Matheuristics for Combinatorial Optimization problems

PhD in Computer Science

Roberto Cordone

DI - Università degli Studi di Milano

Phone nr.: 02 503 16235

E-mail: roberto.cordone@unimi.it

Web page: https://homes.di.unimi.it/cordone/courses/2022-mh/2022-mh.html

Lezione 8: Interazioni fra *B&B* e metaeuristiche Milano, A.A. 2021/22

Euristiche e metaeuristiche classiche

Le euristiche classiche per problemi di Ottimizzazione Combinatoria (*solution-based*) ragionano in termini di soluzioni,
cioè di operazioni su sottoinsiemi di B

A grandi linee, si possono classificare in

1. euristiche costruttive/distruttive:
   - partono da un sottoinsieme estremamente semplice (rispettivamente, $\emptyset$ or B)
   - aggiungono/tolgono elementi fino ad ottenere la soluzione desiderata
2. euristiche di scambio:
   - partono da un sottoinsieme ottenuto in qualche modo
   - scambiano elementi fino ad ottenere la soluzione desiderata
3. euristiche di ricombinazione:
   - partono da una popolazione di sottoinsiemi ottenuta in qualche modo
   - ricombinano sottoinsiemi diversi producendo una nuova popolazione

Si possono poi combinare creativamente elementi di classi diverse

Metaeuristiche

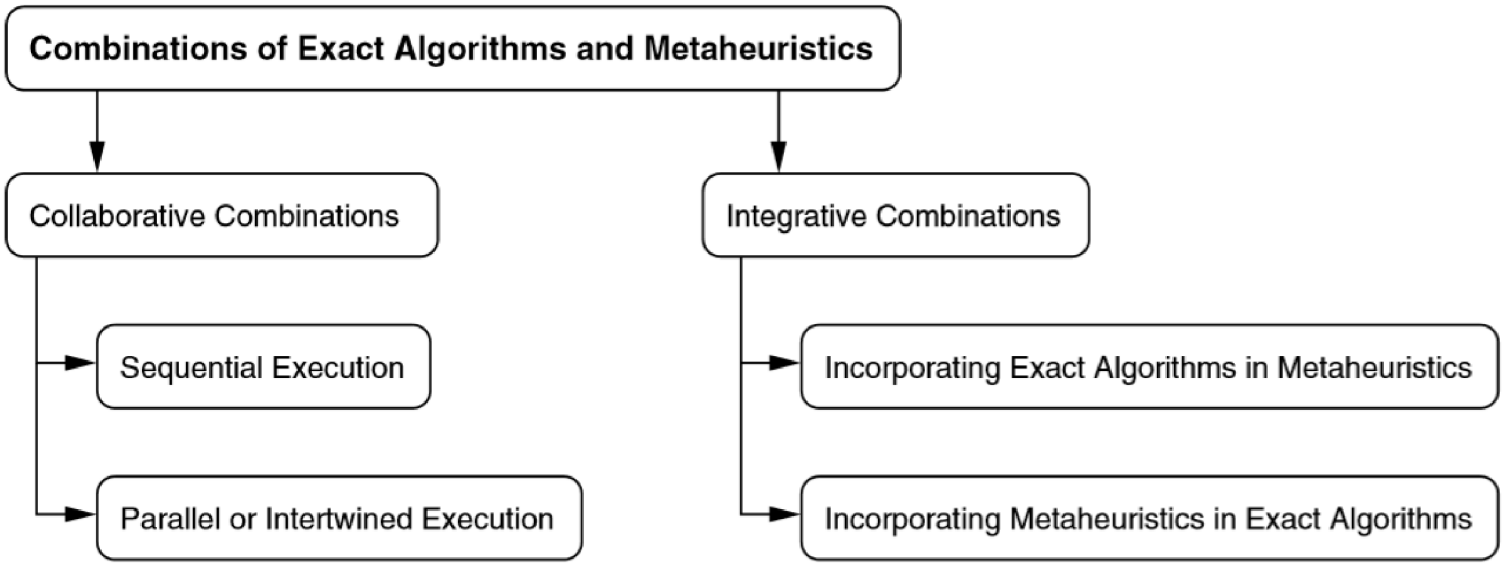
Due altre distinzioni riguardano

- l'uso della casualità:
  - euristiche puramente deterministiche
  - euristiche casualizzate, cioè algoritmi che operano in base a numeri pseudocasuali oltre ai dati
- l'uso della memoria:
  - euristiche che operano solo in base ai dati del problema
  - euristiche che operano ache in base a soluzioni generate in precedenza

Queste distinzioni si applicano a tutte le classi sopra elencate

Metaeuristiche (dal greco, "oltre le euristiche") è il nome comune per le euristiche che utilizzano casualità e/o memoria

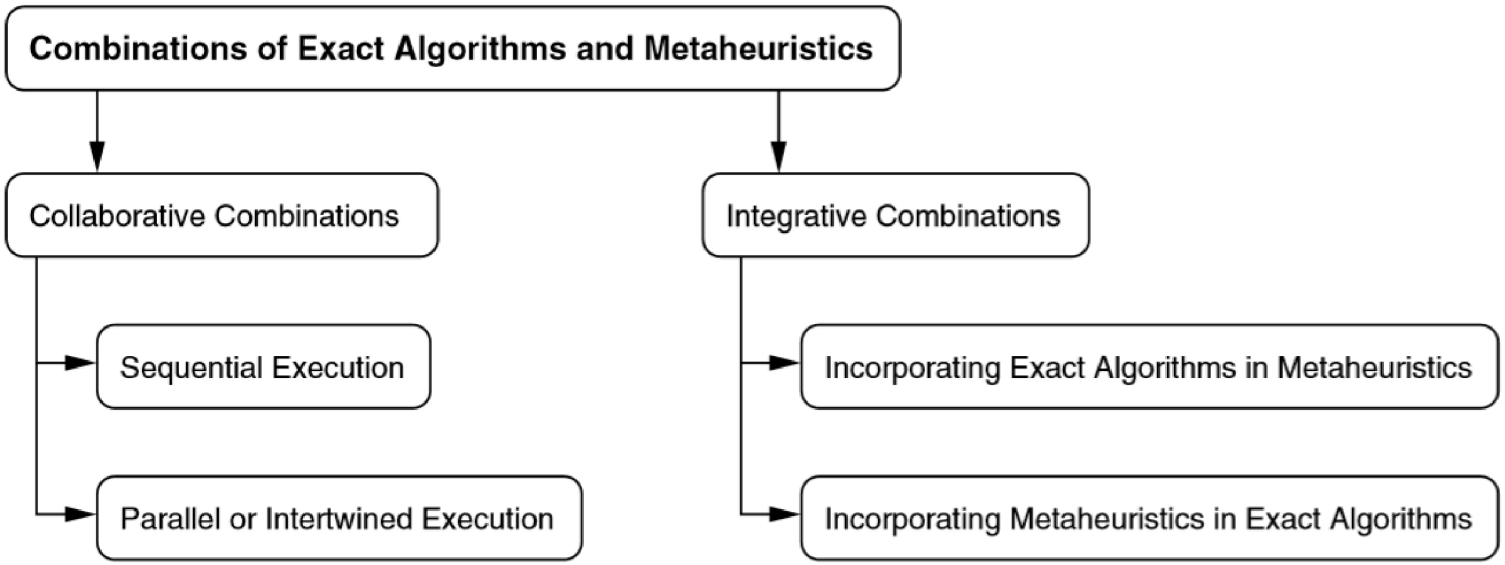
Combinazioni fra branch-and-bound e metaeuristiche



Branch-and-bound e metaeuristiche possono

- collaborare scambiandosi informazioni come programmi distinti
  1. in modo sequenziale
  2. in modo parallelo
- interagire integrandosi in un solo programma
  1. con il B&B controllato dalla metaeuristica
  2. con la metaeuristica controllata dal B&B

Combinazioni collaborative



Branch-and-bound e metaeuristiche possono scambiarsi informazioni

1. in modo sequenziale:
   - la metaeuristica dà al B&B il miglior valore noto $f(\bar{x})$ (`Cutoff`) o soluzioni euristiche $\bar{x}$ per troncare/guidare la ricerca (`InputFile`)
   - il B&B dà alla metaeuristica soluzioni iniziali o da ricombinare
2. operando parallelamente
   - ciascuno migliora le soluzioni dell'altro e gliele comunica

Restringere la regione ammissibile

Oltre alla miglior soluzione nota $\bar{x}$, una metaeuristica può fornire al B&B un intero insieme $\bar{X}$ di soluzioni di buona qualità, che probabilmente

- non sono ottime
- ma nel complesso contengono gli elementi della soluzione ottima

$$\bigcup_{x \in \bar{X}} x \supseteq x^*$$

Se questo è vero, si può

- fissare $x_j = 0$ per ogni $j \in B \setminus \bigcup_{x \in \bar{X}} x$
- risolvere il problema ridotto con un branch-and-bound

L'idea è stata proposta più volte in letteratura con nomi diversi

(*Heuristic Concentration, Fixed Set Search, . . .*)

Heuristic Concentration

Rosing e ReVelle hanno affrontato il problema delle p-mediane

1. generando k soluzioni con la scelta casuale di p nodi come mediane
2. eseguendo su ognuna un'euristica *steepest-descent* che valuta scambi fra una mediana e un altro nodo con riassegnamento dei nodi
3. costruendo due concentration set
   - CS_{free} con le mediane in almeno una delle $k_{\text{free}} \leq k$ migliori soluzioni
   - CS_{open} con le mediane in tutte le $k_{\text{open}} \leq k_{\text{free}} \leq k$ migliori soluzioni
4. risolvendo il problema ridotto

$$\min \sum_{i \in \cancel{N} CS_{\text{free}}} \sum_{j \in \cancel{N} N \setminus CS_{\text{free}}} d_{ij} x_{ij}$$

$$\sum_{i \in N} x_{ij} = 1 \qquad j \in N$$

$$\sum_{i \in N} x_{ii} = p$$

$$x_{ij} \leq x_{ii} \qquad i \in \cancel{N} CS_{\text{free}}, j \in \cancel{N} N \setminus CS_{\text{free}}$$

$$x_{ii} = 1 \qquad i \in CS_{\text{open}}$$

$$x_{ii} \in \{0,1\} \qquad i \in CS_{\text{free}}$$

$$x_{ij} \in \{0,1\} \qquad i \in \cancel{N} CS_{\text{free}}, j \in \cancel{N} N \setminus CS_{\text{free}}$$

Heuristic Concentration e affini

Ovviamente, il metodo richiede di tarare

- il numero k di soluzioni generate
- i numeri k_{free} e k_{open} di soluzioni che definiscono i concentration set

con il solito compromesso fra efficacia ed efficienza

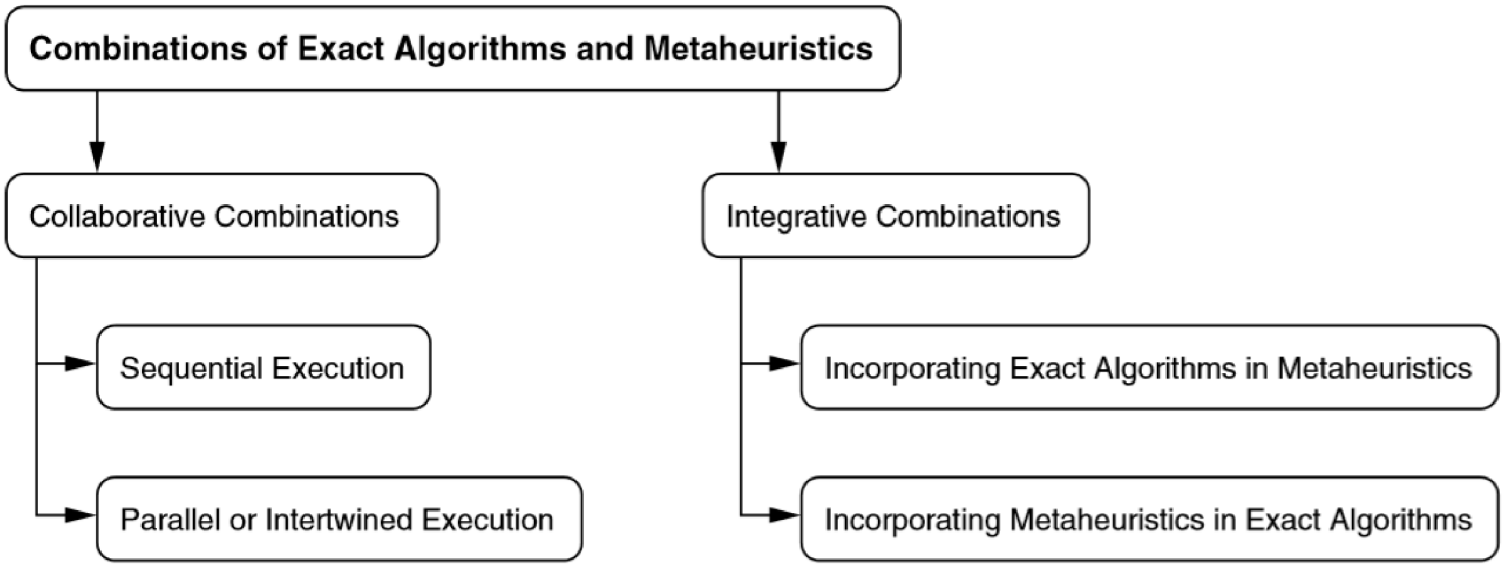
Il vantaggio fondamentale della Heuristic Concentration sta nel poter usare una formulazione buona, ma non pratica per grandi dimensioni

Metodi affini sono stati proposti per problemi di altro genere

- l'euristica Tour merging genera buone soluzioni per il *TSP* e risolve all'ottimo il problema sul grafo ridotto cancellando gli archi non usati
- l'euristica di Taillard per il *VRPTW* genera buone soluzioni, raccoglie i circuiti usati nelle migliori soluzioni e risolve la formulazione estesa con variabili y_p associati a tali circuiti (*una specie di Restricted Master problem generato euristicamente*)

Un problema di una certa rilevanza è come arricchire il problema ridotto se non contiene soluzioni migliori di quelle di partenza

Combinazioni integrative



Branch-and-bound e metaeuristiche possono integrarsi dando luogo a un algoritmo ibrido

1. con il B&B controllato dalla metaeuristica
2. con la metaeuristica controllata dal B&B

Euristiche costruttive

In Ottimizzazione Combinatoria ogni soluzione x è un sottoinsieme di B

Un'euristica costruttiva aggiorna passo per passo un sottoinsieme $x^{(t)}$

1. parte da un sottoinsieme vuoto: $x^{(0)} = \emptyset$
   (*è ovvio che sia sottoinsieme di una soluzione ottima*)
2. si ferma se vale una condizione di fine opportunamente definita
   (*i sottoinsiemi successivi non possono essere soluzioni ottime*)
3. ad ogni passo t, sceglie l'elemento $i^{(t+1)} \in B \setminus x^{(t)}$ "migliore" fra quelli "ammissibili" in base a un opportuno criterio di scelta $\varphi(i, x)$
   (*si cerca di tenere $x^{(t)}$ dentro una soluzione ammissibile e ottima*)
4. aggiunge $i^{(t)}$ al sottoinsieme corrente $x^{(t)}$: $x^{(t+1)} := x^{(t)} \cup \left\{ i^{(t+1)} \right\}$
   (*non si torna più indietro nella scelta!*)
5. torna al punto 2

Branch-and-bound e metaeuristiche costruttive

Un branch-and-bound può integrarsi con una metaeuristica costruttiva fornendo informazioni utili per costruire il criterio di scelta

- i valori delle variabili nella soluzione ottima del rilassamento
- i valori dei costi ridotti
- i valori delle variabili duali associate ai vincoli
- l'esistenza di operazioni di presolve prossime a scattare

Le euristiche di diving si possono vedere come algoritmi greedy influenzati dal calcolo della soluzione x_R^* del rilassamento continuo R

Approximate Nondeterministic Tree-Search (*ANTS*)

L'Approximate Nondeterministic Tree-Search (*ANTS*) (Maniezzo, 1999) integra la metaeuristica costruttiva di Ant System col branch-and-bound
È stata applicata al *Quadratic Assignment Problem* (*QAP*)

$$\min f(x) = \sum_{i,j=1}^{n} \sum_{h,k=1}^{n} q_{ijhk} x_{ij} x_{hk} + \sum_{i,j=1}^{n} c_{ij} x_{ij}$$

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

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

$$x_{ij} \in \{0, 1\} \qquad i, j \in \{1, \ldots, n\}$$

L'Ant System costruisce f soluzioni partendo da $x_{ij} = 0$ per ogni i, j

- sceglie in modo casuale una variabile che rispetti i vincoli
  - con probabilità determinate dai costi c_{ij} e q_{ijhk} e dalle tracce τ_{ij}
- fissa la variabile a 1
- se la soluzione non è completa, torna al punto 2

Al termine, aggiorna la traccia tenendo conto delle soluzioni migliori

Approximate Nondeterministic Tree-Search (*ANTS*)

L'euristica *ANTS*

- inizializza le tracce τ_{ij} con la soluzione rilassata x^*_{Rij}
  anziché con valori uniformi
- sceglie la variabile x_{ij} in base al lower bound η_{ij} ottenuto fissandola*
  anziché in base ai costi c_{ij} e q_{ijhk}

$$\pi_{ij} = \frac{\alpha\tau_{ij} + (1-\alpha)\,\eta_{ij}}{\sum_{ij}\left[\alpha\tau_{ij} + (1-\alpha)\,\eta_{ij}\right]}$$

*così dice il testo, ma η_{ij} dovrebbe essere inversamente legato al lower bound

- termina la costruzione appena il lower bound supera $f(\bar{x})$,
  cioè quando la singola procedura non ha speranze di migliorare
- applica il presolve ad ogni passo
- aggiorna la traccia in base al gap

$$\Delta\tau_{ij}^{(t+1)} = \rho\tau_{ij}^{(t)} + (1-\rho)\,\tau_0\left(1 - \frac{f(x) - f_R^*}{f(\bar{x}) - f_R^*}\right)$$

dove $f(\bar{x})$ è una media dei valori delle ultime soluzioni trovate

Branch-and-bound per risolvere sottoproblemi

Un altro ibrido fra euristiche costruttive e branch-and-bound è

- aggiungere un sottoinsieme $I^{(t+1)} \subset B \setminus x^{(t)}$ anziché un elemento
- risolvere con un branch-and-bound la scelta di $I^{(t+1)}$

Vediamo un esempio non molto efficace

Nel Bin Packing Problem (BPP) abbiamo

- un insieme O di oggetti elementari
- una funzione $v : O \to \mathbb{N}$ che ne descrive il volume
- un insieme C di contenitori
- una capacità $V \in \mathbb{N}$ per i contenitori

e si cerca il minimo numero di contenitori che includano tutti gli oggetti rispettando la capacità

Nel Knapsack Problem (KP) abbiamo

- un insieme O di oggetti elementari
- una funzione $v : O \to \mathbb{N}$ che ne descrive il volume
- una capacità $V \in \mathbb{N}$ per lo zaino
- una funzione $\phi : O \to \mathbb{N}$ che descrive il valore degli oggetti

e si cerca l'insieme di oggetti di valore massimo che rispetti la capacità

I due problemi hanno chiaramente un che di simile

Branch-and-bound per risolvere sottoproblemi

Si potrebbe

1. partire da un insieme vuoto di assegnamenti
2. risolvere con un branch-and-bound (o programmazione dinamica) il *KP* su un contenitore con valori uniformi ($\phi_o = 1$), in modo da assegnare il massimo numero di oggetti
3. finché ci sono ancora oggetti, considerare il contenitore successivo e tornare al punto 2

È una cattiva idea:

- gli oggetti più grandi restano ultimi
- molti contenitori restano semivuoti

Ma si potrebbe invece minimizzare lo spazio residuo

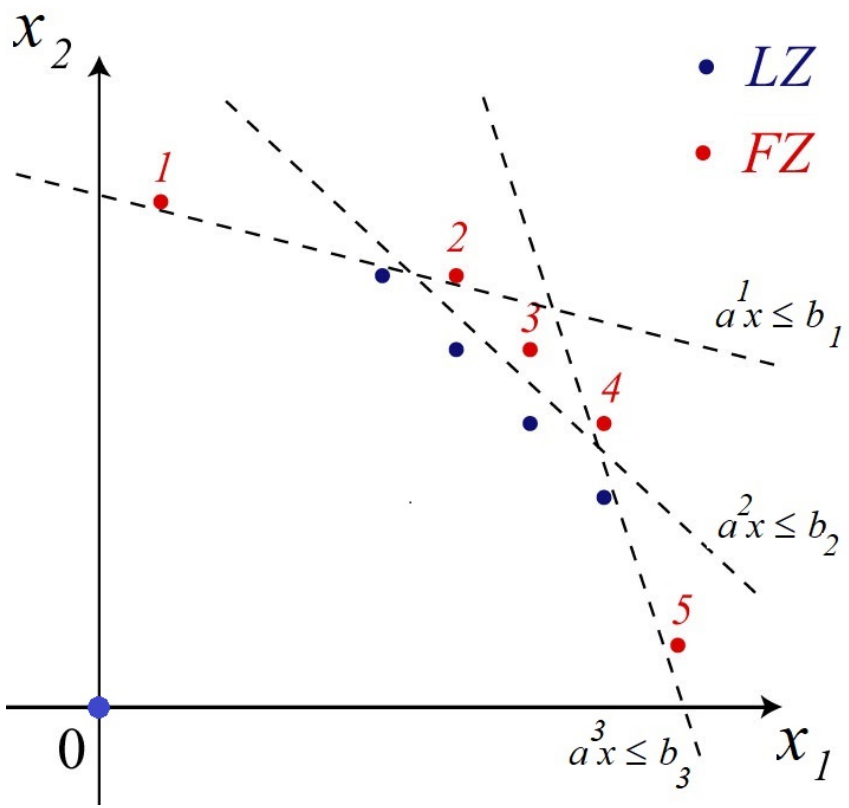
Minimum linear separation

Un problema simile si incontra però nel campo dei classificatori lineari

- un insieme di punti "cattivi" *FZ* (forbidden zone)
- un insieme di punti "buoni" *LZ* (legal zone)

Si cerca il minimo numero di vincoli affini $A^{(i)}x \leq b_i$ (iperpiani) tali che

- sono tutti rispettati da tutti i punti buoni
- almeno uno è violato da ciascun punto cattivo



Minimum linear separation

Un possibile approccio euristico è trovare iterativamente il vincolo violato dal massimo numero di punti cattivi e da nessun punto buono

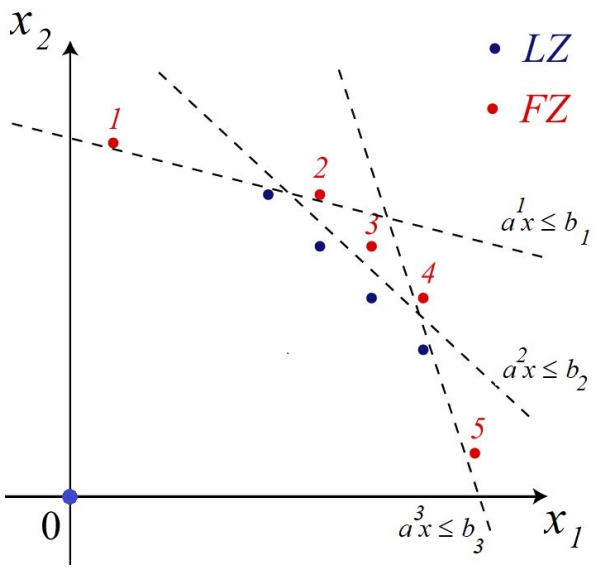
$$\max \sum_{j \in FZ} z_j$$

$$\sum_{i=1}^{n} a_i x_i^{(j)} \geq b + \epsilon - M(1 - z_j) \qquad x^{(j)} \in FZ$$

$$\sum_{i=1}^{n} a_i x_i^{(j)} \leq b \qquad x^{(j)} \in LZ$$

$$z_j \in \{0, 1\} \qquad j \in FZ$$

$$b \in \mathbb{R}, a_i \in \mathbb{R} \qquad i \in \{1, \ldots, n\}$$



Euristiche di scambio

In Ottimizzazione Combinatoria ogni soluzione x è un sottoinsieme di B

Un'euristica di scambio aggiorna via via la soluzione corrente $x^{(t)}$

1. parte da una soluzione ammissibile $x^{(0)} \in X$ trovata in qualche modo
2. genera una famiglia di sottoinsiemi ammissibili ottenuti da $x^{(t)}$ aggiungendo un sottoinsieme esterno A e cancellando un sottoinsieme interno D

$$x'_{A,D} = x \cup A \setminus D \text{ con } A \subseteq B \setminus x \text{ e } D \subseteq x$$

3. sceglie i sottoinsiemi A e D con un criterio di selezione $\varphi(x, A, D)$

$$(A^*, D^*) = \arg \min_{(A,D)} \varphi(x, A, D)$$

(*quasi sempre il criterio è l'obiettivo:* $\varphi(x, A, D) = f(x \cup A \setminus D)$)

4. esegue lo scambio scelto per generare la nuova soluzione corrente

$$x^{(t+1)} := x^{(t)} \cup A^* \setminus D^*$$

5. se vale una condizione di termine, si arresta;
   altrimenti, torna al punto 2

Intorno

È del tutto equivalente definire

- le coppie di sottoinsiemi (A, D) da aggiungere e togliere da x
- direttamente le soluzioni risultanti $x \cup A \setminus D$

Intorno $N : X \to 2^X$ è una funzione che associa ad ogni soluzione ammissibile $x \in X$ un sottoinsieme di soluzioni ammissibili $N(x) \subseteq X$

Il modo più semplice di definire un intorno è attraverso la distanza di Hamming $d_H(\xi, \xi') = |\xi \setminus \xi'| + |\xi' \setminus \xi|$ vale a dire

$$d_H(x, x') = \sum_{i \in B} |x_i - x'_i|$$

e consiste nell'imporre una distanza massima k dalla soluzione corrente x

$$N_{H_k}(x) = \{x' \in X : d_H(x, x') \leq k\}$$

Questi intorni sono particolarmente naturali quando si usa la *PLI*, perché corrispondono al vincolo della Local Branching

$$\sum_{j \in B} [\bar{x}_j (1 - x_j) + (1 - \bar{x}_j) x_j] \leq 2k$$

Euristiche *steepest descent* (*hill-climbing*)

Le euristiche di scambio *steepest descent*

- usano come criterio di selezione l'obiettivo
  $\varphi(x, A, D) = f(x \cup A \setminus D)$ (*quasi tutte*)
- ad ogni passo muovono da $x^{(t)}$ alla miglior soluzione di $N\left(x^{(t)}\right)$
  (*salvo limitazioni euristiche*)
- terminano quando non trovano soluzioni miglioranti allo scopo di evitare comportamenti ciclici

Algorithm SteepestDescent$\left(I, x^{(0)}\right)$

$x := x^{(0)}$;

Stop := *false*;

While Stop = *false do*

$\tilde{x} := \arg \min_{x' \in N(x)} f(x')$;

If $f(\tilde{x}) \geq f(x)$ *then* Stop := *true*; *else* $x := \tilde{x}$;

EndWhile;

Return $(x, f(x))$;

Branch-and-bound e metaeuristiche di scambio

Il punto critico delle euristiche di scambio è la definizione dell'intorno

- un intorno piccolo è veloce da esplorare, ma contiene soluzioni mediamente peggiori
- un intorno grande contiene soluzioni di qualità migliore, ma richiede tempi di calcolo lunghi

È un compromesso difficile da modulare

Vi sono in letteratura esempi di

- intorni contenenti un numero esponenziale di soluzioni
- visitabili in tempo polinomiale risolvendo un problema ausiliario

ma sono esempi molto legati alla struttura del problema

Un branch-and-bound può integrarsi con una metaeuristica di scambio

- esplorando l'intorno in modo implicito, dunque piuttosto veloce, anche se l'intorno è esponenziale o polinomiale di ordine elevato, indipendentemente dalla struttura del problema
- fornendo meccanismi di intensificazione o diversificazione più efficaci

Metodi di *destroy-and-repair* (*ruin-and-recreate*)

Siccome qualsiasi passaggio fra soluzioni x e x' si può vedere come

- eliminazione da x di un sottoinsieme $D = x \setminus x'$
- aggiunta a $x \setminus D$ di un sottoinsieme $A = x' \setminus x$

si possono definire intorni con sottoinsiemi A e D "grandi"

La loro esplorazione è

- impraticabile per via esaustiva
- ma formulabile come problema di *PLI*
  da risolvere con un branch-and-bound

Prescott-Gagnon, Desaulniers e Rousseau hanno affrontato il *VRPTW*

1. generando una soluzione euristica iniziale
2. rimuovendo dalla soluzione un dato numero di nodi
   - particolarmente costosi da servire
   - consecutivi lungo i circuiti correnti
   - vicini in termini di orario di servizio
   - vicini fra loro in termini di costi o finestre temporali (Shaw removal)
3. fissando i pezzi di circuito non toccati dalle rimozioni
4. risolvendo il *VRPTW* ridotto con un branch-and-price limitato

Dynasearch

Un altro modo di generare intorni esponenziali è

- definire mosse elementari, in numero polinomiale
  (*ad es., spostare un nodo in altra posizione o altro circuito*)
- definire mosse composte come insiemi di mosse elementari
  (*ad es., spostare contemporaneamente tanti nodi*)

Molti problemi di Ottimizzazione Combinatoria e molte mosse godono della proprietà che molte mosse sono indipendenti tra loro

- se ciascuna mossa elementare è ammissibile, è ammissibile anche la loro composizione
- l'effetto della mossa composta è la somma di quelli delle componenti

Esempi:

- scambi di nodi o di archi fra circuiti differenti nei problemi di routing
- scambi di vertici o i lati fra sottoalberi diversi nei problemi di foreste ricoprenti
- scambi di job fra macchine diverse nei problemi di scheduling
- scambi di archi fra segmenti disgiunti di un circuito hamiltoniano

Dynasearch

La situazione è modellabile da una matrice di miglioramento A con

- righe che rappresentano componenti della soluzione
  (ad esempio, sottoalberi, circuiti, segmenti di circuito hamiltoniano)
- colonne che rappresentano mosse elementari, valutate attraverso un premio v_k pari al miglioramento dell'obiettivo dopo la mossa k
- $a_{hk} = 1$ se la mossa k modifica il componente h; altrimenti $a_{hk} = 0$

Il problema di trovare la miglior combinazione di mosse indipendenti è

$$\max \phi(y) = \sum_{k \in K} v_k y_k$$

$$\sum_{k \in K} a_{hk} y_k \leq 1$$

$$y_k \in \{0, 1\}$$

cioè un Set Packing Problem

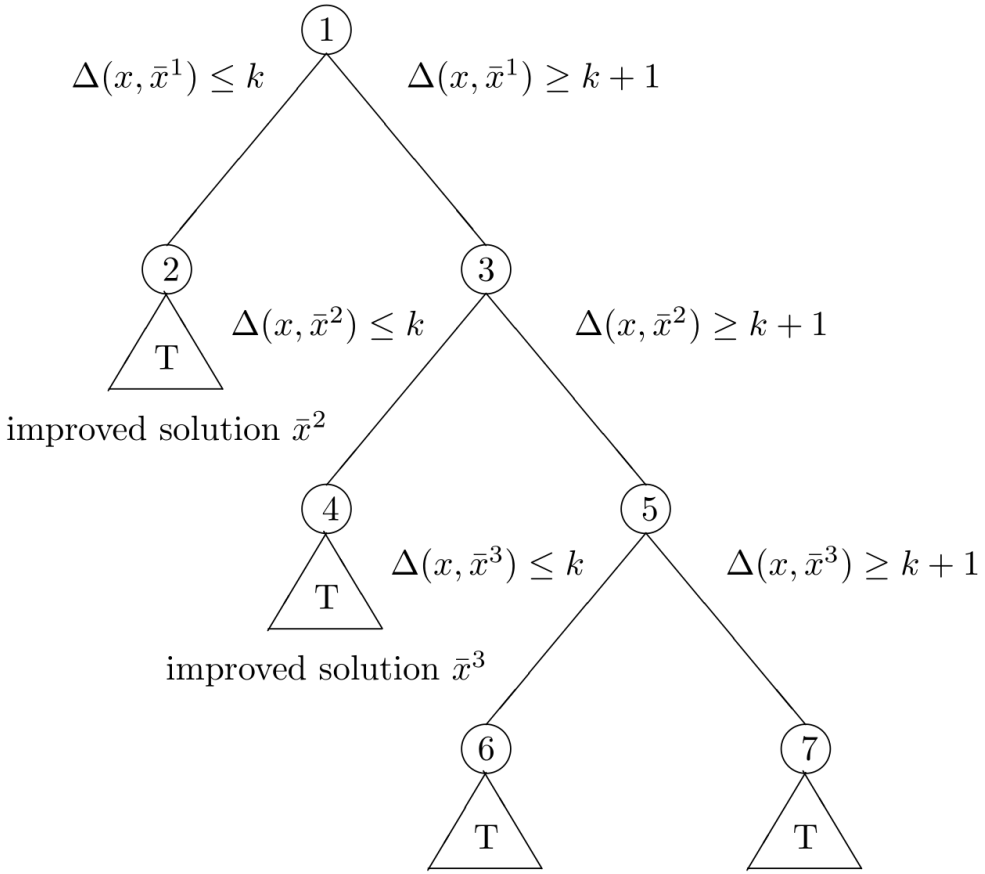
- in generale $\mathcal{NP}$-difficile, ma risolubile con un branch-and-bound anche per dimensioni consistenti
- per alcuni problemi e mosse specifiche polinomiale
  (*se ogni colonna copre al massimo 2 righe, diventa un matching*)

Local Branching e Local Search

La strategia Local Branching genera una sequenza di soluzioni

- miglioranti passo per passo
- a distanza di Hamming $\leq k$ ciascuna dalla precedente

come se fosse un'euristica *steepest-descent* con un intorno N_{H_k}



Trovato un ottimo locale, passa all'altro ramo e cerca soluzioni lontane

Variable Neighbourhood Search

La Variable Neighbourhood Search (*VNS*) applica ciclicamente

- un'euristica *steepest descent* che genera ottimi locali
- una procedura di perturbazione che genera nuove soluzioni iniziali
  - estraendole casualmente da un intorno N_{H_k}
  - regolando l'ampiezza k dell'intorno in modo adattivo:
    se la soluzione migliora, k cala per intensificare;
    se la soluzione peggiora, k cresce per diversificare

Algorithm VariableNeighbourhoodSearch$(I, x^{(0)})$

$x :=$ SteepestDescent$(x^{(0)})$; $x^* := x$;

$k := k_{\min}$;

For $l := 1$ *to* ℓ *do*

  $x' :=$ ExtractNeighbour(x^*, k);

  $x' :=$ SteepestDescent(x');

  If $f(x') < f(x^*)$

    then $x^* := x'$; $k := k_{\min}$;

    else $k := k + \delta k$;

  If $k > k_{\max}$ *then* $k := k_{min}$;

EndWhile;

Return $(x^*, f(x^*))$;

Uso euristico della Local Branching

È facile simulare la *VNS* con la Local Branching:

- si esplora $N_{H_k}(\bar{x})$ col branch-and-bound in modo esaustivo implicito
- passo per passo si migliora $\bar{x}$ fino a giungere in un ottimo locale
- anziché cercare una generica soluzione a distanza $\geq k+1$, si aggiunge il vincolo

$$k+1 \leq H(x,\bar{x}) = \sum_{j \in B} (1-\bar{x}_j)\, x_j + \bar{x}_j\, (1-x_j) \leq k + \delta k$$

  per generare la nuova soluzione iniziale a distanza controllata
- si esplora l'intorno della nuova soluzione
  - se si trova una soluzione migliorante, si riprende con la ricerca locale
  - altrimenti, si torna nell'ottimo locale precedente e si cerca nella cornice seguente di ampiezza δk

È un'euristica perché il ramo $\geq k+1$ non viene esplorato interamente

Anche altre metaeuristiche sono state adattate alla Local Branching (ad es. la Variable Depth Search)

Euristiche di ricombinazione

Le euristiche di ricombinazione si basano sull'idea che

- soluzioni buone hanno elementi comuni con l'ottimo globale
- soluzioni diverse hanno in comune elementi diversi
- ricombinando soluzioni diverse si possono raccogliere gli elementi ottimi meglio che costruendoli o scambiandoli un passo per volta

Lo schema tipico delle euristiche di ricombinazione è

- costruire in qualche modo una popolazione iniziale di soluzioni
- estrarre sottoinsiemi di individui dalla popolazione
- applicare operazioni di ricombinazione agli individui di ogni sottoinsieme generando nuovi individui
- applicare operazioni di scambio agli individui singoli
- scegliere quali nuovi individui inserire nella popolazione
- se una condizione di termine non è ancora valida, tornare al punto 2

Path Relinking

Il *Path Relinking* (*PR*) è una procedura di intensificazione

- raccoglie in un insieme di riferimento R le migliori soluzioni generate da un'euristica di scambio (soluzioni di élite)
- per ogni coppia di soluzioni x e y in R
  - costruisce un cammino da x a y nello spazio di ricerca dell'intorno N applicando a $z^{(0)} = x$ l'euristica di scambio

$$z^{(k+1)} := \arg \min_{z \in N(z^{(k)})} [D(z, y) + \epsilon f(z)]$$

    con un obiettivo D che minimizza la distanza dalla destinazione y e, secondariamente, l'obiettivo f del problema
  - restituisce la miglior soluzione z^*_{xy} lungo il cammino $(z^{(0)}, \ldots, z^{(h)})$

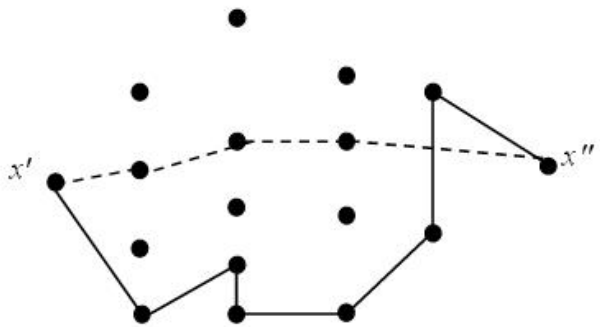
$$z^*_{xy} := \arg \min_{k \in \{1,\ldots,h-1\}} f(z^{(k)})$$

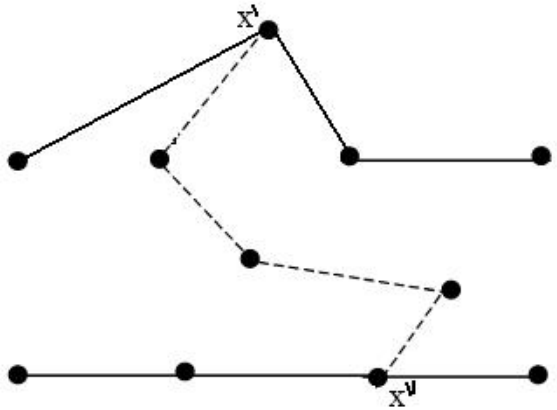
    e la migliora ulteriormente con l'euristica di scambio
  - aggiorna R inserendovi z^*_{xy} se è migliore di almeno una soluzione d'élite e non è un duplicato di alcuna

Cammini di *relinking*

I cammini esplorati in questo modo

- intensificano la ricerca, perché collegano soluzioni buone
- diversificano la ricerca, perché in genere sono diversi da quelli seguiti dall'euristica di scambio (soprattutto se gli estremi sono lontani)





- poiché la distanza di $z^{(k)}$ da y cala via via, si possono esplorare
  - soluzioni peggioranti senza il rischio di comportamenti ciclici
  - sottoinsiemi inammissibili senza il rischio di non riuscire a riottenere soluzioni ammissibili

(*utili non in sé, ma per aprire la strada a miglioramenti*)

RINS e Path Relinking

La *RINS* si può vedere come un'euristica *destroy-and-repair* che distrugge la parte della soluzione $\bar{x}$ discorde dalla soluzione rilassata x^*

È un operatore di distruzione

- non del tutto casuale, dunque probabilmente più efficace
- indipendente dalla struttura specifica del problema, dunque generale

Ma si può anche vedere la *RINS* come un'estensione del Path Relinking: fissare gli elementi comuni a $\bar{x}$ e x^* e ottimizzare gli altri significa

- esplorare l'intero sottospazio compreso fra $\bar{x} \cap x^*$ e $\bar{x} \cup x^*$
- anziché solo uno o più cammini in esso

Il fatto che $\bar{x}$ è inammissibile, ma significativa può essere decisivo per raggiungere soluzioni circondate da molte soluzioni inammissibili

Esistono varianti della *RINS* che usano due soluzioni euristiche $\bar{x}$ e $\bar{x}'$ anziché una euristica e una rilassata

Integrazione di metaeuristiche nel branch-and-bound

Usare metaeuristiche di scambio nei branch-and-bound è tradizionale
e consente vantaggi reciproci

- il branch-and-bound riceve soluzioni $\bar{x}$ migliorate
  (*in tutti i nodi o in alcuni, con condizioni di terminazione varie*)
  cioè la metaeuristica lo aiuta a chiudere nodi
- la metaeuristica sfrutta x_R^* per l'inizializzazione
  (*specialmente nei rilassamenti lagrangiani*)
  e quindi i vincoli di branching aiutano la diversificazione

Un problema interessante è se convenga

1. usare i vincoli di branching solo nel costruire la soluzione iniziale
2. imporre i vincoli di branching anche nella metaeuristica,
   limitando la ricerca, ma forzando evoluzioni diverse nei vari nodi

Tipicamente, conviene la prima opzione (la diversificazione iniziale basta)

Integrazione di metaeuristiche nel branch-and-bound

Rothberg (2007) integra euristiche di ricombinazione (algoritmi genetici) nel branch-and-bound

- la mutazione riottimizza le singole soluzioni euristiche
- il crossover ricombina soluzioni trovate in nodi diversi

e ancora una volta il vantaggio è reciproco perché
i meccanismi di mutazione e crossover sono basati sulla *PLI*

- la mutazione fissa un numero casuale di variabili in una soluzione
  e risolve il problema rimanente (*come nel diving con sub-MIP*)
- il crossover fissa le variabili che hanno ugual valore in due soluzioni
  e risolve il problema rimanente (*come nella RINS*)

Bibliografia

- K. Rosing and C. ReVelle (1997). Heuristic Concentration: Two stage Solution Construction. *European Journal ofOperational Research*, 97 (1): 75–86.
- V. Maniezzo, (1999). Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. *INFORMS Journal on Computing*, 11(4): 358–369.
- E. Prescott-Gagnon, G. y Desaulniers, L.-M. Rousseau, (2009). A Branch-and-Price-Based Large Neighborhood Search Algorithm for the Vehicle Routing Problem with Time Windows. *Networks*, 54 (4): 190–204.
- F. Della Croce, R. Tadei, A. Grosso, (2004). An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters*, 32 (1): 68–72.
- P. Hansen, N. Mladenović, D. Urošević, (2006). Variable neighborhood search and local branching. *Computers & Operations Research*, 33 (10): 3034–3045.