DM841

Discrete Optimization

Lecture 2

Solution Methods

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Last Time

1. Course Introduction

2. Combinatorial Optimization and Terminology

3. Basic Concepts from Algorithmics
   (Review slides and Cormen, Leiserson, Rivest and Stein. *Introduction to algorithms*. 2001)
   Graphs • Notation and runtime • Machine model • Pseudo-code • Computational Complexity • Analysis of Algorithms

Outline

Outline

General vs Instance

General problem *vs* problem instance:

General problem Π:

- Given *any* set of points X in a square, find a shortest Hamiltonian cycle
- *Solution:* Algorithm that finds shortest Hamiltonian cycle for any X

Problem instantiation $\pi = \Pi(I)$:

- Given a specific set of points I in the square, find a shortest Hamiltonian cycle
- *Solution:* Shortest Hamiltonian cycle for I

Problems can be formalized on sets of problem instances $\mathcal{I}$ (instance classes)

Traveling Salesman Problem

Types of TSP instances:

- Symmetric: For all edges uv of the given graph G, vu is also in G, and $w(uv) = w(vu)$.
  Otherwise: asymmetric.
- Euclidean: Vertices = points in an Euclidean space,
  weight function = Euclidean distance metric.
- Geographic: Vertices = points on a sphere,
  weight function = geographic (great circle) distance.

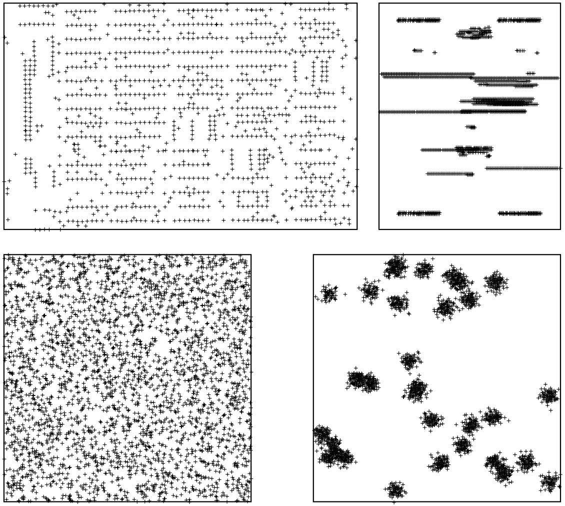
TSP: Benchmark Instances

Instance classes

- Real-life applications (geographic, VLSI)
- Random Euclidean
- Random Clustered Euclidean
- Random Distance

Available at the TSPLIB (more than 100 instances upto 85.900 cities) and at the 8th DIMACS challenge

TSP: Instance Examples



Outline

Methods and Algorithms

A Method is a general framework for the development of a solution algorithm. It is not problem-specific.

An Algorithm (or algorithmic model) is a problem-specific template that leaves only some practical details unspecified.
The level of detail may vary:

- minimally instantiated (few details, algorithm template)
- lowly instantiated (which data structure to use)
- highly instantiated (programming tricks that give speedups)
- maximally instantiated (details specific of a programming language and computer architecture)

A Program is the formulation of an algorithm in a programming language.

An algorithm can thus be regarded as a class of computer programs (its implementations)

Solution Methods

- **Exact methods** (complete)
  guaranteed to find (optimal) solution,
  or to determine that no solution exists (eg, systematic enumeration)
  - Search algorithms (backtracking, branch and bound)
  - Dynamic programming
  - Constraint programming
  - Integer programming
  - Dedicated Algorithms

- **Approximation methods**
  worst-case solution guarantee
  http://www.nada.kth.se/~viggo/problemlist/compendium.html

- **Heuristic (Approximate) methods** (incomplete)
  not guaranteed to find (optimal) solution,
  and unable to prove that no solution exists

Exact Methods

Problem specific methods:

- Dynamic programming (knapsack)
- Dedicated algorithms (greedy, shortest path)

General methods:

- Integer (Mathematical) Programming
- Constraint Programming

Generic methods:

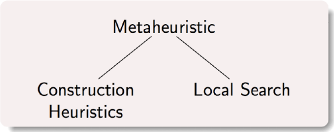
👍 Allow to save development time

👎 Do not achieve same performance as specific algorithms

Heuristics

Get inspired by approach to problem solving in human mind
[A. Newell and H.A. Simon. "Computer science as empirical inquiry: symbols and search." Communications of the ACM, ACM, 1976, 19(3)]

- effective rules
- trial and error



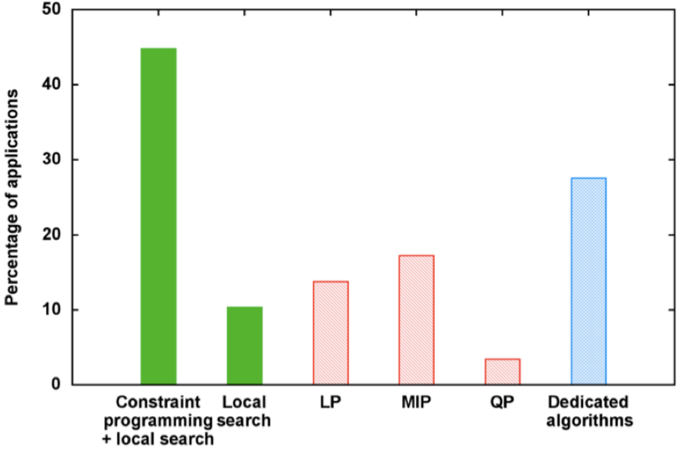
Applications:

- Optimization
- But also in Psychology, Economics, Management [Tversky, A.; Kahneman, D. (1974). "Judgment under uncertainty: Heuristics and biases". Science 185]

Basis on empirical evidence rather than mathematical logic. Getting things done in the given time.

Applications

Distribution of technology used at Google for optimization applications developed by the operations research team



[Slide presented by Laurent Perron on OR-Tools at CP2013]

Outline

Outline

1. Combinatorial Optimization
   - Solution Methods

2. Exact Methods: Examples
   - SAT
   - Mathematical Programming
   - Backtracking

SAT Problem

Satisfiability problem in propositional logic

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

SAT Problem

Satisfiability problem in propositional logic

$$
\begin{aligned}
&(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
&(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
&(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
&(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
&(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
&(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
&(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
&(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
&(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
&(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
&(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
&(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{aligned}
$$

Does there exist a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

Motivation

- From 100 variables, 200 constraints (early 90s)
  to 1,000,000 vars. and 20,000,000 cls. in 20 years.
- Applications:
  Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.
- SAT used to solve many other problems!

SAT Problem
Satisfiability problem in propositional logic

Definitions:

- Formula in propositional logic: well-formed string that may contain
  - propositional variables $x_1, x_2, \ldots, x_n$;
  - truth values $\top$ ('true'), $\bot$ ('false');
  - operators $\neg$ ('not'), $\wedge$ ('and'), $\vee$ ('or');
  - parentheses (for operator nesting).
- Model (or satisfying assignment) of a formula F: Assignment of truth values to the variables in F under which F becomes true (under the usual interpretation of the logical operators)
- Formula F is satisfiable iff there exists at least one model of F, unsatisfiable otherwise.

From Propositiaonl Logic to SAT

Propositional logic: operators: $\neg P, P \wedge Q, P \vee Q, P \implies Q, P \Leftrightarrow Q$

To conjunctive normal form:

- replace $\alpha \Leftrightarrow$ with $(\alpha \implies \beta) \wedge (\beta \implies \alpha)$
- replace $\alpha \implies \beta$ with $\neg\alpha \vee \beta$
- $\neg$ must appear only in literals, hence move $\neg$ inwards
- distributive law for $\vee$ over $\wedge$:

$$(\alpha \vee (\beta \vee \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma))$$

SAT Problem (decision problem, search variant):

- **Given:** Formula F in propositional logic
- **Task:** Find an assignment of truth values to variables in F that renders F true, or decide that no such assignment exists.

SAT: A simple example

- **Given:** Formula $F := (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
- **Task:** Find an assignment of truth values to variables x_1, x_2 that renders F true, or decide that no such assignment exists.

Definitions:

- A formula is in conjunctive normal form (CNF) iff it is of the form

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{ij} = (l_{11} \vee \ldots \vee l_{1k_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mk_m})$$

  where each literal l_{ij} is a propositional variable or its negation. The disjunctions $c_i = (l_{i1} \vee \ldots \vee l_{ik_i})$ are called clauses.

- A formula is in k-CNF iff it is in CNF and all clauses contain exactly k literals (*i.e.*, for all i, $k_i = k$).

- In many cases, the restriction of SAT to CNF formulae is considered.
- For every propositional formula, there is an equivalent formula in 3-CNF.

Example:

$$
\begin{aligned}
F := \quad & \wedge (\neg x_2 \vee x_1) \\
& \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \\
& \wedge (x_1 \vee x_2) \\
& \wedge (\neg x_4 \vee x_3) \\
& \wedge (\neg x_5 \vee x_3)
\end{aligned}
$$

- F is in CNF.
- Is F satisfiable?
  Yes, *e.g.*, $x_1 := x_2 := \top$, $x_3 := x_4 := x_5 := \bot$ is a model of F.

Special Cases

Not all instances are hard:

- Definite clauses: exactly one literal in the clause is positive. Eg:

$$\neg\beta \lor \neg\gamma \lor \alpha$$

- Horn clauses: at most one literal is positive.
  - Easy interpretation: $\alpha \land \beta \Longrightarrow \gamma \rightsquigarrow \neg\alpha \lor \neg\beta \lor \gamma$
  - inference is easy by forward checking, linear time

Outline

Mathematical Programming Models

- How to model an optimization problem
  - choose some decision variables
    they typically encode the result we are interested into
  - express the problem constraints in terms of these variables
    they specify what the solutions to the problem are
  - express the objective function
    the objective function specifies the quality of each solution
- The result is an optimization model
  - It is a declarative formulation
    specify the "what", not the "how"
  - There may be many ways to model an optimization problem

IP model

Standard IP formulation: Let x_l be a 0–1 variable equal to 1 whenever the literal l takes value true and 0 otherwise.
Let c^+ be the set of literals in clause $c \in C$ that appear as positive and c^- the set of variables that appear as negated.

$$
\begin{aligned}
\min \quad & 1 \\
\text{s.t.} \quad & \sum_{l \in c^+} x_l + \sum_{l \in c^-} (1 - x_l) = 1, && \forall c \in C, \\
& x_l \in \{0, 1\}, && \forall l \in L
\end{aligned}
$$

Outline

Max SAT

Definition

(Maximum) K-Satisfiability (SAT)
Input: A set U of variables, a collection C of disjunctive clauses of at most k literals, where a literal is a variable or a negated variable in U. k is a constant, $k > 2$.
Task: A truth assignment for U or a truth assignment that maximizes the number of clauses satisfied.

MAX-SAT (optimization problem)

Which is the maximal number of clauses satisfiable in a propositional logic formula F?

DPLL algorithm

Davis Putam, Logenmann & Loveland (DPLL)
Recursive depth-first enumeration of possible models

1. Early termination:
   a clause is true if any of its literals are true
   a sentence is false if any of its clauses are false, which occurs when all its literals are false

2. Pure literal heuristic:
   pure literal is one that appears with same sign everywhere.
   it can be assigned so that it makes the clauses true. Clauses already true can be ignored.

3. Unit clause heuristic
   consider first unit clause with just one literal or all literal but one already assigned. Generates cascade effect (forward chaining)

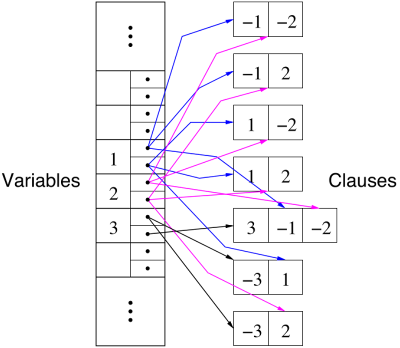
DPLL algorithm

Function `DPLL`(C, L, M)**:**

Data: C set of clauses; L set of literals; M model;
Result: *true* or *false*
if every clause in C is true in M **then return** *true*;
if some clause in C is false in M **then return** *false*;
$(l, val) \leftarrow$`FindPureLiteral`(L, C, M);
if l is non-null **then return** `DPLL`($C, L \setminus l, M \cup \{l = val\}$);
$l \leftarrow$`First`(L); $R \leftarrow$`Rest`(L);
return `DPLL`($C, R, M \cup \{l = true\}$) or
`DPLL`($C, R, M \cup \{l = false\}$)

Speedups

- Component analysis
- Variable value ordering
- Intelligent backtracking
- Random restarts
- Clever indexing (data structures)



Summary