

# Algorithm Engineering

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering /  
Experimentelle Algorithmen

LS11

1. VO

SS 2007

2. April 2007

# Kurz-Vorstellung

- **Studium an Univ. Augsburg (WiMa/Math) 1983--1990**
  - Wiss. Mitarb. an Rice University, Houston (TX)
  - Wiss. Mitarb. an FU Berlin
  - **Promotion an Univ. zu Köln (Inf) 1994**
  - **Habilitation am Max-Planck-Institut für Informatik, Saarbrücken 1999**
- 
- Vertr.-Professur (C3) an Univ. Heidelberg (Inf) 1999
  - **Lehrstuhl für Algorithmen und Datenstrukturen, TU Wien 1999--2004**
  - seit Dezember 2004:  
**Lehrstuhl für Algorithm Engineering, LS11**

# Forschungsinteressen

- Algorithmen und Datenstrukturen
- Graphenalgorithmen
- Kombinatorische Optimierung

anwendungsorientiert

- Graphenlayout
- Netzwerkdesign
- Bioinformatik

# Algorithm Engineering

- Design,
- theoretische Analyse,
- Implementierung, und
- experimentelle Evaluation

von Algorithmen und Datenstrukturen

anwendungsorientiert

# Algorithmmentheorie traditionell

- Entwurf von Algorithmen und Datenstrukturen für einfache und abstrakte Problem- und Maschinenmodelle
- Hauptergebnis: beweisbare Leistungsgarantien für alle möglichen Eingaben
- → Elegante, zeitlose, an viele konkreten Anwendungen anpassbare Lösungen
- → Zuverlässig hohe Effizienz auch für zur Implementierungszeit unbekannte Typen von Eingaben

Vorstellung: Anwender greifen Ergebnisse auf, Implementierung, Einbau in Anwendungen

Dies klappt meist nicht!

**Große Lücke zwischen Theorie und Praxis!**

# Symptome der Lücke

Theorie		Praxis
	<b>Problemmodelle</b>	
	<b>Rechnermodelle</b>	
	<b>Algorithmen</b>	
	<b>Datenstrukturen</b>	
	<b>Komplexitätsmaße</b>	
	<b>Effizienz</b>	

# Historie der Algorithmik

# 50er/60er: Pionierzeit

- Pioniere der Algorithmik (Knuth, Floyd,...) gaben Implementierungen an
- aber selten: Tests oder Vergleiche

Knuth:

„Beware of bugs in the above code;  
I have only proved it correct,  
not tried it.“

# 70er/80er: Papier-und Bleistift Jahre

- Algorithmen werden nie implementiert
- Viele Schichten komplexer, unimplementierter Algorithmen und Datenstrukturen
- Sehr abstrakte Beschreibungen auf hohem Level

If you don't make mistakes, you're not working on hard enough problems [F. Wikzek]

- Fehler werden nicht entdeckt → viele nicht-korrekte Algorithmen werden publiziert, z.B.
  - Planare Einbettung bei Planaritätstest, Hopcroft & Tarjan 1974, Mehlhorn 1982
  - Dreizusammenhangszerlegung, Hopcroft & Tarjan 1973
  - Maximaler Planarer Untergraph, Paperreihe von Jayakumar et al. 1986, 1989, Kant 1992

# 70er/80er: Papier-und Bleistift Jahre

- Asymptotische Worst-Case Laufzeit wird überschätzt (vs. tatsächliche Laufzeit in Praxis), z.B.
  - Fredman & Tarjan für MST vs. Prim: „crossover“ bei [redacted] Knoten und mehrere hundert Mio. Kanten
  - Simplex-Algorithmus: theoretisch exponentiell, praktisch viel schneller

- Sehr hohe versteckte Konstanten, z.B.
  - Robertson & Seymour 1985:  $O(|E|^3)$  Algorithmus für „Ist  $G_1$  Minor von  $G_2$ ?“, Konstante bei ca. [redacted]

- Viele praktisch guten Algorithmen konnten nicht veröffentlicht werden, weil es bessere theoretische gab!

# 70er/80er: Papier-und Bleistift Jahre

- Vereinfachte Annahmen, die nicht praxistauglich sind (z.B. allgemeine Lage in Computational Geometry)
- Numerische Probleme werden nicht behandelt

- Algorithmentheorie entfernt sich immer weiter von der Praxis!
- Hinwendung zu praxisfernen Problemen
- Rückzug in wissenschaftlichen „Elfenbeinturm“
- Implementierung = „Finger schmutzig machen“

# Disclaimer: nicht in Mathematischer Optimierung

- Dort gab es immer gute experimentelle Tradition
- Pioniere in der Diskreten Mathematik:
  - Dantzig, Fulkerson, Johnson (TSP, 1954)
  - Gomory (Ganzzahlige Programmierung, 1958)
  - Grötschel (TSP, seit 1980)
  - Jünger, Reinelt (Linear Ordering 1984)
  - Padberg, Rinaldi (TSP, seit 1980)
  - Bixby (Lineare Programmierung, seit 1990)
  - Cook (TSP, seit 1990)

# 80er: Erste experimentelle Vergleiche

- Bentley 1983: „Programming Pearls“
- Jones 1986: Exp. Vergleich von Datenstrukturen für Prioritätswarteschlangen
- Stasko & Vitter 1987: Pairing heaps

- Aber: viele Experimente sind unzureichend:
- Es existieren keine Benchmark-Bibliotheken → Instanzen bei denen man selbst gut abschneidet
- Selten Vergleiche mit „state-of-the-art“ Algorithmen

# 90er: Beginn des AE

- Johnson 1991: Erster DIMACS Computational Challenge: network flow & shortest path → Benchmark Libraries, Datenformate, beste Verfahren

- Moret & Shapiro 1991: Sortierverfahren, 1994: MST
- Cherkassky, Goldberg, Radzig 1996: Kürzeste Wege
- Mehlhorn & Näher: Beginn von LEDA: C++-Library

Pioniere: Goldberg, Johnson, Karp, McGeoch, Mehlhorn, Moret, Orlin, Pevzner

# 90er: Beginn von Algorithm Engineering

- Aho, Johnson, Karp, Kosaraju, McGeoch, Papadimitriou, Pevzner 1996 (seminal NSF-paper): „Emerging Opportunities for Theoretical Computer Science“:

„Efforts must be made to ensure that promising algorithms discovered by the theory community are implemented, tested and refined to the point where they can be usefully applied in practice.“

„...to increase the impact of theory on key application areas.“

# Mitte der 90er: Konferenzen & Zeitschriften

- Seit 1996: ACM Journal of Experimental Algorithmics (JEA)
- Seit 1997: Workshop on Algorithm Engineering (WAE, Italiano) → European Symposium on Algorithms / Applied Track
- Seit 1999: Workshop on Algorithm Engineering and Experiments (ALENEX) → co-located mit ACM Symposium on Discrete Algorithms (SODA)
- Seit 2001: Workshop on Experimental Algorithmics
- ACM Symposium on Computational Geometry: Applied Track (inzwischen integriert)
- Viele Konferenzen haben mehr „Praxis“ im „Call for Papers“

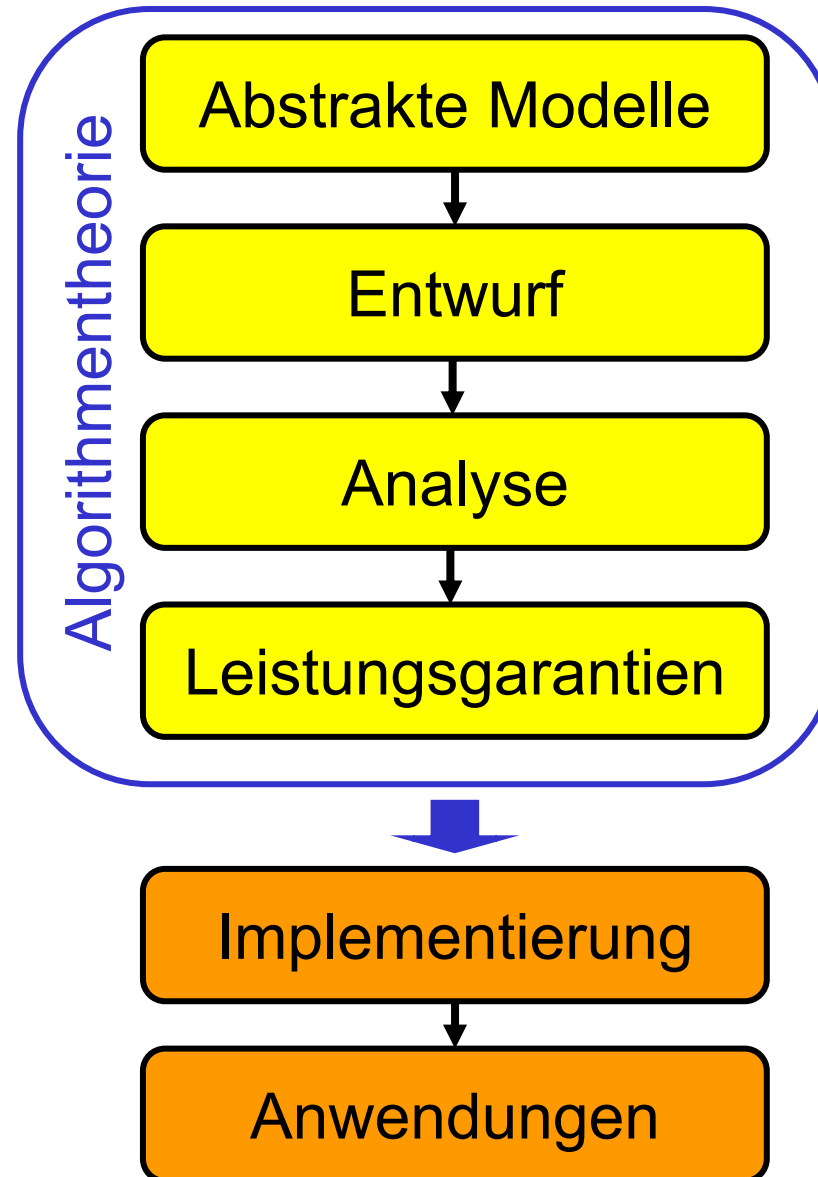
# Ende der 90er: Rechnerarchitektur

- LaMarca & Ladner 1996: „Cache-Optimierung ist machbar, algorithmisch interessant, lohnenswert (auch für Sortierverfahren)
- Vitter 2001: External Memory Modelle und Algorithmen: tiefere Speicherhierarchien als vor 40 Jahren, weit größere Datenmengen

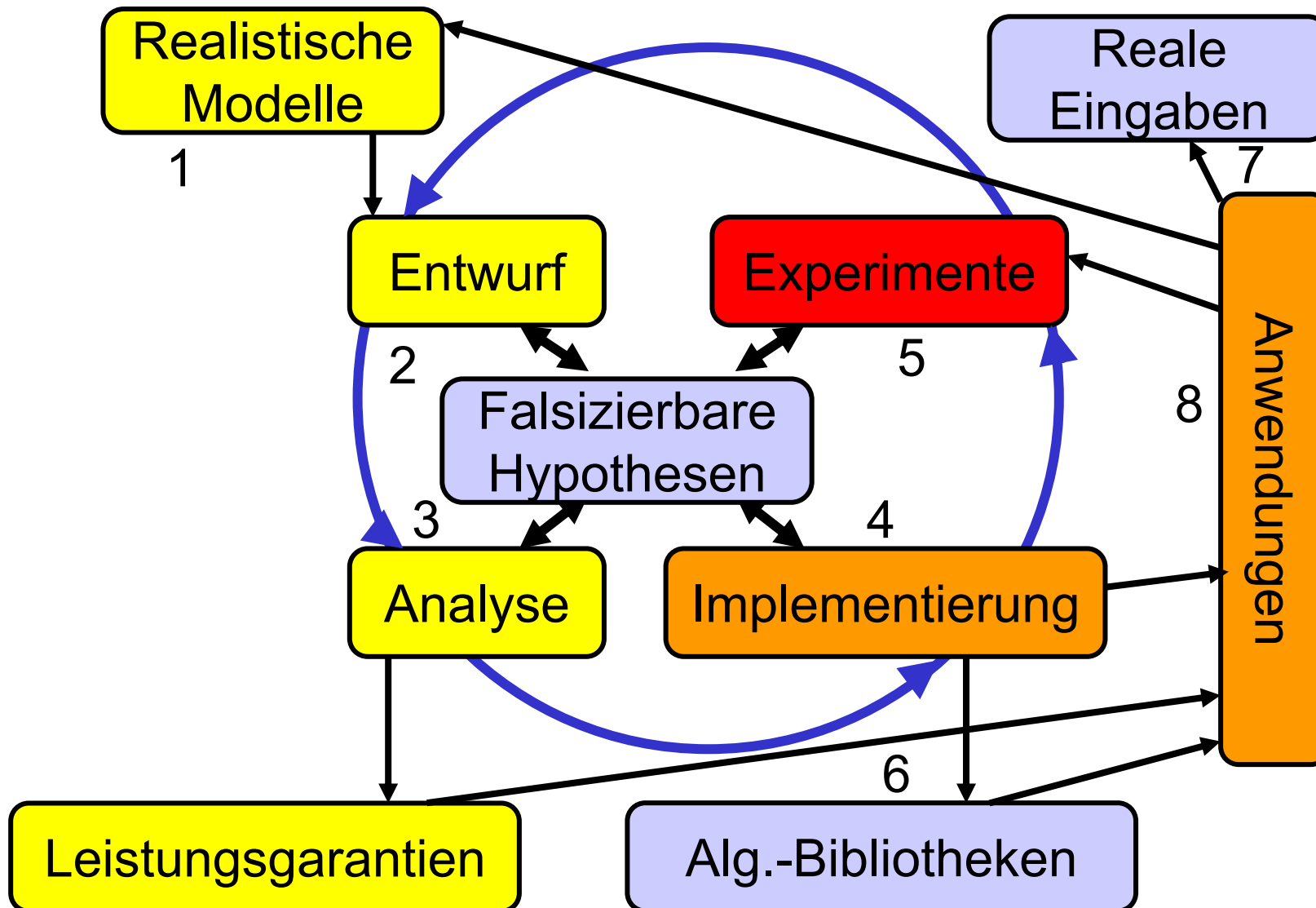
# Einige Erfolgsgeschichten

- Erfolgreiche Algorithmenbibliotheken:
  - LEDA, CGAL, AGD, CPLEX, ABACUS
- Viele vergleichende experimentelle Studien über „beste“ Algorithmen und Datenstrukturen
  - Z.B. Prioritätswarteschlangen, Suchbäume, Hashtabellen, TSP, MST, kürzeste Wege, Konvexe Hülle, Delaunay Triangulierung, Matching, Flüsse)
- Große Instanzen NP-schwerer Probleme gelöst (TSP, Set Cover,...)
- Spektakuläre Speed-Ups über „Everyday“ Code

# Traditionelle Algorithmik



# Algorithm Engineering



# Algorithm Engineering

- Engerer Bezug zu Anwendern
- Überbrückung von Lücken zwischen Theorie und Praxis
- Schneller Transfer von algorithmischem know-how in Anwendungen
- Anregungen aus der Praxis: Probleme und Lösungen
- Verbesserte Algorithmen (Theorie und Praxis)

# Aufgaben des AE

1. Studium von realistischen Modellen für algorithmische Probleme.
2. Studium von realistischen Modellen für realistische Maschinen.
3. Entwurf von einfachen und auch in der Realität effizienten Algorithmen.
4. Analyse praktikabler Algorithmen zwecks Etablierung von Leistungsgarantien, die Theorie und Praxis näher bringen.
5. Implementierungen, die Lücken zwischen bestem theoretischen Algorithmus und bestem implementierten Algorithmus verkleinern.

# Aufgaben des AE

6. Systematische, reproduzierbare Experimente, die der Widerlegung oder Stärkung aussagekräftiger, falsifizierbarer Hypothesen dienen.
7. Entwicklung und Ausbau von Algorithmenbibliotheken, die Anwendungsentwicklungen beschleunigen und algorithmisches Know-how breit verfügbar machen.
8. Sammeln und verfügbar machen von großen und realistischen Probleminstanzen sowie Entwicklung von Benchmarks.
9. Einsatz von algorithmischem Know-how in konkreten Anwendungen.

# Themen der VO

- Ausgewählte Publikationen
  - Algorithmenentwurf, Realisierungen, Experimente, Verbesserungen
  - Datenstrukturen (Prioritätswarteschlangen, SPQR-Bäume)
  - Preprocessing
  - Externspeicheralgorithmen
- Anwendungsprobleme u.a.
  - (Dynamische) Kürzeste Wege
  - Graph Layoutprobleme (z.B. Kreuzungsminimierung)
  - Netzwerkdesignprobleme (Min Cut, Steinerbäume)
  - Bioinformatik (Suffix Arrays)

# Organisatorisches

- Zeiten: Di 12:15-13:45, Do 14:15-15:45

- Übung:

- Projektarbeit in Gruppen, Präsentationen
- 2-wöchig, d.h. 6 Termine
- meist Originalartikel und Experimentieraufgaben, aber auch Programmieraufgaben, DIMACS Challenge
- Termine: Di (notfalls auch Do) ab 16 Uhr?
- Einteilung und Anmeldung: Do 5.4. in VO

- Material:

- Folien, Originalpaper, Mitschriften(?)

# Organisatorisches

- **Schwerpunktgebiete**
  - Algorithmen, Komplexität und formale Modelle
  - Computational Intelligence und Natural Computing
  - Intelligente Systeme

# Prüfungselemente

- Mündliche Fachprüfung:
  - Über VO 4 inkl Ü 2: 9LP
  - Anforderungen:
    - Zusammenhänge des Gebiets
    - Spezielle Fragestellungen einordnen und bearbeiten
    - (Regelmäßige aktive Mitarbeit in Übungen, u.a. mind. drei erfolgreiche Präsentationen)
    - Mündliche Prüfung: Stoff der VO und Ü (20 Min.)

# Prüfungselemente

- Leistungsnachweis:
  - Über VO 4 inkl. Ü 2: 9LP
  - Anforderungen:
    - Regelmäßige aktive Mitarbeit in Übungen, u.a. mind. drei erfolgreiche Präsentationen

# Literatur für diese VO

- B.M.E. Moret und H.D. Shapiro: Algorithms and Experiments: The New (and Old) Methodology, Journal of Universal Computer Science 7 (5) 434-446, 2001
- P. Sanders, K. Mehlhorn, R. Möhring, B. Monien, P. Mutzel, P. Sanders und D. Wagner: Schwerpunktprogramm SPP1307 bei DFG zum Thema „Algorithm Engineering“, <http://www.algorithm-engineering.de>
- P. Sanders: Algorithm Engineering für grundlegende Datenstrukturen und Algorithmen, Vorlesungsfolien WS04/05, <http://www.mpi-sb.mpg.de/~sanders/courses/algen04/>

