Durchführung von Experimenten

siehe auch
„A Theoretician‘s Guide to the Experimental Analysis of Algorithms“
von David S. Johnson, AT&T Labs Research

Ziel der Experimente

Vor ausführlichen Testläufen:

Welche (interessanten!) Fragen sollen die Experimente beantworten?

Zum Beispiel:

- Wie wirken sich Implementierungsdetails (Parameter, Datenstrukturen,…) auf die Laufzeit aus?
- Wie skaliert die Laufzeit mit Instanzgröße?
  Abhängigkeit von der Struktur der Instanzen?
- Wo sind die Bottlenecks in der Praxis? Vrgl. mit WC-Analyse?
- Wie gut ist die Laufzeit „vorhersagbar“?
- Laufzeit abhängig von der Rechnerarchitektur?
- Vergleich mit bestem bekannten Algorithmus:
  Instanzgröße, Struktur, Architektur → erklärbar?
- Neue Aussagen zu bekannten Algorithmen bzgl. neuer InstanzKlassen?

Welche Testinstanzen sind dafür sinnvoll / notwendig? (z.B. Größe?)

→ **Erkundungsexperimente** um gute Fragen/Instanzgrößen zu finden!

Angemessen effiziente Implementierung

Unzureichende Zeit / Fähigkeit zum Programmieren ist keine zulässige Begründung für schlechte Implementierungen!

Vorteile effizienter Implementierungen:

- Nachweis von Praxistauglichkeit, Konkurrenzfähigkeit
- deutlich schlechtere Implementierung können ein verzerrtes Bild abgeben
- schnelle Implementierung → Zeitersparnis bei Experimenten

Aber:

- Es ist nicht notwendig, jeden Codeteil bis ins letzte Detail zu optimieren
- Finde und optimiere Bottlenecks
- Ziel: in etwa gleiche Effizienz wie in Praxis

Testinstanzen – Quelle

Neue Instanzen vs. **Etablierte Instanzen**
(Neue reale Quellen, neuer Generator,…) (aus der Literatur)

Real-World Instanzen vs. **Generierte Instanzen**
(aus Anwendungen) (meist mit Zufallsgenerator)

Generierter Instanzen:

+ Beliebig große Instanzen generierbar.
– Zufällig generierte Instanzen sind oft viel leichter als Real-World Instanzen.
  → Wenig Aussagekraft für die Anwendung.

Wichtig bei zufällig generierten Instanzen:
Benutze immer **die selben** Instanzen für alle Algorithmen!
(Nicht nur den selben Generator)

Testinstanzen – Größe

Vermeide „zu leichte“ Instanzen.

„Millisekunden-Benchmarkset“ = Extrem kurze Laufzeiten

- Laufzeitunterschiede für Praxis nicht relevant
- keine allgemeinen Aussagen,
  keine Aussagen über größere Instanzen
- Ergebnisse sind stark verfälscht: Hintergrundjobs am Rechner, Genauigkeit der Messuhr (typischerweise 1-2 Hundertstel Sek.),…
- Falls Tests unbedingt nötig: Mit **einer** Zeitmessung „1000“ gleichartige Jobs rechnen. Durchschnittliche Einzellaufzeit durch Division durch 1000.

Nicht nur Instanzen mit bekanntem Optimum testen!

- (Vermeintlicher) Vorteil: Aussage über Qualität von Lösungen eines Approximationsalgorithmus bzw. einer Heuristik möglich
- Problem: Aussage ist nur sehr eingeschränkt, nämlich über einfache/kleine Instanzen. Keine Aussage über Skalierung für große Instanzen

Durchführung der Experimente

Auf jeden Fall:

- Alle Experimente auf dem selben Rechner
- Falls Laufzeiten, Cache-Misses etc. relevant:
  keine sonstige Last auf dem Rechner
- **Infrastruktur**, inkl. Auswertbarkeit der Ergebnisdateien(!), an einem kleinen Testset **testen,** **bevor** man die großen/langwierigen Tests macht.

Dringend empfohlen:

- Gleichartige Ausgabeformate für alle Algorithmen/Varianten
- Ausgabe lesbar für Auswertungstools (z.B. eine Textzeile pro Instanz/Algorithmus, einzelne Werte Tab-separiert,…)

Reproduzier- und Vergleichbarkeit

Echte Reproduzierbarkeit ist in der Praxis **schwer** zu erreichen (neuere Hardware, Compiler, …, leicht andere Implementierung, …).

- Ausführliche Experimente und exakte Beschreibung der Algorithmen, Instanzen, Testumgebung, Resultate,…
- Ergebnisdaten, Testinstanzen, Code bereitstellen!

→ **Vergleichbarkeit**

Alles so aufschreiben, dass später **andere** sich damit vergleichen können:

Häufiges Problem beim Vergleich: Unterschiedliche Hardware

- **Nach Möglichkeit:** Alle Algorithmen am selben Rechner laufen lassen
- **Zur Not:** Benchmarks um Geschwindigkeitsunterschiede der Rechner abzuschätzen

Vergleichsstandards

Mehrere Möglichkeiten, alle mit Vor- und Nachteilen!

Lösungswert angeben

- keine allgemeine Aussage zur Qualität des Algorithmus‘ möglich

Relativer Abstand zur besten bekannten Lösung

- Instanz verfügbar & Wert der besten bekannten Lösung angeben
- Qualität der „besten bekannten“ Lösung unklar

Relativer Abstand zu wohldefinierter unteren Schranke

- Qualität der Schranke?

Relative Verbesserung bzgl. einer anderen Heuristik

- „andere Heuristik“ muss selbst wieder exakt spezifiziert sein

Gefährliche Abbruchkriterien

Beste Lösung nach X Minuten

- **Vorteil:** Man kann die Praxistaulichkeit gut abschätzen
- **Nachteil:** **nicht gut reproduzierbar!**
  Schnellerer Computer: wahrscheinlich insges. bessere Ergebnisse, aber Rangliste der Algorithmen kann unterschiedlich sein
- Oft jedoch nicht vermeidbar ☹

(Bessere?) Alternativen:

messbare **Anzahl** als Abbruchkriterium, z.B. # Iterationen, # Branchings, und
zusätzlich: Tabelle Anzahl → Laufzeit

Abbrechen bei Erreichen der optimale Lösung

- Heuristik bricht ab, sobald optimale Lösung gefunden
- **unrealistisch:** der Wert ist in der Praxis unbekannt
- Performance / Qualität in der Praxis unklar

Alternative: Abbruch, wenn Lösung sich nicht mehr wesentlich verbessert (fixer Prozentsatz ggü. Lösungen der letzten Iterationen)

zusätzlich: Tabelle wann beste Lösung schon erreicht war

Parametereinstellungen

Vermeide manuelles, nicht-regelbasiertes Tunen von Parametereinstellungen

Metaheuristiken mit sehr vielen Parametern
(z.B. Simulated Annealing, genetische Algorithmen etc.)

- Feste Wahl der Parameter für alle Testläufe:
  Parameter in der Dokumentation genau angeben
- Parameter-Settings pro Instanz / Klasse von Instanzen
  Wie kommen diese zu Stande?
  „After experimentation, we found the following settings to work well for this instance/class of instances.“
  Zeit zum Auffinden der Parameter nicht Teil der Laufzeit
  → unfairer Vergleich?

Darstellung der Resultate

Ergebnisse möglichst vollständig aber verdaubar darstellen!

- Tabellen, Diagramme, etc… (→ *Präsentation von experim. Auswertungen*)
- Alle Auswertungskriterien müssen einen Sinn ergeben! (→ Ziel der Exp.)
- **Laufzeit** ist **immer** ein wichtiges Auswertungskriterium!

Anomalien sollten **erklärt** (müssen mindestens erwähnt) werden!
→ sonst fragt sich der Leser: Druckfehler? Bug? Echte Anomalie?

IMMER:

- **Daten mit Interpretation.** Welche Schlüsse kann man ziehen?
- **Schlussfolgerungen** müssen durch die Daten **begründet** sein!
- Aus Diagrammen **kein asymptotisches Verhalten ableiten**!
  Laufzeiten&Lösungsqualitäten von kleinen Instanzen meist
  nicht extrapolierbar

Gerne: Profiling, um Laufzeiten zu verstehen, Bottlenecks zu finden, etc.
→ besseres Verständnis & Ansätze um Algorithmus zu verbessern