Enumeration of Circuits and Minimal Forbidden Sets

Frederik Stork
ILOG GmbH, Germany

Marc Uetz
Universiteit Maastricht, Netherlands
Independence Systems

Given ground set $V$, $|V| = n$, then $\mathcal{I} \subseteq 2^V$ is independence system if

$$W \subseteq U \in \mathcal{I} \Rightarrow W \in \mathcal{I}$$

**Bases $\mathcal{B}$:** $B \in \mathcal{I}$ ⊆-maximal

**Circuits $\mathcal{C}$:** $C \not\in \mathcal{I}$ ⊆-minimal

*Example.* $\mathcal{I} =$ forests in a graph: bases = spanning trees circuits = simple cycles

Given membership oracle for $\mathcal{I}$, we want to enumerate all circuits of $\mathcal{I}$. 
Complexity of Enumeration Problems

Given $V$, and an implicit, say poly($n$) description of $C \subseteq 2^V$.

- **Total polynomial time:** Enumeration of $C$ in time poly($n, |C|$)

- **Incremental polynomial time:** Given $\mathcal{X} \subseteq C$, either compute $U \in C \setminus \mathcal{X}$ or decide $\mathcal{X} = C$ in time poly($n, |\mathcal{X}|$)

[Valiant’79, Johnson, Yannakakis, Papadimitriou’88]
Maximal Independent Set Enumeration is Hard

**Theorem.** Unless P=NP, no algorithm exists that enumerates the bases $B$ of any independence system $I$ in total polynomial time.

[Lawler, Lenstra, Rinnooy Kan ’80]

*Proof.* Reduction from **Satisfiability.**
Likewise... 

**Theorem.** Unless P=NP, no algorithm exists that enumerates the circuits $C$ of any independence system $I$ in total polynomial time.

*Proof.* Given $I$, suppose $A$ is such an algorithm. Define $I^D$ by

$$W \in I^D : \iff V \setminus W \notin I.$$ 

$U$ max. indep. for $I^D \iff V \setminus U$ min. cover for $I$. Then $A$ enumerates max. indep. sets of any $I^D$.

**Question.** Particular independence systems?
Motivation: Resource Constrained Scheduling

\[ i, j \text{ jobs } \in V := \{1, \ldots, n\} \]
\[ \prec \text{ partial order on } V \] (precedence constraints)
\[ b_k \text{ availability of resource } k \]
\[ a_{kj} \text{ resource } k \text{ as required by job } j \]

Resource constraints: \( \sum_{j \in V(t)} a_{kj} \leq b_k \) for all \( k \) and \( t \)
Circuits $=\text{Minimal Forbidden Sets}$

**trivial:** \[ F = \{i, j\} \text{ with } i \prec j, \text{ or} \]

**non-trivial:**
- $F$ anti-chain for poset $(V, \prec)$
- \[ \sum_{j \in F} a_{kj} > b_k \text{ for some } k \]
- $F \subseteq \text{–minimal}$

**Required for:**
- policies in stochastic scheduling
- cutting planes for IP formulations
- $\ldots$
**Independence System Hypergraph**

- nodes $V =$ jobs
- hyperedges $C =$ min. forb. sets

jobs $U \subseteq V$ feasible $\iff U$ independent set in $(V, C)$

**Problem:** List of all (non-trivial) minimal forbidden sets $C$ required

**But:** The (non-trivial) minimal forbidden sets $C$ not given explicitly
Given Explicitly: Linear System $Ax \leq b$

Minimally infeasible vectors $x \in \{0, 1\}^n = \text{minimal forbidden sets } F$
The Enumeration Problem

Assume $A \geq 0$, $b \geq 0$, integral.

<table>
<thead>
<tr>
<th>INPUT:</th>
<th>Linear inequality system $Ax \leq b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT:</td>
<td>All circuits: minimal infeasible ${0, 1}$-vectors $x$</td>
</tr>
</tbody>
</table>
Theorem. Unless P=NP, no total poly-time algorithm exists for the enumeration problem.

Proof. Take graph \( G = (V, E) \), decision problem \( \exists \text{ indep. set} \geq t \)? Define

\[
\begin{pmatrix}
k & i & j \\
\{k, j\} & \cdots & 1 \\
\{i, j\} & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n \\
\end{pmatrix}
\leq
\begin{pmatrix}
1 \\
1 \\
\vdots \\
1 \\
\end{pmatrix}
\begin{pmatrix}
t - 1 \\
\end{pmatrix}
\]

Assume total poly-time algorithm \( A \), with \( p(n, |C|) \). Let \( A \) run for \( p(n, |E|) + 1 \) time.

1. \( A \) outputs min. infeas. \( x \) with \( \geq t \) one's \( \rightarrow G \) ‘Yes’

2. \( A \) terminates w/o min. infeas. \( x \) with \( \geq t \) one's \( \rightarrow G \) ‘No’

3. \( A \) outputs no min. infeas. \( x \) with \( \geq t \) one's, and doesn't terminate \( \rightarrow G \) ‘Yes’

\[ \cdots \text{hence } A \text{ decides NP-hard IN-DEP.SET} \]
Poly-Time Enumeration for the Dual is easier

The ‘Dual’:

**INPUT:** Linear inequality system $Ax \leq b$

**OUTPUT:** All maximal feasible $\{0, 1\}$–vectors $x$

**Theorem.** There is an incremental (quasi-)poly-time algorithm [order $t^{o(\log t)}$]

[Boros, Elbassoni, Gurvich, Khachiyan, Makino '02]

**Proof idea.** $\# \text{min. infeas. } x \leq nm \# \text{max. feas. } x$, so generate both sets, solved by reduction to (generalized) hypergraph dualization [Fredman & Khachiyan '96]

**Note.** Such an algorithm is not likely to exist to for min. infeasible $x$ for $Ax \leq b$. 
The Counting Problem

**Input:** Linear inequality system $Ax \leq b$

**Output:** # of min. infeasible $\{0, 1\}$–vectors $x$
# Counting is Hard

**Theorem.** The counting problem is \#P-complete.

**Proof.** (\#P-hardness) Given a poset \((V, \prec)\). Consider **MaxAC**: Compute maximum cardinality anti-chain

- **MaxAC** is \#P-complete
  
  \[ \text{[Provan & Ball '83]} \]

- max. cardinality \( t \) in \( \text{poly}(n) \)
  
  \[ \text{[Max-Flow reduction]} \]

Then:

\[
\text{# of min. infeasible } \{0, 1\}-\text{vectors } x
\]

\[
= |\prec| + \text{ # of MaxAC's}
\]
Implementation of Enumeration Algorithm

\[
\begin{pmatrix}
3 & 2 & 1 & 1 \\
1 & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{pmatrix}
\leq
\begin{pmatrix}
3 \\
1
\end{pmatrix}.
\]

\(C = \{\{1,2\},\{1,3\},\{1,4\},\{2,3,4\}\}\)

\(C = \text{leaves of tree } T(C)\) (several heuristics to fathom the tree)
Empirical: Comparison to Divide & Conquer

1. for any row $k$ of $Ax \leq b$, compute $C_k$ (incr. poly time)

2. store only $\subseteq$–minimal sets of $\bigcup C_k$

[Lawler, Lensta, Rinnooy Kan '80, Bartusch '84]

<table>
<thead>
<tr>
<th></th>
<th>$\emptyset$ CPU (sec.)</th>
<th>max. CPU (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Divide &amp; Conquer</td>
<td>2.19</td>
<td>102.0</td>
</tr>
<tr>
<td>Computation of $T(C)$</td>
<td>0.01</td>
<td>0.2</td>
</tr>
</tbody>
</table>

480 scheduling instances with 30 jobs and 4 resources
Analysis: Special cases

Scheduling application with one resource type

\[
\begin{pmatrix}
  k & i & j \\
  1 & \cdots & 1 \\
  1 & \cdots & 1 \\
  a_1 & \cdots & \geq a_n
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{pmatrix}
\leq
\begin{pmatrix}
  1 \\
  1 \\
  \cdots \\
  1
\end{pmatrix}
\]

Task: Enumerate all min. infeas. \( x \in \{0, 1\}^n \)

Theorem. Our enumeration algorithm can be implemented to do this in incremental polynomial time.
Analysis [contd.]

1) Any infeas. set in $T(C)$ is min. infeasible.
2) At any node, poly($n$) test decides if infeasible descendant exists (max. weight anti-chain!)
3) put all together...
Discussion

• **Surprising:** minimal infeasible \( x \) harder than maximal feasible \( x \) for \( Ax \leq b \)

• **Related:** (generalized) dualization of hypergraphs

  [e.g., Endre Boros et. al.]

• **Open:** \( \exists \) Poly-time enumeration algorithm for general scheduling problem?
Complexity of Counting Problems

Given $V$ and an implicit description of $C \subseteq 2^V$, $|C| = c$.

- **Membership in \#P**: $c$ can be determined by counting the accepting computations of a nondet. poly-time Turing machine.

- **\#P-hardness**: Computation of $c$ yields a solution for any problem in \#P (typically: parsimonious reduction).

- **\#P-complete problems**: Counting hamiltonian cycles or perfect matchings in a bipartite graph.