Additional Cutting Planes?

Combining column generation with the cutting-plane approach allows us to solve LPs that are both, "long" and "wide".

Now, given an optimal solution of the LP 14.1, this is in general NOT the characteristic vector of a tour. What next?

- Either, we could stop with a (pretty good) lower bound and go on to branch-and-bound (see below),
- or, we could try to find some other classes of cutting planes to add, and continue with our cutting-plane algorithm.

Example: ...

Comb Inequalities

Definition 14.3.

A comb consists of a nonempty handle $H \subseteq V$, $H \neq V$ and $2k+1$ pairwise disjoint, nonempty teeth $T_1, T_2, \ldots, T_{k+1} \subseteq V$ for some $k \geq 1$ such that $T_i \cap H \neq \emptyset \neq T_i \cap (V \setminus H)$ for each $i = 1, \ldots, 2k+1$.

Theorem 14.4 (Chvátal'73, Grötschel & Padberg'79).

Let C be a comb with handle H and teeth $T_1, T_2, \ldots, T_{2k+1}$ for $k \geq 1$. Then the characteristic vector x of any tour satisfies

$$x(\gamma(H)) + \sum_{i=1}^{2k+1} x(\gamma(T_i)) \leq |H| + \sum_{i=1}^{2k+1} (|T_i| - 1) - (k+1).$$

Proof: ... □

Branch & Bound

Branch & Bound (B&B) is a technique that simulates a complete enumeration of all possible tours in $\mathcal{T}$ w/o considering them one by one.

We illustrate B&B on the example TSP, but the same principles carry over to almost every $\mathcal{NP}$-hard problem.

For many of these problems, B&B is the best known framework for optaining optimum solutions.

Basic B&B algorithm:

1. split the problem into smaller subproblems ("branching");
2. compute a (local) lower bound for each of these subproblems ("bounding");
3. apply B&B on all subproblem whose local lower bound is smaller than the best (global) upper bound found so far

Example: ...

The running time strongly depends on

- good lower bounding techniques,
- good heuristics to find upper bounds,
- good branching techniques.

Chapter 15: Matroids

(cp. Cook, Cunningham, Pulleyblank & Schrijver, Chapter 8)

Greedy Algorithm

Let us recall Kruskal's algorithm to find a forest of maximum weight in $G = (V, E)$.

Let $\mathcal{F} := \{F \subseteq E \mid (V, F) \text{ forest}\}$ denote the set of all forests in G.

Note: The following Greedy Algorithm finds a forest J of maximum weight for every weight function $c \in \mathbb{R}^E$.

Greedy Algorithm:

1. Set $J := \emptyset$;
2. While $\exists e \in E \setminus J$ with $c_e > 0$ and $J \cup \{e\} \in \mathcal{F}$
3. Choose such e with c_e maximum;
4. Replace J by $J \cup \{e\}$;

Q: Can we apply the same algorithm to other discrete structures $\mathcal{F} \subseteq 2^E$ (e.g. matchings)?

Example: ...

Independence Systems and Matroids

Definition 15.1.

Given a finite set E and $\mathcal{F} \subseteq 2^E$, system $(E, \mathcal{F})$ is called independence system if

(i) $\emptyset \in \mathcal{F}$, and

(ii) $X, Y \in \mathcal{F}$, $X \subseteq Y$ implies $X \in \mathcal{F}$.

A set $F \in \mathcal{F}$ is called independent, a set $G \in 2^E \setminus \mathcal{F}$ is called dependent. The minimal dependent sets are called circuits, a maximal independent set is called basis.

Maximization Problem for Independence Systems:
Given: Independence system $(E, \mathcal{F})$ and $c \in \mathbb{R}^E$.
Task: Find $F \in \mathcal{F}$ with $c(F) := \sum_{e \in F} c_e$ maximum.

Minimization Problem for Independence Systems:
Given: Independence system $(E, \mathcal{F})$ and $c \in \mathbb{R}^E$.
Task: Find a basis $B \in \mathcal{F}$ with $c(B)$ minimum.

Examples:...

Matroids

We assume that $\mathcal{F} \subseteq 2^E$ is given via some oracle (given $F \subseteq E$, is $F \in \mathcal{F}$?)

Definition 15.2.

An independence system $(E, \mathcal{F})$ is a matroid if

$$X \subseteq Y \in \mathcal{F},\ |X| > |Y| \quad \Longrightarrow \quad \exists e \in X \setminus Y \text{ with } Y \cup \{e\} \in \mathcal{F}.$$

Examples:...

Theorem 15.3 (Rado'57).

Let $(E, \mathcal{F})$ be an independence system. Then the Greedy Algorithm finds an optimal independent set for *every* $c \in \mathbb{R}^E$ if and only if $(E, \mathcal{F})$ is a matroid.

Proof:... □