Algoritmi Euristici

Corso di Laurea in Informatica e Corso di Laurea in Matematica

Roberto Cordone

DI - Università degli Studi di Milano

Lezioni: Lunedì 13.30 - 15.30 in Aula G30
Giovedì 13.30 - 15.30 in Aula G30
Ricevimento: su appuntamento
Tel.: 02 503 16235
E-mail: roberto.cordone@unimi.it
Web page: http://homes.di.unimi.it/~cordone/courses/2018-ae/2018-ae.html

Proseguire la ricerca locale senza peggiorare

Anziché ripetere la ricerca locale, si può proseguirla oltre l'ottimo locale

Per continuare a migliorare, si cambia la regola di scelta della soluzione

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

Due strategie consentono di farlo senza accettare peggioramenti

- la *Variable Neighbourhood Descent* prosegue cambiando l'intorno N
- la *Dynamic Local Search* prosegue cambiando la funzione obiettivo f

La prima strategia garantisce un'esplorazione priva di cicli,
la seconda no (la funzione obiettivo originale peggiora)

Variable Neighbourhood Descent (VND)

La *Variable Neighbourhood Descent* di Hansen e Mladenović (1997) sfrutta il fatto che gli ottimi locali sono relativi all'intorno scelto

- cambiando intorno, in genere un ottimo locale non è più tale

Definito un insieme di intorni $N_1, \ldots, N_{k_{\max}}$

1. si parte con $k := 1$
2. si trova un ottimo locale $\bar{x}$ rispetto a N_k con un'euristica *steepest descent*
3. si aggiorna k
4. se $\bar{x}$ è ottimo locale per tutti gli N_k, si termina; altrimenti, si torna al punto 2

Algorithm VariableNeighbourhoodDescent($I, x^{(0)}$)
$K := \{1, \ldots, k_{\max}\}$;
$\bar{x} := x^{(0)}$; $x^* := x^{(0)}$; $k := 1$;
Repeat
  $\bar{x} :=$ SteepestDescent($\bar{x}, k$);
  If $f(\bar{x}) < f(x^*)$
    then $x^* := \bar{x}$; $K := \{1, \ldots, k_{\max}\}$;
    then $K := K \setminus \{k\}$;
  If $K \neq \emptyset$ *then* $k :=$ *Update*(k, K);
until $K = \emptyset$;
Return $(x^*, f(x^*))$;

VND e *VNS*

C'è ovviamente una stretta relazione fra *VND* e *VNS*
(in effetti, furono proposte negli stessi lavori)

Le differenze fondamentali sono che nella *VND*

- ad ogni passo la soluzione corrente è la migliore nota
- gli intorni sono esplorati, anziché usati per estrarre soluzioni casuali

  Quindi non sono mai enormi
- gli intorni non formano necessariamente una gerarchia

  Quindi l'aggiornamento di k può non essere un incremento
- quando si raggiunge un ottimo locale per ogni N_k, si termina

Strategie di scorrimento degli intorni nella *VND*

Ci sono due categorie principali di metodi *VND*

- nei metodi a intorno gerarchico ($N_1 \subset \ldots \subset N_{k_{\max}}$) si vuole
  - sfruttare a fondo gli intorni piccoli e rapidi
  - ricorrere a quelli grandi e lenti solo per uscire dagli ottimi locali

  Di conseguenza, l'aggiornamento di k funziona come nella *VNS*
  - quando non si trovano miglioramenti in N_k, si incrementa k
  - quando si trovano miglioramenti in N_k, k torna a 1
- nei metodi a intorno eterogeneo si vuole
  - sfruttare le potenzialità di intorni topologicamente diversi tra loro (per es., scambiare vertici anziché lati)

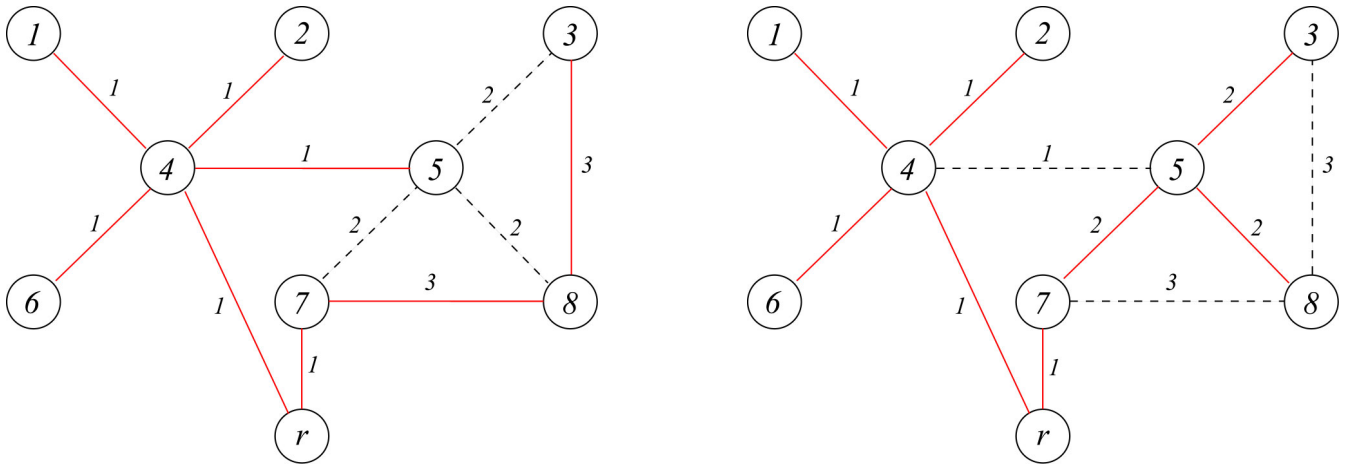
  Di conseguenza, k scorre progressivamente i valori da 1 a $k_{\max}$ (eventualmente permutando la sequenza ad ogni ripetizione)

Si termina quando la soluzione corrente è ottimo locale per tutti gli N_k

- nel caso gerarchico, si termina quando fallisce $N_{k_{\max}}$
- nel caso eterogeneo, occorre un segnalatore di miglioramento (*flag*)

Esempio: il CMSTP

Data un'istanza di *CMSTP* con $n = 9$ vertici, pesi uniformi ($w_v = 1$), capacità $W = 5$ e i costi indicati in figura (per i lati mancanti, $c_e \gg 3$)



Considerando l'intorno N_{S_1} della prima soluzione:

- l'albero di sinistra non può ricevere sottoalberi, ma solo perderne: non si possono cancellare lati nell'albero di destra
- se si cancella un lato nell'albero di sinistra, il costo totale cresce: la soluzione è di ottimo locale per N_{S_1}

L'intorno N_{V_1} (spostamenti di vertici) ha invece uno scambio migliorante, che porta il vertice 5 dall'albero di sinistra a quello di destra

Dynamic Local Search (DLS)

La *Dynamic Local Search* è un approccio complementare alla *VND*

- conserva l'intorno iniziale
- modifica la funzione obiettivo

Si usa spesso nei problemi in cui l'obiettivo è poco utile (ampi *plateau*)

L'idea fondamentale è di

- associare all'insieme base una funzione penalità $w : B \to \mathbb{N}$
- costruire una funzione ausiliaria $\tilde{f}\left(f\left(x\right), w\left(x\right)\right)$
  che combina la funzione obiettivo f con la penalità w
- applicare un'euristica di scambio *steepest descent* che ottimizza $\tilde{f}$
- aggiornare la penalità w in base ai risultati e riapplicare l'euristica

Schema della *DLS*

Algorithm DynamicLocalSearch($I, x^{(0)}$)
$w :=$ PenalitaIniziali(I);
$\bar{x} := x^{(0)}$; $x^* := x^{(0)}$;
While Fine() $=$ *false do*
  $(\bar{x}, x_f) :=$ SteepestDescent($\bar{x}, f, w$);
  If $f(x_f) < f(x^*)$ *then* $x^* := x_f$;
  $w :=$ *AggiornaPenalita*($w, \bar{x}, x^*$);
EndWhile;
Return $(x^*, f(x^*))$;

Si noti che l'euristica *steepest descent*

- considera w oltre a f
- restituisce due soluzioni:
  1. una soluzione finale $\bar{x}$, che è di ottimo locale rispetto a $\tilde{f}$
  2. una soluzione x_f, che è la migliore visitata rispetto a f

Varianti

L'idea è abbastanza generale da avere moltissime varianti

- si possono applicare
  - penalità additive:
    $$\tilde{f}(x) = f(x) + \sum_{i \in x} w_i$$
  - penalità moltiplicative (con obiettivi additivi $f(x) = \sum_{i \in x} \phi_i$):
    $$\tilde{f}(x) = \sum_{i \in x} w_i \, \phi_i$$
- le penalità possono subire
  - un aggiornamento casuale: perturbazione "rumorosa" dei costi
  - un aggiornamento basato sulla memoria, che favorisca gli elementi più frequenti (intensificazione) o rari (diversificazione)
- l'aggiornamento può avvenire
  - ad ogni singola esplorazione di intorno
  - quando si arriva a un ottimo locale per $\tilde{f}$
  - quando la miglior soluzione nota x^* non cambia per parecchio

Esempio: *DLS* per il *MCP*

Dato un grafo non orientato, si cerca una clique di cardinalità massima

- L'euristica di scambio è una *VND* che usa gli intorni
  1. N_{A_1} (aggiunta di un vertice): la soluzione migliora sempre, ma l'intorno è molto piccolo e spesso vuoto
  2. N_{S_1} (scambio di un vertice interno con uno esterno): l'intorno è più grande, ma forma un *plateau* (l'obiettivo non cambia)
- L'obiettivo non fornisce una direzione utile in nessuno dei due intorni
- Si associa a ogni vertice i una penalità w_i inizialmente nulla
- L'euristica di scambio minimizza la penalità totale (dentro l'intorno!)
- Si aggiorna la penalità
  1. quando termina l'esplorazione di N_{S_1}: la penalità dei vertici della clique corrente aumenta di 1
  2. dopo un dato numero di esplorazioni: tutte le penalità non nulle diminuiscono di 1

La logica del metodo consiste nel tendere a

- espellere i vertici interni (diversificazione)
- in particolare, quelli rimasti interni più a lungo (memoria)

Esempio: *DLS* per il *MAX-SAT*

Date m disgiunzioni logiche dipendenti da n variabili logiche, si cerca un assegnamento di verità che soddisfi il massimo numero di formule

- L'intorno N_{F_1} è generato invertendo il valore di una variabile (1-flip)
- Si associa a ogni formula logica una penalità w_j inizialmente pari a 1 (w *qui non è definita sull'insieme base*)
- L'euristica di scambio massimizza il numero di formule soddisfatte
- La funzione modificata massimizza il numero più la penalità totale
- Si aggiorna la penalità
  1. quando si raggiunge un ottimo locale

     $$w_j := \alpha_{\mathrm{us}}\ w_j \text{ per ogni } j \in U(x) \quad \text{(formule insoddisfatte)}$$

     con $\alpha_{\mathrm{us}} > 1$ per favorire le formule attualmente insoddisfatte
  2. con una certa probabilità o dopo un certo numero di aggiornamenti

     $$w_j := (1-\rho)\ w_j + \rho \text{ per ogni } j$$

     per riuniformare le penalità al valore minimo unitario

Esempio: *DLS* per il *MAX-SAT*

La logica del metodo consiste nel tendere a

- soddisfare le formule attualmente insoddisfatte (diversificazione)
- in particolare, quelle rimaste insoddisfatte più a lungo e più di recente (memoria)
- valori bassi di ρ (oblio) conservano le penalità (intensificazione), valori alti le cancellano (diversificazione)

La taratura dei parametri puè essere reattiva

- applicare per lo stesso tempo diverse configurazioni
- replicare il procedimento dando più tempo alle configurazioni sperimentalmente migliori