Modulo 2 del corso di “Ricerca operativa e Logistica” (3 cfu)

Il Modulo 2 del corso di “Ricerca operativa e Logistica” è un possibile approfondimento del Modulo 1 cioè del corso di “Ricerca operativa” (RO).

Come abbiamo già visto, la RO si occupa di modellare e risolvere problemi di ottimizzazione [cioè problemi in cui un decisore deve scegliere una azione, in un certo insieme di azioni ammissibili, che massimizzi una certa utilità] come problemi di Programmazione Matematica; naturalmente ciò non è sempre possibile; in particolare abbiamo visto due classi basilari di problemi di Programmazione Matematica, cioè la Programmazione Lineare (PL) e la Programmazione Lineare Intera (PLI), le quali comunque sembrano costituire un riferimento per molti problemi di ottimizzazione di natura pratica.

Riguardo la Logistica riportiamo di seguito la definizione data dall’Associazione Italiana di Logistica (AILOG): essa è “*l’insieme delle attività organizzative, gestionali e strategiche che governano nell’azienda i flussi di materiali e delle relative informazioni dalle origini presso i fornitori fino alla consegna dei prodotti finiti ai clienti e al servizio post-vendita*”.

Allora nel Modulo 2 proveremo a vedere alcuni problemi di ottimizzazione – in possibili contesti di Logistica – e a modellarli come problemi di PL o di PLI in accordo con la Nota 2 di seguito.

Nel corso di RO abbiamo già visto alcuni problemi di ottimizzazione (con il relativo modello) che potrebbero rientrare in possibili contesti di Logistica; ad esempio: produzione che avviene componendo/decomponendo risorse, trasporti, cammino di costo minimo, programmazione della produzione (cioè gestione delle scorte), localizzazione di impianti.

Nel Modulo 2 vedremo i seguenti problemi:

A. Alcuni esempi
B. Problema del minimo albero ricoprente (Minimum Spanning Tree - MST)
C. Il problema dei trasporti: due generalizzazioni
D. Il problema del commesso viaggiatore (Traveling Salesman Problem - TSP)
E. Il problema della distribuzione (Vehicle Routing Problem - VRP)

Il materiale scritto per il Modulo 2 è dato da: (i) il libro del Prof. Fischetti (cfr. riferimento [2] nel Programma), (ii) questo file con appunti [in cui è presente anche una Appendice, alla fine, con cenni sui grafi], (iii) e un eventuale nuovo file con esercizi.

Come abbiamo già visto in dettaglio, nel file Modelli del corso di RO, una possibile descrizione di una possibile utilità della RO è il seguente schema:

problema di ottimizzazione (informale) $\rightarrow$ modello matematico (formale) $\rightarrow$ risoluzione

che sintetizza quanto segue:

- primo passaggio: si parte da un dato problema di ottimizzazione (in termini informali) e si arriva possibilmente a un suo modello matematico (in termini formali); tale passaggio si ottiene grazie all'abilità e all'esperienza umana;

- secondo passaggio: si parte dal modello matematico di sopra e si arriva possibilmente alla sua risoluzione cioè al calcolo di una "soluzione ottima" del problema di ottimizzazione dato; tale passaggio si ottiene grazie a software specifici.

Allora, come nel corso di RO, alcune ore del Modulo 2 saranno dedicate al secondo passaggio tramite il solutore Excel; per la natura del corso, questo aspetto avrà possibilmente un peso maggiore, nel contesto del corso e (in prospettiva) dell'esame.

Nota 1: Come abbiamo già visto, modellare e risolvere un problema di ottimizzazione come un problema di Programmazione Matematica ha come scopo principale quello di essere un "supporto alle decisioni"; infatti spesso il modello che si produce è solo una approssimazione della realtà; in particolare, la soluzione ottenuta può dare indicazioni sia su come procedere riguardo la decisione direttamente legata al problema, sia su come procedere riguardo decisioni indirettamente legate al problema [ad esempio abbiamo visto che i *prezzi ombra*, nel contesto di una produzione che avviene componendo risorse, individuano direzioni e condizioni per un eventuale sviluppo di quella produzione].

Nota 2: Come abbiamo già visto – in modo approssimativo per il nostro solo scopo, rimandando il lettore a testi di Informatica per una trattazione formale – ricordiamo alcuni cenni di complessità computazionale. I problemi di Programmazione Matematica sono divisi [nei limiti del possibile] in *facili* e *difficili*, in base alla complessità computazionale del migliore algoritmo di soluzione [definito a priori] noto per risolvere una generica istanza del

problema. I problemi facili – detti anche “polinomiali” – comprendono quelli di PL; ad esempio il problema della pianificazione dei progetti è facile; inoltre, anche certi problemi che non sono inizialmente modellati come PL sono infine riconducibili alla PL [e quindi sono facili], tramite risultati sulle matrici TUM; ad esempio i problemi: Assegnamento, Trasporti, Cammino di costo minimo, Massimo Flusso. I problemi difficili – detti anche “NP-hard” – comprendono quelli di PLI; ad esempio il problema di localizzazione degli impianti è difficile. Riguardo i problemi di PL, spesso vengono comunque introdotti “metodi esatti” per trovare una soluzione ottima del problema [senza ricorrere quindi ai metodi usuali per la PL]; questo perché comunque è utile disporre di metodi veloci soprattutto nel caso di grandi istanze del problema. Riguardo i problemi di PLI, spesso vengono introdotti “metodi euristici” per trovare una soluzione presumibilmente buona ma non necessariamente ottima del problema [senza ricorrere quindi ai metodi usuali per la PLI]; questo poiché un metodo usuale per la PLI può essere lento, anche per piccole istanze del problema, così che si può preferire un “metodo euristico” che pur garantendo una soluzione presumibilmente buona ma non necessariamente ottima del problema ha il pregio di essere veloce.

A. Alcuni esempi

Di seguito vediamo sette problemi di ottimizzazione – in possibili contesti di Logistica – che possono essere modellati come problemi di Programmazione Lineare (Intera); in particolare, nel corso di RO abbiamo già visto come modellare situazioni simili, in altri contesti.

A1. Caricamento di un camion

Una Ditta ha a disposizione un camion, per consegnare alcuni oggetti, il quale ha una capacità pari a b in termini di *ingombro* (che può essere riferito al peso o al volume o ad altro). La Ditta dovrebbe consegnare n oggetti, ma il totale dell'ingombro degli oggetti eccede la capacità b, così la Ditta per il primo viaggio del camion dovrà scegliere gli oggetti da consegnare. In particolare, per $i = 1, \ldots, n$, ogni oggetto i ha un *ingombro* a_i (nel senso specificato sopra) e ha una *utilità* u_i (che può essere riferita al valore dell'oggetto o all'utilità per la Ditta riguardo la consegna dell'oggetto).

Il problema è scegliere gli oggetti da consegnare (cioè da caricare sul camion), con il vincolo di ingombro, in modo da massimizzare l'utilità totale degli oggetti scelti.

Variabili decisionali:

x_i per $i = 1, \ldots, n$;

x_i avrà valore 1 se l'oggetto i è scelto;

x_i avrà valore 0 se l'oggetto i non è scelto.

Modello: max $u_1x_1 + u_2x_2 + \ldots + u_nx_n$

$a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$

$x_i \in \{0, 1\}$ per $i = 1, \ldots, n$

A2. Bilanciamento di carico fra camion

Una Ditta ha a disposizione due camion, siano C_1 e C_2, per consegnare n oggetti. Per $i = 1, \ldots, n$, ogni oggetto i ha un *ingombro* a_i (che può essere riferito al peso o al volume o ad altro). Si assuma che i due camion abbiano una capacità comunque sufficiente in termini di ingombro.

Il problema è scegliere gli oggetti da caricare rispettivamente su C_1 e su C_2 in modo da bilanciare il più possibile i rispettivi ingombri totali.

Variabili decisionali:

x_i per $i = 1, \ldots, n$

$x_i = 1$ se l'oggetto i è caricato su C_1

$x_i = 0$ se l'oggetto i non è caricato su C_1 (cioè è caricato su C_2)

Modello: $\min \sum_{i=1}^{n} a_i x_i$

$$\sum_{i=1}^{n} a_i x_i \geq \sum_{i=1}^{n} a_i \ / 2$$

$x_i \in \{0, 1\}$ per $i = 1, \ldots, n$

Di seguito consideriamo una generalizzazione al caso di m camion.

Una Ditta ha a disposizione m camion, siano $C_1, \ldots, C_m$, per consegnare n oggetti. Per $i = 1, \ldots, n$, ogni oggetto i ha un *ingombro* a_i (che può essere riferito al peso o al volume o ad altro). Si assuma che i camion abbiano una capacità comunque sufficiente in termini di ingombro.

Il problema è scegliere gli oggetti da caricare rispettivamente su $C_1, \ldots, C_m$ in modo da bilanciare il più possibile i rispettivi ingombri totali.

Variabili decisionali:

x_{ij} per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

x_{ij} avrà valore 1 se l'oggetto i è caricato sul camion C_j

x_{ij} avrà valore 0 se l'oggetto i non è caricato sul camion C_j

y_{max} = ingombro totale del camion che avrà il massimo ingombro totale

y_{min} = ingombro totale del camion che avrà il minimo ingombro totale

Modello: $\min \ y_{max} - y_{min}$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \text{per } i = 1, \ldots, n \quad \text{(ogni oggetto è caricato)}$$

$$\sum_{i=1}^{n} a_i x_{ij} \leq y_{max} \qquad \text{per } j = 1, \ldots, m$$

$$\sum_{i=1}^{n} a_i x_{ij} \geq y_{min} \quad \text{per } j = 1, \ldots, m$$

$$x_i \in \{0, 1\} \quad \text{per } i = 1, \ldots, n$$

A3. Utilizzo del minimo numero di camion

Una Ditta ha a disposizione m camion, siano 1, …, m, per consegnare n oggetti. Per $i = 1, \ldots, n$, ogni oggetto i ha un *ingombro* a_i (che può essere riferito al peso o al volume o ad altro). Per $j = 1, \ldots, m$, ogni camion j ha una capacità K_j in termini di ingombro.

Il problema è stabilire quali camion utilizzare (per questa consegna), con i vincoli di ingombro, in modo da minimizzare il numero totale dei camion utilizzati.

Variabili decisionali:

x_{ij} per $i = 1, \ldots, n$, e per $j = 1, \ldots, m$

$x_{ij} = 1$ se l'oggetto i è caricato sul camion j

$x_{ij} = 0$ altrimenti

y_j per $j = 1, \ldots, m$;

$y_j = 1$ se il camion j è utilizzato

$y_j = 0$ altrimenti.

Modello: min $\sum_{j=1}^{m} y_j$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \text{per } i = 1, \ldots, n \quad \text{(ogni oggetto è caricato)}$$

$$\sum_{i=1}^{n} a_i x_{ij} \leq K_j \quad \text{per } j = 1, \ldots, m$$

$$\sum_{i=1}^{n} x_{ij} \leq M y_j \quad \text{per } j = 1, \ldots, m$$

(dove M è uno scalare molto grande)

$$x_{ij} \in \{0, 1\} \quad \text{per } i = 1, \ldots, n \text{ e per } j = 1, \ldots, m$$

$$y_j \in \{0, 1\} \quad \text{per } j = 1, \ldots, m$$

A4. Rete di distribuzione - ridefinizione

Una Ditta rileva una rete di distribuzione (già attiva) di un certo bene, formata da m magazzini che riforniscono n negozi, in particolare ogni negozio è rifornito da un solo magazzino.

Problema: la Ditta vuole ridefinire i magazzini della rete, cioè, vuole studiare la possibilità di ridurre il numero dei magazzini così da confermarne (attivi) solo alcuni. In particolare: riguardo i costi di conferma dei magazzini e i costi di rifornimento dai magazzini ai negozi, la Ditta considera un orizzonte temporale fissato [ad esempio di tre anni e mezzo]; riguardo le capacità dei magazzini e le richieste dei negozi, la Ditta considera un orizzonte temporale di un mese.

Parametri:

:: $M = \{1, \ldots, m\}$ = insieme dei magazzini

:: $N = \{1, \ldots, n\}$ = insieme dei negozi

:: d_j = capacità mensile del magazzino j

:: r_i = richiesta mensile del negozio i

:: C_j = costo per la conferma del magazzino j (nell'orizzonte temporale fissato)

:: c_{ij} = costo per il rifornimento dal magazzino j al negozio i (nell'orizzonte temporale fissato)

Variabili decisionali:

x_{ij} per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

x_{ij} = 1 se il negozio i è rifornito dal magazzino j

x_{ij} = 0 altrimenti

y_j per $j = 1, \ldots, m$

y_j = 1 se il magazzino j è da confermare

y_j = 0 altrimenti

Modello: $\min \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} + \sum_{j=1}^{m} C_j y_j$

$\sum_{j=1}^{m} x_{ij} = 1$ per $i = 1, \ldots, n$ (ogni negozio è rifornito da un magazzino)

$\sum_{i=1}^{n} r_i x_{ij} \leq d_j$ per $j = 1, \ldots, m$

$$\sum_{i=1}^{n} x_{ij} \leq M\, y_j \qquad \text{per } j = 1, \ldots, m$$

(dove M è uno scalare molto grande)

$x_{ij} \in \{0, 1\}$ per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

$y_i \in \{0, 1\}$ $j = 1, \ldots, m$

A5. Rete di servizio - ridefinizione

Una Regione vuole ridefinire i reparti della sua rete ospedaliera a seguito di tagli di risorse.

Per semplicità fissiamo un solo tipo di reparto, ad esempio, il tipo di reparto di Ortopedia.

Parametri:

:: la Regione fissa un orizzonte temporale, ad esempio, di un anno;

:: $D = \{D_1, \ldots, D_n\}$ = insieme di distretti [indicativi] in cui è divisa la Regione;

:: $R = \{R_1, \ldots, R_m\}$ = insieme dei reparti di Ortopedia della Regione;

:: d_i = numero di pazienti (attesi) che si stima siano da ricoverare in uno fra i reparti in R (cioè in un reparto di Ortopedia) dal distretto D_i (per $i = 1, \ldots, n$) in un anno;

:: r_j = numero massimo di pazienti che si stima sia possibile ricoverare nel reparto R_j (per $j = 1, \ldots, m$) in un anno;

:: C_j = "costo fisso" = costo stimato per la sola conferma del reparto R_j (per $j = 1, \ldots, m$) in un anno;

:: c_{ij} = "costo marginale" = costo stimato per ricoverare un paziente del distretto D_i (per $i = 1, \ldots, n$) nel reparto R_j (per $j = 1, \ldots, m$) [tale costo può essere ad esempio inteso come $c_j + q_{ij}$ dove: c_j è il costo stimato sostenuto dalla Regione per ricoverare nel reparto R_j un generico paziente, e q_{ij} è il costo stimato sostenuto dal paziente del distretto D_i per essere ricoverato nel reparto R_j].

Problema: individuare quali reparti confermare fra $R_1, \ldots, R_m$, con il vincolo di una gestione dei ricoveri che garantisca a ogni paziente (atteso) il ricovero presso uno fra i reparti in R, in modo da minimizzare la somma dei "costi fissi" e dei "costi marginali".

Variabili decisionali:

x_{ij} per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

x_{ij} = numero di pazienti (attesi) ricoverati dal distretto D_i nel reparto R_j

y_j per $j = 1, \ldots, m$

$y_j = 1$ se il reparto R_i è da confermare

$y_j = 0$ altrimenti

Modello: min $\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} + \sum_{j=1}^{m} C_j y_j$

$\sum_{j=1}^{m} x_{ij} \geq d_i$ per $i = 1, \ldots, n$

$\sum_{i=1}^{n} x_{ij} \leq r_j$ per $j = 1, \ldots, m$

$\sum_{i=1}^{n} x_{ij} \leq M y_j$ per $i = 1, \ldots, n$

(dove M è uno scalare molto grande)

$x_{ij} \geq 0$ per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

$y_j \in \{0, 1\}$ per $j = 1, \ldots, m$

A6. Fornitura – ridefinizione

Uno Stato del Medio Oriente vuole ridefinire la sua fornitura di gas al fine di colmare un deficit di fornitura dovuto a cause di forza maggiore. In particolare lo Stato vuole definire un piano, per la sua fornitura, su un orizzonte temporale di n mesi tenendo conto che può acquistare il gas da m (possibili) fornitori.

Parametri:

:: lo Stato fissa un orizzonte temporale di n mesi;

:: $M = \{M_1, \ldots, M_n\}$ = insieme dei mesi dell'orizzonte temporale;

:: $F = \{F_1, \ldots, F_m\}$ = insieme dei (possibili) fornitori;

:: d_i = quantità di gas necessarie allo Stato nel mese M_i (per $i = 1, \ldots, n$);

:: b_{ij} = quantità massima di gas acquistabile nel mese M_i (per $i = 1, \ldots, n$) dal fornitore F_j (per $j = 1, \ldots, m$) [ad esempio, riguardo un fornitore per il quale è necessaria la costruzione di un gasdotto, tale quantità massima può essere nulla il primo mese];

:: C_j = "costo fisso" = costo fisso stimato nel caso di acquisto di una qualsiasi quantità non-nulla di gas dal fornitore F_j (per $j = 1, \ldots, m$) nel corso dell'orizzonte temporale [ad esempio per la costruzione di un gasdotto];

:: c_{ij} = "costo marginale" = costo marginale stimato per l'acquisto di una unità di gas nel mese M_i (per $i = 1, \ldots, n$) dal fornitore F_j (per $j = 1, \ldots, m$).

Il problema è scegliere le quantità di gas da acquistare ogni mese presso ogni fornitore, con il vincolo di necessità, al fine di minimizzare il totale dei costi.

Variabili decisionali:

x_{ij} per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

x_{ij} = unità di gas da acquistare nel mese M_i (per $i = 1, \ldots, n$) dal fornitore F_j (per $j = 1, \ldots, m$)

y_i per $j = 1, \ldots, m$

$y_i = 1$ se il (possibile) fornitore F_j è scelto come fornitore

$y_i = 0$ altrimenti

Modello: min $\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} + \sum_{j=1}^{m} C_j y_j$

$\sum_{j=1}^{m} x_{ij} \geq d_i$ per $i = 1, \ldots, n$

$x_{ij} \leq b_{ij}$ per $j = 1, \ldots, m$

$\sum_{i=1}^{n} x_{ij} \leq M y_j$ per $i = 1, \ldots, n$

(dove M è uno scalare molto grande)

$x_{ij} \geq 0$ per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

$y_j \in \{0, 1\}$ per $j = 1, \ldots, m$

Nota : il modello dell'esempio A6 è (quasi) identico al modello dell'esempio A5.

A7. Un viaggio in autostrada

Questo esempio è tratto liberamente dal riferimento https://slideplayer.it/slide/935618/ del Prof. Claudio Arbib (si consiglia di vederlo).

Una persona che abitualmente va in automobile (via autostrada) da Pescara a Roma vuole minimizzare il consumo del carburante fissando comunque un tempo massimo di percorrenza. Allora osservando che il consumo di carburante dipende dalla pendenza (della strada) e dalla velocità (della macchina), pensa di partizionare il percorso (da casello a casello) in n tratti in modo che ogni tratto abbia una pendenza uniforme, e di considerare per semplicità solo m possibili velocità (ad esempio 70 km/h, 80 Km/h, …, 130 Km/h). Infine calcola e fissa i seguenti parametri:

:: sia c_{ij} il consumo/costo per percorrere il tratto i (per $i = 1, \ldots, n$) a velocità j (per $j = 1, \ldots, m$);

:: sia t_{ij} il tempo per percorrere il tratto i (per $i = 1, \ldots, n$) a velocità j (per $j = 1, \ldots, m$);

:: sia T il tempo massimo di percorrenza fissato.

Il problema è stabilire per ogni tratto a quale velocità andare, con il vincolo del tempo massimo di percorrenza, in modo da minimizzare il consumo/costo totale.

Variabili decisionali:

x_{ij} per $i = 1, \ldots, n$ e per $j = 1, \ldots, m$

$x_{ij} = 1$ se il tratto i è percorso a velocità j

$x_{ij} = 0$ altrimenti

Modello: min $\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij}$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \text{per } i = 1, \ldots, n$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} x_{ij} \leq T$$

$$x_{ij} \in \{0, 1\} \qquad \text{per } i = 1, \ldots, n \text{ e per } j = 1, \ldots, m$$

B. Il problema del minimo albero ricoprente (Shortest Spanning Tree - SST)

Per favore, il riferimento scritto è il libro del Prof. Fischetti (cfr. il riferimento [2] del Programma) cercando nell'indice "Alberi ci costo minimo: Algoritmi efficienti, Algoritmo di Kruskal.", grazie.

Si consiglia comunque di vedere la dispensa relativa al corso del Prof. Vincenzo Acciaro:

https://www.sci.unich.it/~acciaro/MinimoAlberoRicoprente.pdf

Il problema SST sembra avere molte applicazioni di natura pratica.
Un esempio classico è quello di definire una rete elettrica per un insieme di caseggiati (o in generale di località) che dovranno essere tutti riforniti da una centrale elettrica; allora la centrale elettrica e i caseggiati possono essere visti come i vertici di un grafo, i cui archi rappresentano la possibilità (con relativo costo) di un collegamento diretto per i corrispondenti vertici; così il problema di definire una rete elettrica a costo minimo è proprio il problema SST.
Similmente, altre applicazioni possono essere ipotizzate per la definizione di reti, come ad esempio la definizione di una prima rete di autostrade o di una prima rete di linee ferrovie ad alta velocità.

C. Il problema dei trasporti: due generalizzazioni

Riportiamo il problema dei trasporti, che abbiamo già visto nel corso di RO, insieme con due generalizzazioni: in particolare il problema dei traporti è *facile* [come abbiamo già visto] e queste due generalizzazioni sono ancora problemi *facili*.

Problema dei trasporti

Ci sono r depositi (o impianti di produzione) e s outlet. Ogni deposito $F_i \in \{F_1, \ldots, F_r\}$ ha a disposizione $A_i \geq 0$ unità di un certo bene, e ogni outlet $O_j \in \{O_1, \ldots, O_s\}$ richiede almeno $B_j \geq 0$ unità dello stesso bene. Per ogni coppia (F_i, O_j), per $i \in \{1, \ldots, r\}$ e per $j \in \{1, \ldots, s\}$, è inoltre stabilito il costo unitario di trasporto c_{ij}.

Il problema è pianificare i trasporti in modo da minimizzare il costo totale.

Variabili decisionali:

x_{ij} per $i = 1, \ldots, r$ e per $j = 1, \ldots, s$

x_{ij} indica la quantità di bene trasportata dal deposito F_i alla destinazione O_j

Modello: min $\sum_{i=1}^{r}\sum_{j=1}^{s} c_{ij} x_{ij}$

$\sum_{j=1}^{s} x_{ij} \leq A_i$ per $i = 1, \ldots, r$ (vincoli di disponibilità)

$\sum_{i=1}^{r} x_{ij} \geq B_j$ per $j = 1, \ldots, s$ (vincoli di richiesta)

$x_{ij} \geq 0$ per $i = 1, \ldots, r$ e per $j = 1, \ldots, s$

x_{ij} intero per $i = 1, \ldots, r$ e per $j = 1, \ldots, s$ (eventualmente)

Di seguito riportiamo (in inglese) due generalizzazioni del problema dei trasporti:

:: la prima è il “Flusso a costo minimo (Min-Cost-Flow)” che si presta a molte applicazioni di natura pratica, sia nella sua forma diretta, come generalizzazione del problema dei trasporti in cui il collegamento fra i depositi e gli outlet non è necessariamente diretto ma è costituito da una rete, sia nella sua forma più astratta, come ad esempio il processo di raccolta e di smaltimento dei rifiuti;

:: la seconda è una generalizzazione multi-periodo del problema dei trasporti; essa è essenzialmente un esercizio di teoria, tuttavia può avere verosimili applicazioni di natura

pratica, considerando che i costi di traporto sono spesso stabiliti in anticipo e dipendono spesso dalla variabile tempo (come ad esempio i prezzi dei biglietti aerei).

Ricordiamo che: "(Hitchcock-) transportation problem" è il problema dei trasporti, "digraph" è un grafo orientato, "vertex" è un vertice, "edge" è un arco.

C1. Flusso a costo minimo (Min-Cost-Flow)

A network $N = (\sigma, \tau, V, E, w, c)$ is a digraph (V, E) together with a source $\sigma \in V$ with 0 indegree, with a terminal $\tau \in V$ with 0 outdegree, with an edge-capacity function $w : E \to \mathbb{R}$, and with an edge-cost function $c : E \to \mathbb{R}$.
A flow f in N is a vector in $\mathbb{R}^{|E|}$ [one component $f(u, v)$ for each $(u, v) \in E$] such that:

(i) $0 \leq f(u, v) \leq w(u, v)$ for every $(u, v) \in E$

(ii) $\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$ for every $v \in V \setminus \{\sigma, \tau\}$.

The *value* of f is the quantity: $\sum_{(\sigma,y)\in E} f(\sigma, y)$.

The *cost* of f is the quantity: $\sum_{(u,v)\in E} c(u,v)\, f(u,v)$

Given a network N and a flow-value v_f, the *Min-Cost Flow problem* with respect to pair (N, v_f) is that of computing in N a flow f of value v_f that has minimum cost.

Let us show that (Hitchcock-) transportation problem can be represented as a Min-Cost-Flow problem as follows with reference to a generic instance of the (Hitchcock-) transportation problem [cfr. la definizione del "Problema dei trasporti"]:

- construct digraph G such that:

$G = (V, E)$ where

$V = \{a_i : i = 1, \ldots, r\} \cup \{b_j : j = 1, \ldots, s\}$;

$E = \{(a_i, b_j) : i = 1, \ldots, , r\ ; j = 1, \ldots, s\}$;

Comment: vertex a_i represents F_i , for $i = 1, \ldots, r$; vertex b_j represents O_j , for $j = 1, \ldots, s$; edge (a_i, b_j) represents the possibility of transporting good from F_i to O_j , for $i = 1, \ldots, r$, for $j = 1, \ldots, s$.

- add vertex σ and edges $\{(\sigma, a_i) : i = 1, \ldots, r\}$;

- add vertex τ and edges $\{(b_j, \tau) : j = 1, \ldots, s\}$;

- the elements of V and E are defined as above;

- the function $w : E \to \mathbb{R}$ of edge-capacities is defined as follows:

$w(\sigma, a_i) = A_i$, for $i = 1, \ldots, r$;

$w(b_j, \tau) = B_j$, for $j = 1, \ldots, s$;

$w(e) = \infty$, for the remaining edges $e \in E$;

- the function $c : E \to \mathbb{R}$ of edge-costs is defined as follows:

$c(a_i, b_j) = c_{ij}$, for $i = 1, \ldots, r$, for $j = 1, \ldots, s$

$c(e) = 0$, for the remaining edges $e \in E$;

Then (Hitchcock-) transportation problem can be solved as a Min-Cost-Flow problem with

$$v_f = \sum_{j=1}^{s} B_j$$

C2. Una generalizzazione multi-periodo del problema dei trasporti

Di seguito riportiamo una parte dell'articolo:
"A time expanded version of the (Hitchcock-) transportation problem"

Abstract

Certain quantities of goods are produced in the production facilities in each period of a planning horizon and also the demand values of the outlets are different for each time period. Moreover, the transportation costs from i to j also vary over time. The different planning periods are connected by the fact that goods may be stored at a facility or at an outlet to exploit cheaper transportation costs as long as the demand is met in every time period. We

describe the problem by two models, showing that it can be solved in polynomial time by standard software packages. Then we describe some generalizations of the problem by adapting the two models, pointing out that each model seems to provide different benefits.

Introduction

The (Hitchcock-) transportation problem is a well-known problem in operations research and can be solved in polynomial time by fast algorithms. By its applicative nature, several versions of this problem have been considered, such as dynamic, time-dependent versions.

Let P be the following problem: Given r production facilities $F_1,\ldots, F_r$, let A_{ik} ($i = 1, \ldots, r$, and $k = 1, \ldots, t$) denote the number of units of an indivisible good produced by F_i during the k-th period of a planning horizon of t periods. Assume that there are s outlets $O_1,\ldots, O_s$, each one demanding a certain number of units per period, say B_{ik} ($i = 1, \ldots, r$, and $k = 1, \ldots, t$). Let $c_{ijk} \geq 0$ be the cost associated with transporting one unit from F_i to O_j, during the k-th period, let $h_{ik} \geq 0$ (let $h'_{jk} \geq 0$) be the cost associated with storing one unit in F_i (in O_j) at the end of the k-th period. Then one wishes to minimize the total cost of transportation and of storage along the planning horizon of t periods.

Let us denote as:
x_{ijk} the amount of good which is sent from F_i to O_j in the k-th period;
z_{ik} the amount of good which is stored in F_i at the end of the k-th period;
z'_{ik} the amount of good which is stored in O_j at the end of the k-th period.

A *solution* of P is determined by the values of
x_{ijk} , z_{ik} , z'_{ik} for $i = 1, \ldots, r$, for $j = 1, \ldots, s$, for $k = 1, \ldots, t$.

A Min-Cost Flow model for P

A network $N = (\sigma, \tau, V, E, w, c)$ is a digraph (V, E) together with a source $\sigma \in V$ with 0 indegree, with a terminal $\tau \in V$ with 0 outdegree, with an edge-capacity function $w : E \to \mathbb{R}$, and with an edge-cost function $c : E \to \mathbb{R}$.
A flow f in N is a vector in $\mathbb{R}^{|E|}$ (one component $f(u, v)$ for each $(u, v) \in E$) such that:

(i) $0 \leq f(u, v) \leq w(u, v)$ for every $(u, v) \in E$

(ii) $$\sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u) \text{ for every } v \in V \setminus \{\sigma, \tau\}.$$

The *value* of f is the quantity: $\sum_{(\sigma,y)\in E} f(\sigma, y)$.

The *cost* of f is the quantity: $\sum_{(u,v)\in E} c(u,v)\, f(u,v)$

Given a network N and a flow-value v_f, the *Min-Cost Flow problem* with respect to pair (N, v_f) is that of computing in N a flow f of value v_f that has minimum cost.

In the sequel, let us try to model P as a Min-Cost-Flow problem in the following network, which is constructed by considering that P implicitly contains t instances of the Hitchcock transportation problem and by adding an origin vertex, a destination vertex, and edges for the links between consecutive periods (see Figure 1):

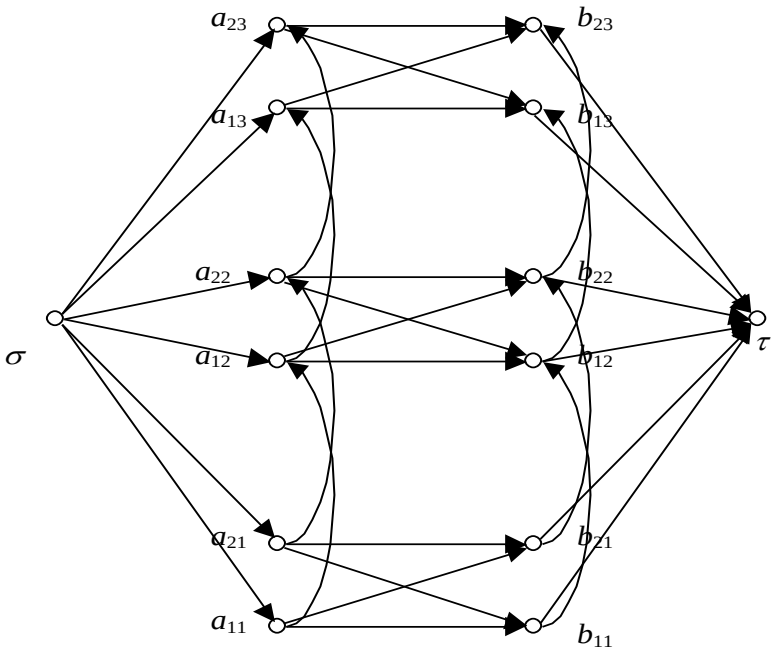

Figure 1

- construct t disjoint digraphs $G_1, \ldots, G_t$ such that:

$G_k = (V_k, E_k)$, for $k = 1, \ldots, t$, where

$V_k = \{a_{ik} : i = 1, \ldots, r\} \cup \{b_{jk} : j = 1, \ldots, s\}$;

$E_k = \{(a_{ik}, b_{jk}) : i = 1, \ldots, , r\ ; j = 1, \ldots, s\}$;

Comment: vertex a_{ik} represents F_i at period k, for $i = 1, \ldots, r$, for $k = 1, \ldots, t$; vertex b_{jk} represents O_j at the period k, for $j = 1, \ldots, s$, for $k = 1, \ldots, t$; edge (a_{ik}, b_{jk}) represents the possibility of transporting good from F_i to O_j at period k, for $i = 1, \ldots, r$, for $j = 1, \ldots, s$, for $k = 1, \ldots, t$.

- add vertex σ and edges $\{(\sigma, a_{ik}) : i = 1, \ldots, r\ ; k = 1, \ldots, t\ \}$;
- add vertex τ and edges $\{(b_{jk}, \tau) : j = 1, \ldots, s\ ; k = 1, \ldots, t\ \}$;
- add edges $\{(a_{ik}, a_{ik+1}) : i = 1, \ldots, r\ ; k = 1, \ldots, t\ \}$;
- add edges $\{(\ b_{jk}, b_{jk+1}) : j = 1, \ldots, s\ ; k = 1, \ldots, t\ \}$;
- the elements of V and E are defined as above;
- the function $w : E \to \mathbb{R}$ of edge-capacities is defined as follows:

$w(\sigma, a_{ik}) = A_{ik}$, for $i = 1, \ldots, r$, for $k = 1, \ldots, t$;

$w(b_{jk}, \tau) = B_{jk}$, for $j = 1, \ldots, s$, for $k = 1, \ldots, t$;

$w(e) = \infty$, for the remaining edges $e \in E$;

- the function $c : E \to \mathbb{R}$ of edge-costs is defined as follows:

$c(a_{ik}, b_{jk}) = c_{ijk}$, for $i = 1, \ldots, r$, for $j = 1, \ldots, s$, for $k = 1, \ldots, t$;

$c(a_{ik}, a_{ik+1}) = h_{ik}$, for $i = 1, \ldots, r$, for $k = 1, \ldots, t$;

$c(b_{jk}, b_{jk+1}) = h_{ik}$, for $j = 1, \ldots, s$, for $k = 1, \ldots, t$;

$c(e) = 0$, for the remaining edges $e \in E$;

Then Problem P can be solved as a Min-Cost-Flow problem with

$$v_f = \sum_{j=1}^{s} \sum_{k=1}^{t} B_{jk}$$

D. Il problema del commesso viaggiatore (Traveling Salesman Problem - TSP)

Questi appunti sono tratti liberamente dal libro del Prof. Sassano (cfr. il riferimento [4] del Programma) e dal riferimento http://groups.di.unipi.it/optimize/Courses/RO2IG/aa1415/8-tsp_vrp.pdf *della Prof.ssa Laura Galli (si consiglia di vederli).*

Riguardo questo problema, in letteratura esistono diverse definizioni (più o meno generali), così di seguito vediamo una definizione forse la più generale possibile.

Ricordiamo che un *cammino* di un grafo G è una sequenza di archi $e_1, ..., e_k$ di G consecutivi, cioè del tipo $e_1 = (v_1, v_2)$, $e_2 = (v_2, v_3)$, ..., $e_k = (v_k, v_{k+1})$; tale cammino *collega* v_1 a v_{k+1}; in particolare, se $v_1 \equiv v_{k+1}$ allora esso è detto *ciclo*; notiamo che, in base a tale definizione, un cammino (così come un ciclo) può passare più volte per uno stesso vertice.

Definizione del TSP: Sia $G = (V, E)$ un grafo orientato, tale che per ogni coppia di vertici esiste un cammino orientato fra loro, e sia $c_{ij} \geq 0$ una assegnazione di costi per ogni arco (i, j) di G. Il problema è determinare un ciclo orientato di costo minimino [dove il costo di un ciclo è dato dalla somma dei costi degli archi che lo compongono] che passi per tutti i vertici di G.

Tale problema è NP-hard, cioè, *difficile* (cfr. Introduzione).

Nell'Introduzione abbiamo visto che determinare il cammino di costo minimo fra due vertici di un grafo è un problema *facile*. Allora, ricordando che un grafo si dice *completo* se contiene tutti gli archi possibili [cioè, per ogni $i, j \in V$, si ha $(i, j) \in E$], il TSP è spesso ri-definito come segue, intendendo ogni peso c^*_{ij} come il costo di un cammino di costo minimo fra i vertici i e j nel grafo originario G :

Definizione (alternativa) del TSP: Sia $G^* = (V, E^*)$ un grafo orientato completo e sia $c^*_{ij} \geq 0$ una assegnazione di costi per ogni arco (i, j) di G^*. Il problema è determinare un ciclo orientato di costo minimino [dove il costo di un ciclo è dato dalla somma dei costi degli archi che lo compongono] che passi per tutti i vertici di G.

Nel seguito vedremo due modelli di Programmazione Lineare Intera per il TSP [i quali conducono a una soluzione esatta del problema] e tre metodi euristici per il TSP [i quali conducono a una soluzione euristica del problema].

Modelli di PLI per il TSP

Modello 1

Variabili decisionali:

x_{ij} per ogni $(i, j) \in E$

$x_{ij} = 1$ se l'arco (i, j) appartiene al ciclo

$x_{ij} = 0$ altrimenti

Modello: min $\sum_{(i,j)\in E} c_{ij} x_{ij}$

$$\sum_{(i,j)\in E} x_{ij} = 1 \quad \text{per ogni } i \in V \qquad (1)$$

$$\sum_{(j,i)\in E} x_{ij} = 1 \quad \text{per ogni } i \in V \qquad (2)$$

$$\sum_{(i,j)\in E: i\in S, j\notin S} x_{ij} \geq 1 \quad \text{per ogni } S \subseteq V, \text{ con } S \neq \emptyset, V \qquad (*)$$

$$x_{ij} \in \{0, 1\} \quad \text{per ogni } (i, j) \in E$$

Correttezza: la correttezza del modello è abbastanza evidente; infatti, i gruppi (1) e (2) di vincoli garantiscono che per ogni vertice esiste (con riferimento al ciclo cercato) esattamente un arco entrante ed esattamente un arco uscente, il gruppo (*) di vincoli garantisce che la soluzione trovata non sia una collezione di più cicli disgiunti ma sia un unico ciclo.

Nota bene: il gruppo (*) di vincoli è formato da un numero esponenziale di vincoli, cioè, da $2^{|V|-1} - 2$ vincoli; questo è un inconveniente nel caso in cui una persona dovesse implementare il Modello 1 su un solutore; il Modello 2, introdotto di seguito, eliminerà questo inconveniente.

<u>Modello 2</u>

Variabili decisionali:

x_{ij} per ogni $(i, j) \in E$

$x_{ij} = 1$ se l'arco (i, j) appartiene al ciclo

$x_{ij} = 0$ altrimenti

u_i per ogni $i \in V$

$u_i \geq 1$ e $u_i = k$ se i è il k-esimo vertice visitato nel ciclo

Modello: min $\sum_{(i,j)\in E} c_{ij} x_{ij}$

$$\sum_{(i,j)\in E} x_{ij} = 1 \quad \text{per ogni } i \in V \qquad (1)$$

$$\sum_{(j,i)\in E} x_{ij} = 1 \quad \text{per ogni } i \in V \qquad (2)$$

$$|V|\, x_{ij} + u_i - u_j \leq |V| - 1 \quad \text{per ogni } (i, j) \in E, \text{ con } j \neq 1 \qquad (**)$$

$u_1 = 1$
$2 \leq u_i \leq |V|$ per ogni $i \in V$, con $i \neq 1$
$x_{ij} \in \{0, 1\}$ per ogni $(i, j) \in E$

Correttezza: la correttezza del modello non è evidente (la vediamo di seguito); i gruppi (1) e (2) di vincoli garantiscono che per ogni vertice esiste (con riferimento al ciclo cercato) esattamente un arco entrante ed esattamente un arco uscente; resta il problema di garantire che la soluzione trovata non sia una collezione di più cicli disgiunti ma sia un unico ciclo; vediamo di seguito che il gruppo (**) di vincoli risolve tale problema:
:: come preliminare osserviamo che:
se $x_{ij} = 0$, allora il corrispondente vincolo in (**) diventa $u_i - u_j \leq |V| - 1$, cioè diventa un vincolo che è sempre soddisfatto [considerando gli altri vincoli sulle variabili u_i];
se $x_{ij} = 1$, allora il corrispondente vincolo in (**) diventa $u_i - u_j \leq -1$, cioè $u_j \geq u_i + 1$;
:: da un lato, se la soluzione è un unico ciclo, sia formato dai nodi 1, 2, …, $|V|$ in questa sequenza (senza perdita di generalità) cioè individuato da $x_{12} = 1$, $x_{23} = 1$, …, $x_{|V|1} = 1$, allora i valori $u_1 = 1$, $u_2 = 2$, …, $u_{|V|} = |V|$ garantiscono che la soluzione è ammissibile [infatti per ciascuna di tali variabili il corrispondente vincolo in (**) è soddisfatto];
:: dall'altro lato, se la soluzione non è un unico ciclo [cioè è una collezione di cicli disgiunti], allora esiste un ciclo sia C della soluzione che non contiene il vertice 1; cioè il ciclo C è individuato dalle variabili $x_{ij} = 1$ per ogni arco (i, j) di C; allora la soluzione non è ammissibile [infatti, per ciascuna di tali variabili si ottengono le corrispondenti disequazioni $u_j \geq u_i + 1$ che messe a sistema producono una incompatibilità cioè una contraddizione, considerati i vincoli sulle variabili u_i e il fatto che il vertice 1 non appartiene al ciclo C].

Nota bene: il gruppo (**) di vincoli è formato da un numero polinomiale di vincoli, cioè, da al più $|V|^2$ vincoli; questo risolve l'inconveniente osservato per il Modello 1 e fa sì che il Modello 2 [nonostante introduca $|V|$ variabili in più rispetto al Modello 1] risulti il modello da utilizzare nel caso una persona dovesse implementare un modello per il TSP su un solutore [per avere una idea si può fissare ad esempio $|V| = 10$ e simulare (in astratto) una implementazione dei due modelli].

<u>Osservazione – Rilassamento:</u>

Osserviamo che un *rilassamento* del Modello 1 e del Modello 2 è dato dal seguente problema:

Modello: min $\sum_{(i,j)\in E} c_{ij} x_{ij}$

$$\sum_{(i,j)\in E} x_{ij} = 1 \quad \text{per ogni } i \in V \qquad (1)$$

$$\sum_{(j,i)\in E} x_{ij} = 1 \quad \text{per ogni } i \in V \qquad (2)$$

$$x_{ij} \in \{0, 1\} \quad \text{per ogni } (i, j) \in E$$

che corrisponde a un problema di Assegnamento, cioè, a un problema *facile* [cfr. Introduzione].
La soluzione ottima, sia *S*, di tale *rilassamento* fornisce comunque un *lower bound* sulla soluzione ottima [con riferimento al costo di *S*]; inoltre: se *S* è un unico ciclo, allora *S* è anche soluzione ottima del TSP; se *S* non è un unico ciclo, allora *S* può essere utilizzata come punto di partenza, per trovare una (buona) soluzione ammissibile del TSP, nel contesto di un metodo euristico.

Metodi euristici per il TSP

Di seguito vediamo tre metodi euristici per il TSP. Tali metodi sono via via più elaborati. In particolare, assumiamo che per ognuno di questi metodi l'input sia una istanza del TSP in accordo con la definizione alternativa, cioè assumiamo che per ognuno di questi metodi si abbia:

Input: un grafo completo $G = (V, E)$, con $V = \{v_1, \ldots, v_n\}$, e con $c_{ij} \geq 0$ per ogni $(i, j) \in E$.
Output: un ciclo di *G* che passa per tutti i vertici di *G*.

Allora, per definizione dell'Input, cerchiamo un ciclo che passa una sola volta per ogni vertice di *G*.

• Metodo del vertice più vicino

Si parte da un vertice qualsiasi e lo si marca (nel senso che si memorizza il fatto che il vertice è stato visitato); ad ogni iterazione si procede verso il vertice più vicino [in dettaglio, un vertice più vicino, nel senso che possono esserci più vertici alla stessa distanza] fra quelli non marcati; infine ci si ferma quando tutti i vertici sono marcati.

begin
--- scegli un qualsiasi vertice di *G*, sia *a*, e poni $\pi[1] := a$
--- **for** $i = 1$ to $n - 1$ **do**
--- --- scegli un vertice $v \in V \setminus \{\pi[1], \ldots, \pi[i]\}$ tale che $c_{\pi[i]v}$ sia minimo e poni $\pi[i+1] := v$
end

• Metodo dell'inserimento graduale

Si considera all'inizio un qualsiasi ciclo di G formato da tre vertici di G; poi si itera il seguente procedimento: inserisci un nuovo vertice in tale ciclo (ad esempio, un vertice che minimizza il costo del suo inserimento, come vedremo) in modo da ottenere un nuovo ciclo con un vertice in più di G; infine ci si ferma quando tutti i vertici sono stati inseriti.

begin
--- sia C un qualsiasi ciclo di G che passa per tre vertici di G con vertici $\pi[1]$, $\pi[2]$, $\pi[3]$;
--- **for** $i = 3$ to $n - 1$ **do**
--- **begin**
--- --- sia $(c_{\pi[j]v} + c_{v\pi[j+1]}) - c_{\pi[j]\pi[j+1]} = argmin \{ [(c_{\pi[j]v} + c_{v\pi[j+1]}) - c_{\pi[j]\pi[j+1]}] :$
$v \in V \setminus \{\pi[1], \ldots, \pi[i]\}, j \in \{1, \ldots, i\} \text{ (mod } i)\}$
--- --- poni $\pi[j+1] := v$ (cioè inserisci v)
--- --- poni $\pi[j+2] := \pi[j+1]$; $\pi[j+3] := \pi[j+2]$; …, $\pi[i+1] := \pi[i]$
(cioè ridefinisci il nuovo ciclo con i +1 vertici)
--- end
end

Notiamo che questo metodo può essere comunque "personalizzato" ad esempio fissando una particolare scelta del ciclo iniziale oppure fissando un alternativo criterio di inserimento.

• Metodo 2-Opt

Si considera all'inizio un qualsiasi ciclo di G che passa per tutti i vertici di G; poi si itera il seguente procedimento: prova a sostituire due archi (non-consecutivi) di tale ciclo con altri due archi del grafo in modo da ottenere un nuovo ciclo che passa per tutti i vertici di G e che sia di costo minore; infine ci si ferma quando questa operazione non è più possibile.

begin
--- sia C un qualsiasi ciclo di G che passa per tutti i vertici di G (ad esempio $v_1, \ldots, v_n$);
--- **while** esistono $(i, j), (k, l) \in C$ tali che $c_{ij} + c_{kl} > c_{il} + c_{kj}$ **do**
--- --- $C := C \setminus \{(i, j), (k, l)\} \cup \{(i, j), (k, j)\}$
--- **endwhile**
end

Osservazione – Rilassamento:

In accordo con la precedente osservazione sul rilassamento del Modello 1 e del Modello 2, riportiamo di seguito un metodo euristico detto "Algoritmo delle toppe", senza entrare nei dettagli (solo per dare una idea):

Passo 1. Calcola il rilassamento del TSP come indicato nella precedente osservazione; sia $F = \{C_1, \ldots, C_p\}$ la famiglia dei cicli orientati così ottenuti; se F contiene un solo ciclo, allora STOP [tale ciclo è una soluzione ottima per il TSP].

Passo 2. Per ogni coppia di cicli C_i , $C_j \in F$, calcola l'*incremento di costo* γ_{ij} corrispondente alla *fusione* di C_i e C_j nel modo più conveniente possibile.

Passo 3. Effettua la *fusione* dei due cicli C_i e C_j ai quali corrisponde il minimo valore di γ_{ij} . Aggiorna F.

Passo 4. Se F contiene un solo ciclo, allora STOP [tale ciclo è una soluzione euristica per il TSP]. Altrimenti torna al Passo 2.

E. Il problema della distribuzione (Vehicle Routing Problem - VRP)

Questi appunti sono tratti liberamente dal libro del Prof. Sassano (cfr. il riferimento [4] del Programma) e dal riferimento http://groups.di.unipi.it/optimize/Courses/RO2IG/aa1415/8-tsp_vrp.pdf *della Prof.ssa Laura Galli* (*si consiglia di vederli).*

Sia $G = (V, E)$ un grafo orientato completo, dove $V = \{v_1, \ldots, v_n\}$, e sia $c_{ij} \geq 0$ una assegnazione di costi per ogni arco (i, j) di G [cfr. la definizione alternativa del TSP, cioè, c_{ij} rappresenta il costo del cammino minimo orientato dal vertice i al vertice j].
Il vertice v_1 rappresenta un *deposito* (magazzino, punto di partenza), da cui si originano le spedizioni di un determinato bene, mentre ogni vertice dell'insieme $V \setminus \{v_1\}$ rappresenta un *punto di vendita*.
Per ogni punto di vendita $v_i \in V \setminus \{v_1\}$ è nota una *domanda* di una quantità d_i del bene. Per effettuare le consegne è disponibile una flotta di m *veicoli* caratterizzati ciascuno da una (stessa) *portata massima* Q espressa nella stessa unità di misura della domanda.

Definizione base del VRP: Il problema è assegnare ciascun punto di vendita a un solo veicolo, con il vincolo che la domanda complessiva dei punti di vendita assegnati a ciascun veicolo non ecceda la portata massima Q, in modo da minimizzare la distanza complessiva percorsa dai veicoli per effettuare (partendo e tornando al deposito) tutte le consegne nei punti di vendita a loro assegnati.

Tale problema è NP-hard, cioè, *difficile* (cfr. Introduzione).

Una *soluzione ammissibile* per il VRP è identificata da una partizione $\{V_1, \ldots, V_m\}$ di $V \setminus \{v_1\}$ [cioè dell'insieme dei punti di vendita], tale che per ogni $h \in \{1, \ldots, m\}$ la somma delle domande dei punti vendita in V_h non eccede la portata massima Q, ed è rappresentata da una famiglia di m cicli orientati $\{C_1, \ldots, C_m\}$, tale che per ogni $h \in \{1, \ldots, m\}$ il ciclo C_h passa per tutti i vertici di V_h e per il vertice v_1 [cioè per il deposito].

Modello di PLI per il VRP

Modello

Sia $V = \{v_1, \ldots, v_n\} = \{1, \ldots, n\}$.

Variabili decisionali:

x_{ij} per ogni $(i, j) \in E$

$x_{ij} = 1$ se l'arco (i, j) appartiene alla famiglia di m cicli

$x_{ij} = 0$ altrimenti

u_i per ogni $i \in V$

u_i = carico del veicolo prima di visitare il vertice i

Modello: $\min \sum_{(i,j)\in E} c_{ij} x_{ij}$

$$\sum_{(i,j)\in E} x_{ij} = 1 \qquad \text{per ogni } i \in V \setminus \{1\} \qquad (1)$$

$$\sum_{(j,i)\in E} x_{ij} = 1 \qquad \text{per ogni } i \in V \setminus \{1\} \qquad (2)$$

$$\sum_{(1,j)\in E} x_{1j} = m \qquad (3)$$

$$u_j - u_i + Q\,x_{ij} \leq Q - d_i \qquad \text{per ogni } (i,j) \in E, \text{ con } j \neq 1 \qquad (***)$$

$$d_i \leq u_i \leq Q \qquad \text{per ogni } i \in V, \text{ con } i \neq 1$$

$$x_{ij} \in \{0, 1\} \qquad \text{per ogni } (i,j) \in E$$

Correttezza: la correttezza del modello non è evidente (la vediamo di seguito); i gruppi (1) e (2) di vincoli garantiscono che per ogni vertice diverso dal deposito esiste (con riferimento ai cicli cercati) esattamente un arco entrante ed esattamente un arco uscente; d'altro canto il vincolo (3) garantisce che dal deposito partiranno e torneranno (in base ai precedenti gruppi di vincoli) esattamente m veicoli; resta il problema di garantire che la soluzione trovata sia una soluzione ammissibile; questo può essere verificato applicando al gruppo (***) di vincoli le stesse considerazioni del gruppo (**) di vincoli e considerando i rimanenti vincoli.

Nota bene: il gruppo (***) di vincoli è formato da un numero polinomiale di vincoli, cioè, da al più $|V|^2$ vincoli; questo, così come nel caso del gruppo di vincoli (**), garantisce che il numero totale di vincoli per tale modello è polinomiale.

Varianti: le varianti del VRP (base) possono naturalmente essere molteplici, di seguito ne riportiamo solo alcune, per aderenza con la realtà; comunque tali varianti possono essere in genere considerate/introdotte nel modello di PLI di sopra per il VRP (base).
:: *numero di automezzi*: può essere fissato a priori ma anche costituire una variabile decisionale;
:: *tipo di automezzi*: possono avere tutti la stessa portata massima oppure non necessariamente;
:: *numero dei depositi*: può essere maggiore di uno;
:: *vincoli temporali*: ci possono essere dei vincoli temporali per effettuare le consegne;
:: *funzioni obiettivo*: ci possono essere diversi obiettivi da perseguire, spesso contrastanti e difficilmente rappresentabili da una unica funzione obiettivo, comunque in letteratura sono state proposte funzioni obiettivo che considerano la distanza percorsa, il numero di veicoli utilizzati, il costo complessivo del servizio, ecc.

Metodi euristici per il VRP

Esistono molti metodi euristici per il VRP. Tuttavia qui non li riportiamo – se non in termini indicativi come di seguito – rimandando comunque il lettore ai metodi euristici per il TSP per avere una idea su possibili metodi euristici.

I metodi euristici per il VRP proposti in letteratura possono essere raggruppati in tre categorie.

:: *Metodi costruttivi*: una soluzione ammissibile viene costruita in modo graduale, i vertici non ancora assegnati a un veicolo vengono inseriti in uno degli insiemi, secondo un qualche criterio "greedy".

:: *Metodi in due fasi*: nella prima fase si stabilisce la partizione $\{V_1, \ldots, V_m\}$, nella seconda fase si stabiliscono i cicli $\{C_1, \ldots, C_m\}$.

:: *Metodi migliorativi o Metodi di ricerca locale*: prima si trova una soluzione ammissibile, poi si cerca di migliorarla esplorando un intorno, fino a trovare una soluzione ottima locale.

Appendice

Cenni su grafi

Informalmente, un grafo può essere definito come una figura formata da un insieme di punti per ogni coppia dei quali può esistere una linea che li collega.

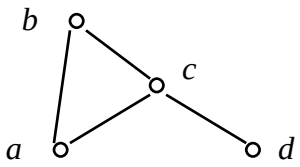

Figura 1 : un grafo

Più formalmente, un *grafo* è una coppia di insiemi $G = (V, E)$ dove E è una famiglia di coppie non-ordinate di elementi di V. L'insieme V è detto insieme dei *vertici*, mentre l'insieme E è detto insieme degli *archi*. Ad esempio, nel grafo $G = (V, E)$ di Figura 1, si ha $V = \{a, b, c, d\}$ e $E = \{(a, b), (a, c), (b, c), (c, d)\}$. Due vertici x, y si dicono *adiacenti* se $(x, y) \in E$; altrimenti, si dicono *non adiacenti*. Due archi si dicono *adiacenti* se incidono su uno stesso vertice; altrimenti si dicono *non adiacenti*.

Un grafo può essere *orientato* oppure *non-orientato.* Un grafo è *orientato* quando gli archi del grafo hanno un orientamento (come se fossero dei sensi unici); così, in un grafo orientato, un arco (a, b) è un arco che parte dal vertice a e arriva al vertice b (ed è indicato con una freccia che va dal vertice a al vertice b); si noti quindi che, in un grafo orientato, l'arco (a, b) è differente dall'arco (b, a). Un grafo è *non-orientato* quando non c'è nessuna specifica sull'orientamento degli archi (come se fossero a doppio senso): in particolare, quando non si specifica se un grafo è orientato oppure non-orientato, allora si intende che il grafo è non-orientato. Secondo altri formalismi, si dice che un grafo è orientato se ogni arco è una *coppia ordinata*, mentre un grafo è non-orientato se ogni arco è una *coppia non-ordinata*.

Nel seguito indicheremo sempre con $n = |V|$ ed $m = |E|$ il numero dei vertici e degli archi di G, rispettivamente. Un grafo $G = (V, E)$ può essere rappresentato anche mediante:

--- *matrice di adiacenza* (*vertici-vertici*) cioè una matrice $\mathbf{A} = [a_{ij}]$ di dimensione $n \times n$ tale che $a_{ij} = 1$ se il vertice i è adiacente al vertice j, mentre $a_{ij} = 0$ altrimenti.

--- *matrice di incidenza* (*vertici-archi*) cioè una matrice $\mathbf{A} = [a_{ij}]$ di dimensione $n \times m$ tale che $a_{ij} = 1$ se il vertice i appartiene all'arco j, mentre $a_{ij} = 0$ altrimenti.

--- *liste di adiacenza*, cioè per ogni $i \in V$ si esplicita l'insieme $A(i) = \{j \in V : (i, j) \in E\}$.

Un grafo $G = (V, E)$ è *bipartito* se esiste una bipartizione $\{V_1, V_2\}$ di V, cioè $V = V_1 \cup V_2$ e $V_1 \cap V_2 = \varnothing$, tale che ogni arco di G contenga un elemento di V_1 e uno di V_2 (cioè, i vertici in V_1 sono mutuamente non adiacenti, e i vertici in V_2 sono mutuamente non adiacenti). Data la bipartizione $\{V_1, V_2\}$, un grafo bipartito è anche indicato anche come $G = (V_1, V_2, E)$. Vale la seguente proprietà:

Proposizione 1
La matrice di incidenza di un grafo bipartito $G = (V_1, V_2, E)$ è totalmente unimodulare (TUM).

dim. Segue dal Teorema 5.2.3 del libro del Prof. Fischetti, indicando con I_1 l'insieme delle righe corrispondenti ai vertici di V_1, e indicando con I_2 l'insieme dalle righe corrispondenti ai vertici di V_2. □

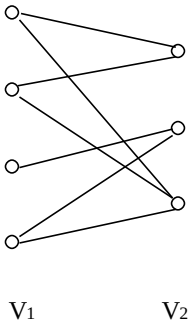

Figura 2 : un grafo bipartito

Un *cammino* di un grafo G è una sequenza di archi $e_1, ..., e_k$ di G consecutivi, cioè del tipo $e_1 = (v_1, v_2)$, $e_2 = (v_2, v_3)$, ..., $e_k = (v_k, v_{k+1})$; tale cammino *collega* v_1 a v_{k+1} ; in particolare, se $v_1 \equiv v_{k+1}$ allora esso è detto *ciclo*. Un grafo G è detto *connesso* se per ogni coppia di vertici di G esiste un cammino di G che li collega; altrimenti è detto *sconnesso*.