DM841

Discrete Optimization

Lecture 1

Course Introduction
Constraint Programming

Combination

Simplification

Contradiction

Redundancy

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Course Introduction

Outline

Discrete and Combinatorial Optimization

- Discrete optimization emphasizes the difference to continuous optimization, solutions are described by integer numbers or discrete structures
- Combinatorial optimization is a subset of discrete optimization.
- Combinatorial optimization is the study of the ways discrete structures (eg, graphs) can be selected/arranged/combined: Finding an optimal object from a finite set of objects.
- Discrete/Combinatorial Optimization involves finding a way to efficiently allocate resources in mathematically formulated problems.

Combinatorial Problems

Combinatorial problems

They arise in many areas of
Computer Science, Artificial Intelligence, Operations Research...:

- allocating register memory
- planning, scheduling, timetabling
- Internet data packet routing
- protein structure prediction
- auction winner determination
- portfolio selection
- ...

Combinatorial Problems

Simplified models are often used to formalize real life problems

- finding models of propositional formulae (SAT)
- finding variable assignment that satisfy constraints (CSP)
- partitioning graphs or digraphs
- partitioning, packing, covering sets
- finding shortest/cheapest round trips (TSP)
- coloring graphs (GCP)
- finding the order of arcs with minimal backward cost
- ...

Example Problems

- They are chosen because conceptually concise, intended to illustrate the development, analysis and presentation of algorithms
- Although real-world problems tend to have much more complex formulations, these problems capture their essence

Elements of Combinatorial Problems

Combinatorial problems are characterized by an input,
i.e., a general description of conditions (or constraints) and parameters,
and a question (or task, or objective) defining
the properties of a solution.

They involve finding a grouping, ordering, or assignment
of a discrete, finite set of objects that satisfies given conditions.

Candidate solutions are combinations of objects or solution components that need not satisfy all given conditions.

Feasible solutions are candidate solutions that satisfy all given conditions.

Optimal Solutions are feasible solutions that maximize or minimize some criterion or objective function.

Approximate solutions are feasible candidate solutions that are not optimal but good in some sense.

Applied Character

Optimization problems are very challenging, seldom solvable exactly in polynomial time and no single approach is likely to be effective on all problems.
Solving optimization problems remains a very experimental endeavor: what will or will not work in practice is hard to predict. [HM]

Hence the course has applied character:

- We will learn the theory
- but also implement some solvers $\rightsquigarrow$ programming in C++
- We will learn how to analyze the experimental results

Course Aims

Basically two Parts:

Part I: Constraint Programming (CP)
Part II: Heuristics

Part I: ($\approx$ 12 classes)

- Modeling Problems in CP
- Local Consistency
- Constraint Propagation
- Search
- Symmetry Breaking

Part II: ($\approx$ 12 classes)

- Local Search
- Metaheuristics
- Implementation Framework
- Efficiency issues
- Experimental Analysis

Schedule

- Class schedule:
  - See course web page (get the ical-feed).
  - mitsdu.sdu.dk, SDU Mobile

- Working load:
  - Intro phase (Introfase): 48 hours, 24 classes
  - Skills training phase (Træningsfase): 20 hours, 10 classes
  - Study phase: (Studiefase) 30 hours

  We have 42 classes scheduled.

Evaluation

- Obligatory Assignments:
  Part I:
  Two preparation assignments with pass/fail
  One midterm with 7-grade scale + external censor
  Part II:
  One/Two preparation assignments with pass/fail
  One final assignment with 7-grade scale + external censor
- All assignments must be passed.
- Final grade is weighted(?) average of midterm and final assignments.
- Preparation assignments can be prepared in pairs but individual submission $\rightsquigarrow$ Feedback
- Midterm and final assignments are individual and communication not allowed.

Content of the Graded Assignments

- Algorithm design
- Modeling
- Implementation (deliverable and checkable source code)
- Written description
- (Analytical) and experimental analysis
- Performance counts!

Web submission with automatic check, execution and comparison.

Communication media

- Course Public Webpage (WWW) $\Leftrightarrow$ BlackBoard (BB)
  (link from http://www.imada.sdu.dk/~marco/DM841/)
- Announcements in BlackBoard
- Course Documents (for photocopies) in (BB)
- Discussion Board (anonymous) in (BB)
- (A-bit-earlier-than) Mid term evaluation in class
- Personal email
- Office visits

Literature

- Part I (on Constraint Programming):
  - RBW F. Rossi, P. van Beek and T. Walsh (ed.), Handbook of Constraint Programming, Elsevier, 2006
  - STL C. Schulte, G. Tack, M.Z. Lagerkvist, Modelling and Programming with Gecode 2013
- Part II (on Local Search):
  - HM P.V. Hentenryck and L. Michel. Constraint-Based Local Search. The MIT Press, Cambridge, USA, 2005. (In BlackBoard)
  - MAK W. Michiels, E. Aarts and J. Korst. Theoretical Aspects of Local Search. Springer Berlin Heidelberg, 2007
  - HS H. Hoos and T. Stuetzle, Stochastic Local Search: Foundations and Applications, 2005, Morgan Kaufmann
- Other sources: articles, slides, lecture notes
- https://class.coursera.org/optimization-001

Software

Under development:
`http://www.minizinc.org/challenge2014/results2014.html`

Here, we will use *free* and open-source software:

- Constraint Programming: Gecode (C++) – MIT license
- Local Search: C++
- Experimental Analysis: R – The R project

Many others, some commercial

Knowledge in Programming and Algorithm and Data Structures is assumed.
C/C++ Language

Class format

Be prepared for:

- Flipped classes: learn content at home, engage with material in class
- Problem solving in class
- Hands on experience with programming
- Experimental analysis of performance
- Discussion on exercises for home

These activities will be announced

They require study phase (= work outside the classes)

Former students' feedback (1/2)

On the course:

- the course builds on a lot of knowledge from previous courses
- programming
- practical drive
- taught on examples
- no sharp rules are given and hence more space left to creativity
- unexpected heavy workload
- the assignments are really an important preparation to the final projects
- Group work and practical examples were good and usable
- The course was intellectually stimulating
- It is not always easy to know the standard of work expected assignments were too open
- Better with separation between submission of code and report

Former students' feedback (2/2)

On the exam:

- hardest part is the design of the heuristics
  the content of the course is vast $\rightsquigarrow$ many possibilities without clue on what will work best.

In general:

- Examples are relevant, would be nice closer look at source code.

From my side, mistakes I would like to see avoided:

- non competitive local search procedures
- bad descriptions
- mistaken data aggregation in instance set analysis.

Good/bad examples and rubric of comments will be made available

You

- Whole course or a part?
- Background
  education line
  programming skills
  DM554/DM545, integer and linear
- Expectations