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

Il corso di Algoritmi Euristici

È un corso relativamente nuovo (terza edizione) e non molto frequente

- rarissimo nelle università italiane, poco frequente in quelle straniere
- spesso ricade nei corsi di Algoritmi o Ricerca Operativa (con limitazioni di spazio e di solito procedendo per esempi)
- spesso gli algoritmi euristici per un'applicazione sono discussi nel corso dedicato all'applicazione (perdendo di vista il quadro generale)

Questo corso di propone di

- evitare l'idea degli algoritmi euristici come ricette pronte e specifiche
- coglierne gli aspetti comuni e generali
- imparare a progettarli creativamente e a valutarne le prestazioni

Il corso ricade nell'indirizzo di Analytics e ottimizzazione

- questo influisce sul taglio con cui la materia verrà presentata
- ma il corso si presta a essere seguito da studenti di indirizzi diversi

Analytics e ottimizzazione

Quando occorre prendere una decisione nella quale

- i dati da considerare sono molti
- le scelte possibili sono molte
- i costi di una scelta errata sono alti

la strategia corretta è prima modellare e calcolare, poi decidere

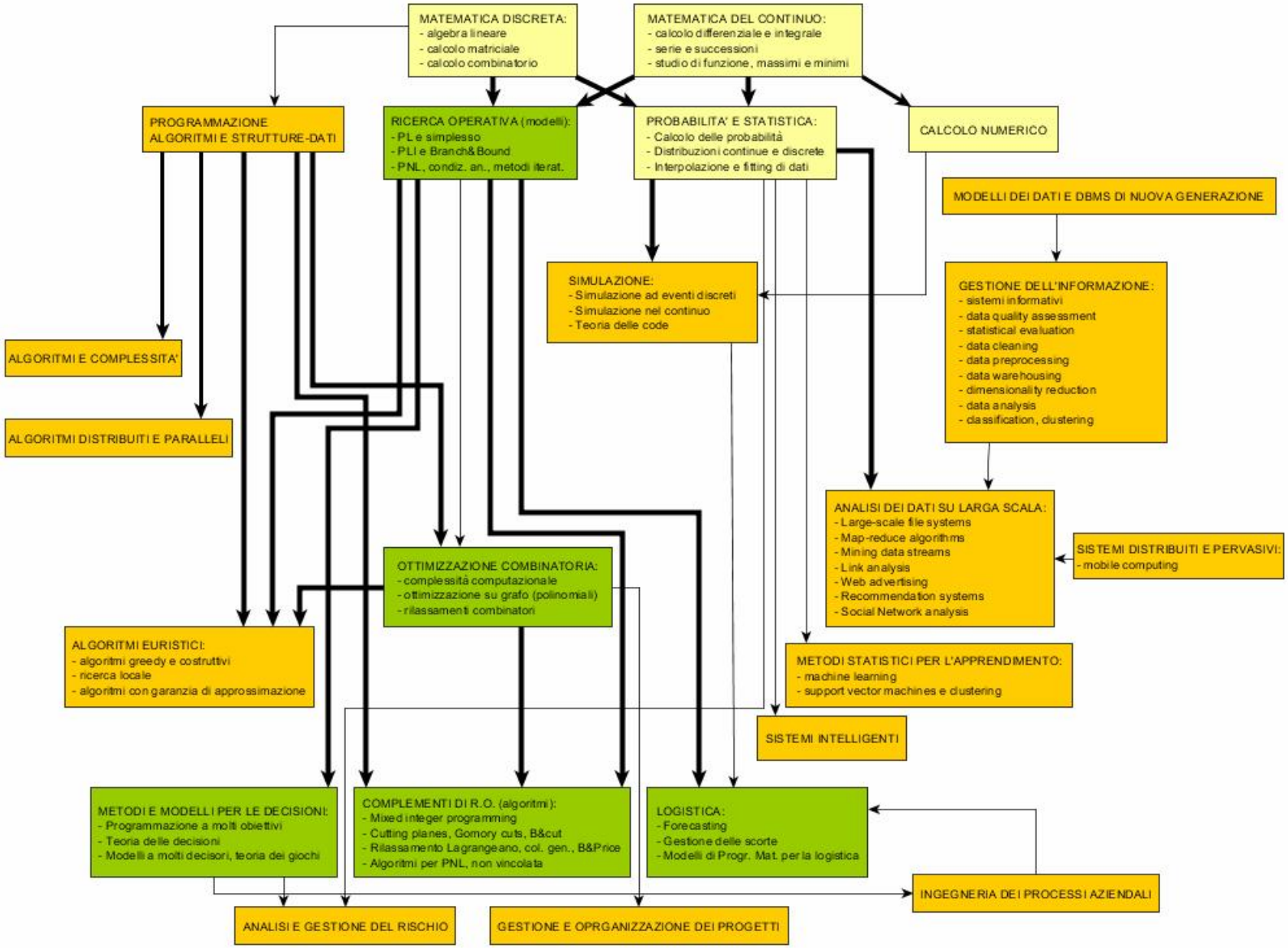
Questo si sa dagli anni '40... è un nome nuovo per una cosa vecchia?

Solo in parte!

Oggi abbiamo

- Big Data: enormi quantità di dati precisi, strutturati e poco costosi, che aspettano solo di essere sfruttati per estrarne informazione
- Cloud Computing: capacità di accesso ai dati e di elaborazione distribuite ovunque a basso costo
- Business Analytics: una cultura aziendale aperta all'uso dei modelli (*anche in Italia?*)
- nuova teoria: online, stochastic, robust programming, etc...

L'indirizzo


MATEMATICA DISCRETA:
- algebra lineare
- calcolo matriciale
- calcolo combinatorio
MATEMATICA DEL CONTINUO:
- calcolo differenziale e integrale
- serie e successioni
- studio di funzione, massimi e minimi
PROGRAMMAZIONE ALGORITMI E STRUTTURE-DATI
RICERCA OPERATIVA (modelli):
- PL e simplesso
- PLI e Branch&Bound
- PNL, condiz. an., metodi iterat.
PROBABILITA' E STATISTICA:
- Calcolo delle probabilità
- Distribuzioni continue e discrete
- Interpolazione e fitting di dati
CALCOLO NUMERICO
MODELLI DEI DATI E DBMS DI NUOVA GENERAZIONE
SIMULAZIONE:
- Simulazione ad eventi discreti
- Simulazione nel continuo
- Teoria delle code
GESTIONE DELL'INFORMAZIONE:
- sistemi informativi
- data quality assessment
- statistical evaluation
- data cleaning
- data preprocessing
- data warehousing
- dimensionality reduction
- data analysis
- classification, clustering
ALGORITMI E COMPLESSITA'
ALGORITMI DISTRIBUITI E PARALLELI
ANALISI DEI DATI SU LARGA SCALA:
- Large-scale file systems
- Map-reduce algorithms
- Mining data streams
- Link analysis
- Web advertising
- Recommendation systems
- Social Network analysis
SISTEMI DISTRIBUITI E PERVASIVI:
- mobile computing
OTTIMIZZAZIONE COMBINATORIA:
- complessità computazionale
- ottimizzazione su grafo (polinomiali)
- rilassamenti combinatori
ALGORITMI EURISTICI:
- algoritmi greedy e costruttivi
- ricerca locale
- algoritmi con garanzia di approssimazione
METODI STATISTICI PER L'APPRENDIMENTO:
- machine learning
- support vector machines e clustering
SISTEMI INTELLIGENTI
METODI E MODELLI PER LE DECISIONI:
- Programmazione a molti obiettivi
- Teoria delle decisioni
- Modelli a molti decisori, teoria dei giochi
COMPLEMENTI DI R.O. (algoritmi):
- Mixed integer programming
- Cutting planes, Gomory cuts, B&cut
- Rilassamento Lagrangeano, col. gen., B&Price
- Algoritmi per PNL, non vincolata
LOGISTICA:
- Forecasting
- Gestione delle scorte
- Modelli di Progr. Mat. per la logistica
INGEGNERIA DEI PROCESSI AZIENDALI
ANALISI E GESTIONE DEL RISCHIO
GESTIONE E OPRGANIZZAZIONE DEI PROGETTI

E per chi non è interessato all'indirizzo?

In molti campi applicativi si cercano oggetti o strutture caratterizzati da valori molto alti o molto bassi di un'opportuna funzione di valutazione

- *bioinformatica*: i farmaci più attivi legano con le proteine in configurazioni di energia potenziale minima
- *reti sociali*: il target migliore per una campagna sono i gruppi di individui più influenzabili, più influenti e più scorrelati
- *apprendimento automatico*: i sistemi più efficaci sono quelli che producono le classificazioni più semplici e con le violazioni minime
- *telecomunicazioni*: gli instradamenti più robusti sono quelli che producono la minima saturazione della banda e il minimo delay
- *stima parametrica*: i modelli fisici che forniscono le previsioni migliori sono quelli che riproducono nel modo più preciso le osservazioni
- *finanza*: gli algoritmi più efficaci per la gestione di portafogli sono quelli che riproducono nel modo più preciso le serie storiche

Gli algoritmi euristici possono essere la strategia migliore: un'ottimizzazione perfetta non è sempre richiesta, o persino desiderabile

Euristica

eurisko = io trovo

È una parola di derivazione greca

- ispirata al celebre aneddoto su Archimede e la corona d'oro

- coniata nel XIX secolo
- mai usata dagli antichi Greci

Qualche cenno storico

- IV secolo d.C.: Pappo di Alessandria discute l'*analyòmenos* (*tesoro dell'analisi*), cioè come si costruisce una dimostrazione matematica
  - come si passa dall'ipotesi alla tesi di un teorema
  - come si passa dai dati alla soluzione di un problema geometrico
- XVII secolo: Cartesio, Leibnitz e altri discutono l'ars inveniendi (*arte del trovare*), cioè la costruzione della verità attraverso la matematica
- XIX secolo: Bernard Bolzano discute in dettaglio le strategie più comuni per costruire dimostrazioni matematiche
- XIX-XX secolo: filosofi, psicologi ed economisti definiscono le euristiche come regole di decisione pratiche, semplici, che non mirano a un risultato ottimo, ma a uno soddisfacente (Simon, 1957)
- 1945: il saggio breve *How to solve it* di György Pólya torna all'accezione matematica di euristica come procedimento informale che porta a costruire una tesi o trovare una soluzione

Ma che c'entrano gli *algoritmi euristici*?

Algoritmi ed euristiche

Alcuni settori scientifici usano le due parole come contrari:

- algoritmo è una procedura formale, deterministica, fatta di passi elementari, in sequenza finita
- euristica è una regola informale, creativa, aperta

E ancora

- un algoritmo riceve i dati e produce la soluzione
- un'euristica riceve i dati e produce "qualcosa"

Infatti, un algoritmo ha una dimostrazione di correttezza, un'euristica no

Si potrebbe anche dire che

- un algoritmo è una dimostrazione di correttezza
- un'euristica è un insieme di considerazioni di buon senso

L'espressione algoritmo euristico è un ossimoro per certi aspetti

E allora che cos'è?

Algoritmi euristici

Un algoritmo euristico è un algoritmo che non garantisce la soluzione

Ma allora non serve a niente!

Non siamo drastici. Può servire se

1. "costa" molto meno di un algoritmo corretto: questa condizione richiede di definire la complessità computazionale di un algoritmo
   - tempo
   - spazio
2. fornisce "spesso" una soluzione "vicina" a quella corretta: questa condizione richiede di definire nello spazio delle soluzioni
   - una metrica, come "distanza soddisfacente tra soluzioni"
   - una distribuzione probabilistica, come "frequenza soddisfacente" delle soluzioni a distanza soddisfacente dalla soluzione corretta

Dimostrazioni e algoritmi

Dimostrazioni matematiche e algoritmi sono strettamente legati

- ogni algoritmo ha/è una dimostrazione di correttezza
- entrambi sono trasformazioni simboliche meccaniche, da un inizio (ipotesi/dati) a una fine (tesi/soluzione)
- la dimostrazione di incompletezza di Gödel si riflette nella dimostrazione di indecidibilità di Turing

Euristica è sia la costruzione di dimostrazioni sia quella di algoritmi

- se si ha successo, si butta l'euristica e si conserva la dimostrazione
- altrimenti, può essere che l'euristica porti a destinazione l'algoritmo spesso e bene, anziché sempre e perfettamente

Ecco gli algoritmi euristici

L'oggetto del corso

Il campo degli algoritmi euristici è enorme (*se un problema ammette più algoritmi corretti, figuriamoci quanti algoritmi scorretti!*)

Per limitare il campo, ci concentriamo su

- una famiglia di problemi: i problemi di Ottimizzazione Combinatoria
- una famiglia di algoritmi: gli algoritmi non basati su Programmazione Matematica

Rimane comunque un campo molto esteso

Classificazione dei problemi

Un problema è una domanda su un sistema di oggetti matematici

Si possono classificare i problemi in base alla natura della loro soluzione:

- problema di decisione: la soluzione è *Vero* o *Falso*
- problema di conteggio: la soluzione è il numero dei sottosistemi che soddisfano certe condizioni
- problema di ottimizzazione: la soluzione è il valore minimo o massimo di una funzione obiettivo definita sui sottosistemi che soddisfano certe condizioni
- problema di ricerca: la soluzione è la descrizione formale di un sottosistema che soddisfa certe condizioni
- problema di enumerazione: la soluzione è la descrizione formale di tutti i sottosistemi che soddisfano certe condizioni

A noi interessa la combinazione ottimizzazione/ricerca,
cioè cerchiamo il valore ottimo e un sottosistema che ha tale valore

Problemi di ottimizzazione/ricerca

Un problema di ottimizzazione/ricerca si può rappresentare come

$$\begin{aligned} &\text{opt} \ \ f(x) \\ &x \in X \end{aligned}$$

dove

- x, detta soluzione, descrive ciascun sottosistema del problema
- X, detta regione ammissibile o spazio delle soluzioni ammissibili, raccoglie i sottosistemi che soddisfano le condizioni date
- $f : X \to \mathbb{R}$, detta funzione obiettivo, misura quantitativamente la qualità di ogni sottosistema ($\text{opt} \in \{\min, \max\}$)

Il problema consiste nel determinare

- ottimizzazione: il valore ottimo f^*

$$f^* = \underset{x \in X}{\text{opt}} \, f(x)$$

- ricerca: almeno una soluzione ottima, ovvero un elemento $x^* \in X^*$

$$X^* = \arg \underset{x \in X}{\text{opt}} \, f(x) = \left\{ x^* \in X : f(x^*) = \underset{x \in X}{\text{opt}} \, f(x) \right\}$$

1. Definizione elementare: X è un insieme finito
2. Definizione sofisticata: dato un insieme base B finito

$$\underset{x \in X}{\text{opt}} \; f(x) \quad \text{con } X \subseteq 2^B$$

cioè la regione ammissibile è la collezione di tutti i sottoinsiemi dell'insieme base che soddisfano opportune condizioni

Le due definizioni sono equivalenti:

- se l'insieme base B è finito, ovviamente qualsiasi $X \subseteq 2^B$ è finito
- se X è finito, si può porre $B = X$, definire come regione ammissibile X' la collezione dei singoletti di B e chiamarla X per semplicità (cioè definire ammissibili i sottoinsiemi con un solo elemento)

La definizione sofisticata consente considerazioni più interessanti

L'algoritmo esaustivo risolve questi problemi esattamente in tempo finito

Programmazione Matematica

La Programmazione Matematica introduce alcune ipotesi descrittive

$$\begin{array}{l} \text{opt } f(x) \\ x \in X \end{array} \longrightarrow \begin{array}{ll} \min f(x) & \\ g_i(x) \leq 0 & i = 1, \ldots, m \end{array}$$

dove

- $x \in \mathbb{R}^n$, cioè una soluzione è un vettore di n valori reali
- $X = \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, \ldots, m\}$, cioè la regione ammissibile è l'insieme di vettori che soddisfano tutte le disuguaglianze (vincoli)

La programmazione matematica risolve il problema sfruttando le proprietà analitiche delle funzioni f e g_i $(i = 1, \ldots, m)$

Alcuni algoritmi euristici sfruttano informazioni ricavate da tali proprietà

Noi non useremo questi strumenti

Una classificazione degli algoritmi euristici (no PM)

Poiché ogni soluzione x è un sottoinsieme di B, abbiamo

- euristiche costruttive/distruttive:
  - partono da un sottoinsieme ovvio (rispettivamente, $\emptyset$ o B)
  - aggiungono/tolgono elementi fino a ottenere la soluzione desiderata
- euristiche di ricerca locale:
  - partono da una soluzione ottenuta in qualsiasi modo
  - scambiano elementi fino a ottenere la soluzione desiderata
- euristiche di ricombinazione:
  - partono da una popolazione di soluzioni ottenuta in qualsiasi modo
  - ricombinano soluzioni diverse producendo una nuova popolazione
- euristiche a convergenza:
  - associano a ogni elemento $j \in B$ un valore frazionario $x_j \in [0, 1]$
  - lo aggiornano iterativamente finché converge a $\{0, 1\}$

$$\begin{aligned} x_j = 1 &\quad \Leftrightarrow \quad j \in x \\ x_j = 0 &\quad \Leftrightarrow \quad j \in B \setminus x \end{aligned}$$

Spesso si combinano creativamente componenti di categorie diverse

Casualità e memoria

Altre due distinzioni fondamentali riguardano

- l'uso della casualità:
  - euristiche puramente deterministiche
  - euristiche che eseguono passi casuali,
    cioè determinati da generatori di numeri pseudocasuali
- l'uso della memoria:
  - euristiche che considerano solo i dati e le soluzioni correnti
  - euristiche che considerano anche le soluzioni generate in precedenza

Queste distinzioni sono ortogonali alla classificazione precedente

I rischi dell'approccio euristico

1. atteggiamento reverenziale o modaiolo, cioè farsi dettare l'approccio dal contesto sociale, anziché dal problema
2. l'atteggiamento magico, cioè credere all'efficacia di un metodo per semplice analogia con fenomeni fisici e naturali
3. l'ipercomplicazione, cioè introdurre componenti e parametri pletorici, come se potessero portare solo miglioramenti
4. il number crunching, cioè fare calcoli pesanti e sofisticati con numeri di dubbia utilità
5. l'overfitting, cioè sovradattare componenti e parametri dell'algoritmo allo specifico insieme di dati usati nella valutazione sperimentale
6. l'effetto SUV, cioè confidare nella potenza dell'hardware
7. l'integralismo euristico, cioè usare euristiche per problemi che si possono risolvere bene con metodi esatti

È fondamentale

- liberarsi dai pregiudizi (fermo restando il valore dell'esperienza)
- valutare le prestazioni in maniera scientificamente fondata
- distinguere il contributo di ciascuna componente dell'algoritmo
- implementare efficientemente le singole componenti