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

Efficacia di un algoritmo euristico

L'efficacia di un algoritmo euristico può essere sottoposta ad

- analisi teorica (a priori): dimostrare che l'algoritmo fornisce soluzioni con una data garanzia di qualità, sempre o con una data frequenza
- analisi sperimentale (a posteriori): misurare le prestazioni dell'algoritmo su un campione di istanze di benchmark

L'analisi teorica è complicata dal fatto che

- le operazioni raramente hanno un effetto semplice sulla soluzione (*per il VCP: un lato, due vertici; per il TSP: un lato, due archi*)
- caso medio e randomizzazione richiedono un trattamento statistico

Le conclusioni dell'analisi teorica possono essere insoddisfacenti in pratica perché fondate su assunzioni poco rappresentative

- un caso pessimo infrequente (istanze rare, ma ostiche)
- una distribuzione di probabilità delle istanze irrealistica

These slides are partly based on those provided with the book "*Stochastic Local Search*" by H.H. Hoos and T. Stützle (Morgan Kaufmann, 2004), see www.sls-book.net for information

Analisi sperimentale

L'approccio sperimentale è molto comune nella scienza

- la matematica è un'eccezione, fondata sull'approccio formale
- ma l'algoritmica è un'eccezione nell'eccezione

Perciò si tende a dimenticare le basi del metodo sperimentale scientifico

1. partire dall'osservazione
2. formulare un modello (ipotesi di lavoro)
3. finché non si ottiene un modello soddisfacente
   a. progettare esperimenti computazionali per validare il modello
   b. condurre gli esperimenti e raccoglierne i risultati
   c. analizzare i risultati con metodi quantitativi
   d. rivedere il modello in base ai risultati

Che cosa si intende con "modello" nello studio degli algoritmi?

Obiettivi dell'analisi

L'analisi sperimentale indaga

- in fisica le leggi che regolano il comportamento dei fenomeni
- in algoritmica le leggi che regolano il comportamento degli algoritmi

Si tratta di

1. ottenere indici sintetici di efficacia e di efficienza di un algoritmo
2. descrivere il legame fra gli indici di efficacia o efficienza e valori parametrici delle istanze (dimensione n, ecc. . .)
3. confrontare l'efficacia di algoritmi diversi per indicare il migliore
4. suggerire miglioramenti all'algoritmo

Benchmark

Non potendo testare tutte le istanze, occorre definire un campione

Un campione significativo deve rappresentare diverse

- dimensioni, in particolare per l'analisi sperimentale della complessità
- caratteristiche strutturali (per i grafi: densità, grado, diametro, ...)
- tipologie
  - di applicazione: logistica, telecomunicazioni, produzione, ...
  - di generazione: realistiche, artificiali, trasformazioni di altri problemi
  - di distribuzione probabilistica: uniforme, normale, esponenziale, ...

Non ha senso cercare un campione strettamente "equiprobabile" perché

- conviene concentrarsi sulle istanze più frequenti in pratica
  - avranno un'utilità maggiore per le applicazioni
- conviene concentrarsi sulle istanze più difficili
  - avranno un impatto più forte (anche ipotizzando l'equiprobabilità)
  - sono quelle da cui si può imparare di più
- conviene evitare le istanze che richiedono troppo tempo
  - bisogna raccogliere dati in quantità sufficiente a fare estrapolazioni

Riproducibilità

Il metodo scientifico richiede risultati riproducibili e controllabili

- per quanto riguarda le istanze, occorre usare
  - istanze già pubblicamente disponibili
  - nuove istanze rese disponibili
- per quanto riguarda l'algoritmo, occorre specificare
  - esattamente e completamente i dettagli implementativi
  - il linguaggio di programmazione
  - il compilatore
- per quanto riguarda l'ambiente, occorre specificare
  - la macchina usata
  - il sistema operativo
  - la memoria disponibile
  - ...

La riproduzione di risultati ottenuti da altri è estremamente difficile

Analisi dell'efficienza di un algoritmo

Consideriamo l'esecuzione dell'algoritmo A come un esperimento casuale

Un'analisi significativa è possibile solo tenendo fissa la dimensione n
(e altri parametri rilevanti suggeriti dall'analisi del caso pessimo)

- l'insieme delle istanze $\mathcal{I}_n$ costituisce lo spazio degli esiti
- il tempo di calcolo $T_A(I)$ è la variabile aleatoria sotto indagine

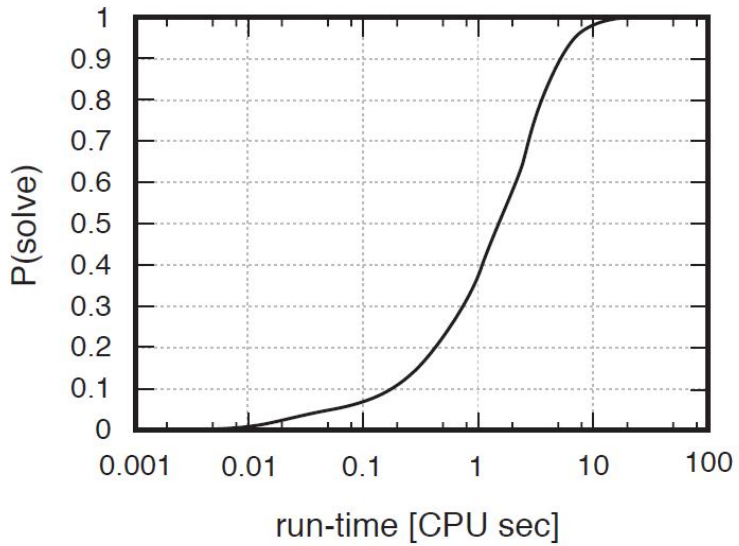
Le prestazioni di A sono descritte dalle proprietà statistiche di $T_A(I)$

- il valore atteso di $T_A(I)$ in $\mathcal{I}_n$
- la varianza di $T_A(I)$ in $\mathcal{I}_n$

Per algoritmi polinomiali, se tutti i parametri rilevanti sono fissati,
la media tende ad essere un indice affidabile

Analisi del tempo di calcolo (diagramma *RTD*)

La funzione di distribuzione della variabile aleatoria $T_A(I)$ in $\mathcal{I}_n$ fornisce il diagramma *Run Time Distribution* (*RTD*)



Se tutti i parametri influenti sono stati individuati, il diagramma *RTD* degenera in un gradino (*il tempo è quasi uguale per le istanze di* $\mathcal{I}_n$)

Se è molto distribuito, possono esistere altri parametri interessanti

La dipendenza del tempo di calcolo da n si studia con un altro diagramma

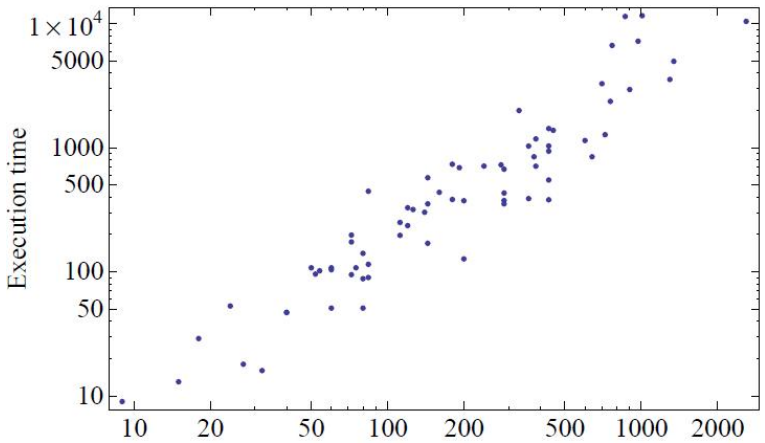
Analisi del tempo di calcolo (diagramma di *scaling*)

La complessità asintotica indica per il caso pessimo (*gli altri stan sotto*) una fascia di andamenti di ampiezza imprecisata (*non dà* c_1 *e* c_2)

Lo studio empirico al variare di n fornisce le informazioni mancanti

Per eseguirlo, bisogna:

- estrarre una sequenza di campioni $\mathcal{I}_n$ per valori crescenti di n
- applicare l'algoritmo e registrare i tempi $T_A(I)$
- tracciare la nuvola dei punti $(n, T(I))$ o delle medie $\left(n, \frac{\sum_{I \in \mathcal{I}_n} T(I)}{|\mathcal{I}_n|}\right)$
- ipotizzare una famiglia di interpolanti con l'aiuto dell'analisi teorica
- tarare i parametri dell'interpolante



Interpolazione

Per indovinare la famiglia di interpolanti, si può valutare per tentativi quale tipo di scala genera un diagramma di *scaling* lineare:

- scala lineare (su ascisse e ordinate): la funzione interpolante è lineare

$$T(n) = \alpha n + \beta \Leftrightarrow T(n) = c_1 n + c_2$$

- scala logaritmica (su ascisse e ordinate): la funzione interpolante è polinomiale

$$\log T(n) = \alpha \log n + \beta \Leftrightarrow T(n) = 2^{\beta} n^{\alpha}$$

- scala semilogaritmica (lineare sulle ascisse e logaritmica sulle ordinate): la funzione interpolante è esponenziale

$$\log T(n) = \alpha n + \beta \Leftrightarrow T(n) = 2^{\beta} (2^{\alpha})^{n}$$

Analisi dell'efficacia di un algoritmo

Consideriamo l'esecuzione dell'algoritmo A come un esperimento casuale

- l'insieme delle istanze $\mathcal{I}$ costituisce lo spazio degli esiti
- la differenza relativa $\delta_A(I)$ è un'altra variabile aleatoria

Le prestazioni di A sono descritte dalle proprietà statistiche di $\delta_A(I)$

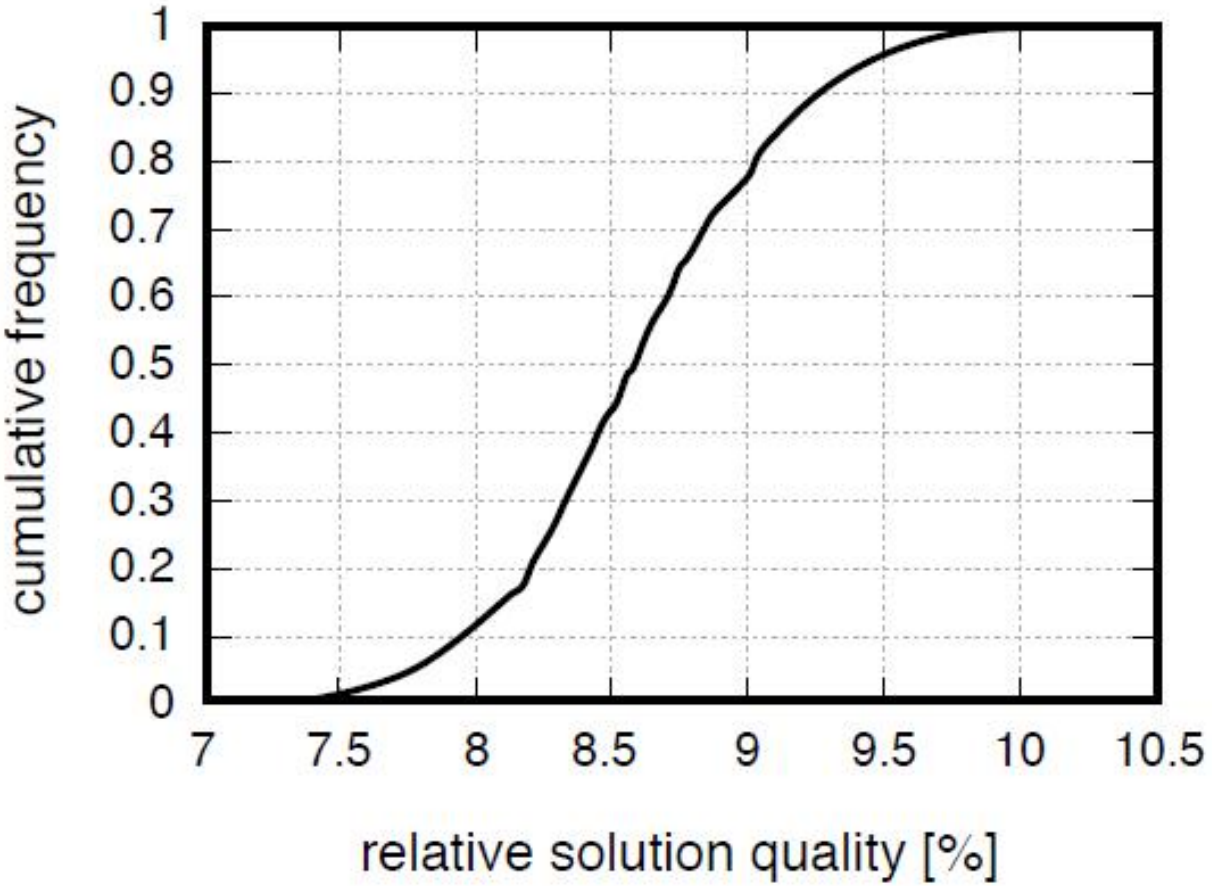
- i parametri di uso più comune e classico
  - il valore atteso di $\delta_A(I)$
  - la varianza di $\delta_A(I)$

  tendono a "subire" l'influsso dei valori estremi (*specie quelli cattivi*)
- di recente, si cominciano a preferire altre statistiche descrittive
  - più dettagliate
  - più "stabili"

Solution Quality Distribution (*SQD*)

Una descrizione completa di $\delta_A(I)$ è data dalla funzione di distribuzione

$$F_A(\alpha) = Pr[\delta_A(I) \leq \alpha] \text{ per ogni } \alpha \in \mathbb{R}$$



Il suo andamento è il diagramma *SQD* (Solution Quality Distribution)

Solution Quality Distribution (*SQD*)

Per qualsiasi algoritmo, la funzione di distribuzione di $\delta_A(I)$

- è monotona non decrescente
- è nulla per $\alpha < 0$
- tende a 1 per $\alpha \to +\infty$

Inoltre, se si considerano

- algoritmi esatti, è una funzione a gradino, pari a 1 per ogni $\alpha \geq 0$
- algoritmi $\bar{\alpha}$-approssimati, è una funzione pari a 1 per ogni $\alpha \geq \bar{\alpha}$

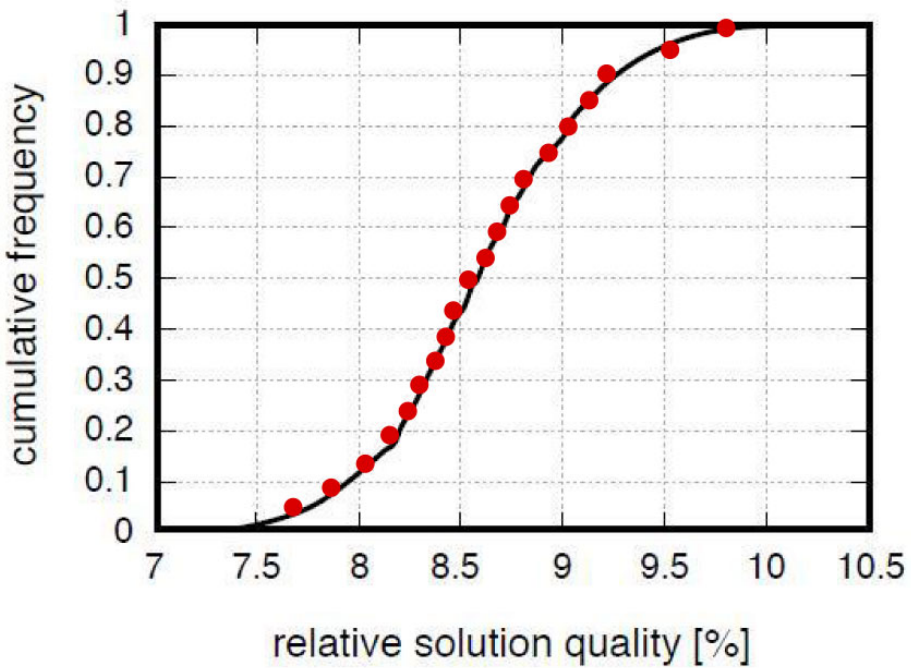
Procedura per generare il diagramma *SQD*

In pratica, si ricostruisce un diagramma campionario della funzione

1. Si genera un campione $\bar{\mathcal{I}} \subset \mathcal{I}$ abbastanza grande di istanze indipendenti
2. Si applica l'algoritmo a ogni istanza $I \in \bar{\mathcal{I}}$ e si costruisce l'insieme

$$\Delta_A\left(\bar{\mathcal{I}}\right) = \left\{\delta_A\left(I\right) : I \in \bar{\mathcal{I}}\right\}$$

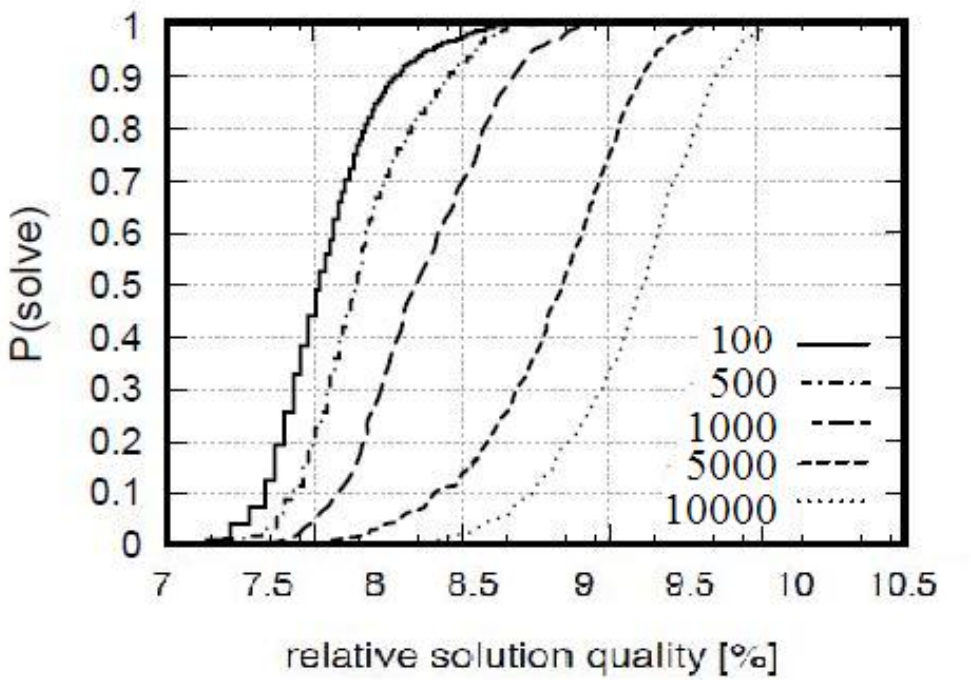
3. Si ordina per valori non decrescenti $\Delta_A\left(\bar{\mathcal{I}}\right) = \left\{\delta_1, \ldots, \delta_{|\bar{\mathcal{I}}|}\right\}$
4. Si riportano nel grafico i punti $\left(\delta_j, \frac{j}{|\bar{\mathcal{I}}|}\right)$ per $j = 1, \ldots, |\bar{\mathcal{I}}|$



Diagrammi *SQD* parametrici

Dati i problemi teorici e pratici nella definizione di un campione significativo spesso il diagramma è parametrizzato rispetto a

- un parametro strutturale delle istanze (dimensione n, ...)
- un parametro probabilistico della distribuzione assunta per le istanze (valor medio o varianza dei costi, ...)

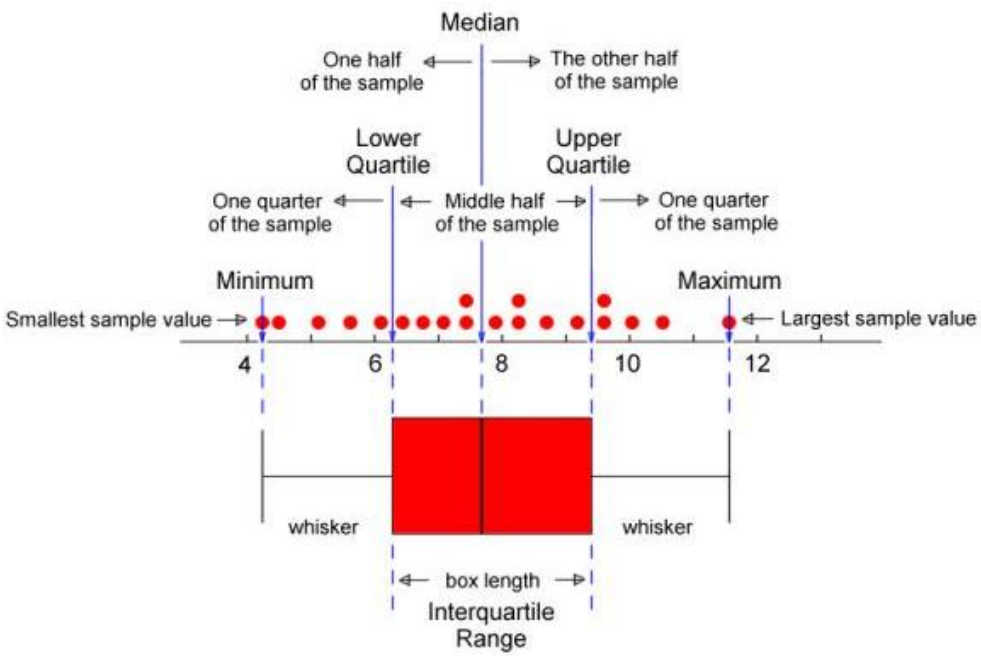


Questo rende più limitate le conclusioni, ma più significativo il campione
Inoltre, permette di individuare delle tendenze generali

Statistiche descrittive e boxplot

Una rappresentazione più compatta è data da statistiche descrittive come

- il valore atteso e la varianza campionarie (*più classici*)
- la mediana e i vari quantili campionari (*più "stabili"*)
- il diagramma *box-plot* (o *box and whiskers plot*), che riporta
  - mediana campionaria ($q_{0.5}$)
  - quartile inferiore e superiore campionari ($q_{0.25}$ e $q_{0.75}$)
  - valori estremi campionari (spesso indicando a parte gli "outlier")



Confronto fra algoritmi con i diagrammi *SQD*

Come si fa a dire che un algoritmo è meglio di un altro?

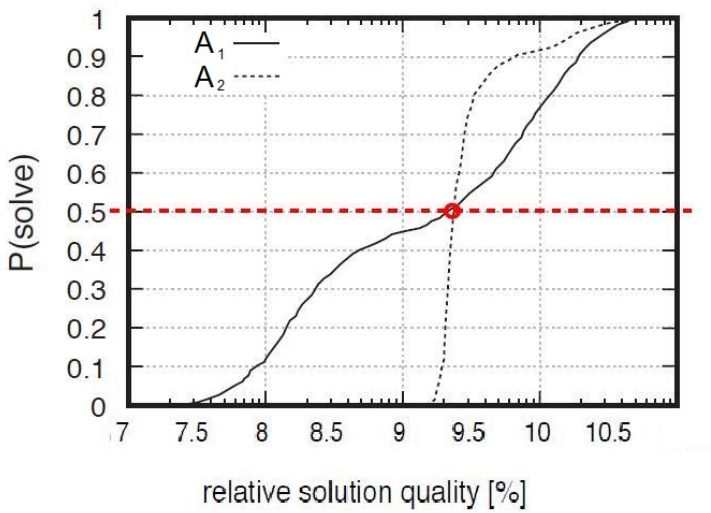
- dominanza stretta: ottiene risultati migliori su tutte le istanze

$$\delta_{A_2}(I) \leq \delta_{A_1}(I) \qquad \text{per ogni } I \in \mathcal{I}$$

  Di solito, avviene solo in casi banali (per es., A_2 "contiene" A_1)
- dominanza probabilistica: la funzione di distribuzione ha valori più alti per ogni valore di α

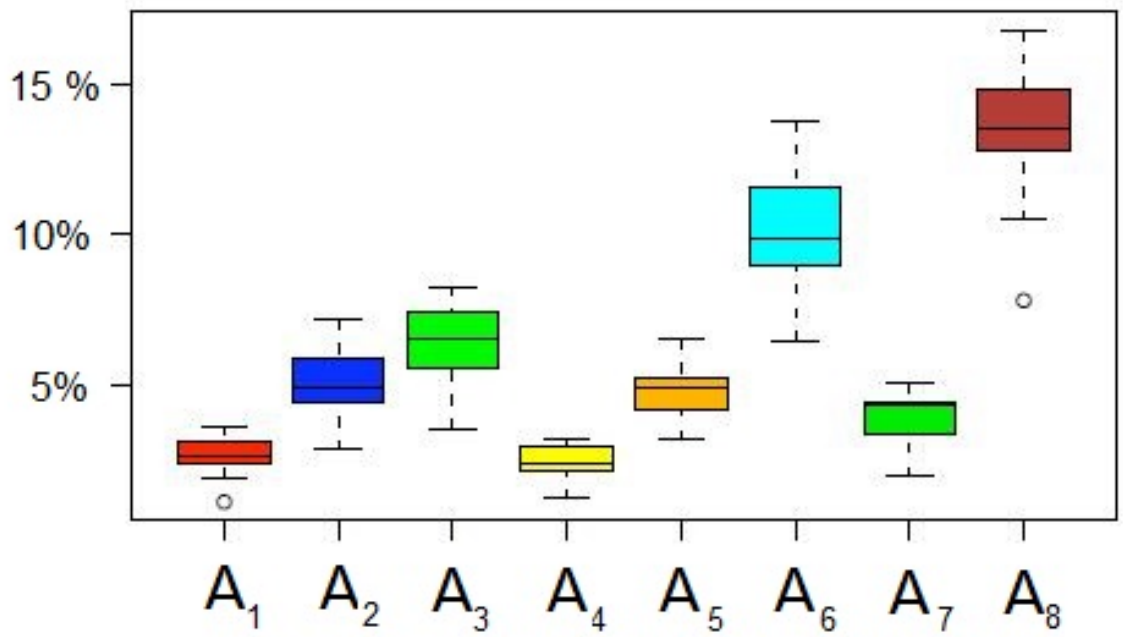
$$F_{A_2}(\alpha) \geq F_{A_1}(\alpha) \qquad \text{per ogni } \alpha \in \mathbb{R}$$

Controesempio: l'algoritmo A_1 trova con alta probabilità differenze minori di A_2, ma anche maggiori. Non c'è dominanza, ma A_1 è meno "robusto"



Confronto fra algoritmi con i diagrammi *boxplot*

Un confronto più sintetico si può condurre con i diagrammi *boxplot*



Se i diagrammi sono completamente separati, c'è dominanza stretta

La dominanza probabilistica si può intuire dalla posizione relativa dei cinque quartili disponibili (q_0, $q_{0.25}$, $q_{0.5}$, $q_{0.75}$, q_1)

Test statistici

Il giudizio sui diagrammi però è qualitativo: come sapere se la differenza empirica fra due algoritmi, A_1 e A_2, è quantitativamente significativa?

Per confrontare due distribuzioni esiste il test di Kolmogorov-Smirnov

Il test di Wilcoxon valuta la differenza fra i risultati dei due algoritmi $\delta_{A_1}(I) - \delta_{A_2}(I)$ (variabile aleatoria definita sullo spazio degli esiti $\mathcal{I}$)

- si formula l'ipotesi nulla H_0: la differenza ha mediana nulla
- si estrae un campione di istanze $\bar{\mathcal{I}}$ e gli si applicano i due algoritmi, ottenendo un campione di coppie di valori $(\delta_{A_1}, \delta_{A_2})$
- si calcola la probabilità p di ottenere il risultato osservato o uno più "estremo", supponendo vera H_0
- si sceglie un livello di significatività $\bar{p}$ (spesso $\bar{p} = 5\%$ o $\bar{p} = 1\%$):
  - probabilità massima accettabile di rifiutare H_0 qualora sia vera,
  - cioè di considerare non nulla una mediana che invece lo è
  - cioè di considerare diversamente efficaci due algoritmi equivalenti
- se $p \leq \bar{p}$, si rifiuta H_0

Test di Wilcoxon (ipotesi)

Il test non fa ipotesi sulla distribuzione di probabilità dei valori testati
Quindi è un test non parametrico

Questo lo rende adatto a valutare le prestazioni di algoritmi euristici, dato che la distribuzione della differenza relativa $\delta_A(I)$ non è nota

Si basa sulle seguenti ipotesi:

- i dati sono coppie di valori riferiti alla stessa popolazione
  (*si estraggono istanze da $\mathcal{I}$ e si applicano A_1 e A_2 a ciascuna*)
- ogni coppia di dati è estratta indipendentemente dalle altre
  (*le istanze estratte non sono correlate fra loro in alcun modo*)
- i dati sono misurati su scale almeno ordinali
  (*non contano i valori ottenuti, ma il loro ordine relativo*)

Test di Wilcoxon (esecuzione)

Il test esegue le seguenti operazioni

1. calcola le differenze assolute $|\delta_{A_1}(I) - \delta_{A_2}(I)|$ per $i = 1, \ldots, |\bar{\mathcal{I}}|$
2. le ordina per valori crescenti e attribuisce un rango R_i a ciascuna
3. somma separatamente i ranghi delle coppie con differenza positiva e quelli delle coppie con differenza negativa

$$\begin{cases} W^+ = \sum_{i:\delta_{A_1}(I_i) > \delta_{A_2}(I_i)} R_i \\ W^- = \sum_{i:\delta_{A_1}(I_i) < \delta_{A_2}(I_i)} R_i \end{cases}$$

   Se l'ipotesi nulla H_0 fosse vera, le due somme dovrebbero coincidere

4. la differenza fra W^+ e W^- consente di calcolare il valore di p: dati i $2^{|\bar{\mathcal{I}}|}$ esiti, corrispondenti a $|\bar{\mathcal{I}}|$ differenze positive o negative, si contano quelli con $|W^+ - W^-|$ pari o superiore all'osservazione
5. se $p < \bar{p}$, la differenza è significativa e un algoritmo prevale sull'altro
   - se $W^+ < W^-$, A_1 è meglio di A_2
   - se $W^+ > W^-$, A_1 è peggio di A_2

Conclusioni possibili

Il test di Wilcoxon può suggerire

- che uno dei due algoritmi sia significativamente migliore dell'altro
- che i due algoritmi siano statisticamente equivalenti

Se il campione è composto da classi di istanze parametricamente distinte, algoritmi complessivamente equivalenti potrebbero non esserlo sulle singole classi di istanze; lo studio rivelerebbe

- classi di istanze a cui conviene applicare A_1
- classi di istanze a cui conviene applicare A_2
- classi di istanze sulle quali i due algoritmi si equivalgono

Compromesso fra qualità e tempo

Il confronto di algoritmi che richiedono tempi diversi è in generale iniquo (*a meno che risparmiare tempo non sia esplicitamente definito inutile*)

Un algoritmo è migliore di un altro quando contemporaneamente

1. ottiene risultati migliori
2. richiede un tempo inferiore

Analisi che prescindono dal tempo di calcolo possono valere se

- si considera un algoritmo singolo senza fare confronti
- si confrontano varianti di un algoritmo che richiedono ugual tempo di calcolo (tipicamente, ottenute variando un parametro numerico)
- si confrontano algoritmi che hanno molte operazioni in comune e differiscono solo per operazioni di durata relativamente trascurabile

Relazione fra qualità e tempo di calcolo

È in generale scorretto confrontare

- algoritmi lenti con risultati buoni
- algoritmi veloci con risultati cattivi

Molti algoritmi trovano più di una soluzione durante l'esecuzione, e quindi possono fornire un risultato anche se terminati prematuramente

Definiamo $\delta_A(I, t)$ la differenza relativa della miglior soluzione trovata dall'algoritmo A dopo un tempo t

- per convenzione, poniamo $\delta_A(I, t) = +\infty$ quando all'istante t l'algoritmo non ha ancora trovato soluzioni ammissibili
- $\delta_A(I, t)$ è una funzione monotona non crescente a gradini di t
- quando l'algoritmo termina, $\delta_A(I, t)$ rimane costante

Questo riguarda soprattutto gli algoritmi con passi casuali o memoria, dato che il loro tempo di calcolo è fissato dall'utente

Algoritmi randomizzati

Un algoritmo con passi casuali può in teoria proseguire indefinitamente

La differenza relativa $\delta_A(I, \omega, t)$ dipende allora

1. dall'istanza $I \in \mathcal{I}$ su cui l'algoritmo è eseguito
2. dall'esito $\omega \in \Omega$ dell'esperimento casuale che guida l'algoritmo
   cioè il seme della sequenza pseudocasuale
3. dal tempo di esecuzione t

Siccome i primi due sono casuali, si valuta la robustezza dell'algoritmo

1. a seme fissato su un campione di istanze
2. a istanza fissata su un campione di semi

I risultati dello studio rispetto al seme si possono riassumere in due modi:

- con il valor medio della differenza relativa $\delta_A(I)$ e del tempo $T_A(I)$
- con il valor minimo della differenza relativa $\delta_A(I)$ e il valore totale del tempo $T_A(I)$

Classificazione

La relazione fra qualità del risultato e tempo di calcolo consente di classificare gli algoritmi (sia deterministici sia randomizzati) in

- completi: per ogni istanza $I \in \mathcal{I}$ trovano l'ottimo in tempo finito

$$\text{Per ogni } I \in \mathcal{I}, \exists t_I \in \mathbb{R}^+ : \delta_A(I,t) = 0 \text{ per ogni } t \geq t_I$$

  È un altro modo di definire gli algoritmi esatti

- probabilisticamente approssimativamente completi: per ogni istanza $I \in \mathcal{I}$, la probabilità che trovino l'ottimo tende a 1 per $t \to +\infty$

$$\text{Per ogni } I \in \mathcal{I}, \lim_{t \to +\infty} Pr[\delta_A(I,t) = 0] = 1$$

  Esempi: molte metaeuristiche randomizzate

- essenzialmente incompleti: per alcune istanze $I \in \mathcal{I}$, la probabilità che trovino l'ottimo è strettamente < 1 per $t \to +\infty$

$$\exists I \in \mathcal{I} : \lim_{t \to +\infty} Pr[\delta_A(I,t) = 0] < 1$$

  Esempi: gli algoritmi *greedy*, gli algoritmi di ricerca locale, ...

Generalizzazione a soluzioni approssimate

Un'ovvia generalizzazione sostituisce la ricerca dell'ottimo con quella di un dato livello di approssimazione

$$\delta_A(I,t) = 0 \;\rightarrow\; \delta_A(I,t) \leq \alpha$$

- algoritmi α-completi: per ogni istanza trovano una soluzione α-approssimata in tempo finito (sono gli algoritmi α-approssimati)
- algoritmi probabilisticamente approssimativamente α-completi: per ogni istanza $I \in \mathcal{I}$, la probabilità che trovino una soluzione α-approssimata tende a 1 per $t \rightarrow +\infty$
- algoritmi essenzialmente α-incompleti: per alcune istanze $I \in \mathcal{I}$, la probabilità che trovino una soluzione α-approssimata è strettamente < 1 per $t \rightarrow +\infty$

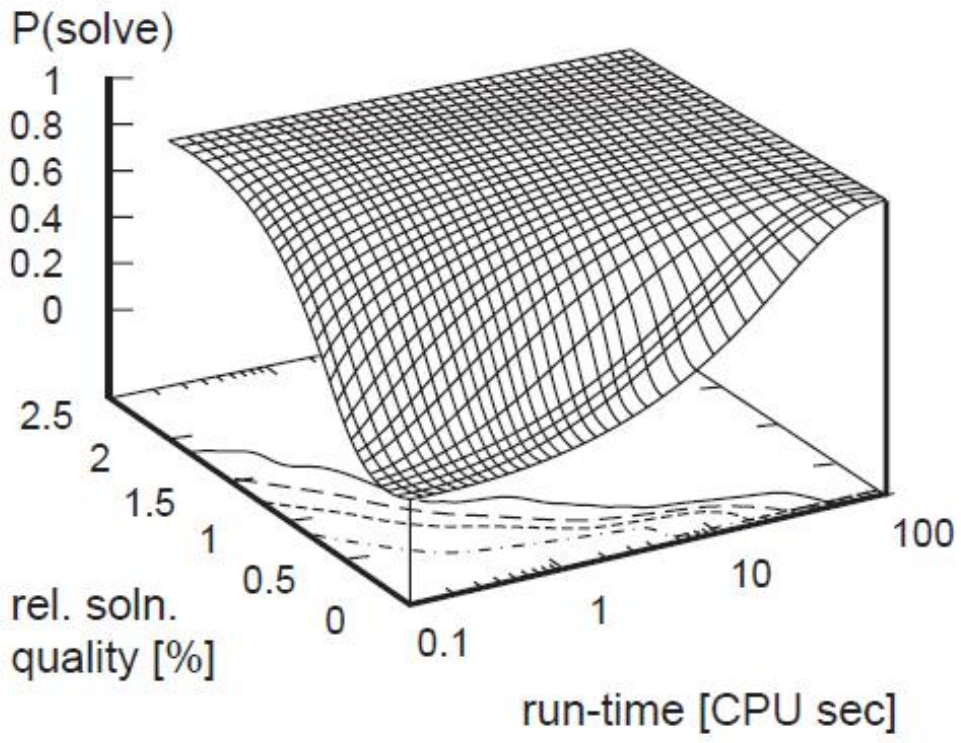
Considerando l'intero spazio campionario bidimensionale (istanze, esiti), si può descrivere statisticamente un algoritmo come compromesso fra

- la qualità del risultato, descritta da una soglia α
- il tempo richiesto per ottenerlo, descritto da una soglia t

La probabilità di successo

Definiamo probabilità di successo $\pi_{A,n}(\alpha, t)$ la probabilità che l'algoritmo A trovi in tempo $\leq t$ una soluzione di livello $\leq \alpha$

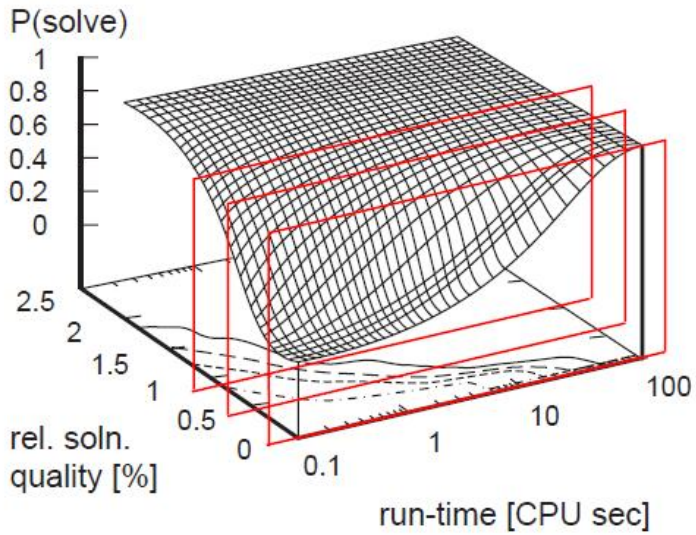
$$\pi_{A,n}(\alpha, t) = Pr\left[\delta_A(I, t) \leq \alpha | I \in \mathcal{I}_n\right]$$

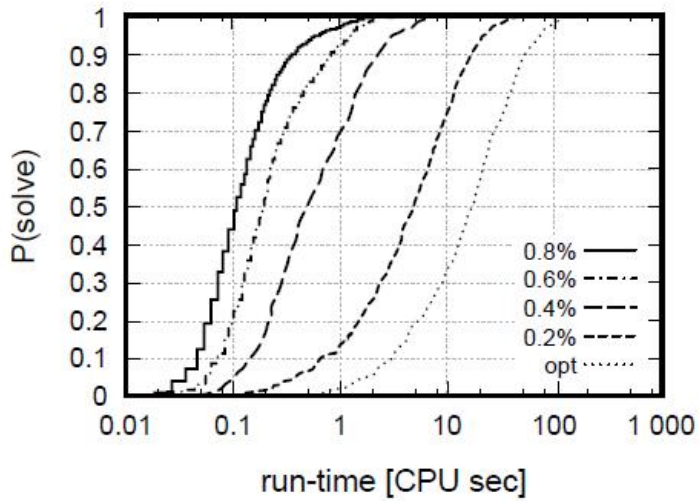


Ne derivano diversi diagrammi secondari

Diagrammi *Qualified Run Time Distribution* (*QRTD*)

I diagrammi *QRTD* descrivono l'andamento del tempo necessario a raggiungere prefissati livello di qualità





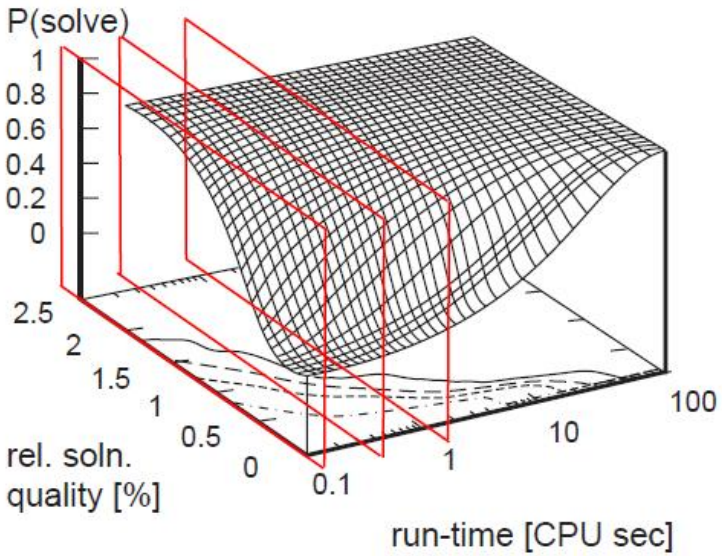
Sono utili quando il tempo di calcolo non è una risorsa stringente

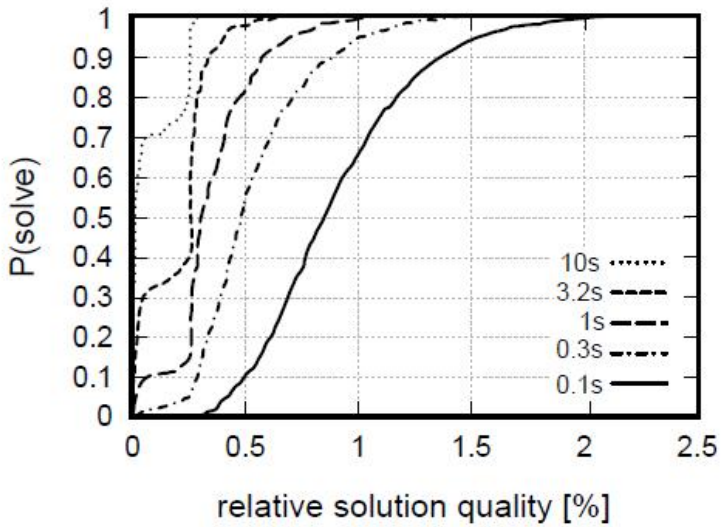
Se l'algoritmo è completo, arriva a 1 per ogni α in tempo finito

Se l'algoritmo è $\bar{\alpha}$-approssimato, arriva a 1 per ogni $\alpha \geq \bar{\alpha}$ in tempo finito

Diagrammi *Solution Quality Distribution* (*SQD*)

I diagrammi *SQD* descrivono l'andamento del livello di qualità raggiunto in un dato tempo di calcolo





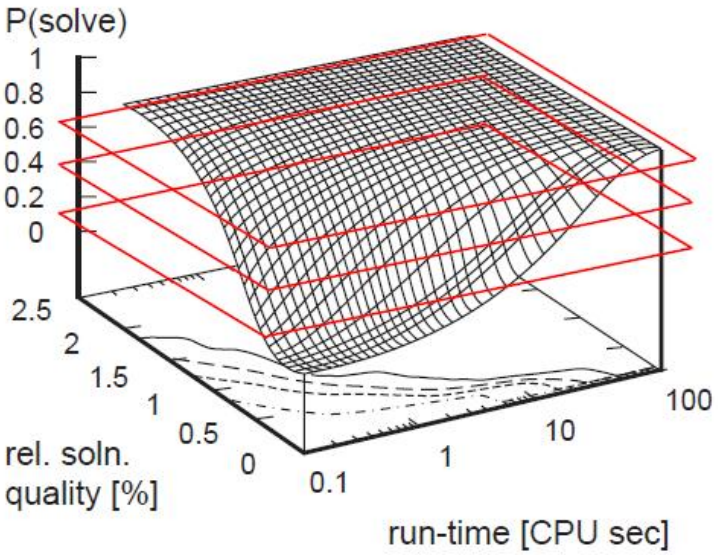
Sono utili quando il tempo di calcolo è una risorsa stringente

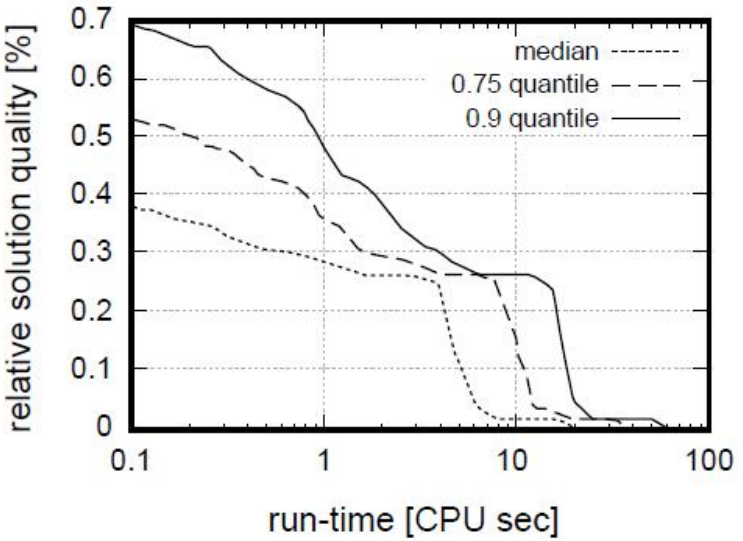
Se l'algoritmo è completo, per t abbastanza grande è un gradino alto 1

Se l'algoritmo è α-completo, per t abbastanza grande arriva a 1 entro α

Diagrammi *Solution Quality statistics over Time* (*SQT*)

Infine si possono tracciare le linee di livello associate ai diversi quantili





Queste descrivono il compromesso fra qualità e tempo di calcolo

Un algoritmo robusto avrà linee molto ravvicinate fra loro