Experimentelle Algorithmik: Design, Planung, Durchführung und Auswertung von Studien zum Verhalten von Algorithmen

Überblick:

- Experimentelle Algorithmik
- “Gute” empirische Arbeiten
- Design
- Testbeispiele
- Durchführung
- Auswertung
- Literaturhinweise

Experimentelle Algorithmik

- junges, bisher vernachlässigtes Forschungsfeld
- Lücke zwischen Theorie und Praxis
- Fehlen von etablierten Tests für Vergleichsstudien

analytische Ansatz:
Beweis von Sätzen über Algorithmen aufgrund eines math. Berechnungsmodells

empirischer Ansatz:
Ziehen von Schlußfolgerungen über Algorithmen durch "Anschauen" von echten Implementationen

analytische Ziele:

- worst-case, asymptotisches Verhalten

empirische Ziele:

- Verständnis der durchschnittlichen Güte oder Laufzeit in Abhängigkeit von Inputparametern
- Robustheitsstudien:
  Verteilungseigenschaften der Beobachtungen (Ausreißer, Standardabweichung etc.); Korrelation zu Eigenschaften von Input oder Algorithmus
- Bottleneck-Analyse

Was macht gute empirische Arbeit aus?

- "The purpose of mathematical programming is insight, not numbers"
  (A. M. Geoffrion)
- zielt auf allgemeine algorithmische Resultate, nicht auf spezifische Implementation auf spezieller Hardware
- Resultate sollen nicht offensichtlich sein
- Performance-Unterschied sollte groß sein, nicht Artefakt des experimentellen Aufbaus
- Negativresultate nicht inhärent schlecht

Definitionsversuch: Ein experimentelles Resultat ist interessant, wenn es nichttriviale Einsichten in das "In-Praxis"-Verhalten von Algorithmen für wichtige Probleme liefert.

Testbeispiele:

- echte Anwendungsinstanzen
- generische Instanzen
  ermöglichen partiell die Untersuchung
  asymptotischer Aussagen

Ideal: Kombination von beidem

Planung des Versuchsaufbaus

- iteratives Vorgehen
- zunächst sinnvolle Parameterbereiche suchen (möglichst groß!)
- Auswahl von Designpunkten (Kombination von Parametern)
- Berücksichtigung von logarithmischen Skalen
- Laufzeit $\gg$ Meßgenauigkeit
- genügend Instanzen pro Designpunkt

Durchführung

- Reproduzierbarkeit sichern
- penibles Protokollieren der Experimente
- Überwachung
  — Speicherplatz (Limits setzen!)
  — Vollständigkeit der Experimente
- geeignete Rechner auswählen
  (Vergleichbarkeit der Messungen!)
- Korrektheitskontrolle

Messungen

- Laufzeit (CPU)
- Speicherplatz
- Güte (bei Approximationen)
- Zählung repräsentativer Operationen
  (representative operation counts)

  ermöglicht:
  asymptotische Bottleneck-Analyse
  "virtual CPU-time"

Zeitmessung: ohne I/O; optimierte Version

Auswertung

- deskriptive Statistik (log-log-Plots)
- Standardabweichung kontrollieren
- Regressionsanalyse
- Residualplot, Konfidenzintervalle

Beschreibung der Experimente

- Rechnerumgebung, Programmiersprache, Compiler
- Details der Implementation (benutzte Datenstrukturen; wie wurden Freiheitsgrade ausgenutzt?)
- Ziel des Experiments
- Aufbau des Experiments
- Auswertung des Experiments
- Schlußfolgerungen

Literaturhinweise

1. *Toward an Experimental Method for Algorithm Simulation.* Catherine McGeoch, INFORMS J. on Computing, vol. 8, pp. 1–15, 1996.

2. *Reporting Computational Experiments in Mathematical Programming.* Crowder, Dembo, Mulvey, Math. Programming 15 (1997), pp. 316–329.

3. *The role of experiment in the theory of algorithms.* Richard Anderson, draft, 1997.

4. *Use of Representative Operation Counts in Computational Testing of Algorithms*, Ahuja, Orlin, INFORMS J. on Computing, vol. 8, pp. 318–330, 1996.