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

Superare gli ottimi locali

Le euristiche di scambio *steepest descent* forniscono solo ottimi locali

Per migliorare, si può

- ripetere la ricerca (*Come evitare di seguire lo stesso percorso?*)
- proseguire la ricerca (*Come evitare di ricadere nello stesso ottimo?*)

Negli algoritmi costruttivi era possibile solo la ripetizione

Gli strumenti disponibili allo scopo sono i soliti:

- passi casuali
- memoria

Negli algoritmi costruttivi influenzavano $\mathrm{Ext}_A(x)$ e il criterio $\varphi_A(i, x)$

Negli algoritmi di scambio influenzano

1. la soluzione iniziale $x^{(0)}$ (multi-start, *ILS*, *VNS*)
2. l'intorno $N(x)$ (*VND*)
3. il criterio di scelta $\varphi(x, A, D)$ (*DLS*, *SA*, *TS*, *GLS*)

Condizione di termine

Se la ricerca si ripete o prosegue anziché terminare in un ottimo locale, idealmente potrebbe avere durata anche infinita

In pratica, si usano condizioni di termine "assolute"

1. un dato numero totale di ripetizioni della ricerca locale oppure
   un dato numero totale di esplorazioni dell'intorno
2. un dato tempo totale di esecuzione
3. un dato valore dell'obiettivo
4. un dato miglioramento dell'obiettivo rispetto alla soluzione iniziale

oppure "relative"

1. un dato numero di ripetizioni o esplorazioni dell'intorno dopo l'ultimo miglioramento del risultato f^*
2. un dato tempo di esecuzione dopo l'ultimo miglioramento
3. un dato valore minimo del rapporto fra miglioramento dell'obiettivo e numero di esplorazioni o tempo di esecuzione
   (*Es.:* f^* *migliora meno del* 1% *nelle ultime* 1 000 *esplorazioni*)

Per confrontare algoritmi si usano le assolute (tempo o numero di esplorazioni)

Modificare la soluzione iniziale

Si possono creare soluzioni iniziali diverse tra loro

- generandole casualmente
- applicando diverse euristiche costruttive
- modificando soluzioni generate dall'algoritmo di scambio

I vantaggi della generazione casuale sono

- semplicità concettuale
- rapidità per i problemi in cui è facile garantire l'ammissibilità
- controllo sulla distribuzione di probabilità nello spazio X in base a
  - costo degli elementi (per es., favorire gli elementi meno costosi)
  - frequenza degli elementi durante la ricerca passata, per favorire gli elementi più frequenti (intensificazione) o più rari (diversificazione)

  In questo modo si combinano i passi casuali e la memoria
- convergenza all'ottimo in tempo infinito

Gli svantaggi della generazione casuale sono

- qualità scarsa delle soluzioni iniziali (*non quelle finali!*)
- tempi lunghi prima di raggiungere l'ottimo locale

  Questo dipende dalla complessità dell'algoritmo di scambio
- lentezza per i problemi in cui l'ammissibilità è $\mathcal{NP}$-completa

Metodi multi-start

I metodi multi-start costituiscono l'approccio classico

- si progettano più euristiche costruttive
- ogni euristica costruttiva genera una soluzione iniziale
- ogni soluzione iniziale viene migliorata dall'algoritmo di scambio

Gli svantaggi sono

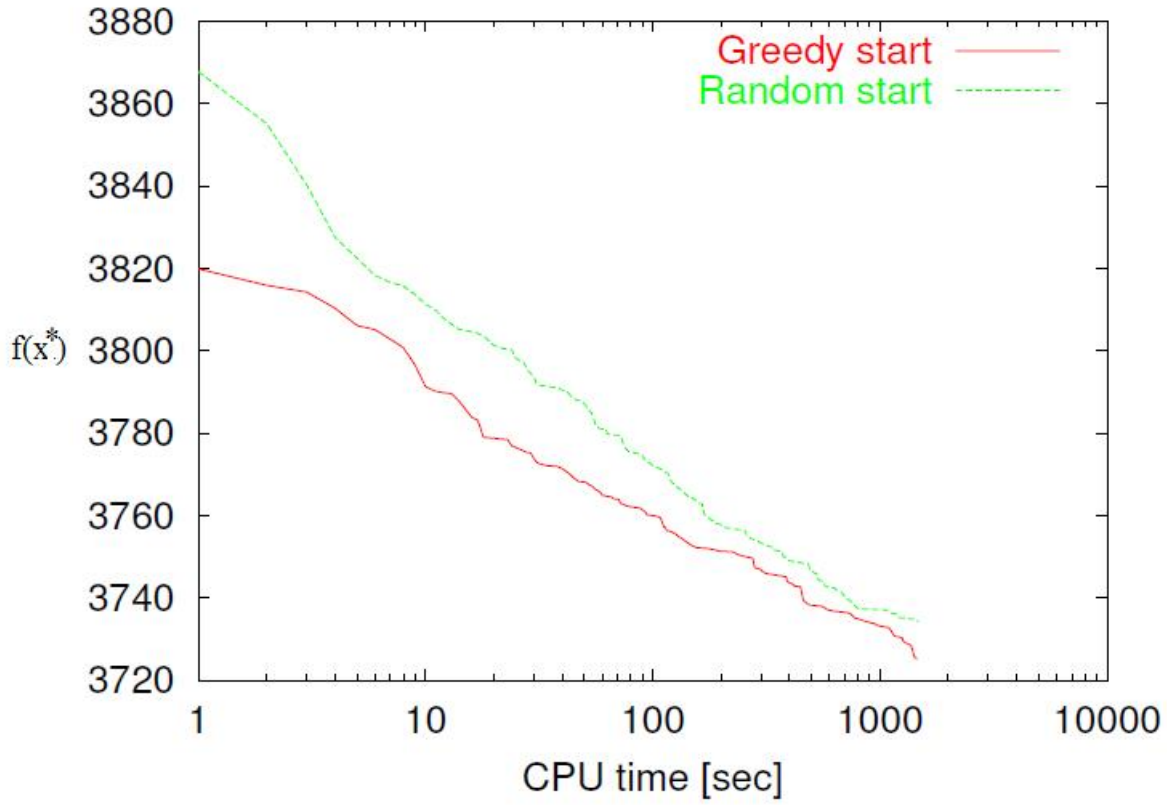
1. scarso controllo: le soluzioni generate tendono a somigliarsi
2. impossibilità di proseguire a oltranza: il numero di ripetizioni è fisso
3. sforzo di progetto elevato: bisogna inventare molti algoritmi diversi
4. nessuna garanzia di convergenza, nemmeno in tempo infinito

Per superare gli svantaggi, oggi si preferiscono metaeuristiche costruttive con memoria o passi casuali

GRASP e Ant System includono per definizione una procedura di scambio

Influenza della soluzione iniziale

Se l'euristica di scambio e il meccanismo di reinizializzazione sono buoni, la soluzione iniziale ha influenza solo nelle fasi iniziali della ricerca



Generare $x^{(0)}$ casualmente o con un'euristica costruttiva cambia poco

Sfruttare le soluzioni precedenti

L'idea è sfruttare la memoria delle soluzioni già visitate

- conservare delle soluzioni di riferimento, tipicamente l'ottimo locale migliore trovato sinora ed eventualmente altri
- generare la nuova soluzione iniziale modificando quelle di riferimento

I vantaggi di questo approccio sono

- controllo: si può rendere la modifica piccola o grande a piacere
- buona qualità: la soluzione di partenza è molto buona
- semplicità concettuale
- semplicità implementativa: la modifica può derivare dalle operazioni che definiscono l'intorno
- convergenza all'ottimo sotto opportune condizioni

Iterated Local Search (*ILS*)

La Iterated Local Search (*ILS*) richiede

- un'euristica di scambio *steepest descent* che produce ottimi locali
- una procedura di perturbazione che genera le soluzioni iniziali
- una condizione di accettazione che indica se cambiare la soluzione di riferimento x
- una condizione di terminazione

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

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

For $l := 1$ *to* ℓ *do*

  $x' :=$ Perturbate(x);

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

  If Accept(x', x) *then* $x := x'$;

  If $f(x') < f(x^*)$ *then* $x^* := x'$;

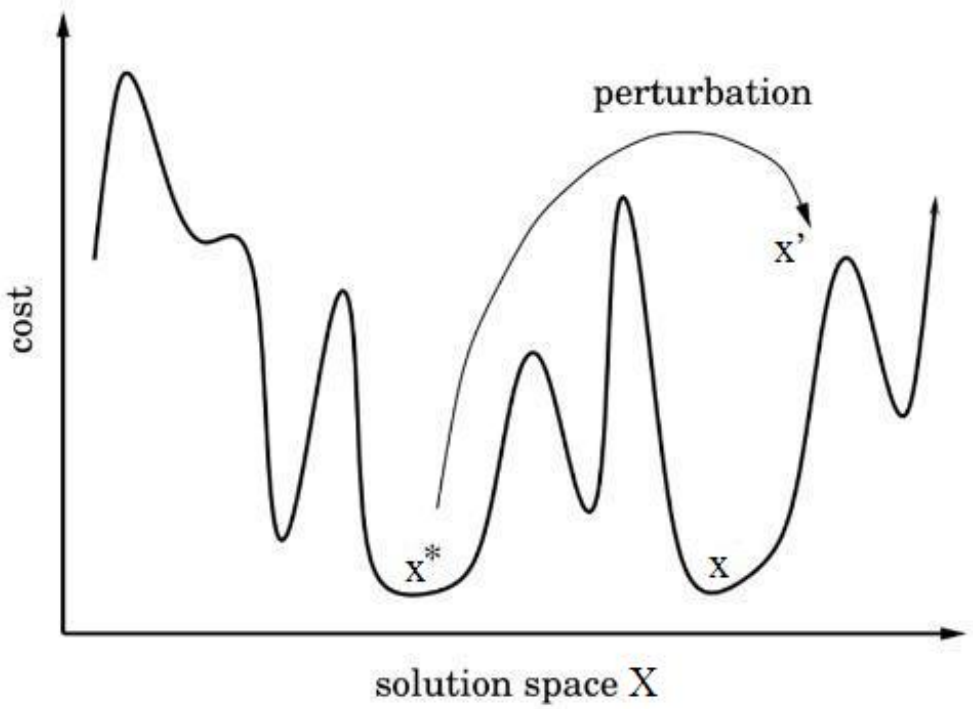
EndWhile;

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

Iterated Local Search (*ILS*)

L'idea è che

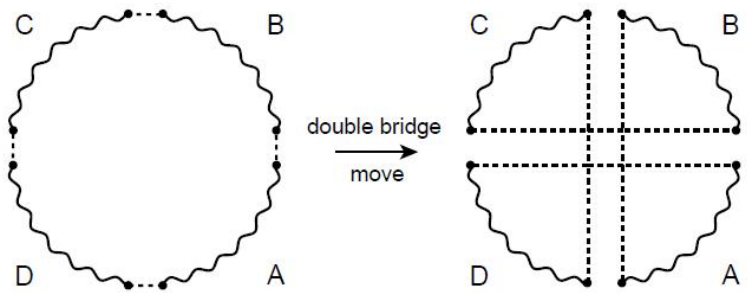
- l'euristica di scambio esplora rapidamente un bacino di attrazione, terminando in un ottimo locale
- la procedura di perturbazione passa a un altro bacino di attrazione
- la condizione di accettazione valuta se il nuovo ottimo locale è un punto di partenza promettente per la successiva perturbazione



Esempio: *ILS* per il *TSP*

Una classica applicazione della *ILS* al *TSP* usa

- euristica di scambio: *steepest descent* con l'intorno $N_{\mathcal{R}_2}$ o $N_{\mathcal{R}_3}$
- procedura di perturbazione: una mossa *double-bridge* che è un tipo particolare di 4-scambio



- criterio di accettazione: si migliora la miglior soluzione nota

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

Procedura di perturbazione

Sia $\mathcal{O}$ l'insieme che definisce l'intorno $N_{\mathcal{O}}$ dell'euristica di scambio

La procedura di perturbazione esegue un'operazione $o \in \mathcal{O}'$ scelta a caso

- deve essere $\mathcal{O}' \not\subseteq \mathcal{O}$, altrimenti l'euristica di scambio riporterebbe la soluzione x' all'ottimo locale iniziale x

Due tipiche definizioni di $\mathcal{O}'$ sono

- sequenze di $k > 1$ operazioni di $\mathcal{O}$
  (costa poco generare una sequenza casuale)
- operazioni concettualmente diverse
  (ad es., scambi di vertici anziché di archi)

La difficoltà principale della *ILS* è nel graduare la perturbazione

- troppo forte, trasforma la ricerca in un restart casuale
- troppo debole, riporta sempre la ricerca all'ottimo iniziale
  - perdendo la convergenza all'ottimo globale
  - sprecando tempo

Idealmente si vuole poter entrare in ogni bacino e uscire da ogni bacino

Criterio di accettazione

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

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

For $I := 1$ *to* ℓ *do*

  $x' :=$ Perturbate(x);

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

  If Accept(x', x) *then* $x := x'$;

  If $f(x') < f(x^*)$ *then* $x^* := x'$;

EndWhile;

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

Il criterio di accettazione bilancia intensificazione e diversificazione

- accettare solo soluzioni miglioranti favorisce l'intensificazione

  $$\text{Accept}(x', x) := (f(x') < f(x^*))$$

  La soluzione di riferimento è sempre la migliore trovata: $x = x^*$

- accettare qualsiasi soluzione favorisce la diversificazione

  $$\text{Accept}(x', x) := \textit{true}$$

  La soluzione di riferimento è sempre l'ultimo ottimo trovato: $x = x'$

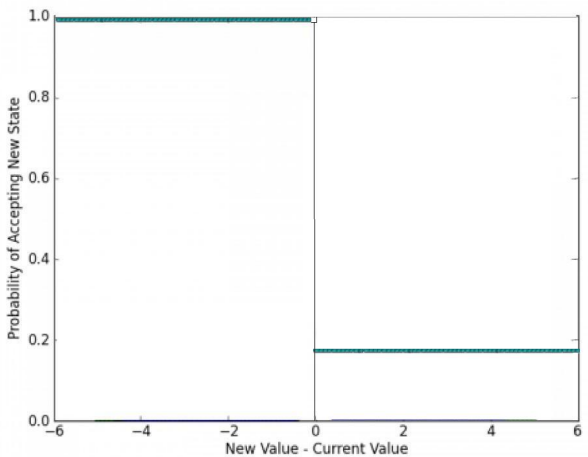
Criterio di accettazione

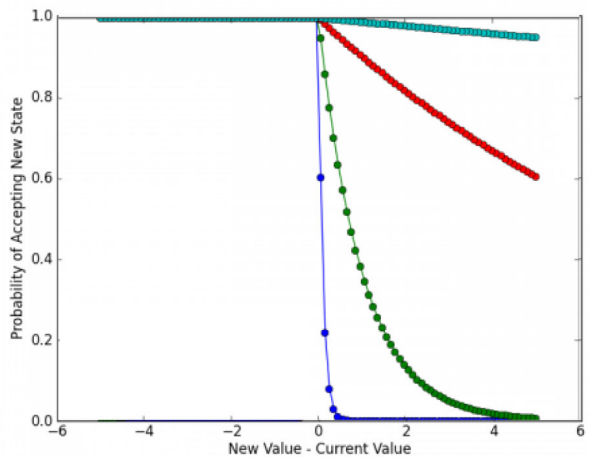
Strategie intermedie si possono definire in base a $\delta f = f(x') - f(x^*)$

- se $\delta f < 0$, si accetta x' sempre
- se $\delta f \geq 0$, si accetta x' con probabilità $\pi(\delta f)$, dove $\pi(\cdot)$ è una funzione non crescente

I casi più tipici sono:

- probabilità costante: $\pi(\delta f) = \bar{\pi} \in (0; 1)$ per ogni $\delta f \geq 0$
- probabilità monotona decrescente con $\pi(0) = 1$ e $\lim_{\delta f \to +\infty} \pi(\delta) = 0$





Si può usare anche la memoria, accettando x' più facilmente se molte iterazioni sono passate dall'ultimo miglioramento di x^*

Variable Neighbourhood Search (*VNS*)

Un metodo molto simile alla *ILS* è la *Variable Neighbourhood Search* proposta da Hansen e Mladenović (1997)

Le differenze principali tra *ILS* e *VNS* stanno nell'usare

- il criterio di accettazione stretto: $f(x') < f(x^*)$
- un meccanismo di perturbazione adattivo anziché fisso

Inoltre la *VNS* introduce spesso anche tecniche di modifica dell'intorno

Il meccanismo di perturbazione si basa su una gerarchia di intorni, cioè una famiglia di intorni d'ampiezza crescente rispetto a un parametro k

$$N_1 \subset N_2 \subset \ldots \subset N_k \subset \ldots N_{k_{\max}}$$

Tipicamente si usano

- gli intorni N_{H_k} basati sulla distanza di Hamming fra sottoinsiemi
- gli intorni $N_{\mathcal{O}_k}$ basati sulle sequenze di k operazioni tratte da un insieme $\mathcal{O}$ di operazioni elementari

La soluzione iniziale viene estratta casualmente da uno di questi intorni

Meccanismo adattivo di perturbazione

Si parla di *variable neighbourhood* perché l'intorno da cui si estrae la nuova soluzione iniziale varia in base ai risultati dell'euristica di scambio

- se trova soluzioni migliori, si usa l'intorno più piccolo, generando una soluzione iniziale molto vicina (intensificazione)
- se non trova soluzioni migliori, si usa un intorno un po' più grande, generando una soluzione iniziale po' più lontana (diversificazione)

Il metodo ha tre parametri

1. $k_{\min}$ individua l'intorno più piccolo da cui si generano nuove soluzioni
2. $k_{\max}$ individua l'intorno più grande da cui si generano nuove soluzioni
3. δk è la variazione del parametro k fra due intorni consecutivi usati

L'euristica di scambio adotta per efficienza l'intorno più piccolo possibile (N_1, *o comunque* N_k *con* $k \leq k_{\min}$)

Schema generale della *VNS*

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

- la soluzione di riferimento è sempre la miglior soluzione nota x^*
- la soluzione iniziale si ottiene estrandone una a caso dall'intorno corrente della soluzione di riferimento $N_k(x^*)$
- l'euristica di scambio produce un ottimo locale rispetto all'intorno base N
- se la miglior soluzione nota migliora, l'intorno corrente diventa $N_{k_{\min}}$
- altrimenti, si passa a un intorno più ampio $N_{k+\delta k}$, senza mai superare $N_{k_{\max}}$

Taratura dei parametri

Il valore di k_{min} deve essere

- abbastanza alto da far uscire dal bacino di attrazione corrente
- non così alto da far saltare i bacini di attrazione adiacenti

In genere si parte con $k_{\text{min}} = 1$, e poi si modula sperimentalmente

Il valore di k_{max} deve essere

- abbastanza alto da raggiungere qualsiasi bacino di attrazione utile
- non così alto da raggiungere regioni inutili dello spazio delle soluzioni

In genere si parte col diametro dello spazio di ricerca per l'intorno base: $\min(m, n - m)$ per l'*MDP*, n per *MAX-SAT* e *TSP*, ecc...
e poi si modula sperimentalmente

Il valore di δk deve essere

- abbastanza alto da raggiungere k_{max} in tempi ragionevoli
- non così alto da impedire a ogni valore di k di svolgere il suo ruolo

In genere si parte con $\delta k = 1$, e poi si modula sperimentalmente

Skewed VNS

Un criterio di accettazione più raffinato è accettare x' quando

$$f(x') < f(x) + \alpha d_H(x', x)$$

dove

- x è l'ottimo locale di partenza (soluzione di riferimento)
  (*non più sempre uguale a* x^*!)
- $d_H(x', x)$ è la distanza di Hamming fra x' e x
- $\alpha > 0$ è un opportuno parametro

Si favorisce la diversificazione accettando soluzioni peggioranti se lontane

- con $\alpha \approx 0$, si tende ad accettare solo soluzioni miglioranti
- con $\alpha \gg 0$, si tende ad accettare qualsiasi soluzione

Ovviamente, si possono anche adottare le strategie viste per la *ILS*
(*i due metodi sono sostanzialmente varianti*)