Emerging Trends in Optimization

Daniel Bienstock

Columbia University

I. Integer programming
II. Large-scale linear programming
What is (mixed-) integer programming?

\[ \min c^T x \]

\[ Ax + By \geq b, \; x \text{ integral} \]

Dramatic improvements over the last ten years

- More mature methodology: branch-and-cut (theory and implementation)
- Vastly better linear programming codes
- Better computing platforms

→ Can now solve problems once thought impossible

But ...
Network design problem

- We have to build a directed network where each node has small out- and in-degree
- In the network, we have to route given multicommodity demands
- We have to carry out both tasks at the same time, so that the maximum total flow on any edge is minimized

Input data:

- A list of traffic demands. Traffic demand $k$ specifies that $d^k$ units must be routed from node $s^k$ to node $t^k$; $k = 1, 2, \ldots, K$.
- The network has out-degrees and in-degrees at most $p$.

Mixed-integer programming formulation:

- For every pair of nodes $i, j$: $x_{ij}$, to indicate whether the logical network contains edge $(i, j)$
- For every demand $k$ and every pair of nodes $i, j$: $f^k_{ij}$, the amount of flow of commodity $k$ that is routed on edge $(i, j)$.
Complete formulation

\[ f_{ij}^k \leq d^k x_{ij}, \text{ for all } k \text{ and } (i, j) \]

Route all traffic (Flow conservation)

\[ \sum_{j \neq s} f_{skj}^k = d^k, \text{ for all } k \]
\[ \sum_{j \neq i} f_{ij}^k = 0, \text{ for all } k \text{ and } i, j \neq s, t^k \]

Degree constraint

\[ \sum_{j \neq i} x_{ij} \leq p, \text{ for all } i \]
\[ \sum_{j \neq i} x_{ji} \leq p, \text{ for all } i \]

Congestion

\[ \sum_k f_{ij}^k - \lambda \leq 0, \text{ for all } (i, j) \]

Objective: Minimize \( \lambda \)

(\( f, x \geq 0 \))
Instance **danoint** in the **MIPLIB**

**1992**

- Linear programming relaxation has value $\sim 12.0$
- Strong formulation has value (lower bound) $\sim 60.0$
  (Two minutes CPU on SUN Sparc 2)
- Branch-and-bound (Cplex 3.0) improves lower bound to $\sim 63.0$;
  best upper bound has value $\sim 65.6$

**2002**

- Cplex 8.0 solves **danoint** in one day of CPU time on 2.0 GHz P4 while enumerating approximately two million branch-and-bound nodes.

**Progress?**
Speedup is due to:

1. Vastly faster computers

2. Much faster linear programming solver (esp. Cplex)

3. Faster integer programming solver (Branch-and-cut)

A thought experiment:

Use the 2002 branch-and-cut module together with the 1992 LP solver + machine

Why does this matter?

→ The network in `danoint` has 8 nodes: `danoint` has 56 0-1 variables and ~ 200 continuous variables.

So?

dano3mip is `danoint`'s big brother.

• 24 node network; ~ 550 0-1 variables and ~ 13500 continuous variables.

• **1992:** strong formulation gives error bound of ~ 25% which cannot be improved by branch-and-bound. Problem considered unsolvable.

• **2003:** Cplex 8.0 cannot improve 1992 bound after one week. `dano3mip` is now the only unsolved problem in MIPLIB.
Starting point

Balas, Pulleyblank, Barahona, others (pre 1990).

A polyhedron $P \subseteq \mathbb{R}^n$ can be the **projection**

of a *simpler* polyhedron $Q \subseteq \mathbb{R}^N$ ($N > n$)

**More precisely:**

There exist polyhedra $P \subseteq \mathbb{R}^n$, such that

- $P$ has exponentially (in $n$) many facets, and
- $P$ is the projection of $Q \subseteq \mathbb{R}^N$, where
- $N$ is polynomial in $n$, and $Q$ has polynomially many facets.
Given $\mathcal{F} = \{ x \in \{0, 1\}^n : Ax \geq b \}$

**Question:** Given $x^* \in \mathbb{R}^n_+$, is $x^* \in \text{conv}(\mathcal{F})$?

**Idea:** Let $N \gg n$.

Consider a function (a “lifting”) that maps each

$$v \in \{0, 1\}^n \text{ into } \hat{z} = \hat{z}(v) \in \{0, 1\}^N$$

with $\hat{z}_i = v_i$, $1 \leq i \leq n$.

Let $\hat{\mathcal{F}}$ be the image of $\mathcal{F}$ under this operator.

**Question:** Can we find $y^* \in \text{conv}(\hat{\mathcal{F}})$, such that $y^*_i = x^*_i$, $1 \leq i \leq n$?
Concrete Idea

\( v \in \{0, 1\}^n \) mapped into \( \hat{v} \in \{0, 1\}^{2^n} \), where

(i) the entries of \( \hat{v} \) are indexed by subsets of \( \{1, 2, \ldots, n\} \), and

(ii) For \( S \subseteq \{1, \ldots, n\} \), \( \hat{v}_S = 1 \) iff \( v_j = 1 \) for all \( j \in S \).

**Example:** \( v = (1, 1, 1, 0)^T \) mapped to:

\[
\hat{v}_0 = 1, \quad \hat{v}_1 = 1, \quad \hat{v}_2 = 1, \quad \hat{v}_3 = 1, \quad \hat{v}_4 = 0,
\]

\[
\hat{v}_{\{1,2\}} = \hat{v}_{\{1,3\}} = \hat{v}_{\{2,3\}} = 1,
\]

\[
\hat{v}_{\{1,4\}} = \hat{v}_{\{2,4\}} = \hat{v}_{\{3,4\}} = 0,
\]

\[
\hat{v}_{\{1,2,3\}} = 1, \quad \hat{v}_{\{1,2,4\}} = \hat{v}_{\{1,3,4\}} = \hat{v}_{\{2,3,4\}} = 0,
\]

\[
\hat{v}_{\{1,2,3,4\}} = 0.
\]
Take \( v \in \{0, 1\}^n \). The \( 2^n \times 2^n \) matrix \( \hat{v} \hat{v}^t \) is symmetric, and its main diagonal = \( \emptyset \)-row.

Further, suppose \( x^* \in R^n \) satisfies

\[
x^* = \Sigma_i \lambda_i v_i
\]

where each \( v_i \in \{0, 1\}^n \), \( 0 \leq \lambda \), and \( \Sigma_i \lambda_i = 1 \).

Let \( W = W(x^*) = \Sigma_i \lambda_i \hat{v}_i \hat{v}_i^t \) and \( y = \Sigma_i \lambda_i \hat{v}_i \).

\- \( y_{\{j\}} = x^*_j \), for \( 1 \leq j \leq n \).
\- \( W \) is symmetric, \( W_{\emptyset,\emptyset} = 1 \), diagonal = \( \emptyset \)-column = \( y \).
\- \( W \succeq 0 \).
\- \( \forall p, q \subseteq \{1, 2, \cdots, n\}, W_{p, q} = y_{p \cup q} \)

So we can write \( W = W^y \).
\[ x^* = \sum \lambda_i v_i, \quad 0 \leq \lambda \text{ and } \sum \lambda_i = 1. \]

\[ y = \sum \lambda_i \hat{v}_i, \quad W^y = \sum \lambda_i \hat{v}_i \hat{v}_i^t, \quad \text{(cont’d)} \]

Assume each \( v_i \in \mathcal{F} \).

**Theorem**

Suppose \( \sum_{j=1}^n \alpha_j x_j \geq \alpha_0 \quad \forall x \in \mathcal{F} \).

Let \( p \subseteq \{1, 2, \cdots, n\} \).

Then:

\[ \sum_{j=1}^n \alpha_j W^y_{\{j\},p} - \alpha_0 W^y_{\emptyset,p} \geq 0 \]

e.g. the \( p \)-column of \( W \) satisfies

**every constraint valid for** \( \mathcal{F} \), **homogenized**.

→ just show that for every \( i \),

\[ \sum_{j=1}^n \alpha_j [\hat{v}_i \hat{v}_i^t]_{\{j\},p} - \alpha_0 [\hat{v}_i \hat{v}_i^t]_{\emptyset,p} \geq 0 \]

Also holds for the \( \emptyset \)-column minus the \( p^{th} \)-column.
Lovász-Schrijver Operator, for $\mathcal{F} = \{ x \in \{0, 1\}^n : Ax \geq b \}$

1. Form an $(n + 1) \times (n + 1)$-matrix $W$ of variables

2. **Constraint:** $W_{0,0} = 1$, $W$ symmetric, $W \succeq 0$.

3. **Constraint:** $0 \leq W_{i,j} \leq W_{0,j}$, for all $i, j$.

4. **Constraint:** The main diagonal of $W$ equals its 0-row.

5. **Constraint:** For every column $u$ of $W$,

   $$\sum_{h=1}^{n} a_{i,h} u_h - b_i u_0 \geq 0, \quad \forall \text{ row } i \text{ of } A$$

   and

   $$\sum_{h=1}^{n} a_{i,h} (W_{h,0} - u_h) - b_i (1 - u_0) \geq 0, \quad \forall \text{ row } i \text{ of } A$$

Let $C' = \{ x \in R^n : 0 \leq x \leq 1, Ax \geq b \}$

and $N_+(C') = \text{set of } x \in R^n$, such that

there exists $W$ satisfying 1-5, with $W_{j,0} = x_j$, $1 \leq j \leq n$.

**Theorem.** $C \supseteq N_+(C') \supseteq N^2_+(C') \supseteq \cdots \supseteq N^n_+(C') = \text{conv}(\mathcal{F})$.
Lovász-Schrijver revisited

$v \in \{0, 1\}^n$ lifted to $\hat{v} \in \{0, 1\}^{2^n}$, where

(i) the entries of $\hat{v}$ are indexed by subsets of $\{1, 2, \ldots, n\}$, and
(ii) For $S \subseteq \{1, \ldots, n\}$, $\hat{v}_S = 1$ iff $v_j = 1$ for all $j \in S$.

→ this approach makes statements about sets of variables that simultaneously equal 1

How about more complex logical statements?
Subset algebra lifting

For $1 \leq j \leq n$, let

$$Y_j = \{z \in \{0, 1\}^n : z_j = 1\}, \quad N_j = \{z \in \{0, 1\}^n : z_j = 0\}$$

Let $\mathcal{A}$ denote the set of all set-theoretic expressions involving the $Y_j$, the $N_j$, and $\emptyset$.

Note:
(i) $\mathcal{A}$ is isomorphic to the set of subsets of $\{0, 1\}^n$.
(ii) $|\mathcal{A}| = 2^{2^n}$
(iii) $\mathcal{A}$ is a lattice under $\supseteq$; containing an isomorphic copy of the lattice of subsets of $\{1, 2, \ldots, n\}$.

Lift $v \in \{0, 1\}^n$ to $\bar{v} \in \{0, 1\}^\mathcal{A}$

where for each $S \subseteq \{0, 1\}^n$, $\bar{v}_S = 1$ iff $v \in S$. 
Example

\[ v = (1, 1, 1, 0, 0) \in \{0, 1\}^5 \] is lifted to

\[ \tilde{v} \in \{0, 1\}^{2^{32}} \] which satisfies

\[ \tilde{v}[(Y_1 \cap Y_2) \cup Y_5] = 1 \]

\[ \tilde{v}[Y_3 \cap Y_4] = 0 \]

\[ \tilde{v}[Y_3 \cap (Y_4 \cup N_5)] = 1 \]

\[ \cdots \]

\[ \tilde{v}[S] = 1 \text{ iff } (1, 1, 1, 0, 0) \in S \]

**Note:** if \( v \in \mathcal{F} \) then \( \tilde{v}[\mathcal{F}] = 1 \).

→ Family of algorithms that generalize Lovász-Schrijver, Sherali-Adams, Lasserre
1. **Form a family of set-theoretic indices, \( \mathcal{V} \).**
These include, for \( 1 \leq j \leq n \):
- \( Y_j \), to represent \( \{ x \in \{0, 1\}^n : x_j = 1 \} \)
- \( N_j \), to represent \( \{ x \in \{0, 1\}^n : x_j = 0 \} \)

Also \( \emptyset , \mathcal{F} \).

2. **Impose all constraints known to be valid for \( \mathcal{F} \):**
e.g.
\[
x_1 + 4x_2 \geq 3 \text{ valid } \rightarrow X[Y_1] + 4X[Y_2] - 3 \geq 0
\]
Also set theoretic constraints, e.g. \( X[N_5] \geq X[Y_2 \cap N_5] \).

3. **Form a matrix** \( U \in \mathbb{R}^{\mathcal{V} \times \mathcal{V}} \) **of variables, with**
- \( U \) symmetric, main diagonal = \( \mathcal{F} \)-row = \( X \)
- For \( p, q \in \mathcal{V} \),
  \[
  U_{p,q} = X[p \cap q] \text{ if } p \cap q \in \mathcal{V}
  \]
  \[
  U_{p,q} = \text{a new variable, otherwise}
  \]
- All columns of \( U \) satisfies every constraint; optionally \( U \succeq 0 \).

How do we algorithmically choose small (polynomial-size) \( \mathcal{V} \)?
Example (level 2):

\[ x_1 + 5x_2 + x_3 + x_4 + x_5 - 2x_6 \geq 2 \quad \rightarrow \quad N_1 \cap N_2 \cap Y_6 \]

\[-x_2 + 2x_3 + x_4 + x_6 \leq 3 \quad \rightarrow \quad N_2 \cap Y_3 \cap Y_4 \cap Y_6 \]

\[ x_1 + x_2 + x_3 - x_4 \geq 1 \quad \rightarrow \quad N_1 \cap N_2 \cap N_3 \cap Y_4 \]

\[ \rightarrow \text{ construct } \omega = N_1 \cap N_2 \cap Y_4 \cap Y_6 \]

also \[ Y_1 \cap Y_2 \cap Y_4 \cap Y_6 \] (a negation of order 2 of \( \omega \))

and all other negations of order 2 (e.g. \( N_1 \cap N_2 \cap N_4 \cap N_6 \)).

also:

\[ \omega^{>2} = \bigcup_{t>2} \{ \omega' : \omega' \text{ is a negation of order } t \text{ of } \omega \} \]

In general, the \( w^{>r} \) expressions are unions (disjunctions) of exponentially many intersections.
Constraints

“Box” constraints:

\[ 0 \leq X, \quad X[F] = 1, \quad X[p] - X[F] \leq 0 \]

Also, say:

\[ \omega = N_1 \cap N_2 \cap Y_4 \cap Y_6. \]

Then e.g.

\[ X[N_1] - X[\omega] \geq 0. \]

Also,


Finally,

\[ X[\omega] + X[Y_1 \cap N_2 \cap Y_4 \cap Y_6] + X[N_1 \cap Y_2 \cap Y_4 \cap Y_6] + \]
\[ + X[N_1 \cap N_2 \cap N_4 \cap Y_6] + X[N_1 \cap N_2 \cap Y_4 \cap N_6] + \]
\[ + X[\omega^>1] = 1 \]

→ Implications for “matrix of variables”
Set covering problems

\[ \min \ c^T x \]

\[ \text{s.t. } x \in \mathcal{F} \]

\[ \mathcal{F} = \{ x \in \{0, 1\}^n : Ax \geq 1 \}, \quad A \text{ a 0–1-matrix.} \]

\[ \rightarrow \text{All nontrivial valid inequalities } \alpha^T x \geq \alpha_0 \text{ satisfy } \alpha \geq 0 \text{ and integral} \]

Theorem

For any integer \( k \geq 1 \), there is a polynomial-size relaxation guaranteed to satisfy all valid inequalities with coefficients in \( \{0, 1, \ldots, k\} \). ●

\[ \rightarrow k = 2 \text{ requires exponential time for Lovász-Schrijver} \]
Chvátal-Gomory cuts

Given constraints:
\[ \sum_j a_{ij} x_j \geq b_i, \quad 1 \leq i \leq m \]
and multipliers \( \pi_i \geq 0, \quad 1 \leq i \leq m \)

\( x \in \mathbb{Z}_+ \) implies that
\[ \sum_j \left\lceil \sum_i \pi_i a_{ij} \right\rceil x_j \geq \left\lceil \sum_i \pi_i b_i \right\rceil \]
is valid
→ a C-G rank 1 cut (ca. 1960)

Example:
\[
\begin{align*}
3x_1 & -2x_2 \quad \geq \quad 2 \times \frac{1}{2} \\
2x_1 & +4x_3 \quad \geq \quad 5 \times \frac{1}{3} \\
x_2 & +x_3 \quad \geq \quad 1 \times 1
\end{align*}
\]
get:
\[
\begin{align*}
\frac{13}{6}x_1 & + \frac{7}{3}x_3 \quad \geq \quad \frac{11}{3}
\end{align*}
\]
round-up:
\[ 3x_1 \quad + 3x_3 \quad \geq \quad 4 \]
the polyhedron defined by all rank-1 cuts is called the rank-1 closure (Cook, Kannan, Schrijver 1990)

Similarly, one can define the rank-2, rank-3, etc. closures ... the convex hull of 0-1 solutions always has finite rank

Recent results: the separation problem over the rank-1 closure of a problem is NP-hard (Eisenbrand, Fischetti-Caprara, Cornuejols-Li)
Chvátal-Gomory cuts and Set Covering Problems

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad x \in \mathcal{F}
\end{align*}
\]

\[\mathcal{F} = \{ x \in \{0, 1\}^n : Ax \geq 1 \}, \quad A \text{ a } 0-1\text{-matrix.}\]

Suppose \( r > 0 \) integral. Let:

- \( \mathcal{F}_r = \) rank-\( r \) closure of \( \mathcal{F} \),
- \( \tau_r = \min \{ c^T x : x \in \mathcal{F}_r \} \)

**Theorem**

For each \( r > 0 \) and \( 0 < \epsilon < 1 \), there is a polynomial-time relaxation with value at least

\[
(1 - \epsilon) \tau_r
\]
PROGRESS IN LARGE-SCALE LINEAR PROGRAMMING
LAST 10-20 YEARS

- THE ELLIPSOID METHOD - first provably good algorithm
- PROBABILISTIC ANALYSIS OF THE SIMPLEX METHOD - is it fast on "average" problems?
- KARMARKAR’S ALGORITHM - provably good interior point methods = logarithmic barrier methods
- STEEPEST-EDGE PIVOTING SCHEMES FOR SIMPLEX METHODS - fewer iterations
- BETTER USE OF SPARSITY IN SIMPLEX METHODS - faster pivots
- MORE EFFECTIVE CHOLESKY FACTORIZATION METHODS - faster, sparser
- IMPROVEMENTS IN NUMERICAL LINEAR ALGEBRA - improved numerical stability
- GREATLY EXPANDED TEST-BEDS WITH LARGER PROBLEMS OF DIFFERENT TYPES
- MUCH FASTER COMPUTERS WITH VASTLY LARGER STORAGE
- MUCH BETTER SOFTWARE DEVELOPMENT ENVIRONMENTS

→ LPs CAN BE SOLVED THOUSANDS OF TIMES FASTER THAN 10 YEARS AGO
ARE THE ALGORITHMS SCALABLE?

EXPERIMENTS USING CONCURRENT FLOW PROBLEMS

WHAT IS THE PROBLEM SIZE WE NEED TO HANDLE?

- TEN YEARS AGO: THOUSANDS TO (SOMETIMES) MILLIONS OF VARIABLES AND CONSTRAINTS
- NOW: TENS OR HUNDREDS OF MILLIONS
On large, difficult problems:

- **Interior Point Methods** – based on polynomial-time algorithms. In practice: “few” iterations, punishingly slow per iteration, could require gigantic amounts of memory

- **Simplex-like methods** – no really satisfactory theoretical basis. In practice: astronomically many iterations, could require a lot of memory

- at least **ten years** have elapsed since last major speedup
Prototypical example: **routing problems**

A special case: **the maximum concurrent flow problem**:

Given a network with multicommodity demands and with capacities, route a maximum common percentage of all the demands without exceeding capacities.
Approximation algorithms

→ Given a tolerance $0 < \epsilon < 1$, find a throughput of value at least $(1 - \epsilon)$ times the optimal

→ Do so with a fast algorithm with low memory requirements

- Shahrokhi and Matula (1989): running time at most $O(\epsilon^{-7})$ (times polynomial)
- Plotkin, Shmoys, Tardos (1990): $\epsilon^{-3}$
- Plotkin, Shmoys, Tardos; Grigoriadis and Khachiyan (1991): $\epsilon^{-2}$
- ⋯
- Young; Garg and Könemann; Fleischer (1999); better methods, still $\epsilon^{-2}$

★ Old heuristic popular among EE crowd: the “Flow Deviation Method”
Fratta, Gerla, Kleinrock (1971)

Theorem: The flow deviation method requires $O(\epsilon^{-2})$ iterations.
[Bienstock and Raskina, 2000]

But is it practical?
## Experiments

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Vars</th>
<th>FD Time (sec.)</th>
<th>Cplex Dual Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>484</td>
<td>1095606</td>
<td>746</td>
<td>20184</td>
</tr>
<tr>
<td>500</td>
<td>1019054</td>
<td>1024</td>
<td>33056</td>
</tr>
<tr>
<td>600</td>
<td>1569446</td>
<td>1303</td>
<td>68842</td>
</tr>
<tr>
<td>700</td>
<td>2131038</td>
<td>2103</td>
<td>227770</td>
</tr>
<tr>
<td>1000</td>
<td>4396604</td>
<td>5487</td>
<td>1734618</td>
</tr>
</tbody>
</table>

[FD using $\epsilon = 10^{-4}$]

☆ On-going work: parallel implementation to handle massively large cases (hundreds of millions of variables)
Basic Idea

- **Penalize** flows according to the degree that they utilize links
- **Reroute** flows according to penalties
- **Scale** flows (and throughput) whenever very slack utilization

Given a flow \( f \), \( \lambda(f) \) is the highest link utilization by \( f \).

Algorithm outline

**A1.** Let \( f \) be a strictly feasible flow, with throughput \( \theta \).

**B1.** Let \( g = \frac{1}{\lambda(f)} f \).

**C1.** Find a feasible flow \( h \) with throughput \( \frac{1}{\lambda(f)} \theta \) and \( \lambda(h) \) substantially smaller than \( \lambda(g) \).

**D1.** Reset \( f \leftarrow h \) and \( \theta \leftarrow \frac{1}{\lambda(f)} \theta \), and go to **B1**.

→ To accomplish **C1**, reduce \( \Sigma e u_e - f_e \)

→ This is done with a “gradient” method