Complexity and Multi-level Optimization

Ted Ralphs[1]

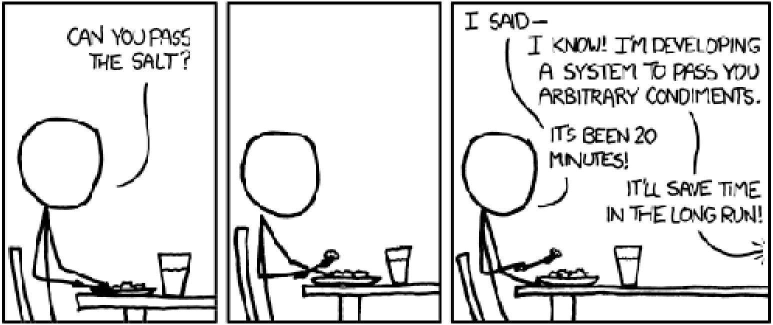
Joint work with:
Aykut Bulut[1], Scott DeNegre[2],
Andrea Lodi[4], Fabrizio Rossi[5], Stefano Smriglio[5], Gerhard Woeginger[6]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University
[2]Technomics, Inc. [4]DEIS, Universitá di Bologna
[5]Dipartimento di Informatica, Universitá di L'Aquila
[6]Department of Mathematics and Computer Science, Eindhoven University of Technology

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Outline

Motivation



What started it all: Proving something "obvious".

Motivation

- The framework traditionally used for complexity analysis of discrete optimization problems does not extend easily to multi-level optimization.
- "Difficult" optimization problems are typically characterized as being *NP*-hard, but this class is far too broad to be useful.
- In the traditional framework, optimization problems are converted into associated decision problems, which
  - results in a less refined classification scheme,
  - does not (directly) include the role of *solutions* and associated *values*, notions that are needed in many settings.
  - is difficult to do with multi-level optimization problems.
- Krentel (1988, 1992) suggested a framework for complexity based on the interpretation of problems as *functions*.
- This point of view is more natural for optimization.
- The point of view adopted here is largely similar to that proposed by Krentel, but there are substantial additions and deviations.

What This Talk is About

- This talk is about questions of complexity that are more general than those that can be asked in the framework traditionally used by discrete optimizers.
- The goal of the talk is to develop notions of complexity that
  - encompass multi-level and multi-stage optimization problems, and
  - are based on a more general framework of function evaluation that is better suited for optimization than the traditional set-based framework.
- We'll discuss two hierarchies that can be used to classify multi-level optimization problems.
  - The *polynomial time hierarchy* classifies multi-level decision problems.
  - The *min-max hierarchy* classifies multi-level optimization problems.
- We'll discuss the complexity of some special classes of optimization problems in light of this framework.
- We'll also re-interpret some well-known results in terms of this framework.
- Finally, we'll discuss the inherent multi-level nature of some optimization problems that arise in the implementation of branch and cut.

Outline

Basic Notions

- The formal complexity framework traditionally used in discrete optimization is for classifying *decision problems* (Garey and Johnson, 1979).
- The formal model of computation is a *deterministic Turing machine* (DTM).

  - A DTM specifies an *algorithm* computing the value of a Boolean function.
  - The DTM executes a program, reading the input from a *tape*.
  - We equate a given DTM with the program it executes.
  - The output is YES or NO.
  - A YES answer is returned if the machine reaches an *accepting state*.

- A problem is specified in the form of a *language*, defined to be the subset of the possible inputs over a given *alphabet* (Γ) that are expected to output YES.
- A DTM that produces the correct output for inputs w.r.t. a given language is said to *recognize the language*.
- Informally, we can then say that the DTM represents an "algorithm that solves the given problem correctly."

Non-deterministic Turing Machines

- A *non-deterministic Turing machine* (NDTM) can be thought of as a Turing machine with an infinite number of parallel processors.
- An NDTM follows all possible execution paths simultaneously.
- It returns YES if an accepting state is reached on *any* path.
- The running time of an NDTM is the *minimum* running time (length) of any execution paths that end in an accepting state.
- The running time is the minimum time required to verify that some path (given as input) leads to an accepting state.

Complexity Classes

- Languages can be grouped into *classes* based on the *best worst-case running time* of any TM that recognizes the language.

  - The class *P* is the set of all languages for which there exists a DTM that recognizes the language in time polynomial in the length of the input.
  - The class *NP* is the set of all languages for which there exists an NDTM that recognizes the language in time polynomial in the length of the input.
  - The class *coNP* is the set of languages whose complements are in *NP*.
  - As we will see, additional classes are formed hierarchically by the use of *oracles*.

- A language L_1 can be *reduced* to a language L_2 if there is an output-preserving polynomial transformation of members of L_1 to members of L_2.
- A language L is said to be *complete* for a class if all languages in the class can be reduced to L.
- This talk primarily addresses time complexity, though space complexity must ultimately also be considered.

Sets and Complexity

- The view of complexity just described is implicitly based on *solutions* and *sets*.
  - A solution (or *certificate*) can be thought of as a path that can be followed in a TM to reach an accepting state.
  - In many cases, we have a notion of solution that is indepdendent of a particular TM.
  - The YES answer means $\exists$ a solution, i.e., a path to an accepting state was found.
  - The NO answer means no solution was found, i.e., the final terminating state $\forall$ paths was a rejecting one.
- We can say, loosely, that problems in *NP* pose existentially quantified questions, whereas problems in *coNP* pose universally quantified questions.
- With any language (and perhaps a TM that recognizes it), we can associate a set of solutions.
  - The set of all possible solutions can be viewed as the *feasible set*, which we shall denote as $\text{feas}(l)$ for an input l.
  - A YES answer can be said to indicate an instance that is “feasible.”
  - A NO answer can be said to indicate “infeasible.”

Outline

Turing Functions

- The complexity framework based on decision problems, sets, and feasibility can be generalized to include *functions* and *optimization*.
- The functions here are not quite the same as mathematical functions.
- We use the term *Turing function* (TF) to refer to this type of "function."

  - A TF f is defined with respect to a given language L.
  - For $l \in L$, there is a (mathematical) function g_l (the *objective function*) that associates each $x \in \text{feas}(l)$ with a value $g_l(x)$.
  - The objective function may depend on the instance and may be encoded as part of the input.
  - Evaluating the TF involves both identifying a solution (if it exists) and computing its associated value.
  - The output of a TF (the solution) is generally not unique—we are allowed to choose any of the alternatives.

- In this framework, decision problems are TFs for which the objective is Boolean.

Metric Turing Machines and Classes of Functions

- A TF can be evaluated by a TM modified to output a numerical value.
- Krentel (1988) called such a TM a *metric Turing machine*, but we use the generic term "Turing machine" to refer to all variants.
- Solutions can be encoded into the single output value.
- Just as with languages, we can group functions into classes based on the best worst-case running time of a TM for evaluating them.
- We can also define notions of *reduction* and *completeness*.

Function Classes

- *FP* is the class of functions for which there exists a DTM that can evaluate the function in time polynomial in the length of the input.
- *FNP* is the class of functions for which there exists a NDTM that can evaluate the function in time polynomial in the length of the input.
- We denote by A^B class of functions that are in complexity class A if we are given an oracle for functions in class B.

Optimization Functions

- Let *MaxA* be the class of TFs for which the accepting states are associated only with solutions of maximum value w.r.t. an underlying TF in class *A*.
- Formally, we define the set *MaxA* of *optimization functions* by

$$f \in MaxA \Leftrightarrow f(l) = (x, g_l(x)) \ \forall l \in L,$$

where $x \in \mathrm{argmax}_{y \in \mathrm{feas}(l)} \, g_l(y)$ and L is a language in class A.
- We can similarly define *MinA* and *MidA* and $OptA = MaxA \cup MinA$.

Relationship of Turing Functions and Decision Problems

- From any TF f, we can construct an associated decision problem as follows.
  - We define the *hypograph* of a TF f as

$$\text{hypo}(f) := \{(l, k) \mid \exists x \in \text{feas}(l) \text{ s.t. } g_l(x) \geq k\}$$

  - This can be interpreted as a language specifying a decision problem.
  - This is the mapping we use to reduce optimization problems to decision problems.
  - We can similarly define the hypograph of classes of functions.
- Similarly, we can either interpret decision problems as TFs with a Boolean objective or specify a different objective function.

Relationship of Complexity Classes

- **Theorem 1** *(Krentel, 1987)* $f \in FP^{NP}$ *if and only if* $f(l) = h(l, g(l))$, *where* $g \in OptNP$ *and* $h \in FP$.
- Roughly, all functions that can be computed in polynomial time with an oracle for a language complete for NP can be reduced to optimization functions.
- It's really true that "everything is optimization"!
- We further have (Vollmer and Wagner, 1995)

$$
\begin{aligned}
NP &= \text{hypo}(MaxNP) \\
coNP &= \text{hypo}(MinNP) \\
PP &= \text{hypo}(MedNP)
\end{aligned}
$$

- Krentel (1987) shows $OptNP$-completeness results for weighted SAT, Max-SAT, TSP, 0-1 IP, and Knapsack.

Outline

The Polynomial Hierarchy

The polynomial hierarchy is a scheme for classifying multi-level and multi-stage decision problems. We have

$$\Delta_0^p := \Sigma_0^p := \Pi_0^p := P,$$

where P is the set of decision problems that can be solved in polynomial time. Higher levels are defined recursively as:

$$\begin{aligned} \Delta_{k+1}^p &:= P^{\Sigma_k^p}, \\ \Sigma_{k+1}^p &:= NP^{\Sigma_k^p}, \textit{and} \\ \Pi_{k+1}^p &:= coNP^{\Sigma_k^p}. \end{aligned}$$

PH is the union of all levels of the hierarchy.

First Three Levels of the Hierarchy

Collapsing the Hierarchy

In general, we have

$$\Sigma_0^p \subseteq \Sigma_1^p \subseteq \dots \Sigma_k^p \subseteq \dots$$
$$\Pi_0^p \subseteq \Pi_1^p \subseteq \dots \Pi_k^p \subseteq \dots$$
$$\Delta_0^p \subseteq \Delta_1^p \subseteq \dots \Delta_k^p \subseteq \dots$$

It is not known whether any of the inclusions are strict. We do have that

$$(\Sigma_k^p = \Sigma_{k+1}^p) \Rightarrow \Sigma_k^p = \Sigma_j^p \ \forall j \geq k$$

In particular, if $P = NP$, then every problem in the PH is solvable in polynomial time. Similar results hold for the Π and Δ hierarchies.

Satisfiability Game

- The canonical complete problem in *PH* is the *k-player satisfiability game*.
  - k players determine the value of a set of Boolean variables with each in control of a specific subset.
  - In round i, player i determines the values of her variables.
  - Each player tries to choose values that force a certain end result, given that subsequent players may be trying to achieve the opposite result.
- Examples
  - $k = 1$: SAT
  - $k = 2$: The first player tries to choose values such that any choice by the second player will result in satisfaction.
  - $k = 3$: The first player tries to choose values such that the second player cannot choose values that will leave the third player without the ability to find satisfying values.
- Note that the odd players and the even players are essentially "working together" and the same game can be described with only two players.

More Formally

- More formally, we are given a Boolean formula with variables partitioned into k sets $X_1, \ldots, X_k$.
- The decision problem

$$\exists X_1 \forall X_2 \exists X_3 \ldots ?X_k$$

is complete for Σ_k^p.
- The decision problem

$$\forall X_1 \exists X_2 \forall X_3 \ldots ?X_k$$

is complete for Π_k^p.
- A more general form of this problem, known as the *quantified Boolean formula problem* (QBF) allows an arbitrary sequence of quantifiers.

Reduction from SAT Game to Multi-level Optimization

- It is easy to formulate SAT games as multi-level integer programs.
- For $k = 1$, SAT can be formulated as the (feasibility) integer program

$$?\exists x \in \{0,1\}^n : \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \ \forall j \in J. \tag{SAT}$$

- (SAT) can be re-formulated as the optimization problem

$$\begin{aligned} \max_{x \in \{0,1\}^n} \ & \alpha \\ \text{s.t.} \ & \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq \alpha \ \forall j \in J \end{aligned}$$

- For $k = 2$, we then have

$$\begin{aligned} \min_{x_{I_1} \in \{0,1\}^{I_1}} \max_{x_{I_2} \in \{0,1\}^{I_2}} \ & \alpha \\ \text{s.t.} \ & \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq \alpha \ \forall j \in J \end{aligned}$$

Complexity of Multi-Level Optimization

- The reductions on the previous slide can be generalized to k levels.
- For the k-level optimization problem, the optimal value is ≥ 1 if and only if the first player has a winning strategy.
- This means the satisfiability game can be reduced to the (decision) problem of whether the optimal value ≥ 1?
- This decision problem is then complete for Σ_k^p.
- More generally, this means that (the decision version of) k-level mixed integer programming is also complete for Σ_k^p.
- By swapping the "min" and the "max," we can get a similar decision problem that is complete for Π_k^p.

$$\min_{x_{N_1} \in \{0,1\}^{N_1}} \max_{x_{N_2} \in \{0,1\}^{N_2}} \alpha$$

$$\text{s.t.} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq \alpha \ \forall j \in J$$

- The question remains whether the optimal value is ≥ 1, but now we are asking it with respect to a minimization problem.

The Min-Max Hierarchy

- The *Min-Max hierarchy* is a hierarchy of function classes defined by Krentel (1992) mirroring the polynomial hierarchy.

$$\Delta_0^{MM} := \Sigma_0^{MM} := \Pi_0^{MM} := FP,$$

$$\begin{aligned} \Delta_{k+1}^{MM} &:= FP^{\Sigma_k^{MM} \cup \Pi_k^{MM}}, \\ \Sigma_{k+1}^{MM} &:= Max\Pi_k^{MM}, \\ \Pi_{k+1}^{MM} &:= Min\Sigma_k^{MM}. \end{aligned}$$

- We can thus more accurately say that k-level maximization integer programs are complete for Σ_{k+1}^{MM}.

Relationship of the Hierarchies

- Many of the earlier results can be generalized. For example, we have (Vollmer and Wagner, 1995)

$$\Sigma_k^p = \text{hypo}(\Sigma_k^{MM})$$

- Also, any language $L \in \Delta_{k+1}^p$ can be expressed as $L = \{x \mid g(x, f(x))\}$ for some $f \in \Sigma_k^{MM}$ and some Boolean function $g \in FP$ Krentel (1992).

Alternating Turing Machines

- An *alternating Turing machine* (ATM) can directly model the computations required to solve multi-level optimization problems.
- In addition to accepting and rejecting states, these machines have two other special classes of state.

  - The "$\vee$" is accepting if there exists some configuration reachable in one step that is accepting and rejecting otherwise ($\exists$).
  - The "$\wedge$" is accepting if all configurations reachable in one step are accepting, and rejecting otherwise ($\forall$).

- Another way of thinking of this is that the final result is obtained by combining the states of all paths using the $\vee$ and $\wedge$ operators.
- Such a machine can switch between existential and universal quantification and is thus capable of solving multi-level decision problems directly.
- Σ_k^{MM} can be defined as languages recognizable on a machine with at most k alternations on any given path.
- The canonical problem that can be solved by an ATM is the aforementioned QBF problem.

Metric ATMs

- A metric version of an ATM is one for which each branch is associated with a "max" or "min" operator.
- The value output by the machine is calculated by combining the values in each accepting state with the "max" and "min" operators.
- Metric ATMs can solve general multi-level optimization problems.
- Subtrees of the execution tree encode the value functions of lower level problems.

Outline

Separation Functions

- The *membership problem* for a set S and a point x is the decision problem of determining whether $x \in S$.
- An optimization version of this problem is

$$\min_{y \in S} \|y - x\| \qquad \text{(SEP)}$$

for norm $\|\cdot\|$.
- We call (SEP) the *separation problem* associated with S.
- The *separation function* associated with $f \in OptA$, defined over a language L, is an optimization function

$$f^p_{\text{sep}}(x, l) = (y^*, \|y^* - x\|_p),$$

where $y^* \in \text{argmin}_{y \in feas(l)} \|y - x\|_p$ for $l \in L$.
- For $f \in OptA$ with convex feasible set, f^2_{sep} is closely related to the usual separation problem.
  - From the point y^*, we can obtain a separating hyperplane.
  - There are a number of alternative objective functions that can be employed.

Equivalence of Optimization and Separation

- The well-known equivalence of optimization and separation was proven by Grötschel et al. (1988).
- This result depends on the interpretation of the separation problem as an optimization problem (we need the separating hyperplane).

Definition 1 *If $f \in OptA$ is an optimization function defined over a language L, f is said to have a linear objective if $\exists d_l \in \mathbb{R}^n$ such that $g_l(x) = d_l^\top x \; \forall x \in \mathrm{feas}(l)$.*

- We conjecture it is possible to state the result of GLS using functions, roughly as follows.

Conjecture 1 *(Grötschel et al., 1988) Let f be an optimization function defined over a language L. If f has a linear objective and $\mathrm{feas}(l)$ is polyhedral for all $l \in L$, then $f \in OptA \Leftrightarrow f^2_{sep} \in OptA$.*

- We assume f^2_{sep} returns the separating hyperplane, so the complexity of f implicitly depends on the *facet complexity*.

Outline

Inverse Problems

- An *inverse problem* is one in which we want to determine the input that would produce a given output.
- To be more formal, let f be a TF defined over a language L.
- For a given partial input $l \in \Gamma^*$ and a solution x, an inverse problem associated with f is of the form

$$?\exists \hat{l} \in \Gamma^* \text{ s.t. } (\hat{l}, l) \in L \text{ and } f(\hat{l}, l) = (x, g(x))$$

- As stated, this is a decision problem with input (l, x).
- In principle, it can be solved by an NDTM accepting the language

$$L_{inv} = \{(l, x) \mid \exists \hat{l} \in \Gamma^* \text{ s.t. } (\hat{l}, l) \in L \text{ and } f(\hat{l}, l) = (x, g(x))\}$$

Conjecture 2 *If L_{inv} is the language arising from an inverse problem associated with a TF $f \in A$, then $L_{inv} \in NP^A$.*

Inverse Functions

- Inverse problems can also be expressed in the form of an optimization problem by requiring a "target" l^* as part of the input.
- The challenge is to find a feasible completion of the input that is as close as possible to the target.
- Formally, we can define an *inverse function* f^p_{inv} over the language L_{inv} by adding the objective function

$$g_{(l,x,l^*)}(\hat{l}) = \|l - \hat{l}\|_p$$

We can generalize the previous conjecture to

Conjecture 3 *If L_{inv} is the language arising from an inverse problem associated with a TF $f \in A$, then $f^{\infty}_{inv}, f^{1}_{inv} \in FNP^A$.*

Special Inverse Problems

- When f has a linear objective function, we assume the objective vector is an explicit part of the input.
- Let a q be the description of a given feasible region, $c \in \mathbb{R}^n$ a given objective function vector, and $x \in \text{feas}(c, q)$.
- Then the inverse problem for the ℓ_∞ norm can be stated as

$$\begin{aligned} &\min \|c - d\|_\infty \\ &\text{s.t. } d^T x \leq d^T y && \forall y \in \text{feas}(c, q) \\ &d \in \mathbb{R}^n \end{aligned}$$

- This can be linearized, as follows

$$\begin{aligned} &\min z \\ &s.t. \\ &c_i - d_i \leq z && \forall i \in \{1, 2, \ldots, n\} \\ &d_i - c_i \leq z && \forall i \in \{1, 2, \ldots, n\} \\ &d^T x \leq d^T y && \forall y \in \text{feas}(c, q) \end{aligned}$$

Complexity of Inverse Functions

Theorem 2 *Let $f \in MaxA$ be a TF defined over a language L such that $\text{feas}(l)$ is polyhedral for all $l \in L$ and f has a linear objective function. Then $f^{\infty}_{inv}, f^{1}_{inv} \in FP^{MaxA} = FP^{A}$.*

Proof: Follows from Theorem 1 (GLS).

Corollary 1 *Inverse integer programming with the ℓ_∞ and ℓ_1 norms is in FP^{OptNP}.*

Outline

Multilevel Nature of Branch and Cut

- Consider an instance of MILP

MILP

$$\min\{c^\top x \mid x \in \mathcal{P} \cap (\mathbb{Z}^p \times \mathbb{R}^{n-p})\}, \qquad \text{(MILP)}$$

where $\mathcal{P} = \{x \in \mathbb{R}^n_+ \mid Ax = b\}$, $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$.

- A *branch-and-cut algorithm* to solve this problem requires the solution of two fundamental problems.

Definition 2 *The separation problem for a polyhedron $\mathcal{Q}$ is to determine for a given $\hat{x} \in \mathbb{R}^n$ whether or not $\hat{x} \in \mathcal{Q}$ and if not, to produce an inequality $(\bar{\alpha}, \bar{\beta}) \in \mathbb{R}^{n+1}$ valid for $\mathcal{Q}$ and for which $\bar{\alpha}^\top \hat{x} < \bar{\beta}$.*

Definition 3 *The branching problem for a set $\mathcal{S}$ is to determine for a given $\hat{x} \in \mathbb{R}^n$ whether $\hat{x} \in \mathcal{S}$ and if not, to produce a disjunction*

$$\bigvee_{h\in\mathcal{Q}} A^h x \geq b^h, \; x \in \mathcal{S} \tag{1}$$

that is satisfied by all points in $\mathcal{S}$, but not satisfied by $\hat{x}$.

Multilevel Structure of the Separation Problem

- Often, we wish to select an inequality that maximizes violation, i.e., $(\alpha, 1)$, where

$$\bar{\alpha} \in \operatorname{argmin}_{\alpha \in \mathbb{R}^n} \{\alpha^\top \hat{x} \mid \alpha^\top x \geq 1 \; \forall x \in \mathcal{Q}\} \tag{2}$$

- To make the problem tractable, we may restrict ourselves to a specific *template class* of valid inequalities with well-defined structure.
- Given a class C, calculation of the right-hand side β required to ensure (α, β) is a member of $\mathcal{C}$ may itself be an optimization problem.
- The separation problem for the class $\mathcal{C}$ with respect to a given $\hat{x} \in \mathbb{R}^n$ can in principle be formulated as the bilevel program:

$$\min \; \alpha^\top \hat{x} - \beta \tag{3}$$

$$\alpha \in C_\alpha \tag{4}$$

$$\beta = \min_{x \in \mathcal{P}_C} \{\alpha^\top x\}, \tag{5}$$

where the set $C_\alpha \subseteq \mathbb{R}^n$ is the projection of C into the space of coefficient vectors and $\mathcal{P}_C$ is the closure over the class C.

Formulating the Cut Generation Problem

- In other words, $\mathcal{C}_\alpha$ is the set of all vectors that are coefficients for some inequality in $\mathcal{C}$.
- The upper-level objective (3) is to find the maximally violated inequality in the class, while the upper-level constraints (4) require that the inequality is a member of the class.
- The lower-level problem (5) is to generate the strongest possible right-hand side associated with a given coefficient vector, i.e., the largest β value among the feasible ones.
- The difficulty of the separation problem depends on the form of the *right-hand side generation problem*.

Example: Disjunctive cuts

- Given a MIP in the form (MILP), Balas (1979) showed how to derive a valid inequality by exploiting any fixed disjunction

$$\pi^\top x \leq \pi_0 \quad \text{OR} \quad \pi^\top x \geq \pi_0 + 1 \; \forall x \in \mathbb{R}^n, \tag{6}$$

where $\pi \in \mathbb{Z}^n$ and $\pi_0 \in \mathbb{Z}$.

- A *disjunctive inequality* is one valid for the convex hull of union of $\mathcal{P}_1$ and $\mathcal{P}_2$, obtained by imposing the two terms of the disjunction.
- The separation problem can be written as the following bilevel program:

$$\min \quad \alpha^\top \hat{x} - \beta \tag{7}$$

$$\alpha \geq u^\top A - u_o \pi \tag{8}$$

$$\alpha \geq v^\top A + v_o \pi \tag{9}$$

$$u, v, u_0, v_0 \geq 0 \tag{10}$$

$$u_0 + v_0 = 1 \tag{11}$$

$$\beta = \min\{\alpha^\top x \mid x \in \mathcal{P}_1 \cup \mathcal{P}_2\} \tag{12}$$

Example: Disjunctive Cuts (cont.d)

- Equation (12) requires β to have the largest value consistent with validity.
- To ensure the cut is valid, we need only ensure that

$$\beta \le \min\{u^\top b - u_0\pi_0, v^\top b + v_0(\pi_0 + 1)\}. \tag{13}$$

- Using the standard modeling trick, we can rewrite (13) as

$$\beta \le u^\top b - u_0\pi_0 \tag{14}$$

$$\beta \le v^\top b + v_0(\pi_0 + 1). \tag{15}$$

- The sense of the optimization ensures that (13) holds at equality.

Theorem 3 *For a fixed disjunction (π, π_0), the separation function associated with the disjunctive closure is in FP.*

Example: Capacity Constraints for CVRP

- In the Capacitated Vehicle Routing Problem (CVRP), the *capacity constraints* are of the form

$$\sum_{\substack{e=\{i,j\}\in E\\ i\in S, j\notin S}} x_e \geq 2b(S) \quad \forall S \subset N,\ |S| > 1, \tag{16}$$

 where $b(S)$ is any lower bound on the number of vehicles required to serve customers in set S.

- By defining binary variables
  - $y_i = 1$ if customer i belongs to $\bar{S}$, and
  - $z_e = 1$ if edge e belongs to $\delta(\bar{S})$,

  we obtain the following bilevel formulation for the separation problem:

$$\min \sum_{e\in E} \hat{x}_e z_e - 2b(\bar{S}) \tag{17}$$

$$z_e \geq y_i - y_j \qquad \forall e \in E \tag{18}$$

$$z_e \geq y_j - y_i \qquad \forall e \in E \tag{19}$$

$$b(\bar{S}) = \max\{b(\bar{S}) \mid b(\bar{S}) \text{ is a valid lower bound}\} \tag{20}$$

Example: Capacity Constraints for CVRP (cont.d)

If the bin packing problem is used in the lower-level, the formulation becomes:

$$\min \sum_{e \in E} \hat{x}_e z_e - 2b(\bar{S}) \tag{21}$$

$$z_e \geq y_i - y_j \qquad \forall e = \{i,j\} \tag{22}$$

$$z_e \geq y_j - y_i \qquad \forall e = \{i,j\} \tag{23}$$

$$b(\bar{S}) = \min \sum_{\ell=1}^{n} h_\ell \tag{24}$$

$$\sum_{\ell=1}^{n} w_i^\ell = y_i \qquad \forall i \in N \tag{25}$$

$$\sum_{i \in N} d_i w_i^\ell \leq K h_\ell \qquad \ell = 1, \ldots, n, \tag{26}$$

where we introduce the additional binary variables

- $w_i^\ell = 1$ if customer i is served by vehicle ℓ, and
- $h_\ell = 1$ if vehicle ℓ is used.

Complexity of the Separation Function for GSECs

Theorem 4 *The optimization function described by* (21)–(26) *is in the complexity class* Σ_2^{MM}.

Proof: Reduction to 2-Quantified 1-in-3 SAT.

Multi-level Structure of the Branching Problem

- A typical criteria for selecting a branching disjunction is to maximize the bound increase resulting from imposing the disjunction.
- The problem of selecting the disjunction whose imposition results in the largest bound improvement has a natural *bilevel structure*.
  - The upper-level variables can be used to model the choice of disjunction (we'll see an example shortly).
  - The lower-level problem models the bound computation after the disjunction has been imposed.
- In strong branching, we are solving this problem essentially by enumeration.
- The bilevel branching paradigm is to select the branching disjunction directly by solving a bilevel program.

Example: Interdiction Branching

The following is a bilevel programming formulation for the problem of finding a smallest branching set in interdiction branching:

$$\max \sum c^\top x \tag{27}$$

$$\text{s.t.} \tag{28}$$

$$c^\top x \leq \bar{z} \tag{29}$$

$$y \in \mathbb{B}^n \tag{30}$$

$$x \in \arg\max\{c^\top x \mid x_i + y_i \leq 1 \forall i \in N^a, x \in \mathcal{F}^a\} \tag{31}$$

where $\mathcal{F}^a$ is the feasible region of a given relaxation of the original problem used for computing the bound.

Conjecture 4 *The optimization function described by* (27)–(31) *is in the complexity class* Σ_2^{MM}.

Further Generalizations and Conclusions

- We can generate separation and branching functions of any level in the complexity hierarhcy by “looking ahead” multiple levels.
- The separation functions for closures of rank > 1 are also likely in higher levels of the hierarchy.
- The framework presented here seems to be promising in terms of analyzing the complexity of these and related multi-level optimization problems.
- This is a first stab at a general framework, but I’m sure it could use tweaking.
- If you have thoughts, feel free to talk to me.

References I

Balas, E. 1979. Disjunctive programming. In *Annals of Discrete Mathematics 5: Discrete Optimization*, pages 3–51. North Holland.

Garey, M. and D. Johnson 1979. *Computers and Intractability: A Guide to the Thoery of NP-Completeness*. W.H. Freeman and Company.

Grötschel, M., L. Lovász, and A. Schrijver 1988. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, New York.

Krentel, M. 1987. *The Complexity of Optimization Problems*. Ph.D. thesis, Cornell University.

— 1988. The complexity of optimization problems. *Journal of Computer and System Sciences* **36**, 490–509.

— 1992. Generalizations of optp to the polynomial hierarchy. *Theoretical Computer Science* **97**, 183–198.

Vollmer, H. and K. Wagner 1995. Complexity classes of optimization functions. *Information and Computation* **120**, 198–219.