Algorithm Engineering

Organisatorisches, Ablauf, Themen

Frank Heitmann
heitmann@informatik.uni-hamburg.de

2. April 2015

Theory vs. Engineering

Algorithmik $\approx$ Programmieren im Kleinen

Theory vs. Engineering

Algorithmik $\approx$ Programmieren im Kleinen

In der Algorithmentheorie eher theoretische Ergebnisse. Nicht sofort klar, wie (und ob) sie praktisch zu benutzen sind. Kurz:

Theory vs. Engineering

Algorithmik $\approx$ Programmieren im Kleinen

In der Algorithmentheorie eher theoretische Ergebnisse. Nicht sofort klar, wie (und ob) sie praktisch zu benutzen sind. Kurz:

Es gibt einen (bisweilen großen) Gap zwischen den Resultaten der Algorithmentheorie und der praktischen Einsetzbarkeit der Algorithmen!

Algorithm Engineering

Hier setzt das Forschungsfeld *Algorithm Engineering* an.

Algorithm Engineering

Hier setzt das Forschungsfeld *Algorithm Engineering* an. Aus einem EATCS Bulletin Beitrag von Demetrescu, Finocchi und Italiano:

Algorithm Engineering 1

Algorithm Engineering is concerned with the design, analysis, implementation, tuning, debugging and experimental evaluation of computer programs for solving algorithmic problems.

Algorithm Engineering

Hier setzt das Forschungsfeld *Algorithm Engineering* an. Aus einem EATCS Bulletin Beitrag von Demetrescu, Finocchi und Italiano:

Algorithm Engineering 1

Algorithm Engineering is concerned with the design, analysis, implementation, tuning, debugging and experimental evaluation of computer programs for solving algorithmic problems.

Oder aus der Wikipedia:

Algorithm Engineering 2

Algorithm Engineering focuses on the design, analysis, implementation, optimization, profiling and experimental evaluation of computer algorithms, bridging the gap between algorithm theory and practical applications of algorithms in software engineering.

Der Projektteil

In diesem **Modul** geht es (zumindest primär und salopp) um die praktische Implementierungen von Algorithmen.

Der Projektteil

In diesem **Modul** geht es (zumindest primär und salopp) um die praktische Implementierungen von Algorithmen.

Im **Projektteil** wollen wir

- verschiedene Algorithmen (in mathematischer Darstellung) verstehen und diese
- implementieren,
- evaluieren, optimieren, ...

Der Projektteil

In diesem **Modul** geht es (zumindest primär und salopp) um die praktische Implementierungen von Algorithmen.

Im **Projektteil** wollen wir

- verschiedene Algorithmen (in mathematischer Darstellung) verstehen und diese
- implementieren,
- evaluieren, optimieren, ...

Klingt erstmal ganz einfach, aber ... ;-)

Der Seminarteil

Zudem wollen wir uns im **Vorlesungs-/Seminarteil** noch

- einige theoretische Hintergründe
  - LH, HH, PH, BH, ...
  - ZPP, RP, BPP, PP, ...
  - AM, MA, ...

  und
- einige (weitere) Algorithmentechniken und
- einige (weitere) Algorithmen

ansehen.

Teils mach ich das, teils macht ihr das.

Der Ablauf 1

Die wöchentlichen Termine:

- Do, 14-16: VL / Seminar, C-221
- Do, 16-18: Projektzeit (insb. Besprechungen), C-221
- Do, 18-20: Individuelle Projektgespräche
- Fr, 16-18: Individuelle Projektgespräche

Der Ablauf 2

Ihr sollt in **Gruppen** von i.A. 2-3 Personen

- im Projektteil Algorithmen in C/C++ oder Java implementieren,
- im Seminarteil (später) einen Vortrag zu z.B. einer Algorithmentechnik halten,
- kleinere Aufgaben bearbeiten, die ich im Vorlesungsteil bzw. ihr selbst im Seminarteil stellt.

Die Gruppen können dabei variieren.

Der Projekt-/Seminarteil ist vergleichbar mit einer Gruppen-Bachelorarbeit.

Ablauf - Zusammenfassung

Also:

- Treffen immer Do 14-16/18 Uhr
- sonst individuell abgesprochene Treffen
- viel Eigenarbeit in den Gruppen

Ablauf - Zusammenfassung

Also:

- Treffen immer Do 14-16/18 Uhr
- sonst individuell abgesprochene Treffen
- viel Eigenarbeit in den Gruppen

Gleich folgen noch

- Themenvorschläge zu den Projekten
- drei kleine Programmieraufgaben zum warm werden

Ablauf - Eure Aufgabe

Eure Aufgabe dann bis nächstes Mal:

Ablauf - Eure Aufgabe

Eure Aufgabe dann bis nächstes Mal:

- alle Themenvorschläge angucken und einen auswählen
  - ggf. schon überlegen, was ihr da genau(er) machen wollt

Ablauf - Eure Aufgabe

Eure Aufgabe dann bis nächstes Mal:

- alle Themenvorschläge angucken und einen auswählen
  - ggf. schon überlegen, was ihr da genau(er) machen wollt
- die Programmieraufgaben bearbeiten (in C/C++ oder Java)

Ablauf - Eure Aufgabe

Eure Aufgabe dann bis nächstes Mal:

- alle Themenvorschläge angucken und einen auswählen
  - ggf. schon überlegen, was ihr da genau(er) machen wollt
- die Programmieraufgaben bearbeiten (in C/C++ oder Java)
- für beides jeweils Gruppen von 2-3 Personen bilden

Ablauf - Eure Aufgabe

Eure Aufgabe dann bis nächstes Mal:

- alle Themenvorschläge angucken und einen auswählen
  - ggf. schon überlegen, was ihr da genau(er) machen wollt
- die Programmieraufgaben bearbeiten (in C/C++ oder Java)
- für beides jeweils Gruppen von 2-3 Personen bilden

Für die Gruppenbildung erstell ich noch einen Commsy-Raum oder ähnliches.

Zum Seminar (und den Themen dort) folgt später im Semester mehr.

Ablauf - Nächstes Mal

Nächstes Mal dann

- Do 14-16 Uhr, VL von mir
- Do 16-18 Uhr, PJ
  - Programmieraufgaben/Lösungen besprechen
  - Themenvorschläge bzw. eure Auswahl besprechen

(So dann auch erstmal die ersten Wochen...)

Fragen?

Alles klar
?!?

Themenvorschläge

Thema: SAT-Solver

Ein **SAT-Solver** lösst das Erfüllbarkeitsproblem der Aussagenlogik.
Wie implementiert man so etwas?

Thema: SAT-Solver

Ein **SAT-Solver** lösst das Erfüllbarkeitsproblem der Aussagenlogik. Wie implementiert man so etwas?

Verschiedene Ansätze wie z.B. Brute-Force, Resolution, Tableaux, BDDs, DLL, ...

Thema: SAT-Solver

Ein **SAT-Solver** lösst das Erfüllbarkeitsproblem der Aussagenlogik. Wie implementiert man so etwas?

Verschiedene Ansätze wie z.B. Brute-Force, Resolution, Tableaux, BDDs, DLL, ...

- Logic in Computer Science, M. Huth und M. Ryan, Cambridge University Press, 2.ed, 2004.
- Mathematical Logic for Computer Science, M. Ben-Ari, Springer, 3.ed, 2012.
- Das Erfüllbarkeitsproblem SAT, U. Schöning und J. Torán, Lehmanns Media, 2012.
- An Extensible SAT-solver, N. Eén und N. Sörensson, in Theory and Applications of Satisfiability Testing, Lecture Notes in Computer Science, Volume 2919, 2004, pp 502-518.
- Searching for Truth: Techniques for Satisfiabiliy of Boolean Formulas, Dissertation von Lintao Zhang, Princeton, 2003.

Thema: Exakte Algorithmen (für Probleme in NP)

Exakte Algorithmen (für Probleme in NP) lösen ein Problem aus NP exakt (nicht randomisiert o.ä), bleiben dabei aber exponentiell. Man versucht die Schranke möglichst klein zu halten.

Thema: Exakte Algorithmen (für Probleme in NP)

Exakte Algorithmen (für Probleme in NP) lösen ein Problem aus NP exakt (nicht randomisiert o.ä), bleiben dabei aber exponentiell. Man versucht die Schranke möglichst klein zu halten.

Man kann verschiedene Probleme betrachten oder sich auf eines wie z.B. **Graphfärbbarkeit** konzentrieren.

Thema: Exakte Algorithmen (für Probleme in NP)

Exakte Algorithmen (für Probleme in NP) lösen ein Problem aus NP exakt (nicht randomisiert o.ä), bleiben dabei aber exponentiell. Man versucht die Schranke möglichst klein zu halten.

Man kann verschiedene Probleme betrachten oder sich auf eines wie z.B. **Graphfärbbarkeit** konzentrieren.

- Exact Algorithms for NP-hard Problems: A Survey, G. J. Woeginger, in Combinatorial Optimization Eureka, You Shrink!, Lecture Notes in Computer Science, Volume 2570, 2003, pp 185-207.
- Exakte Algorithmen fr schwere Graphenprobleme, F. Gurski, I. Rother, J. Rothe, E. Wanke, Springer, 2010.

Thema: Algorithmen in der polynomiellen Hierarchie

Die **polynomielle Hierarchie** (PH) liegt überhalb von NP und in PSpace. Etliche Probleme sind vollständig für die einzelnen Stufen. Wie löst man diese Probleme algorithmisch?

Thema: Algorithmen in der polynomiellen Hierarchie

Die **polynomielle Hierarchie** (PH) liegt überhalb von NP und in PSpace. Etliche Probleme sind vollständig für die einzelnen Stufen. Wie löst man diese Probleme algorithmisch?

Wieder kann man verschiedene Probleme betrachten oder sich auf eines wie z.B. **MIN DNF** konzentrieren.

Thema: Algorithmen in der polynomiellen Hierarchie

Die **polynomielle Hierarchie** (PH) liegt überhalb von NP und in PSpace. Etliche Probleme sind vollständig für die einzelnen Stufen. Wie löst man diese Probleme algorithmisch?

Wieder kann man verschiedene Probleme betrachten oder sich auf eines wie z.B. **MIN DNF** konzentrieren.

- Completeness in the Polynomial-Time Hierarchy: A Compendium. M. Schaefer und C. Umans, 2008. [Insb. mit einer Liste von Problemen!]
- Completeness in the Polynomial-Time Hierarchy: Part II. M. Schaefer und C. Umans, SIGACT News Complexity Theory Column 38, 2002. [Eher Theoretischer Natur.]
- Approximability and Completeness in the Polynomial Hierarchy. Dissertation von Christopher M. Umans, Berkeley, 2000.

Thema: QBF-Solver

Ein **QBF-Solver** löst das Erfüllbarkeitsproblem von QBF-Formeln.
Dieses ist für PSpace das, was für NP SAT ist.

Thema: QBF-Solver

Ein **QBF-Solver** löst das Erfüllbarkeitsproblem von QBF-Formeln. Dieses ist für PSpace das, was für NP SAT ist.

- Introduction to Automata Theory, Languages, and Computation. J. E. Hopcroft, R. Motwani und J. D. Ullman, 2. ed., Addison-Wesley, 2001.
- Algorithm Design. J. Kleinberg und É. Tardos, Pearson, 2006.
- Searching for Truth: Techniques for Satisfiabiliy of Boolean Formulas. Dissertation von Lintao Zhang, Princeton, 2003.

Thema: Reduktionen

Um die mittlerweile recht mächtigen SAT- und QBF-Solver zu nutzen, muss man zunächst eine **Reduktion auf SAT- bzw. QBF** finden und **implementieren**.

Thema: Reduktionen

Um die mittlerweile recht mächtigen SAT- und QBF-Solver zu nutzen, muss man zunächst eine **Reduktion auf SAT- bzw. QBF** finden und **implementieren**.

Grober Ablauf:

- Beliebiges, einen interessierendes Problem aus NP bzw. PSpace nehmen,
- auf SAT bzw. QBF reduzieren
- die Reduktion implementieren
- Ergebnis als Eingabe für SAT- bzw. QBF-Solver.
- Evaluieren.

Thema: ?

Eigene Themenvorschläge sind willkommen! Weitere Ideen sind z.B.

- Randomisierte Algorithmen
- Approximationsalgorithmen
- Heuristiken
- ...

Ich werde über Ostern noch weitere Vorschläge auf die Webseite tun.

Programmieraufgaben

Programmieraufgaben

Implementiert in C/C++ oder Java

Programmieraufgaben

Programmieraufgaben

Implementiert in C/C++ oder Java

1. Den MergeSort-Algorithmus
   - Divide & Conquer-Verfahren

Programmieraufgaben

Programmieraufgaben

Implementiert in C/C++ oder Java

1. Den MergeSort-Algorithmus
   - Divide & Conquer-Verfahren
2. Den Algorithmus von Kruskal zur Bestimmung minimaler Spannbäume
   - Greedy-Verfahren
   - Graph-Datenstruktur nötig
   - an einer Stelle schwierig - welche? (Da könnt ihr ggf. auch abbrechen)

Programmieraufgaben

Programmieraufgaben

Implementiert in C/C++ oder Java

1. Den MergeSort-Algorithmus
   - Divide & Conquer-Verfahren
2. Den Algorithmus von Kruskal zur Bestimmung minimaler Spannbäume
   - Greedy-Verfahren
   - Graph-Datenstruktur nötig
   - an einer Stelle schwierig - welche? (Da könnt ihr ggf. auch abbrechen)
3. 2-Färbbarkeit und 3-Färbbarkeit von Graphen
   - Nutzt obige Graph-Datenstruktur (ggf. anpassen)
   - Greedy-Verfahren und Backtracking-Verfahren
   - *NP*-Vollständigkeit

Programmieraufgaben

Programmieraufgaben

Implementiert in C/C++ oder Java

1. Den MergeSort-Algorithmus
   - Divide & Conquer-Verfahren
2. Den Algorithmus von Kruskal zur Bestimmung minimaler Spannbäume
   - Greedy-Verfahren
   - Graph-Datenstruktur nötig
   - an einer Stelle schwierig - welche? (Da könnt ihr ggf. auch abbrechen)
3. 2-Färbbarkeit und 3-Färbbarkeit von Graphen
   - Nutzt obige Graph-Datenstruktur (ggf. anpassen)
   - Greedy-Verfahren und Backtracking-Verfahren
   - *NP*-Vollständigkeit

Wichtig: Implementiert viel selbst (Datenstrukturen etc.) und nutzt nicht Bibliotheksfunktionen!

Zusammenfassung

Eure Aufgabe bis nächstes Mal:

- alle Themenvorschläge angucken und einen auswählen
  - ggf. schon überlegen, was ihr da genau(er) machen wollt z.B.
  - welches NP-Problem wollt ihr betrachten (exakte Algorithmen)
  - welches PH-Problem wollt ihr betrachten
  - was genau interessiert euch beim SAT- bzw. QBF-Solver
  - ...
- die Programmieraufgaben bearbeiten (in C/C++ oder Java)
- für beides jeweils Gruppen von 2-3 Personen bilden

Nächstes Mal dann

- Do 14-16 Uhr, VL von mir
- Do 16-18 Uhr, PJ
  - Programmieraufgaben besprechen
  - Themenvorschläge bzw. eure Auswahl besprechen

Viel Spaß!