

ALGORITMI EURISTICI

Prof. Roberto Cordone
6 CFU

Luca Cappelletti

Lecture Notes
Year 2017/2018



Magistrale Informatica
Università di Milano
Italy
24 settembre 2018

Indice

1	Introduzione	2
1.1	Problemi tipici	2
1.2	Classificazione delle euristiche	2
1.3	Rischi tipici	2
2	Complessità computazionale	3
2.1	Complessità temporale	3
2.1.1	Notazione Theta grande	3
2.1.2	Notazione O grande	3
2.1.3	Notazione Omega grande	3
2.1.4	Complessità temporale di un algoritmo esaustivo	3

Introduzione

1.1 Problemi tipici

I problemi vengono catalogati in base alla natura della loro soluzione.

Problema di decisione: la soluzione è vero o falso. soddisfano certe condizioni.

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 su sottoinsiemi che

Problema di enumerazione: la soluzione è la descrizione formale di tutti i sottosistemi che soddisfano certe condizioni.

1.2 Classificazione delle euristiche

<p>Euristiche costruttive/distruttive: partono da un sottoinsieme ovvio (l'insieme intero o vuoto) ed aggiungono o tolgono elementi sino ad ottenere la soluzione desiderata.</p>	<p>Euristiche di ricombinazione: partono da una popolazione di soluzioni ottenuta in qualsiasi modo e ricombinano soluzioni diverse producendo una nuova popolazione.</p>
--	--

<p>Euristiche di ricerca locale: partono da una soluzione ottenuta in qualsiasi modo e scambiano elementi fino a ottenere la soluzione desiderata.</p>	<p>Euristiche a convergenza: associano a ogni elemento del set un valore frazionario tra 0 e 1 e lo aggiornano iterativamente finché converge a $\{0, 1\}$</p>
---	--

1.3 Rischi tipici

Atteggiamento referenziale o modaiolo: farsi dettare l'approccio dal contesto sociale.

Overfitting: sovradattare componenti e parametri dell'algoritmo allo specifico insieme di dati usati nella valutazione sperimentale.

Atteggiamento magico: credere all'efficacia di un metodo per pura analogia con fenomeni fisici e naturali.

Iperscomplicazione: introdurre componenti e parametri pleotorici, come se potessero portare solo miglioramenti.

Number crunching: fare calcoli pesanti e sofisticati con numeri di dubbia utilità.

Complessità computazionale

2.1 Complessità temporale

La complessità asintotica di un algoritmo nel caso pessimo fornisce una misura del tempo di calcolo dell'algoritmo attraverso i seguenti passaggi:

1. Misuriamo il tempo col numero T di operazioni elementari eseguite.
2. Scegliamo un valore n che misuri la dimensione di un'istanza.
3. Troviamo il tempo di calcolo massimo per le istanze di dimensione n , denominato con $T(n)$.
4. Approssimiamo $T(n)$ con una funzione $f(n)$ più semplice, di cui interessa solo l'andamento per $n \rightarrow +\infty$.
5. Raccogliamo le funzioni in classi con la stessa approssimante.

2.1.1 Notazione Theta grande

Con la notazione $T(n) \in \Theta(f(n))$ si indica che $f(n)$ stima $T(n)$ a meno di un fattore moltiplicativo:

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}: c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n) \quad \forall n \geq n_0$$

dove c_1 , c_2 e n_0 sono dipendenti da n .

2.1.2 Notazione O grande

Con la notazione $T(n) \in O(f(n))$ si indica che $f(n)$ stima $T(n)$ per eccesso a meno di un fattore moltiplicativo:

$$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}: T(n) \leq c \cdot f(n) \quad \forall n \geq n_0$$

dove c e n_0 sono indipendenti da n .

2.1.3 Notazione Omega grande

Con la notazione $T(n) \in \Omega(f(n))$ si indica che $f(n)$ stima $T(n)$ per difetto a meno di un fattore moltiplicativo:

$$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}: T(n) \geq c \cdot f(n) \quad \forall n \geq n_0$$

dove c e n_0 sono indipendenti da n .

2.1.4 Complessità temporale di un algoritmo esaustivo

Data un'istanza di un problema di ottimizzazione combinatoria, con cardinalità dell'insieme base $n = |B|$, l'algoritmo esaustivo ha complessità temporale almeno esponenziale:

$$T(n) \in \Omega(2^n)$$