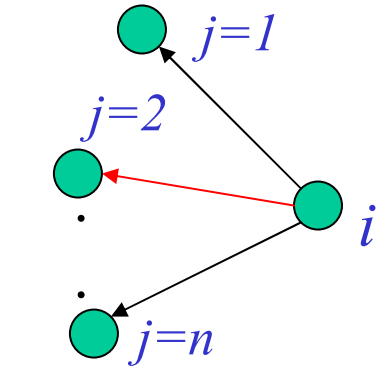
1. *Il veicolo k esca da ogni nodo di C_k*
2. *Il veicolo k entri in ogni nodo di C_k*

Il vettore $[\, x_{ij}^k :\ i=1,\ldots,n;\ \ j=1,\ldots,n]$

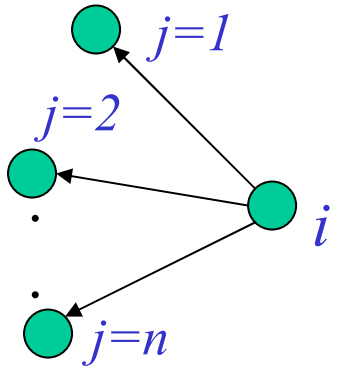
Deve rispettare queste condizioni

$$\sum_{j=1}^{n} x_{ij}^k = 1 \text{ oppure } 0$$

$$\sum_{j=1}^{n} x_{ji}^k = 1 \text{ oppure } 0$$



1 *se* $i \in C_k$



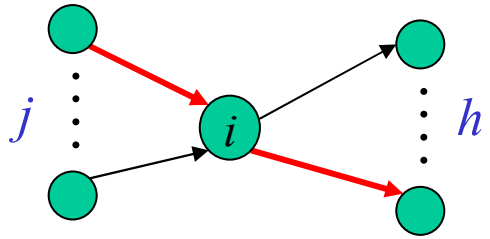
0 *se* $i \notin C_k$

Condizioni di Ammissibilità 2/2

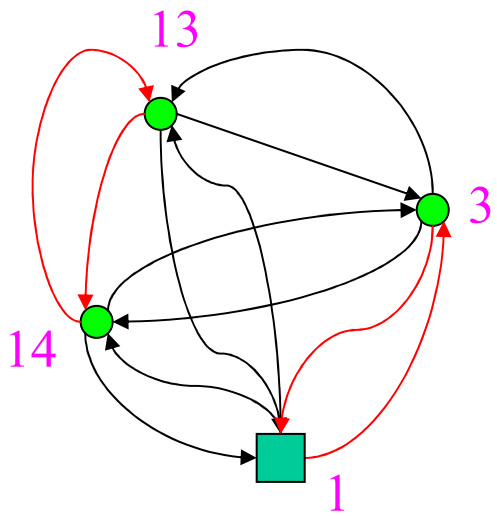
Mettendo tutto assieme possiamo scrivere

$$\sum_{\substack{j\in I\\ j\neq i}} x_{ji}^{\,k} = \sum_{\substack{h\in I\\ h\neq i}} x_{ih}^{\,k} \qquad \begin{matrix} i\in I \\ k\in J \end{matrix}$$

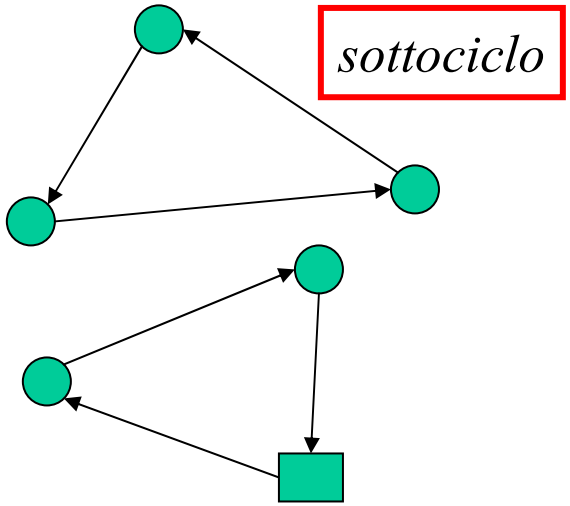
se veicolo k entra nel nodo i deve uscirne



Ma non basta: una soluzione che rispetta questi vincoli può ancora contenere sottocicli, cioè cicli che non passano per il deposito !

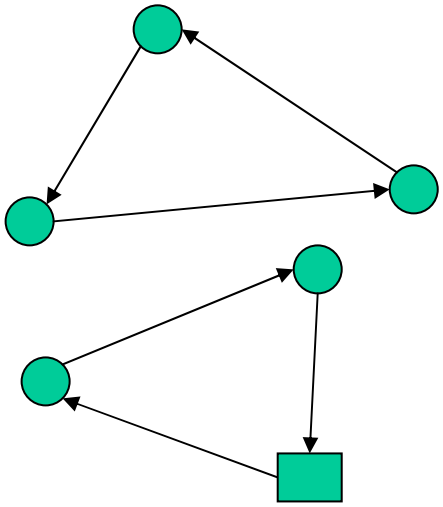


G_2



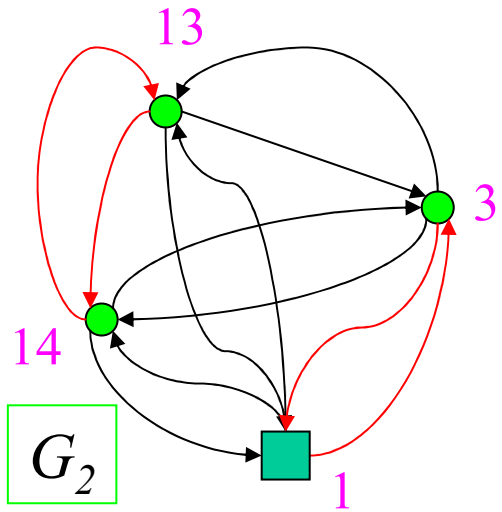
Come si eliminano i sottocicli ?

Eliminazione dei Sottocicli



- *Un sottociclo è un ciclo orientato su un sottoinsieme dei nodi di* C_k
- *Un ciclo orientato su un sottoinsieme* $S \subset C_k$ *contiene* $|S|$ *archi*

ELIMINAZIONE DEI SOTTOCICLI:
Nessun sottoinsieme proprio S *di* C_k *può contenere* $|S|$ *archi*



$$\sum_{(i,j)\in S} x_{ij}^{k} \leq |S|\text{-}1$$

o equivalentemente

$$\sum_{i\in S\, j\notin S} x_{ij}^{k} \geq 1$$

per ogni $S \subset C_k$
$2 \leq |S| \leq |C_k|\text{-}1$
per ogni $k \in J$

Formulazione VRP come PL01

$$\min \sum_{k\in J}\sum_{i\in I}\sum_{\substack{j\in I\\ j\neq i}} c_{ij}\, x_{ij}^{k}$$ minimizzo somma costi

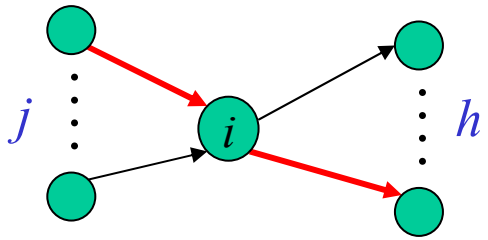
$I=\{1, 2, \ldots, n\}$: insieme dei nodi (tra cui 1 è il deposito)

$J=\{1, 2, \ldots, m\}$: insieme dei veicoli

$$\sum_{k\in J}\sum_{j\in I} x_{1j}^{k} = m$$ dal deposito escono m veicoli

$$\sum_{k\in J}\sum_{\substack{j\in I\\ j\neq i}} x_{ij}^{k} = 1 \qquad i\in\{2,\ldots,n\}$$ da ogni nodo esce 1 veicolo, cioè ogni cliente è servito

$$\sum_{\substack{j\in I\\ j\neq i}} x_{ji}^{k} = \sum_{\substack{h\in I\\ h\neq i}} x_{ih}^{k} \qquad \begin{matrix} i\in I\\ k\in J\end{matrix}$$ se veicolo k entra nel nodo i deve uscirne



$$\sum_{i\in I}\sum_{\substack{j\in I\\ j\neq i\\ j\neq 1}} d_j\, x_{ij}^{k} \leq Q_k \qquad k\in J$$ la capacità è rispettata

$$\sum_{(i,j)\in S} x_{ij}^{k} \leq |S|\text{-}1 \qquad S\subseteq\{2,\ldots,n\},\ |S|\geq 2 \qquad k\in J$$ proibisco ogni possibile sottociclo S

$$x_{ij}^{k} \in \{0,1\} \qquad \begin{matrix} i,j\in I\\ k\in J\end{matrix}$$ variabili x_{ij}^{k} : percorro arco (i,j) col veicolo k

Numero di Veicoli variabile

- Se i veicoli disponibili sono m ma **potrei anche utilizzarne di meno** (spendendo meno), considero per ognuno un costo di attivazione f_k
- Se invece spendo sempre lo stesso anche se lascio dei veicoli fermi, non serve

$$\min \sum_{k\in J}\sum_{i\in I}\sum_{\substack{j\in I\\ j\neq i}} c_{ij}\, x_{ij}^{\,k} + \sum_{k\in J} f_k \sum_{\substack{j\in I\\ j\neq 1}} x_{1j}^{\,k}$$

minimizzo somma costi percorrenza + costi attivazione

$$\sum_{k\in J}\sum_{j\in I} x_{1j}^{\,k} \leq m$$

dal deposito escono fino a m veicoli

… e tutto il resto come prima

- Con meno veicoli occorre fare dei percorsi più lunghi
- Cerco in pratica il **miglior compromesso** tra risparmio per i veicoli ed economicità dei percorsi

Aspetti Temporali

- Se per il generico veicolo k ho un **tempo massimo di utilizzo** T_k (dovuto per es. alla disponibilità del conducente, a vincoli sull'orario di lavoro o alla chiusura del deposito)
- T_k deve essere espresso in una unità di misura facilmente sommabile e a partire dall'inizio delle operazioni (ad esempio in minuti dall'uscita dal deposito)

- Devo **considerare**: (ovviamente nella stessa unità di misura)

$$\begin{cases} \textit{Tempo di transito} \ \text{da} \ i \ \text{a} \ j\text{:} \ t_{ij} \\ \textit{Tempo di servizio} \ \text{di} \ i\text{:} \ t_i \end{cases}$$

- Devo **aggiungere vincoli**:

$$\sum_{i \in I} t_i \sum_{\substack{j \in I \\ j \neq i}} x_{ij}^k + \sum_{i \in I} \sum_{\substack{j \in I \\ j \neq i}} t_{ij} \, x_{ij}^k \leq T_k \qquad k \in J$$

il tempo complessivo di utilizzo del veicolo k non supera il limite

- Tempi di transito e/o di servizio potrebbero variare da veicolo a veicolo
- In tal caso ho t_{ij}^k e t_i^k

Finestre Temporali 1/2

- Se invece per il generico cliente j ho una **finestra temporale** $[\ iniz_j\ \ fine_j\ ]$ (posso visitarlo solo dall'istante $iniz_j$ all'istante $fine_j$, sempre espresso in una opportuna unità di misura a partire dall'inizio delle operazioni

- Devo **considerare**: un nuovo insieme di variabili $a_j \in \mathrm{R}$ con $j \in I$ che rappresentano l'istante di arrivo dal cliente j (dette variabili temporali, ovviamente nella stessa unità di misura)

- Devo **aggiungere i vincoli**:

$$iniz_j \le a_j \le fine_j \qquad j \in I$$

giungo dal cliente j rispettando la sua finestra temporale

$$a_j \in \mathrm{R} \qquad j \in I$$

- Però questo non basta (!): devo anche **aggiungere vincoli** tra le a e le x, altrimenti troverò dei valori per le a che soddisfano tutte le finestre, ma che non è magari possibile ottenere con i percorsi fatti secondo le x

Finestre Temporali 2/2

- Vincoli da aggiungere che legano le a e le x:

se visito il cliente i all'istante a_i e subito dopo vado dal cliente j, deve avere:

$a_j = a_i + t_i^k + t_{ij}^k$ raggiungo il cliente j un istante che è dato da quando arrivo in i, più il tempo di servire i e transitare da i a j

e ciò deve valere sse $x_{ij}^k = 1$

allora separo = in $\leq$ e $\geq$ e aggiungo qualcosa che impone il vincolo sse $x_{ij}^k = 1$:

$$\left.\begin{aligned} a_j &\geq a_i + t_i^k + t_{ij}^k - (1 - x_{ij}^k)\, T \\ a_j &\leq a_i + t_i^k + t_{ij}^k + (1 - x_{ij}^k)\, T \end{aligned}\right\} \quad i,j \in I \quad k \in J$$

T è un valore maggiore di tutti i possibili valori di a, e quindi se $x_{ij}^k = 0$ i vincoli sono banalmente soddisfatti, se invece $x_{ij}^k = 1$ i vincoli devono valere

Cenni a Ulteriori Varianti

- Se in alcuni nodi devo prelevare e in altri consegnare ciò che ho prelevato, si chiama **pickup and delivery** problem

devo imporre vincoli di precedenza tra l'instante in cui visito il cliente i in cui prelievo e l'istante in cui visito il cliente j a cui consegno

$$a_i \leq a_j \quad \text{per ogni coppia } i, j \text{ di pickup e delivery}$$

Oppure, se non ho le variabili temporali a_i, uso variabili intere u_i^k che rappresentano l'ordine di visita del nodo i col veicolo k, vincoli opportuni tra le u e vincoli che colleghino le u alle x (simili a quelli già visti per le a)

Inoltre servono variabili q_i^k che sono il carico totale del veicolo k alla partenza dal cliente i, in modo da imporre che il carico totale non superi mai la capacità

- Esiste anche il **Dial a Ride** Problem: bisogna soddisfare varie richieste di trasporto da un punto ad un altro (eventualmente giunte anche durante l'esecuzione): è il problema dei taxi

Come gestire i Vincoli di Sottocicli?

- I vincoli di "eliminazione" di sottocicli sono in numero **esponenziale**

$$\sum_{ij \in E(S)} x_{ij}^{k} \leq |S| - 1 \quad \text{per ogni} \quad S \subset C_k$$

Allora si può usare l'approccio di **generazione di vincoli** per questo gruppo di vincoli: non li metto e poi ne aggiungo uno alla volta finché l'oracolo di separazione dice che sono tutti soddisfatti (cioè finche non si creano più sottocicli)
Bisogna inventare un oracolo per questi vincoli. Immaginiamo di avere un solo cluster per semplicità

• Oracolo di separazione: dato un punto (anche frazionario) x^* che soddisfa gli altri vincoli, decide *se esiste un vincolo di sottociclo violato da* x^*, cioè trova $S \subset V$, *tale che*

$$\sum_{ij \in E(S)} x_{ij}^{*} > |S| - 1 \qquad \text{(A)}$$

oppure conclude che nessun vincolo di sottociclo è violato da x^*

Separazione Vincoli Sottocicli

Osservazione: dato che ogni cliente $i \in S$ *deve essere visitato,* per ogni possibile S si ha:

$$\sum_{ij \in E(S)} x_{ij}^* + \sum_{ij \in \delta^+(S)} x_{ij}^* = |S|$$

Si ha violazione dei vincoli di sottociclo quando la I sommatoria è $> |S|-1$ (condizione (A)), cioè quando $\sum_{ij \in \delta^+(S)} x_{ij}^* < 1$

Allora possiamo generare vincoli di sottociclo violati cercando per esempio tagli per cui valga questa condizione (è un modo, se ne potrebbero pensare anche altri…)

Oracolo Vincoli Sottocicli

Oracolo di Separazione

Associa a ogni arco (i,j) *capacità* $c_{ij} = x_{ij}^*$

Per ogni coppia di nodi u,v

Calcola l'u-v *taglio a capacità minima* ($u \in W$, $v \notin W$)

Se $c(W) < 1$ $\rightarrow$ vincolo violato e STOP

Fine ciclo

Se esco dal ciclo e la condizione non si è mai verificata

$\rightarrow$ *Non esiste un vincolo violato*

l'u-v *taglio* a capacità minima si trova risolvendo il problema di massimo flusso u-v

Quindi risolvo un problema di massimo flusso per ogni scelta u,v. Se fisso u, ne devo risolvere uno per ogni scelta di v, cioè $|C_k|$-1

Algoritmi di Soluzione

- **Metodi Esatti:**

  Branch-and-cut, cioè Branch-and-Bound con aggiunta di vincoli per migliorare la formulazione, eventualmente con generazione dinamica di vincoli di sottociclo

  raggiunge l'ottimo per problemi di dimensione contenuta, non riesce a risolvere problemi grandi

- **Metodi Approssimati (Euristiche):**

  Euristiche costruttive: Greedy, Clarke & Wright

  Costruttive in due fasi: Cluster First Route Second (CFRS)
  o Route First Cluster Second (RFCS)

  Euristiche migliorative: Ricerca Locale, Taboo Search

  più usate per risolvere problemi pratici (spesso grandi dimensioni)

Euristiche Costruttive

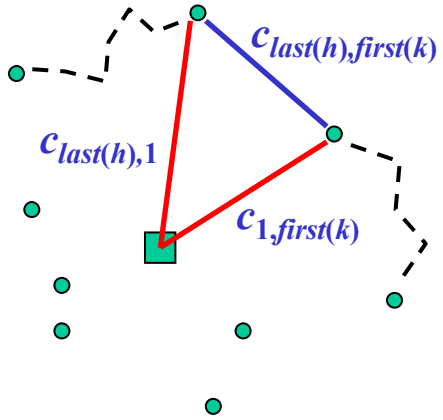
- **Clarke & Wright:**

Passo 0: forma soluzione iniziale $\{T_1^0, \ldots, T_{n-1}^0\}$ con $T_i^0 = \{v_1, v_{i+1}\}$ (una rotta per cliente)

Passo *i*: per ogni coppia $\{T_h^i, T_k^i\}$ tale che la somma delle sue domande è $\leq Q$ (cioè rotte unibili senza violare capacità) calcola

$$c_{last(h),1} + c_{1,first(k)} - c_{last(h),first(k)}$$

(cioè il risparmio ottenuto fondendo la coppia: elimino l'ultimo arco della prima e il primo della seconda e aggiungo l'arco che collega le due rotte, può essere negativo) e scegli il massimo



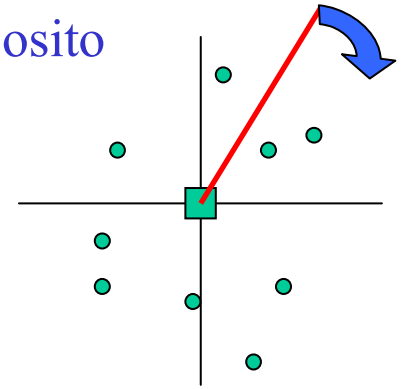
Se il massimo risparmio è ≤ 0 STOP
altrimenti fondi le rotte corrispondenti al massimo, $i=i+1$ e vai al Passo *i*

Se il numero di veicoli è *m*, per avere una sol. ammissibile occorre fondere rotte anche se ho massimo negativo fino ad averne non più di *m*

Metodi Costruttivi Due Fasi

- **Sweep (CFRS):**

Si rappresentano tutti i clienti in un piano dove l'origine è il deposito



Si immagina un raggio uscente dal deposito che ruoti in modo da spazzare tutto il piano

I clienti progressivamente incontrati dal raggio vengono **messi in un cluster** finchè la somma delle loro domande **non supera la capacità**, oltre viene creato un nuovo cluster

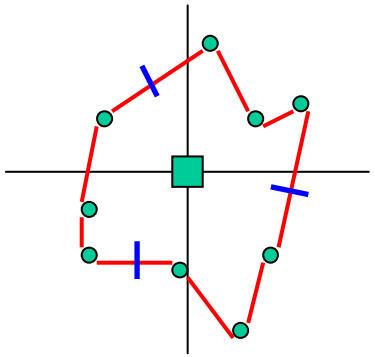
Per ogni cluster si trova poi il routing con una delle euristiche per il TSP

- **Clarke & Wright modificato:**

Il risparmio viene calcolato considerando anche le permutazioni delle due rotte che potrei fondere, e periodicamente riottimizzo le rotte di ogni cluster con una euristica per il TSP

- **RFCS:**

Si risolve **prima il TSP** per tutti i clienti per mezzo di una euristica, poi si **spezza** il ciclo Hamiltoniano trovato in modo da rispettare le capacità dei veicoli



Euristiche Migliorative

- **Ricerca Locale:**

  Calcola un intorno di una soluzione ammissibile (vari possibili criteri: spostamento di cliente da un cluster a un altro, modifica routing)

  Trova la **miglior soluzione dell'intorno** e passa ad essa se conveniente (come visto, ci si arresta in un minimo locale)

- **Tabu Search** (per non arrestarsi in minimi locali)**:**

  Inizializzazione: Definisci una soluzione ammissibile iniziale attraverso un metodo costruttivo

  Iterazione: genera **intorni** eseguendo delle "mosse" (ad esempio spostamento di cliente da un cluster a un altro). Effettuare la "mossa" inversa viene **vietato** (è Tabu) **per un certo numero di passi**

  Ci si sposta in una soluzione dell'intorno accettando anche, sotto opportune condizioni, dei peggioramenti, in modo da non fermarsi su minimi locali

  Il Tabu serve ad evitare di tornare subito sulla soluzione precedente quando ho accettato un peggioramento

  Postottimizzazione: riottimizza il routing di ogni cluster con euristica per TSP

  Ovviamente, tante possibili varianti e miglioramenti dello schema descritto