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

Potenziare gli algoritmi costruttivi

Su molti problemi, gli algoritmi costruttivi hanno forti limitazioni

Che fare, se non si vuole abbandonare del tutto lo schema?

Si può iterare lo schema ℓ volte, modificando gli elementi di base $\mathcal{F}_A$ e φ_A in modo da ottenere soluzioni $x^{[1]}, \ldots, x^{[\ell]}$ potenzialmente diverse

- si perde efficienza: si sommano i tempi di calcolo
- si acquista efficacia: si restituisce la soluzione migliore

È un trade-off da valutare attentamente!

Si può assumere un solo spazio di ricerca $\mathcal{F}_A$ senza perdere generalità

- si fondono i singoli spazi $\mathcal{F}_A^{[l]}$ in uno complessivo $\mathcal{F}_A = \bigcup_{l=1}^{\ell} \mathcal{F}_A^{[l]}$
- si estendono le singole funzioni di scelta da $\mathcal{F}_A^{[l]}$ a $\mathcal{F}_A$ ponendo

$$\varphi_A^{[l]}(i, x) = +\infty \text{ per ogni } (i, x) : x \cup \{i\} \in \mathcal{F}_A \setminus \mathcal{F}_A^{[l]}$$

Principali metaeuristiche costruttive

1. multi-start o restart: esegue una batteria di algoritmi $A^{[l]}$ diversi
   (*porta in fondo tanti tentativi e conserva il risultato migliore*)
2. roll-out: applica l'algoritmo A con $x^{(0)} = x \cup \{i\}$ anziché $x^{(0)} = \emptyset$, trova la soluzione $x_A(x \cup \{i\})$ e usa il valore per la scelta seguente
   (*prova ogni singola mossa lecita, vede dove termina e fa un passo*)
3. Adaptive Research Technique (*ART*) o Tabu Greedy: definisce $\mathcal{F}^{[l]}$ vietando alcune parti di $\mathcal{F}$ e lo modifica via via in base ai risultati
4. semigreedy e GRASP: usano una funzione di scelta casualizzata

$$\varphi_A^{[l]}\left(i, x, \omega^{[l]}\right)$$

5. Ant System (AS): usa una funzione di scelta casualizzata e dipendente dai risultati precedenti

$$\varphi_A^{[l]}\left(i, x, \omega^{[l]}, x_A^{[1]}, \ldots, x_A^{[l-1]}\right)$$

L'*ART* usa la memoria, il *GRASP* la casualità, l'*AS* entrambi

Multi-start

È un approccio classico, molto semplice e naturale:

- si definiscono diversi criteri di scelta $\varphi_A^{[l]}(i, x)$
- si applicano in sequenza gli algoritmi $A^{[l]}$ che ne derivano
- si restituisce la soluzione migliore

Spesso si usa quando $\varphi_A(i, x)$ contiene parametri numerici μ

Esempio: consideriamo il *TSP* su grafo completo

- criterio di inserimento: anziché $c_{s_i,k} + c_{k,s_{i+1}} - c_{s_i,s_{i+1}}$ si può usare

$$\min_{i \in \{1,\ldots,|x|\}} \gamma_{i,k} = \mu_1 \left(c_{s_i,k} + c_{k,s_{i+1}}\right) - (1 - \mu_1)\, c_{s_i,s_{i+1}}$$

  dove $\mu_1 \in [0; 1]$ modula la forza dei due termini in modo controllabile
- criterio di selezione: detta $d(x, k)$ la distanza del nodo k dal ciclo x, anziché $\min_{ik} \gamma_{ik}$ (euristica *CI*) o $\max_k d(k, x)$ (*FI*) si può usare

$$\max_{k \in N \setminus N_x} \varphi_A(k, x) = \mu_2 d(x, k) - (1 - \mu_2)\, \gamma_{i_k^*,k} \qquad \text{con } \mu_2 \in [0; 1]$$

Euristiche roll-out

Furono proposte da Bertsekas e Tsitsiklis (1997)

Data un'euristica costruttiva base A

- si parte con un sottoinsieme vuoto: $x^{(0)} = \emptyset$
- ad ogni passo $t = 1, 2, \ldots$
  - si estende la soluzione in ogni modo lecito: $x^{(t-1)} \cup \{i\}, \forall i \in \mathrm{Ext_A}(x)$
  - ad ogni estensione si applica l'euristica base e si calcola il valore della soluzione $x_A(x^{(t-1)} \cup \{i\})$, che fa da stima del risultato
  - si usa la stima come funzione di scelta $\varphi_A(i, x)$

$$i^{(t)} = \arg \min_{i \in \mathrm{Ext}_A(x^{(t-1)})} f\left(x_A(x^{(t-1)} \cup \{i\})\right)$$

- si termina quando $\mathrm{Ext}_A(x)$ è vuoto, come al solito

Si parla di euristica costruttiva single-step look-ahead

Lo schema è parametrico nell'euristica base scelta

Esempio: roll-out per il *SCP*

$$c = \begin{array}{|ccccc|}\hline 25 & 6 & 8 & 24 & 12 \\ \hline\end{array}$$

$$A = \begin{array}{|ccccc|}\hline 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ \hline\end{array}$$

1. si parte con il sottoinsieme vuoto: $x^{(0)} = \emptyset$
2. per ogni colonna i, si applica l'euristica costruttiva alla soluzione iniziale $x^0 \cup \{i\} = \{i\}$
   - per $i = 1$, si ottiene $x_A(\{1\}) = \{1\}$ di costo $f_A(\{1\}) = 25$
   - per $i = 2$, si ottiene $x_A(\{2\}) = \{2, 3, 5, 4\}$ di costo $f_A(\{2\}) = 50$
   - per $i = 3$, si ottiene $x_A(\{3\}) = \{3, 2, 5, 4\}$ di costo $f_A(\{3\}) = 50$
   - per $i = 4$, si ottiene $x_A(\{4\}) = \{4, 2, 5\}$ di costo $f_A(\{4\}) = 42$
   - per $i = 5$, si ottiene $x_A(\{5\}) = \{5, 2, 3, 4\}$ di costo $f_A(\{5\}) = 50$
3. la migliore è la prima soluzione, quindi $i^{(1)} = 1$
4. le righe sono coperte: l'algoritmo termina con la soluzione ottima

Proprietà delle euristiche roll-out

Il risultato dell'euristica roll-out domina quello dell'euristica base se opportune condizioni sono verificate

È possible ibridare euristiche roll-out e multi-start:
date più euristiche costruttive base $A^{[1]}, \ldots, A^{[\ell]}$

- si parte dal sottoinsieme vuoto
- ad ogni passo t
  - per ogni possible estensione $i \in \mathrm{Ext}_A(x^{(t-1)})$ si esegue ogni algoritmo base $A^{[l]}$ a partire da $x^{(t-1)} \cup \{i\}$
  - si ottiene una stima $f_{A^{[l]}}(x^{(t-1)} \cup \{i\})$
  - si esegue la mossa che produce la stima migliore

$$i^{(t)} = \arg \min_{l=1,\ldots,\ell} \min_{i \in \mathrm{Ext}_A(x^{(t-1)})} f\left(x_{A^{[l]}}(x^{(t-1)} \cup \{i\})\right)$$

- si termina quando $\mathrm{Ext}_A(x)$ è vuoto

L'euristica roll-out ibrida domina ogni euristica base

La complessità cresce fortemente rispetto all'euristica base A, ma resta polinomiale (al limite, si moltiplica per $|B|^2$)

Adaptive Research Technique

Fu proposta da Patterson et al (1998) per il *CMST* e mai ripresa da altri

Spesso nei primi passi un'euristica costruttive include elementi apparentemente buoni che portano a soluzioni finali molto cattive

- le euristiche roll-out provano a imporre vari elementi nella soluzione (*ma dopo un passo quelli ingannevoli potrebbero ancora parer buoni*)
- l'*ART* prova a vietare elementi allo scopo di impedire che elementi ingannevoli guidino il sottoinsieme x sul cammino sbagliato in $\mathcal{F}_A$

Inoltre, vietando elementi delle soluzioni già visitate si impedisce di ottenere soluzioni uguali o simili ad esse (diversificazione), ed è possibile

- ripetere l'euristica quasi indefinitamente
- modulare la distanza delle nuove soluzioni dalle vecchie

Adaptive Research Technique

L'*ART* richiede un'euristica costruttiva base A

Crea una lista di elementi vietati, inizialmente vuota ($l_i = -\infty, \forall i \in B$)

Ad ogni iterazione $l \in \{1, \ldots, \ell\}$

1. applica l'euristica base A evitando gli elementi vietati ($l - l_i \leq d$) e ottenendo quindi una soluzione $x^{[l]}$
2. decide con probabilità π se vietare o no ogni elemento $i \in x^{[l]}$
3. conserva per ogni elemento vietato l'iterazione l_i di inizio del divieto
4. rimuove i divieti più vecchi di d iterazioni (expiration time)
5. conserva la miglior soluzione trovata e le corrispondenti l_i
6. apporta eventuali modifiche ausiliarie alla lista dei divieti

Al termine, restituisce la miglior soluzione trovata

Taratura dei parametri

Il metodo ha almeno tre parametri

- il numero totale di iterazioni ℓ
- la durata d del divieto
- la probabilità π del divieto

Come assegnare loro dei valori sensati?

Valutare sperimentalmente le prestazioni con valori diversi ha svantaggi

1. richiede lunghe fasi sperimentali, perché
   il numero di configurazioni cresce combinatorialmente con
   - il numero dei parametri
   - il numero dei valori testati per ogni parametro

     (*più il risultato è sensibile, più valori bisogna testare*)
2. rischia l'overfitting, cioè di giudicare buoni in senso assoluto
   valori buoni solo sulle istanze di benchmark considerate

L'eccesso di parametri è un aspetto indesiderabile, e spesso rivela uno studio insufficiente del problema e dell'algoritmo

Meccanismi di aggiornamento dei parametri

Un'alternativa alla taratura a priori dei parametri consiste nell'ideare uno schema ragionevole di aggiornamento dei valori dei parametri

Si possono esplorare diversi valori di π e d, per blocchi di ℓ iterazioni

1. abbassare gradualmente d: $d^{(r)} := \mu_d d^{(r-1)}$ con $\mu_d \in (0,1)$
   (*la durata di ogni divieto cala per lasciar spazio ad altri divieti*)
2. aumentare gradualmente π: $\pi^{(r)} := \mu_\pi \pi^{(r-1)}$ con $\mu_\pi > 1$
   (*si vietano sempre più elementi per diversificare sempre più*)

o viceversa (queste sono le strategie proposte da Patterson et al.)

Questi schemi automatici rendono l'euristica più robusta, ma

- allungano i tempi di calcolo (*rispetto alla configurazione singola*)
- sostituiscono pochi parametri con molti
  - il numero di valori testati per i parametri originali
  - il valore iniziale $d^{(1)}$ e il tasso μ_d di aggiornamento di d
  - il valore iniziale $\pi^{(1)}$ e il tasso μ_π di aggiornamento di π

Conviene se il risultato è meno sensibile ai nuovi parametri che ai vecchi

Intensificazione

Una diversificazione eccessiva può ostacolare la scoperta dell'ottimo

Intensificazione è il meccanismo opposto alla diversificazione, cioè la concentrazione della ricerca sui sottoinsiemi più promettenti

I meccanismi di intensificazione possono essere basati sui dati:
se si considerano promettenti gli elementi meno costosi

- si diminuisce la probabilità π del divieto per tali elementi
- si diminuisce la durata d del divieto per tali elementi

Oppure possono essere basati sulla memoria:
se si considerano promettenti gli elementi delle migliori soluzioni note

- si esentano dal divieto i loro elementi comuni
- si avvia ogni blocco di ℓ iterazioni con la lista di divieti l_i associata alla miglior soluzione nota, anziché con una lista vuota

Diversificazione e intensificazione svolgono funzioni complementari

Esempio: *ART* per il *SCP*

$$c = \begin{array}{|ccccc|} \hline 25 & 6 & 8 & 24 & 12 \\ \hline \end{array}$$

$$A = \begin{array}{|ccccc|} \hline 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

1. l'euristica base trova la soluzione $x^{[1]} = \{2, 3, 5, 4\}$ di costo $f\left(x^{[1]}\right) = 50$; si vieta (casualmente) la colonna 2
2. l'euristica base trova la soluzione $x^{[2]} = \{3, 1\}$ di costo $f\left(x^{[2]}\right) = 33$ si vieta (casualmente) la colonna 3
3. l'euristica base trova la soluzione $x^{[3]} = \{1\}$ di costo $f\left(x^{[2]}\right) = 25$, che è ottima
4. ...

Ovviamente, al passo 2 si sarebbe potuta vietare la colonna 1

Euristiche semi-greedy

Un'euristica costruttiva non esatta ha almeno un passo t in cui sceglie un elemento che non porta ad alcuna soluzione ottima

Siccome sceglie sempre l'elemento migliore in base al criterio di scelta

$$i^{(t)} = \arg \min_{i \in \mathrm{Ext}_A(x)} \varphi_A(i, x)$$

ne segue che $\varphi_A(i, x)$ è sbagliata, ma forse non del tutto

L'algoritmo semi-greedy (Hart e Shogan, 1987) si basa sull'idea che l'elemento che mantiene la via verso l'ottimo è fra i migliori per $\varphi_A(i, x)$, anche se non rigorosamente il migliore

Ma come sapere quale?

Se non si può correggere $\varphi_A(i, x)$, si sceglie a caso in $\mathrm{Ext}_A(x)$

$$i^{(t)} = \arg \min_{i \in \mathrm{Ext}_A(x)} \varphi_A(i, x, \omega) \text{ con } \omega \in \Omega_x$$

con una distribuzione di probabilità che favorisca i migliori $\varphi_A(i, x)$

Euristiche semi-greedy

Poiché $\mathrm{Ext}_A(x)$ fa da insieme degli eventi, ed è finito, si assegna

- una probabilità $\pi_A(i,x)$ alla scelta di ciascun $i \in \mathrm{Ext}_A(x)$
- privilegiando gli elementi con valori migliori di $\varphi_A(i,x)$

$$\varphi_A(i,x) \leq \varphi_A(j,x) \Leftrightarrow \pi_A(i,x) \geq \pi_A(j,x)$$

per ogni $i,j \in \mathrm{Ext}_A(x), x \in \mathcal{F}$

Questo approccio ha importanti proprietà

- se c'è un cammino da $\emptyset$ a X^*, può raggiungere una soluzione ottima
  (*questa è una condizione fondamentale*)
- rieseguendo l'euristica più volte, genera soluzioni diverse e
  la probabilità di raggiungere l'ottimo cresce via via
  (*ad ogni tentativo, cala la probabilità di sbagliare sempre strada*)

Grafi di costruzione stocastici

Il grafo di costruzione consente una trattazione formale del procedimento

- i nodi sono i sottoinsiemi validi $x \in \mathcal{F}_A$
- gli archi legano sottoinsiemi separati da una mossa:

$$x \in \mathcal{F}_A \text{ e } x \cup \{i\} \text{ con } i \in \mathrm{Ext}_A(x)$$

o più in generale $x \in \mathcal{F}_A$ e $x \cup B^+ \setminus B^-$ con $(B^+, B^-) \in \mathrm{Ext}_A(x)$

- l'arco $(x, x \cup \{i\})$ ha peso $\varphi_A(i, x)$

I cammini massimali rappresentano l'esecuzione di un'euristica costruttiva

- partono dal sottoinsieme vuoto $\emptyset$
- terminano in un sottoinsieme non ampliabile,
  che spesso è una soluzione ammissibile (non sempre!)

Un'euristica costruttiva con passi casuali associa ad ogni arco $(x, x \cup \{i\})$
la probabilità $\pi_A(i, x)$ di estendere il sottoinsieme x con l'elemento i

- la somma delle probabilità per gli archi uscenti da ogni nodo è 1

Convergenza all'ottimo

La probabilità di seguire un cammino γ è il prodotto di quelle sugli archi

$$\prod_{(y,y\cup\{i\}\in\gamma)} \pi_A(i,y)$$

La probabilità di ottenere un sottoinsieme x è la somma di quelle dei cammini Γ_x che lo visitano

$$\sum_{\gamma\in\Gamma_x} \prod_{(y,y\cup\{i\}\in\gamma)} \pi_A(i,y)$$

Due risultati fondamentali per le euristiche costruttive con passi casuali

1. l'euristica ha probabilità non nulla di raggiungere l'ottimo se e solo se esiste almeno un cammino di probabilità non nulla da $\emptyset$ a X^*
2. la probabilità di raggiungere l'ottimo tende a 1 per $\ell \to +\infty$
   (*la probabilità di non raggiungerlo ciascuna volta va elevata a* ℓ)

Queste euristiche sono probabilisticamente approssimativamente complete

Convergenza all'ottimo

Random walk è un'euristica costruttiva in cui
le probabilità sugli archi uscenti da ogni nodo sono uniformi

- se un cammino verso l'ottimo esiste, la random walk lo trova
- il tempo necessario può essere lunghissimo

Si ricordi che l'algoritmo esaustivo è esatto e richiede tempo finito

Un'euristica costruttiva deterministica azzera tutte le probabilità
tranne quelle sugli archi di un solo cammino

- trova l'ottimo solo se gode di proprietà specifiche

In generale, le euristiche costruttive con passi casuali favoriscono gli archi più promettenti e sfavoriscono gli altri

- accelerano il tempo medio di convergenza
- riducono la garanzia di convergenza nel caso pessimo

C'è un trade-off fra valore medio e pessimo dei risultati

L'introduzione di archi a probabilità zero può sbarrare la via per l'ottimo

Semi-greedy e GRASP

GRASP, cioè Greedy Randomized Adaptive Search Procedure (Feo e Resende, 1989) è una variante sofisticata dell'euristica semi-greedy

- *Greedy* indica che usa un'euristica costruttiva base
- *Randomized* indica che l'euristica base fa passi casuali
- *Adaptive* indica che l'euristica usa un criterio di scelta adattivo $\varphi_A(i, x)$, che dipende anche da x (non è strettamente necessario)
- *Search* indica che alterna all'euristica costruttiva l'esecuzione di euristiche di scambio (diversamente dall'euristica semi-greedy)

L'uso di euristiche ausiliarie di scambio migliora fortemente i risultati

Nel seguito, ignoriamo questo aspetto, che approfondiremo in altre lezioni

Quale funzione di probabilità?

Diverse funzioni $\pi_A(i,x)$ rispettano la monotonia rispetto a $\varphi_A(i,x)$

$$\varphi_A(i,x) \leq \varphi_A(j,x) \Leftrightarrow \pi_A(i,x) \geq \pi_A(j,x)$$

- probabilità uniforme: ogni arco uscente da x ha ugual $\pi_A(i,x)$; l'algoritmo esegue un cammino casuale in $\mathcal{F}_A$ (random walk)
- Heuristic-Biased Stochastic Sampling (*HBSS*):
  - ordina gli archi uscenti da x per valori non crescenti di $\varphi_A(i,x)$
  - assegna probabilità decrescenti secondo la posizione nell'ordine in base a uno schema semplice (lineare, esponenziale, ecc. . .)
- Restricted Candidate List (*RCL*):
  - ordina gli archi uscenti da x per valori non crescenti di $\varphi_A(i,x)$
  - inserisce i primi archi in una lista (*Quanti?*)
  - assegna probabilità uniformi agli archi della lista, nulle agli altri

La strategia più comune è la *RCL*, anche se gli archi di probabilità nulla potenzialmente le fanno perdere la convergenza globale all'ottimo

Definizione della RCL

Quanti elementi entrano a far parte della *RCL*?

Vi sono due strategie principali

- cardinalità: la RCL contiene i migliori μ elementi di $\mathrm{Ext}_A(x)$, dove $\mu \in \{1, \ldots, |B|\}$ è un parametro deciso dall'utente
  - con $\mu = 1$, si ottiene l'euristica costruttiva base
  - con $\mu = |B|$ ($\geq |\mathrm{Ext}(x)|$ per ogni x), si ottiene la *random walk*
- valore: la RCL contiene tutti gli elementi di Ext(x) di valore compreso fra $\varphi_{\min}$ e $\varphi_{\min} + \mu(\varphi_{\max} - \varphi_{\min})$ dove

$$\varphi_{\min}(x) = \min_{i \in \mathrm{Ext}(x)} \varphi_A(i, x) \qquad \varphi_{\max}(x) = \max_{i \in \mathrm{Ext}(x)} \varphi_A(i, x)$$

  e $\mu \in [0; 1]$ è un parametro deciso dall'utente
  - con $\mu = 0$, si ottiene l'euristica costruttiva base
  - con $\mu = 1$, si ottiene la *random walk*

Schema general del *GRASP*

Algorithm GRASP(I)

$x^* := \emptyset$; $f^* := +\infty$; { Miglior soluzione trovata finora }

For $I = 1$ *to* ℓ *do*

{ Euristica costruttiva con passi casuali }

$x := \emptyset$;

While $\mathrm{Ext}(x) \neq \emptyset$ *do*

$\varphi_i := \varphi_A(i, x)$ per ogni $i \in \mathrm{Ext}(x)$

$L :=$ Sort($\mathrm{Ext}(x), \varphi$);

$\pi :=$ AssignProbabilities(L, μ); { o costruisce una RCL }

$i :=$ RandomExtract($\mathrm{Ext}(x), \pi$);

$x := x \cup \{i\}$;

EndWhile;

$x :=$ Search(x);

If $x \in X$ and $f(x) < f^*$ *then* $x^* := x$; $f^* := f(x)$;

EndFor;

Return (x^*, f^*);

Esempio: *GRASP* per il *SCP*

$$c \quad \begin{array}{|ccccc|} \hline 25 & 6 & 8 & 24 & 12 \\ \hline \end{array}$$

$$A \quad \begin{array}{|ccccc|} \hline 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

1. si parte con il sottoinsieme vuoto: $x^{(0)} = \emptyset$
2. si costruisce una *RCL* con $\mu = 2$ candidati: le colonne 2 ($\varphi_2 = 2$) e 3 ($\varphi_3 = 4$); si sceglie (casualmente) la colonna 3
3. si costruisce una *RCL* con $\mu = 2$ candidati: le colonne 2 ($\varphi_2 = 3$) e 1 ($\varphi_3 = 6.25$); si sceglie (casualmente) la colonna 1
4. la soluzione ottenuta è $x = \{2, 1\}$ di costo $f(x) = 33$

Con $\mu = 2$, non si può ottenere la soluzione ottima; con $\mu = 3$ si può

In effetti, si può anche con $\mu = 2$ se si applica una fase distruttiva

Taratura reattiva dei parametri

Ancora una volta ci sono parametri da tarare:

- il numero di iterazioni ℓ
- il valore μ che determina la dimensione della *RCL*

Un'idea è sfruttare la memoria, cioè apprendere dai risultati precedenti

1. si fissa ℓ e si scelgono m configurazioni di parametri $\mu_1, \ldots, \mu_m$
2. si pone $\ell_r = \ell/m$ per ogni $r = 1, \ldots, m$
3. si prova ogni configurazione μ_r per ℓ_r iterazioni
4. si valuta la media campionaria $\bar{f}(\mu_r)$ dei risultati ottenuti con μ_r
5. si rimodula il numero di iterazioni ℓ_r per ogni μ_r in base a $\bar{f}(\mu_r)$

$$\ell_r = \frac{\frac{1}{\bar{f}(\mu_r)}}{\sum_{s=1}^{m} \frac{1}{\bar{f}(\mu_s)}} \ell \text{ per } r = 1, \ldots, m$$

   in modo che le configurazioni più efficaci facciano più iterazioni

6. si ripete l'intero procedimento, tornando al punto 3, per R volte

Metodi di perturbazione dei costi

Anziché proibire o forzare alcune scelte, oppure variarne la probabilità, è possible modificare l'appetibilità delle scelte disponibili

Data un'euristica costruttiva base A, ad ogni passo dell'iterazione l

- si modula il criterio di scelta $\varphi_A(i,x)$ con un fattore $\tau_A^{[l]}(i,x)$

$$\psi_A^{[l]}(i,x) = \frac{\varphi_A(i,x)}{\tau_A^{[l]}(i,x)}$$

- si aggiorna $\tau_A^{[l]}(i,x)$ in base a un insieme $\tilde{X}^{[l]}$ di soluzioni precedenti

Gli elementi con $\varphi_A(i,x)$ migliore tendono a rimanere favoriti, ma $\tau_A^{[l]}(i,x)$ modula questo effetto, spingendo verso la

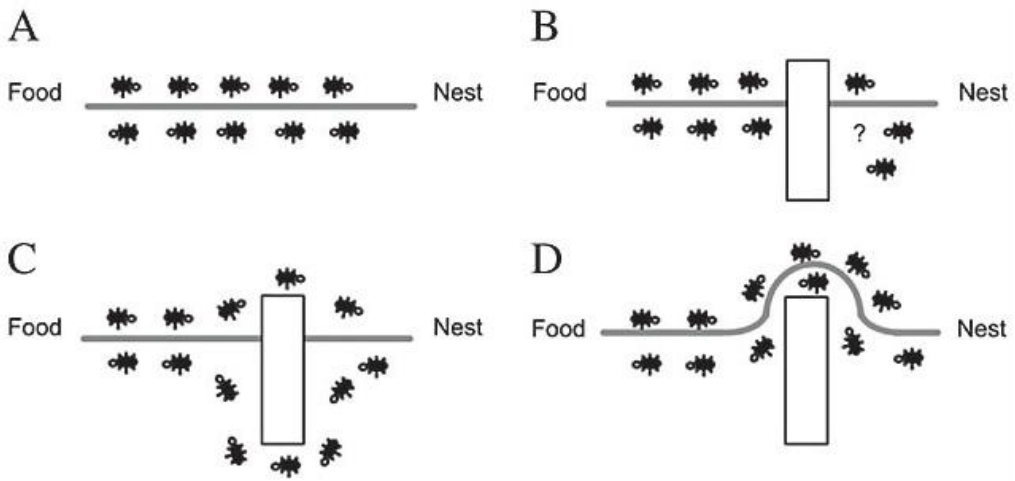
- intensificazione: se si aumenta $\tau_A^{[l]}(i,x)$ per gli elementi più frequenti in modo da ottenere soluzioni simili alle precedenti
- diversificazione se si diminuisce $\tau_A^{[l]}(i,x)$ per gli elementi più frequenti in modo da ottenere soluzioni diverse dalle precedenti

Si possono eseguire più euristiche in parallelo con $\tau_A^{[l,g]}(i,x)$ diversi e/o $\tau_A^{[l,g]}(i,x)$ può essere in parte casuale

Ant Colony Optimization

Fu ideato da Dorigo, Maniezzo e Colorni nel 1991
ispirandosi al comportamento sociale delle formiche

Stigmergia = comunicazione indiretta fra agenti nella quale essi sono stimolati e guidati dai risultati delle azioni proprie e altrui



Ogni agente è un'applicazione dell'euristica costruttiva base

- lascia sui dati una traccia che dipende dalla soluzione prodotta
- compie scelte influenzate dalle tracce lasciate dagli altri agenti

Le scelte dell'agente hanno anche una componente casuale

Traccia

Come nell'euristica semi-greedy

- è data un'euristica costruttiva base A
- ogni passo compie una scelta casuale

Diversamente dall'euristica semi-greedy

- ogni iterazione I lancia f volte l'euristica A (popolazione)
- tutte le scelte di $\mathrm{Ext}_A(x)$ sono lecite (*non c'è RCL*)
- la probabilità $\pi_A(i, x)$ dipende
  1. dal criterio di scelta $\varphi_A(i, x)$
  2. da un'informazione ausiliaria $\tau_A(i, x)$ detta traccia prodotta nelle iterazioni $< I$, e talvolta dagli altri agenti nella stessa iterazione

La traccia è inizialmente uniforme ($\tau_A(i, x) = \tau_0$), e viene poi modulata

- aumentandola per favorire scelte promettenti
- diminuendola per evitare scelte troppo ripetitive

Per semplicità, la traccia $\tau_A(i, x)$ non è associata a ogni arco $(x, x \cup \{i\})$, ma è identica per blocchi di archi (per esempio, dipende solo da i)

Scelta casuale

Anziché scegliere l'elemento migliore secondo il criterio $\varphi_A(i,x)$,
si estrae i da $\mathrm{Ext}_A(x)$ con probabilità

$$\pi_A(i,x) = \frac{\eta_A(i,x)^{\mu_\eta}\,\tau_A(i,x)^{\mu_\tau}}{\sum_{j\in\mathrm{Ext}_A(x)} \eta_A(j,x)^{\mu_\eta}\,\tau_A(j,x)^{\mu_\tau}}$$

dove

- il denominatore normalizza la probabilità
- si definisce visibilità la funzione ausiliaria

$$\eta_A(i,x) = \begin{cases} \varphi_A(i,x) & \text{per problemi di massimizzazione} \\ \dfrac{1}{\varphi_A(i,x)} & \text{per problemi di minimizzazione} \end{cases}$$

Le scelte promettenti hanno visibilità alta

- i parametri μ_τ e μ_η modulano il peso dei due termini

Informazioni date e informazioni apprese

I parametri μ_η e μ_τ regolano il peso dei dati e della memoria

- $\mu_\eta \approx 0$ e $\mu_\tau \approx 0$ spingono verso la casualità
  (*quindi, diversificando*)
- $\mu_\eta \gg \mu_\tau$ favorisce i dati: si tende all'euristica costruttiva base
  (*quindi, intensificando verso le soluzioni più immediate*)
- $\mu_\eta \ll \mu_\tau$ favorisce la memoria: si tende a ritrovare soluzioni di $\tilde{X}^{[l]}$
  (*quindi, intensificando verso le ultime soluzioni trovate*)

La variante detta Ant Colony System sceglie ad ogni passo

- con probabilità q l'elemento che massimizza $\eta_A(i,x)\,\tau_A(i,x)^{\mu_\tau}$
- con probabilità $(1-q)$ un elemento estratto uniformemente a caso

ottenendo gli stessi comportamenti estremi

- con $q \approx 0$, scelte casuali
- con $q \approx 1$ e $\mu_\tau \approx 0$, privilegia i dati
- con $q \approx 1$ e μ_τ alto, privilegia le scoperte fatte via via

Aggiornamento della traccia

Ad ogni iterazione $l \in \{1, \ldots, \ell\}$

1. si eseguono $\mathfrak{f}$ istanze dell'euristica base A
2. si gestisce un sottoinsieme $\tilde{X}^{[l]}$ di soluzioni, i cui elementi saranno favoriti nelle iterazioni seguenti
3. si aggiorna la traccia secondo la formula

$$\tau_A(i,x) := (1-\rho)\,\tau_A(i,x) + \rho \sum_{y \in \tilde{X}^{[l]} : i \in y} F_A(y)$$

dove $\rho \in [0;1]$ è un parametro di oblio

- per $\rho = 1$, la vecchia traccia viene cancellata interamente
- per $\rho = 0$, la vecchia traccia viene conservata senza modifiche

e $F_A(y)$ è una funzione di fitness che esprime la qualità della soluzione y e garantisce $F > \tau$ (spesso $F(y) = Q/f(y)$ con Q costante opportuna)

L'aggiornamento ha due effetti

1. aumenta la traccia sugli elementi inclusi nelle soluzioni di $\tilde{X}^{[l]}$
2. diminuisce la traccia sugli elementi che non vi appartengono

Oblio, diversificazione e intensificazione

Anche il coefficiente di oblio modula il comportamento fra due estremi

- un oblio alto ($\rho \approx 1$) induce diversificazione
  Infatti, cancella la traccia accumulata, indicando che
  - ci si fida poco delle soluzioni ottenute sinora
  - si vogliono esplorare soluzioni diverse da loro
- un oblio basso ($\rho \approx 0$) induce intensificazione
  Infatti, conserva la traccia accumulata, indicando che
  - ci si fida molto delle soluzioni ottenute sinora
  - si vogliono esplorare soluzioni simili a loro

Scelta delle soluzioni che influenzano la traccia

L'insieme $\tilde{X}^{[l]}$ raccoglie le soluzioni vicino a cui si vuole cercare

- nell'Ant System classico, sono tutte le soluzioni dell'iterazione $l-1$
- nei metodi elitari sono le migliori soluzioni note
  - la miglior soluzione dell'iterazione $l-1$
  - la miglior soluzione di tutte le iterazioni $< l$

Con i metodi elitari

- si ottengono risultati migliori nelle prime iterazioni
- si rischia una convergenza prematura, per evitare la quale occorrono accorgimenti aggiuntivi

Alcune varianti dell'Ant System

- $\mathcal{MAX} - \mathcal{MIN}$ Ant System: impone alla traccia un intervallo di valori limitato $[\tau_{\min}; \tau_{\max}]$, tarato sperimentalmente
- HyperCube Ant Colony Optimization (HC-ACO): normalizza la traccia fra 0 e 1
- Ant Colony System: aggiorna la traccia su due livelli
  - l'aggiornamento globale (già visto) si esegue ad ogni iterazione ℓ

    Lo scopo è intensificare la ricerca
  - l'aggiornamento locale si esegue ad ogni applicazione $g \in \{1, \ldots, \mathfrak{f}\}$ dell'euristica base per scoraggiare scelte uguali nelle seguenti

$$\tau_A(i, x) := (1 - \rho)\, \tau_A(i, x) \qquad \text{per ogni } i \in x_A^{[l,g]}$$

Lo scopo è diversificare la ricerca

Schema generale dell'*Ant System*

Algorithm AntSystem(I)
$x^* := \emptyset$; $f^* := +\infty$; { Miglior soluzione trovata finora }
InizializzaTraccia(τ_A);
For $l = 1$ *to* ℓ *do*
  For $g = 1$ *to* $\mathfrak{f}$ *do*
    $x := A(I, \tau_A)$; { Euristica base con passi casuali e memoria }
    $x :=$ Search(x); { Miglioramento con euristica di scambio }
    If $f(x) < f^*$ *then* $x^* := x$; $f^* := f(x)$;
    $\tau_A :=$ AggiornamentoLocale(τ_A, x, ρ); { Aggiornamento locale della traccia }
    $\tilde{X}^{[l]} :=$AggiornaSoluzioniInfluenti($\tilde{X}^{[l]}, x$);
  EndFor;
  $\tau_A :=$ AggiornamentoGlobale($\tau_A, \tilde{X}^{[l]}, \rho$); { Aggiornamento globale della traccia}
EndFor;
Return (x^*, f^*);

Convergenza all'ottimo

Alcune varianti di Ant System convergono all'ottimo con probabilità 1 (Gutjahr, 2002)

L'analisi si basa sul grafo di costruzione

- la traccia $\tau_A(i, x)$ si deposita sugli archi $(x, x \cup \{i\})$
- non si usa nessuna informazione sui dati, cioè $\eta_A(i, x) \equiv 1$ (*questa strana ipotesi semplifica i calcoli, ma non è necessaria*)
- $\tau^{[l]}$ è la funzione traccia all'inizio dell'iterazione l
- $\gamma^{[l]}$ è il miglior cammino nel grafo alla fine dell'iterazione l,
- $\left(\tau^{[l]}, \gamma^{[l-1]}\right)$ è lo stato di un processo di Markov non omogeneo:
  - la probabilità di ogni stato dipende solo dall'iterazione precedente
  - il processo è non omogeneo perché la dipendenza varia con l

La dimostrazione conclude che per $\ell \to +\infty$, con probabilità 1

1. almeno un'applicazione segue un cammino ottimo in $\mathcal{F}$
2. la traccia τ tende a un massimo lungo uno dei cammini ottimi, a zero sugli altri archi

Prima variante con convergenza globale

La traccia viene aggiornata con un coefficiente di oblio variabile

$$\tau^{[l]}(i,x) := \begin{cases} \left(1-\rho^{[l-1]}\right)\tau^{[l-1]}(i,x) + \rho^{[l-1]}\dfrac{1}{|\gamma^{*[l-1]}|} & \text{se } (x, x\cup\{i\}) \in \gamma^{[l-1]} \\ \left(1-\rho^{[l-1]}\right)\tau^{[l-1]}(i,x) & \text{altrimenti} \end{cases}$$

dove $\gamma^{[l-1]}$ è il miglior cammino nel grafo trovato fino all'iterazione $l-1$

Se l'oblio decresce abbastanza lentamente, cioè se

$$\rho^{[l]} \leq 1 - \frac{\log l}{\log(l+1)} \qquad \text{e} \qquad \sum_{l=1}^{+\infty} \rho^{[l]} = +\infty$$

allora con probabilità 1 lo stato converge a (τ^*, γ^*), dove

- γ^* è il cammino ottimo nel grafo di costruzione
- $\tau^*(i,x) = \dfrac{1}{|\gamma^*|}$ per $(x, x\cup\{i\}) \in \gamma^*$, 0 altrimenti

Seconda variante con convergenza globale

Alternativamente, se l'oblio ρ rimane costante,
ma si impone alla traccia una soglia minima lentamente decrescente

$$\tau(i,x) \geq \frac{c_l}{\log(l+1)} \quad \text{e} \quad \lim_{l \to +\infty} c_l = c \in (0,1)$$

allora con probabilità 1 lo stato converge a (τ^*, γ^*)

Qui è la soglia minima a limitare l'azione dell'oblio

In pratica, tutti gli algoritmi finora proposti in letteratura

- associano la traccia non a singoli archi $(x, x \cup \{i\})$, ma a gruppi (per es., al singolo elemento i)
- usano parametri ρ e $\tau_{\min}$ costanti

dunque non garantiscono la convergenza

La traccia τ, e quindi π, può tendere a zero su ogni cammino ottimo troppo velocemente per garantire la convergenza asintotica dell'algoritmo