Problems and applications in Combinatorial Optimization

Andrea Lodi

DEIS, Università di Bologna
http://www.or.deis.unibo.it
alodi@deis.unibo.it

Outline:

Lecture 1:
Solving Discrete Optimization Problems by Branch-and-Cut

Lecture 2:
The Traveling Salesman Problem: Basics, Applications and Variations

Lecture 3:
The Traveling Salesman Problem: Inequalities and Separation

Lecture 1:
Solving Discrete Optimization Problems by Branch-and-Cut

Outline:

1. Integer linear programs (ILPs)
2. Solving ILPs: classical methods
3. Valid inequalities and facets
4. The separation problem(s)
5. Strong cutting plane algorithms
6. Branch-and-cut
7. Conclusions

1.Integer linear programs (ILPs)

We will say that an optimization problem is *discrete* if the decision variables can only take a finite number of values.

The majority of discrete optimization problems can be formulated as *integer linear programs* (ILPS). These are problems of the form:

Minimise $\sum_{i=1}^{n} c_i \, x_i$

Subject to:

$$\sum_{i=1}^{n} a_{ij} \, x_i \leq b_j \qquad (j = 1, \ldots, m)$$

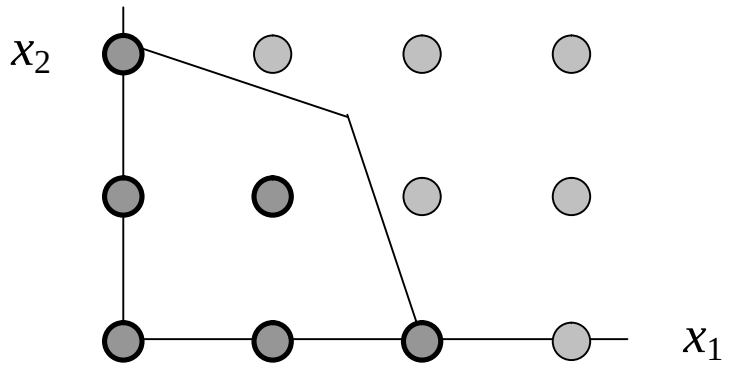
$x_i \geq 0$ and integer $\qquad (i = 1, \ldots, n).$

2. Solving ILPs: classical methods

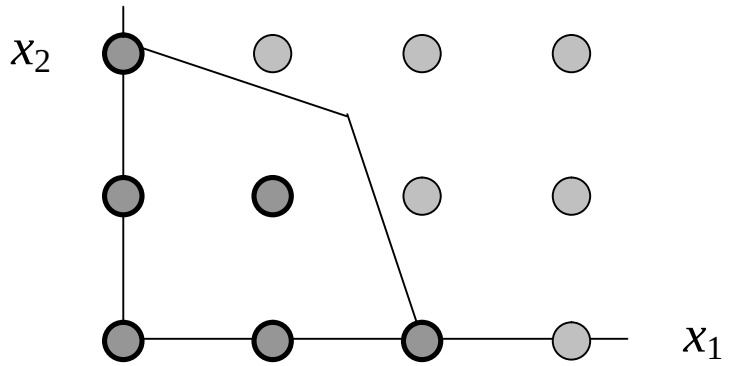
If we remove the integrality condition, we obtain a linear program, which is easy to solve.

In the late 1950s and early 1960s, two methods were proposed to find the *integer* solution.

i) ***cutting*:** adding extra linear inequalities.



ii) ***branching***: dividing into subproblems.



Gomory's cutting plane method

i) Solve the initial LP relaxation by a variant of the simplex method.

ii) Let x^* be the current solution vector. If x^* is integral, stop.

iii) Let x^*_k be fractional. There will be a row of the simplex tableau of the form:

$$x_k + \sum_{i \in NB} \alpha_i x_i = x^*_k.$$

Generate the cutting plane:

$$\sum_{i \in NB} f(\alpha_i) x_i \geq f(x^*_k).$$

iv) Add the cutting plane to the LP and re-optimize via a dual simplex pivot. Return to (ii).

(This was proven to be finitely convergent given a certain rule for choosing the fractional variable in step (iii).)

The problem with Gomory's cutting plane method is:

i) A very large number of cutting planes are typically needed.

ii) Numerical errors can lead to an incorrect solution or even cause the program to crash.

iii) No feasible solution to the problem is obtained until the very end.

The problem with branch-and-bound is:

i) A large number of sub-problems may need to be explored.

ii) If we do *bread-first* search, a huge amount of memory may be necessary.

iii) If we do *depth-first* search, we may spend a long time exploring a "dead-end".

3. Valid inequalities and facets

Why are Gomory's cutting planes weak?

What makes a cutting plane weak or strong?

We can make this more formal as follows.

The *linear programming relaxation* is:

$$\min \{cx: Ax \leq b, x \in \Re^n_+\}.$$

Its *feasible region* is:

$$P = \{x \in \Re^n_+: Ax \leq b\}.$$

The set of *integer solutions* is:

$$\{ x \in Z^n_+: Ax \leq b\}.$$

The *integral hull* is:

$$P_I = \text{conv}\{ x \in Z^n_+: Ax \leq b\}.$$

The strongest cutting planes are the ones defining *facets* of P_I.

Example 1: The perfect matching polytope

Let $G = (V, E)$ be an undirected graph with an even number of vertices.

A *perfect matching* is a set of edges which meets each vertex exactly once.

For each edge $e \in E$, let x_e be a binary variable taking the value 1 if e is in the matching, 0 otherwise. Then the vectors x representing feasible matchings satisfy:

$$\sum_{e \in \delta(i)} x_e = 1 \qquad (i \in V) \qquad \text{(degree equations)}$$

$$x \in \{0, 1\}^{|E|} \qquad \text{(binary condition)}$$

The integral hull is called the *perfect matching polytope*.

The perfect matching polytope is not full-dimensional, because of the degree equations.

(There are no other equations necessary.)

The non-negativity inequalities $x_e \geq 0$ induce facets.

In the 1960s, Jack Edmonds showed that a complete description of the perfect matching polytope is given by the degree equations, the non-negativity inequalities, and the following *odd-cut* inequalities:

$$\sum_{e \in \delta(S)} x_e \geq 1 \qquad (\forall S \subset V\text{: } |S| \text{ odd}).$$

Note that there are an exponential number of odd-cut inequalities.

But, as Edmonds showed, they do not all have to be generated explicitly…!

Example 2: The 0-1 knapsack polytope

A *0-1 knapsack polytope* is a polyhedron of the form:

$$\text{conv}\left\{ x \in \{0, 1\}^n : \sum_{i=1}^{n} a_i x_i \leq b \right\},$$

where $1 \leq a_i \leq b$ for $i = 1, \ldots, n$.

Since the 0-1 knapsack problem is *NP*-hard, we cannot hope to get a complete linear description for general n (unless NP = co-NP: see works by Papadimitriou).

Nevertheless, some useful polyhedral results are known.

E.g., if a set of items $C \subset \{1, \ldots, n\}$ is such that $\sum_{i \in C} a_i x_i > b$, then we have the so-called *cover inequality* $\sum_{i \in C} x_i \leq |C| - 1$.

These can be strengthened in various ways (Balas, Wolsey, Zemel, Savelsbergh…)

Example 3: The Symmetric TSP

Again, we are given a graph G = (V , E).

For each edge $e \in E$, let x_e be a binary variable taking the value 1 if e is in the tour, 0 otherwise. Then the vectors x representing tours satisfy:

$$\sum_{e \in \delta(i)} x_e = 2 \qquad (i \in V) \qquad \text{(degree equations)}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \qquad (S \subset V) \qquad \text{(SECs)}$$

$$x \in \{0, 1\}^{|E|} \qquad \text{(binary condition)}$$

The integral hull is called the *symmetric traveling salesman polytope*.

The STS polytope is not full-dimensional because of the degree equations.

(There are no other equations necessary.)

The trivial bounds $0 \leq x_e \leq 1$ induce facets.

The SECs induce facets. Note that there are an exponential number of them.

Lots of other facets are known:

> 2-matching, comb, clique-tree, hypohamiltonian, chain, ladder, crown, path, star, hyperstar, bipartition, binested, domino-parity… etc.

However, again, we cannot hope to get a complete description for general n unless NP = co-NP.

4. The Separation Problem(s)

In order to use strong valid inequalities in a practical cutting plane algorithm, we need to be able to detect when they are violated by the solution to a given LP relaxation.

That is, given a vector $x^* \in P \setminus P_I$, we would like to find an inequality (preferably facet-inducing) which *separates* x^* from P_I.

The problem of finding such violated inequalities is called the *separation problem* (Grötschel, Lovász & Schrijver, 1988).

Bad News: If the ILP is *NP*-hard then:

i) We do not know all the facets.
ii) Even the known valid inequalities / facets are usually exponential in number.
iii) Testing whether there is a cut separating x^* from P_I is co-*NP*-complete.

Good news: for some particular *classes* of inequalities, we have algorithms for detecting if an inequality *in that class* is violated.

This leads to the following concepts:

An *exact separation algorithm* for a given class of inequalities is a procedure which takes a vector $x^* \in P$ as input, and either outputs one or more inequalities in the class which are violated by x^*, or proves that none exists.

A *heuristic* separation algorithm is similar, except that it outputs either one or more inequalities in the class which are violated by x^*, or a failure message.

E.g. in the case of the TSP there are exact separation algorithms known for the SECs, the 2-matching ineq.s, and certain variants of the comb inequalities, though not the combs themselves.

There are also some fast and effective *heuristics* for various other classes.

5. Strong Cutting Plane Algorithms

In the 1970s and 1980s, the above ideas were used as follows:

Strong Cutting Plane Algorithm

i. Solve an initial LP relaxation.

ii. Let x^* be the solution. If it is integer and feasible, output the optimal solution and stop.

iii. Attempt to find some facet-inducing inequalities which are violated by x^*. If none are found, output the final lower bound and stop (or resort to branch-and-bound).

iv. Add the facet-inducing inequalities to the LP. Resolve the LP and go to (ii).

Using this general scheme, Miliotis, Padberg, Grötschel... were able to solve STSP instances with up to 600 vertices.

Crowder, Johnson & Padberg won an award for applying the scheme to general 0-1 ILPs.

6. Branch-and-cut algorithms

The next big advance was the realization that separation algorithms could be called at *every* node of the branch-and-bound tree.

A primitive version of this idea was already applied to the TSP by Hong (1972) and Miliotis (1976).

Grötschel, Jünger, Reinelt (1984) applied it to the Linear Ordering Problem.

However, a major landmark was the papers by Padberg & Rinaldi (1987, 1991). They devised the full paradigm which is now called *Branch-and-Cut*:

i) LP-based branch-and-bound plus…
ii) separation called at each node of tree
iii) cutting planes globally available via cut pool
iv) reduced-cost fixing
v) LP-based primal heuristics.

Branch-and-cut is *very* effective for the TSP: Applegate et al. solve real-life instances with over 10,000 nodes.

Also successful on:

- other variants of the TSP,
- facility location problems,
- lot-sizing problems
- bus, train, airline scheduling problems
- constrained tree problems,
- large sparse 0-1 integer programs...

But not so successful on tightly constrained *machine scheduling* or *vehicle routing* problems, nor on certain graph theory problems like *max-clique*, *chromatic number* etc.

Other methods which sometimes work better are:

- Branch-and-relax
- Branch-and-price
- Constraint Programming

7. Conclusions

Branch-and-cut provides a general framework for designing solution algorithms for hard discrete optimisation problems:

i) Formulate as an ILP
ii) Study associated polyhedra
iii) Look for separation algorithms
iv) Do computational experiments
v) Play with branching rules, heuristics, etc.

Software is available (CPLEX, XPRESS, ABACUS, etc.) to deal with the LP and branching parts.