Algorithms for Computational Logic

Introduction

Emmanuel Hebrard (adapted from João Marques Silva, Inês Lynce and Vasco Manquinho)

Outline

1. Extensions
Motivation

**Facility Location Problem**

Suppose that a company has to decide where to install new factories from $n$ potential locations in order to be able to serve $m$ clients.

Let $c_i$ denote the cost for opening a factory at location $i$ and let $d_{ij}$ denote the cost of serving client $j$ from location $i$.

Provide a formulation that helps the administration to decide where to open the factories such that the overall costs (factory open and serving clients) are minimized.
Motivation

Facility Location Problem

- Problem variables
  - $x_i$: denotes if a factory is to be open at location $i$
  - $y_{ij}$: denotes if client $j$ is served from location $i$

$$\text{Minimize} \quad \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} y_{ij}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} y_{ij} = 1 \quad \forall j \in \{1 \ldots m\}$$
$$x_i - y_{ij} \geq 0 \quad \forall i \in \{1 \ldots n\}, j \in \{1 \ldots m\}$$
$$x_i \in \{0, 1\}, y_{ij} \in \{0, 1\}$$

Pseudo-Boolean Optimization (PBO)

Formulation

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \begin{cases} \geq, =, \leq \end{cases} b_i$$
$$x_j \in \{0, 1\} \quad \forall j \in \{1, 2, \ldots, n\}$$

- 0-1 Integer Linear Programming (0-1 ILP)
If we identify \{false, true\} to \{0, 1\}, a clause \((x \lor y \lor z)\) is equivalent to \(x + y + z \geq 1\)

\((x \lor \overline{y} \lor z)\) is \(x + (1 - y) + z \geq 1\)

Not quite Integer Programming because the domain is Boolean

Particular case

Algorithmic Solutions

- Integer Programming solvers are very powerful
  - We are not going to discuss Integer Programming
- When there is a linear objective, MaxSAT can be a good approach (we will see MaxSAT)
- In some case, a CDCL-like algorithm can be better than IP
  - Replace clauses by cutting planes
Cutting Planes

Combination of two constraints

\[\delta \left( \sum_{j=1}^{n} a_j x_j \leq b \right)\]
\[\delta' \left( \sum_{j=1}^{n} a'_j x_j \leq b' \right)\]
\[\delta \sum_{j=1}^{n} a_j x_j + \delta' \sum_{j=1}^{n} a'_j x_j \leq \delta b + \delta' b'\]

Rounding can also be applied

\[\sum_{j=1}^{n} a_j x_j \leq b\]
\[\sum_{j=1}^{n} |a_j| x_j \leq |b|\]

- The correctness of the rounding operation follows from \(|x| + |y| \leq |x + y|\)
- Hence, \(\delta\) coefficients in cutting plane operations do not need to be integer. Rounding can be safely applied afterwards
Cutting Planes

### Rounding Example

\[ 0.5(3x_1 + 2x_2 + x_3 + 2x_4 + x_5 \leq 5) \]
\[ \frac{1.5x_1 + x_2 + 0.5x_3 + x_4 + 0.5x_5 \leq 2.5}{} \]

After rounding: \( x_1 + x_2 + x_4 \leq 2 \)

---

Cutting Planes generalize (p-simulate) CNF clause resolution

#### Example

\( (\overline{x}_1 \lor x_2 \lor x_3) \) \n\( (x_2 \lor x_4 \lor \overline{x}_3) \) \n\( \overline{x}_1 \lor x_2 \lor x_4 \) \n
\( (1 - x_1) + x_2 + x_3 \geq 1 \)
\( x_2 + x_4 + (1 - x_3) \geq 1 \)
\( (1 - x_1) \geq 0 \)
\( x_4 \geq 0 \)

\( 2(1 - x_1) + 2x_2 + 2x_4 \geq 1 \) addition
\( (1 - x_1) + x_2 + x_4 \geq 1 \) division
- Cutting planes is a stronger proof system than resolution

Use of Cutting Planes

- Used in branch and bound algorithms for PBO
  - And in the more general case of Integer Linear Programming (ILP)
- Very common at preprocessing (i.e., at the root node of the search tree)
- Algorithms that use cutting plane techniques during the search process are also known as branch and cut algorithms
- Other types of cutting planes exist (e.g., clique cuts)
Backtrack search with Cutting Plane learning

- DPLL-like algorithms for PBO can perform cutting plane learning instead of clause learning
- Replace clause resolution with cutting planes in implication graph analysis
- Important note: It is not guaranteed that the new constraint will be assertive

Consider the following constraints:

\[
\begin{align*}
c_1 : & \quad 3x_1 + x_2 + x_7 - 2x_8 \leq 3 \\
c_2 : & \quad -3x_1 + x_3 + 2x_7 + x_9 \leq 0 \\
c_3 : & \quad -x_2 - x_3 + x_6 \leq -1 - 1
\end{align*}
\]

- Suppose you start with assignment \(x_8 = 0\) at first decision level
- Next, you decide to assign \(x_6 = 1\). What happens?
  - Constraint propagation on \(c_3\) sets \(x_2 = 1, x_3 = 1\)
  - Constraint propagation on \(c_2\) sets \(x_1 = 1\)
  - Constraint \(c_1\) is violated
Backtrack search with Cutting Plane learning

- Conflict in constraint $c_1$
- Start backward traversal of graph

Cutting plane between $c_1$ and $c_2$ to remove $x_1$

\[
\begin{align*}
1(3x_1 + x_2 + x_7 - 2x_8 & \leq 3) \\
1(-3x_1 + x_3 + 2x_7 + x_9 & \leq 0) \\
x_2 + x_3 + 3x_7 - 2x_8 + x_9 & \leq 3
\end{align*}
\]

Cutting plane with $c_3$ to remove $x_3$

\[
\begin{align*}
1(x_2 + x_3 + 3x_7 - 2x_8 + x_9 & \leq 3) \\
1(-x_2 + x_3 + x_6 & \leq -1) \\
x_6 + 3x_7 - 2x_8 + x_9 & \leq 2
\end{align*}
\]

- Backward traversal to the decision variable $x_6$
- Learned constraint:

\[
x_6 + 3x_7 - 2x_8 + x_9 \leq 2
\]

- Backtrack to level 1 and assert $x_7 = 0$