Lecture 7: Constraint satisfaction
Linear and integer programming

- Constraint satisfaction
  - Global constraints
  - Local search
  - Tools for SAT and CSP
- Linear and integer programming
  - Introduction

Global Constraints

- Constraint programming systems often offer constraints with special purpose constraint propagation (filtering) algorithms. Such a constraint can typically be seen as an encapsulation of a set of simpler constraints and is called a global constraint.
- A representative example is the alldiff constraint:

$$\mathrm{alldiff}(x_1,\ldots,x_n) = \{(d_1,\ldots,d_n) \mid d_i \neq d_j, \text{for } i \neq j\}$$

  Example. A tuple (a,b,c) satisfies $\mathrm{alldiff}(x_1,x_2,x_3)$ but (a,b,a) does not.
- $\mathrm{alldiff}(x_1,\ldots,x_n)$ can be seen as an encapsulation of a set of binary constraints $x_i \neq x_j$, $1 \leq i < j \leq n$.

Global Constraints: alldiff

- Global constraints enable compact encodings of problems.
- **Example.** N Queens
  Problem: Place n queens on a $n \times n$ chess board so that they do not attack each other.
  - Variables: $x_1,\ldots,x_n$ (x_i gives the position of the queen on ith column)
  - Domains: [1..n]
  - Constraints: for $i \in [1..n-1]$ and $j \in [i+1..n]$:
    (i) $\mathrm{alldiff}(x_1,\ldots,x_n)$ (rows)
    (ii) $x_i - x_j \neq i - j$ (SW-NE diagonals)
    (iii) $x_i - x_j \neq j - i$ (NW-SE diagonals)
- In addition to compactness global constraints often provide more powerful propagation than the same condition expressed as the set of corresponding simpler constraints.

Global Constraints: Propagation

- Consider the case of alldiff:
  For $\mathrm{alldiff}(x_1,\ldots,x_n)$ there is an efficient hyper-arc consistency algorithm which allows more powerful propagation than hyper-arc consistency for the set of corresponding "$\neq$" constraints.
- **Example.**
  - Consider variables x_1,x_2,x_3 with domains $D_1 = \{a,b,c\}, D_2 = \{a,b\}, D_3 = \{a,b\}$.
  - Now $\mathrm{alldiff}(x_1,x_2,x_3)$ is not hyper-arc consistent and the projection rule removes values a,b from the domain of x_1.
  - However, the corresponding set of constraints $x_1 \neq x_2, x_1 \neq x_3, x_2 \neq x_3$ is hyper-arc consistent and the projection rule is not able to remove any values.

Global Constraints: Other Examples

- When solving a CSP problem often a special purpose (global) constraint and an efficient propagation algorithm for it needs to be developed to make the solution technique more efficient.
- There is a wide range of such global constraints:
  - cumulative
  - diff-n
  - cycle
  - sort
  - alldifferent and permutation
  - symmetric alldifferent
  - global cardinality (with cost)
  - sequence
  - stretch
  - minimum global distance
  - k-diff
  - number of distinct values
  - ...

CSP: Local Search

- GSAT and WalkSAT type of local search algorithms (see Lecture 4) can be generalized to CSPs.
- As an example we consider Min Conflict Heuristic (MCH) algorithm (Minton et al, 1990):
  Given a CSP instance *P*
  - Initialize each variable by selecting a value uniformly at random from its domain.
  - In each local step select a variable x_i uniformly at random from from the conflict set, which is the set of variables appearing in a constraint that is unsatisfied under the current assignment.
  - A new value v for x_i is selected from the domain of x_i such that by assigning v to x_i the number of conflicting constraints is minimized.
  - If there is more than one value with that property, one of the minimizing values is chosen uniformly at random.
- One can add also a random walk step like in NoisyGSAT (WMCH algorithm; Wallace and Freuder, 1995).

CSP: Local Search

- A tabu search algorithm by Galiner and Hao is one of the best performing general local search algorithms for CSPs.
- TS-GH algorithm (Galiner and Hao, 1997):
  - Initialize each variable by selecting a value uniformly at random from its domain.
  - In each local step select among all variable-value pairs (x', v') such that x' appears in a constraint that is unsatisfied under the current assignment and v' is in the domain of v', a pair (x, v) that leads to a maximal decrease in the number of violated constraints.
  - If there multiple such pairs, one of them is chosen uniformly at random.
  - After changing the assignment of x to v, the pair (x, v) is declared tabu for *tt* steps.
- For competitive performance, the evaluation function for variable-value pairs needs to be implemented using caching and incremental updating techniques.

SAT: Local Search

- Local search methods have difficulties with structured problem instances.
- For good performance parameter tuning is essential.
  (For example in WalkSAT: the noise parameter p and the max_flips parameter.)
- Finding good parameter values is a non-trivial problem which typically requires substantial experimentation and experience.
- WalkSAT revised: adding greediness and adaptivity
  $\Longrightarrow$ Novelty+ and AdaptiveNovelty+ algorithms

WalkSAT

function WalkSAT(F,p):
  $t \leftarrow$ initial truth assignment;
  while flips $<$ max_flips **do**
    if t satisfies F **then return** t **else**
      choose a random unsatisfied clause C in F;
      if some variables in C can be flipped without
        breaking any presently satisfied clauses,
        then pick one such variable x at random; else:
      with probability p, pick a variable x in C unif. at random;
      with probability $(1-p)$, do basic GSAT move:
        find a variable x in C whose flipping causes
        largest decrease in $c(t)$;
      $t \leftarrow (t$ with variable x flipped$)$
  end while;
  return t.

Novelty+

- WalkSAT can be made greedier using a history-based variable selection mechanism.
- The age of a variable is the number of local search steps since the variable was last flipped.
- Novelty algorithm (McAllester et al., 1997):
  After choosing an unsatisfiable clause the variable to be flipped is selected as follows:
  - If the variable with the highest score does not have minimal age among the variables within the same clause, it is always selected.
  - Else it is only selected with probability $1-p$, where p is a parameter called noise setting.
  - Otherwise the variable with the next lower score is selected.
  - When sorting variables according to their scores, ties are broken according to decreasing age.
- In Novelty+ (Hoos 1998) a random walk step is added: with probability $1-wp$ the variable to be flipped is selected according to the Novelty mechanism and in the other cases a random walk step is performed.

Adaptive WalkSat and Adaptive Novelty+

- A suitable value for the noise parameter p is crucial for competitive performance of WalkSAT and its variants.
- Too low noise settings lead to stagnation behaviour and too high settings to long running times.
- Instead of a static setting, a dynamically changing noise setting can be used.
- Adaptive WalkSat and Novelty+ (Hoos, 2002):
  Two parameters θ and ϕ are given.
  - At the beginning the search is maximally greedy ($p=0$).
  - There is a search stagnation if no improvement in the evaluation function value has been observed over the last $m\theta$ search steps where m is the number of clauses in the instance.
  - In this situation the noise value is increased by $p := p + (1-p)\phi$ and if after this the search stagnation continues, p is further increased in the same way.
  - If there is an improvement in the evaluation function value, then the noise value is decreased by $p := p - p\phi/2$.

Tools for SAT

- The development of SAT solvers is strongly driven by SAT competitions (`http://www.satcompetition.org/`)
- There is a wide range of efficient solvers also available in public domain.
- See for example `http://www.satcompetition.org/` for solvers that ranked well in previous SAT competitions.
  SAT2005:

  `SatELiteGTI, MiniSAT 1.13, zChaff_rand, HaifaSAT, Vallst, March_dl, kcnf-2004, Dew_Satz1a, Jerusat 1.31 B, Hsat1, ranov, g2wsat, VW`

Tools for CSP

- Constraint programming systems offer a rich set of supported constraint types with efficient propagation algorithms and primitives for implementing search.
- Typically the user needs to program, for example, the search algorithm, splitting technique, and heuristic.
- See, for example, `http://4c.ucc.ie/~tw/csplib/links.html` for available constraint solvers:

  ```
  CLAIRE, ECLiPse, GNU Prolog, Oz,
  Sicstus Prolog, ILOG Solver, ...
  ```


Linear and Integer Programming

- Linear and Integer Programming can be thought to be a subclass of constraint programming where there are
  - two types of variables: real valued and integer valued
  - only one type of constraint: linear (in)equalities.
- Linear Programming (LP): only real valued variables.
- Integer Programming (IP): only integer variables.
- Mixed Integer Programming (MIP): both integer and real valued variables.

Linear and Integer Programming

- Computationally there is a fundamental difference between LP and IP:
  LP problems can be solved efficiently (in polynomial time) but
  IP problems are NP-complete (and all known algorithms have an exponential worst-case running time).
- MIP offers an attractive framework for solving (search and) optimization problems:
  - Continuous variables can be handled efficiently along with discrete variables.
  - Powerful LP solution techniques can be exploited in the IP case through linear relaxation.
  - Bounds on deviation from optimality can be generated even when optimal solutions are not proven.

MIP: Basic Concepts

- Let x be a vector of variables $x = (x_1, \ldots, x_n)$.
- Each variable in a set I of variables is required to take integer values while the remaining variables can take any real value.
  Each variable can have a range represented by vectors $l = (l_1, \ldots, l_n)$ and $u = (u_1, \ldots, u_n)$ such that for all i, $l_i \leq x_i \leq u_i$, that is, $l \leq x \leq u$.
- A linear constraint on the variables is of the form

  $$\Sigma_j a_j x_j = b \qquad \text{or} \qquad ax = b$$

  where a is a vector coefficients $a = (a_1, \ldots, a_n)$ and b is a scalar. The relation symbol '$=$' can also be '$\leq$' or '$\geq$'.
- A linear objective function is represented by a vector of coefficients $c = (c_1, \ldots, c_n)$ with the objective of minimizing (or maximizing) $\Sigma_j c_j x_j = cx$.

MIP: Basic Concepts

- A (mixed) integer program consists of a single linear objective and a set of constraints.
- If we create a matrix $A = (a_{ij})$ where a_{ij} is the coefficient for variable j in the ith constraint, then a MIP can be written as:

$$\begin{aligned} & \min cx \\ s.t. \quad & Ax = b \\ & l \leq x \leq u \\ & x_j \text{ is integer for all } j \in I \end{aligned}$$

An Example. SET COVER

INSTANCE: A family of sets $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U.
QUESTION: Find an l-cover of U (a set of l sets from F whose union is U) with the smallest number l of sets.

- For each set S_i an integer variable x_i such that $0 \leq x_i \leq 1$
- For each element u of U a constraint

$$a_1x_1 + \cdots + a_nx_n \geq 1$$

  where the coefficient $a_i = 1$ if $u \in S_i$ and otherwise $a_i = 0$.
- Objective: $\min x_1 + \cdots + x_n$

MIP: Basic Concepts

- A feasible solution to a MIP is an assignment of values to the variables in the problem such that the assignment satisfies all the linear constraints and range constraints and for each variable in I it assigns an integer value.
- A program is feasible if it has a feasible solution otherwise it is said to be infeasible.
- An optimal solution is a feasible solution that gives the minimal value of the objective function among all feasible solutions.
- A program is unbounded (from below) if for all $\lambda \in R$ there is a feasible solution for which the value of the objective function is at most λ.

Modelling: Logical Constraints

- Use binary integer variables ($0 \leq x \leq 1$).
- Disjunction: x_3 has the value of the boolean expression $x_1 \vee x_2$:

$$\begin{aligned} & x_3 \geq x1 \\ & x_3 \geq x2 \\ & x_3 \leq x_1 + x_2 \end{aligned}$$

- Conjunction: x_3 has the value of the boolean expression $x_1 \wedge x_2$:

$$\begin{aligned} & x_3 \leq x1 \\ & x_3 \leq x2 \\ & x_3 \geq x_1 + x_2 - 1 \end{aligned}$$

Modelling SAT

- Given a SAT instance F in CNF, introduce
- for each Boolean variable in F, a binary integer variable x ($0 \leq x \leq 1$).
- for each clause $l_i \vee \cdots \vee l_n$ in F, a constraint

$$a_1 x_1 + \cdots + a_n x_n \geq 1 - m$$

  where the coefficient $a_i = 1$ if the literal l_i is positive and otherwise $a_i = -1$ and m is the number of negative literals in the clause.
- Then F is satisfiable iff the corresponding set of constraints has a feasible solution.