Seminar on Constraint Solving Meets Data Mining and Machine Learning
Spring 2013

Matti Järvisalo

Practical Arrangements, introduction.
Course Information

Instructor: Dr. Matti Järvisalo
matti.jarvisalo@cs.helsinki.fi

Reception: Contact instructor by email for an appointment

Credit units: 3 ECTS

Language: English (by default)

WWW:
Course Requirements

- Choose a topic (scientific article) to study
- Write a 10-15 page (plus references) report on the topic
- Give a 30-min presentation on the topic
- Give constructive feedback on another student’s report (and draft)
- Act as the opponent of another student’s presentation
- Attend the seminar 1-2 workshop day(s) in May
Choosing a Topic

- List of topics available on the seminar webpage
- If you have not reserved a topic, do this by this Friday March 15
- Each topic consists of one scientific article
- *Can suggest a topic outside the list!*
- The article provides a starting-point for your work
- You may need to read additional articles for necessary background
- Synthesis of multiple related articles is a major plus
Deadlines

- All deadlines are strict — you will fail the course if you do not meet a deadline
- March 15: vote on the workshop dates, choose topic
- April 13: at least 5-page draft report (send to teacher)
- April 20: feedback on another student’s report draft (send to teacher and your opponent)
- One week before the workshop: Full-length report and preliminary presentation slides (send to teacher and your opponent)
- At the workshop: act as an opponent
- One week after the workshop: Final report
Report and Presentation

- A seminar report is a short review paper: you explain some interesting results in your own words.
- A typical seminar report will consist of the following parts:
  - an informal introduction,
  - a formally precise definition of the problem that is studied,
  - a brief overview of very closely related work—here you might cite approx. 3–10 papers and explain their main contributions,
  - a more detailed explanation of one or two interesting results, with examples
  - conclusions.

- Superficially, your report should look like a typical scientific article.
  - However, it will not contain any new scientific results, just a survey of previously published work.

- The presentation is an overview of the report
  - You should understand what you are saying
  - Everyone should understand you
  - The abstraction level should be right
  - Examples are always good to communicate ideas
Agreeing on the workshop day(s)

- We need to agree on two full workshop days during May 6–14
- All presentations take place during the workshop days
- Go to http://www.doodle.com/xwbnvtvabiftmr7c
- Attendance mandatory at least on the day you are scheduled for
What This Course is about

- Interplay between constraint solving and data analysis
- How to use constraint satisfaction and optimization techniques to solve data analysis task
  - Providing optimal solutions?
  - Addressing more general problems that classical approaches?
  - Example: Clustering: from $k$-means style local search to guaranteed optimal clustering?
- How to use machine learning to speed-up constraint solving in practice?
  - Learning to select the best algorithm for solving a given problem instance as input
Declarative Programming and Constraint Solving

Two-step approach to solving hard combinatorial problems:

1. **Encoding**: Domain-specific declarative formulation of problem using chosen (constraint) modelling language
   - Given any problem instance, formulate the instance in terms of mathematical constraints

2. **Solving**: A generic solver—a search algorithm—for the chosen modelling language, which can find a solution (or determine that none exist) to any formulation in the modelling language
   - Found solution mapped back to a solution of the original problem instance

Various approaches based on different modelling languages: integer programming, linear programming, constraint programming, Boolean satisfiability, Boolean optimization (MaxSAT), ...
Constraints: A general view

- A set of variables $X = \{x_1, \ldots, x_n\}$
- Each variable $x_i$ has domain $D_i$
- A constraint $C$ over $X$ is a subset of $D_1 \times \cdots \times D_n$

**Example.** Let $D_1 = D_2 = \{1, 2, 3\}$. The constraint $\neq$ over $x_1, x_2$ is
\[
\{(d_1, d_2) \mid d_1 \in D_1, d_2 \in D_2, d_1 \neq d_2\}
\]

\[
= \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\} \subset D_1 \times D_2.
\]

- The above is an example of a *finite-domain* constraint, where the domain of each variable is a finite set of values
  - A special case are *Boolean constraints* that are defined over Boolean variables, i.e., variables with domain $B = \{0, 1\}$.
    - 1 is the value *true*, 0 is the value *false*
- Depending on the constraint language, the variables may also have infinite domains.
  - Examples: $\mathbb{R}$ (real domains), $\mathbb{Z}$ (integer domains)
Constraint Satisfaction Problems (CSPs)

- Given: a set of variables $X = \{x_1, \ldots, x_n\}$ with domains $D_1, \ldots, D_n$
- a constraint satisfaction problem (CSP) is a set of constraints $C = \{C_1, \ldots, C_m\}$
  - each constraint $C$ is defined over some subset of $X$.
- Value assignment for $x_1, \ldots, x_n$ is a function $T$ that assigns for each $x_i$ a value from the domain $D_i$.
- $T$ satisfies a constraint $C$ over variables $x_{i_1}, \ldots, x_{i_k} \subseteq X$ iff $(T(x_{i_1}), \ldots, T(x_{i_m})) \in C$.
- $T$ is a solution to $C$ iff $T$ satisfies every constraint in $C$.
- If $C$ has a solution, then $C$ is satisfiable, and otherwise unsatisfiable.
CSPs: Example

Quasigroup Completion problem:

- \( n \times n \) matrix
- Some cells have been pre-filled
- Fill each of the cells with integers \( 1, \ldots, n \) so that:
  - each \( 1..n \) appears exactly once in each column and row

A CSP encoding:

- Let \( \text{AllDiff}(x_1, \ldots, x_k) = \{ (v_1, \ldots, v_k) \mid v_1 \in D_1, \ldots, v_k \in D_k, v_i \neq v_j \ \forall i \neq j, i, j \in \{1, \ldots, k\} \} \)
- Introduce variable \( x_{ij} \) for each cell in the \( n \times n \) matrix:
  - \( x_{ij} \) represents the value in cell on row \( i \), column \( j \)
- Domains:
  - \( D_{ij} = \{1, \ldots, n\} \) for each \( ij \) such that the cell \( ij \) is empty
  - \( D_{ij} = \{v_{ij}\} \) for each \( ij \) with a pre-filled value \( v_{ij} \)
- Constraints:
  - For each row \( i \): \( \text{AllDiff}(x_{i1}, \ldots, x_{in}) \)
  - For each column \( j \): \( \text{AllDiff}(x_{1j}, \ldots, x_{nj}) \)
An important special case of CSPs is the *Boolean satisfiability problem* SAT.

In general, SAT the question of whether a given propositional logic formula is satisfiable.

Typically SAT refers to CNF SAT.

- The satisfiability problem of propositional (Boolean) formulas in *conjunctive normal form*, CNF formulas.

Despite its simplicity, SAT is an often used constraint language that provides a highly efficient approach to solving various hard computational problems.
A literal is a Boolean variable \( x \), or the negation \( \neg x \) of \( x \)

- \( \neg x \) is the negative literal of \( x \), \( x \) the positive literal

A clause is the constraint \( \bigvee_{i=1}^{n} l_i \) (i.e., \( l_1 \vee \cdots \vee l_k \)) over distinct literals \( l_i \)

- \( \vee \) is called disjunction, i.e., logical OR

A CNF formula is a set of clauses

- In other words:
  - a CNF formula is a constraint of the form \( \bigwedge_{C \in \mathcal{C}} C \),
  - where each \( C \in \mathcal{C} \) is a clause

A value assignment \( T \) over Boolean variables is a truth assignment

\( T \) satisfies a literal \( l \) iff

- \( l \) is a positive literal \( x \) and \( T(x) = 1 \), or
- \( l \) is a negative literal \( \neg x \) and \( T(x) = 0 \)

\( T \) satisfies a clause \( C = l_1 \vee \cdots \vee l_k \) iff

- there is a literal \( l \in \{l_1, \ldots, l_k\} \) such that \( T(l) = 1 \).
A CNF encoding of the Quasigroup Completion problem:

- Boolean variables $x_{ijk}$, where $i, j, k = 1, \ldots, n$:
  $x_{ijk}$ means “cell at row $i$, column $j$ has value $k$”

- Use clauses to enforce that for each cell $ij$, exactly one of $x_{ij1}, \ldots, x_{ijn}$ is assigned to 1:
  - At least one: $(x_{ij1} \lor \cdots \lor x_{ijn})$
  - At most one: $(\neg x_{ijk} \lor \neg x_{ijk}') \forall i, j \in \{1, \ldots, n\}$, where $i \neq j$

- Similarly, enforce that
  - for each row $i$, exactly one of $x_{i1k}, \ldots, x_{ink}$ is assigned to 1 for each $k$
    (all cells in row $i$ have different values)
  - for each column $i$, exactly one of $x_{1jk}, \ldots, x_{njk}$ is assigned to 1 for each $k$
    (all cells in column $i$ have different values)
  - SAT encodings of AllDiff!
Constraint Optimization Problems (COPs)

- A CSP has some set $S$ of solutions (possible infinite).
- An *objective* (or *cost*) function $f$ is a mapping from $S$ to some set of values (can be reals, integers, etc).
- A *constraint optimization problem* (COP) consists of a set of constraints and a cost function.
- Each element in $S$ is a *feasible* solution.
- An *optimal solution*:
  - as a *minimization problem*:
    - any $s \in S$ such that $f(s') \geq f(s)$ for each $s' \in S$.
  - as a *maximization problem*:
    - any $s \in S$ such that $f(s') \leq f(s)$ for each $s' \in S$.
- Search task: find an optimal solution.
- Different paradigms:
  - Maximum Boolean satisfiability (MaxSAT): maximize the number of satisfied CNF clauses.
  - Integer/Linear programming (ILP, MIP).
  - Some CP solvers can also cope with optimization.
Linear and Integer Programs

- A *linear function* is of the form \( c_1 x_1 + \cdots + c_n x_n \).

- *Linear constraints* over variables \( x_1, \ldots, x_n \) are of the form

  \[
  a_1 x_1 + \cdots + a_n x_n \square b, \]

  where \( a_1, \ldots, a_n \) and \( b \) are constants, and \( \square \) is

  - \( = \) (linear equality/equation), or
  - \( \geq \) or \( \leq \) (linear inequality)

- A *linear program* (LP) is a COP such that
  - each constraint in the problem is linear,
  - the objective function in the problem is linear, and
  - the variable domains are real-valued ranges \([l_i, u_i]\), i.e., \( l_i \leq x_i \leq u_i \).
  - Solvable in polynomial-time.

- *Integer programs* (IPs) are like linear programs, except that
  *the variables can only take integer values*. Capture NP.

- *Mixed integer programs* (MIPs) have both integer and real-valued variables.
Integer Programming: Example

Knapsack problem:
- Given: A knapsack of size $S$ (an integer), items $1, \ldots, n$, and the size $s_i$ (integers) and value $v_i$ (integers) of each item $i$.
- Find a subset of the $n$ items that fits into the knapsack and maximizes the total value of the objects in the knapsack.

IP formulation:
- Take a binary variable $x_i$ for each item $i$.
  - $x_i = 1$ ($x_i = 0$) means that item $i$ is (not) included in the knapsack.

$$\max \sum_{i=1}^{n} v_i x_i$$

$$\sum_{i=1}^{n} s_i x_i \leq S$$

$x_i \in \{0, 1\} \ \forall i \in \{1, \ldots, n\}$

- The above formulation is a 0-1 integer program: all variables have binary domains.
Different paradigms

- Different constraint solving paradigms have different strengths and weaknesses
  - Deciding whether a given finite-domain CSPs, Boolean formula (SAT), or (M)IP has a solution is NP-hard
  - LPs can be solved in polynomial time
  - Tradeoffs between expressiveness (high-level constraints, CP) and fast solver techniques (low-level, SAT)
  - Satisfaction vs optimization: SAT vs MaxSAT, CP, MIP
  - Variable domains (binary, integer, real, ...)

- CP / SAT / MIP solvers are algorithmically different
Topics

- A: Data Mining using constraint solvers / knowledge compilation
- B: Using machine learning to configure / speed-up constraint solving
- C: Clustering with constraints
- D: Learning Bayesian networks using constraint solvers / heuristic search
- E: Learning causal models using constraint solving
- F: Model counting and probabilistic inference
- G: miscellaneous — ask me for further ones if necessary
Background: Propositional logic

Propositional formulas

- Syntax based on:
  Boolean variables $X = \{x_1, x_2, \ldots \}$
  Boolean connectives $\lor, \land, \neg$

- The set of (propositional) formulas is the smallest set such that all Boolean variables are formulas and if $\phi_1$ and $\phi_2$ are formulas, so are $\neg \phi_1$, $(\phi_1 \land \phi_2)$, and $(\phi_1 \lor \phi_2)$.
  For example, $((x_1 \lor x_2) \land \neg x_3)$ is a formula but $((x_1 \lor x_2) \neg x_3)$ is not.

- A formula of the form $x_i$ or $\neg x_i$ is called a literal where $x_i$ is a Boolean variable.

- Usual shorthands:
  $\phi_1 \rightarrow \phi_2 : \neg \phi_1 \lor \phi_2$
  $\phi_1 \leftrightarrow \phi_2 : (\neg \phi_1 \lor \phi_2) \land (\neg \phi_2 \lor \phi_1)$
  $\phi_1 \oplus \phi_2 : (\neg \phi_1 \land \phi_2) \lor (\phi_1 \land \neg \phi_2)$
Semantics

- Boolean variables are either true or false
- A truth assignment $T$ is mapping from a finite subset $X' \subset X$ to the set of truth values $\{1, 0\}$.
- Consider a truth assignment $T : X' \rightarrow \{1, 0\}$ which is appropriate to $\phi$, i.e., $X(\phi) \subseteq X'$ where $X(\phi)$ be the set of Boolean variables appearing in $\phi$.
- $T \models \phi$ (T satisfies $\phi$) is defined inductively as follows:
  - If $\phi$ is a variable, then $T \models \phi$ iff $T(\phi) = 1$.
  - If $\phi = \neg \phi_1$, then $T \models \phi$ iff $T \not\models \phi_1$
  - If $\phi = \phi_1 \land \phi_2$, then $T \models \phi$ iff $T \models \phi_1$ and $T \models \phi_2$
  - If $\phi = \phi_1 \lor \phi_2$, then $T \models \phi$ iff $T \models \phi_1$ or $T \models \phi_2$

Example. Let $T(x_1) = 1$, $T(x_2) = 0$. Then $T \models x_1 \lor x_2$, and $T \not\models (x_1 \lor \neg x_2) \land (\neg x_1 \land x_2)$