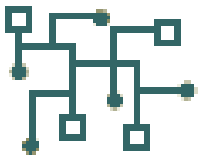
Duality and Meaning in Computation (Some Half-baked Ideas)

Ted Ralphs[1]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University

UG Workshop, 1 October 2021



ISE Industrial and Systems Engineering

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

What Is This Talk About?

- This talk describes some (very) preliminary ideas for thinking about how to "interpret" or "explain" the result of a computation.
- The premise is that existing duality concepts make a good basis for the development of an "explainability" framework.
- The talk will also have an overarching "meta" theme.
  - There many different "languages" for taking about the topics here and the language used can make a big difference to intuition.
  - Within different branches of mathematics, there are many notions of "duality" that can be seen as roughly isomorphic to each other.
  - The theory of computation reflects many of these same notions of duality and interprets them in a computational context.
- Learning to speak another language can open your eyes to many fascinating and relevant ideas!
- Alternatively, we (mathematical optimizers) may also have something unique to add to the conversation.

Caveats

- This is my very first talk on a big topic and the ideas are not fully baked.
- I'll touch on things I'm not an expert in and wave my hands A LOT.
- Realistically, some of what is in this talk was already known by Turing, Gödel, Bourbaki, etc. on some level (think about that...).
- But... the language they used was *very* abstract and the early pioneers developed multiple (equivalent) foundation on which to formalize.
- All of this makes it very difficult to see the big picture.

Shout Out

At a pivotal time in the development of these ideas, I picked up *Gödel, Escher, Bach*. Go read (or re-read) it!

Human versus Machine

- The extreme rigor needed initially to develop viable mathematical/computational theories tends to *really* obscure intuition.
- There is a very fine line to walk between rigor and intuition.
- Interestingly, this reflects the same fine line we need to walk in developing ways to "explain" computations to humans.
- The goal is to develop practical techniques, not just theoretical results.

Lost In Translation

$$\begin{aligned}
\text{model of computation} &\Leftrightarrow \text{computer (hardware), machine language} \\
\text{Turing machine} &\Leftrightarrow \text{computer program, algorithm} \\
\text{"universal" Turing machine} &\Leftrightarrow \text{compiler, interpreter} \\
\text{language} &\Leftrightarrow \text{problem (class), function, set} \\
\text{formal system} &\Leftrightarrow \text{programming language} \\
\text{theorem} &\Leftrightarrow \text{instance}
\end{aligned}$$

Duality Concepts

The following roughly "isomorphic" duality concepts will all appear.

Duality Concepts

- Sets: Projection/complement, intersection/union
- Conic duality: Cones and their duals, convexity/nonconvexity
- Farkas duality: Theorems of the alternative, empty/non-empty
- Complexity: Languages and their complements (NP vs. co-NP)
- Quantifier duality: Existential versus universal quantification
- De Morgan duality: Conjunction versus disjunction
- Weyl-Minkowski duality: V representation versus H representation
- Polarity: Optimization versus separation
- Dual problems: Primal and dual problems in optimization
- Inverses: Functions and inverses, inverse optimization inverses

Decision Problems and Complexity

- We have in mind problems in the *polynomial hierarchy* (PH), which is a widely used scheme for classifying optimization problems.
- This scheme applies to problems for which the result of a computation is "YES" or "NO."
- It is useful, however, to interpret such a problem as that of trying to prove a theorem, which must be either "TRUE" or "FALSE".
- By viewing the proof as part of the output, it is easier to see that this class of problems is in fact very rich.
- The notion of a proof is fundamental to how problems are classified in the PH—higher complexity means longer proofs are expected.
- When considering "explainability," it makes sense to start by contrasting how humans and computers generate proofs.

Formal Proofs and Certificates

- In the theory of computation, the *formal proof* that the answer given by an algorithm is correct is sometimes called a *certificate*.
- Formal proofs are constructed using the logic of a specific *formal system*.
- Such a proof is generally intended to be automatically verified by another algorithm (if at all).
- The mathematical proofs that humans write, on the other hand, are meant to be read by *other humans*.
- They leave out steps that must be filled in and are constructed to be as concise as possible—they are ultimately a *social construct*.
- One view of the "explainability gap" is that it is about bridging this difference.

Possible Goals of Explainability

1. Make a machine-generated proof understandable to a human.
2. Make the proof and intuitive as short as possible.

The Human Computer Interface

- Programming languages are an intermediate layer that allows humans to communicate with machines.
- They are meant to capture both the formality and lack of ambiguity of mathematics, while still being understandable to a human.
- What is effectively missing is the ability to translate in the reverse direction.
- We want the computer to explain what it did, also in some kind of intermediate language.
- This would effectively be a “reverse compiler”.
- What should the intermediate language be?

Theorems About Sets

- In optimization (and even more generally), the "theorems" we wish to prove or disprove can be formulated as statements about sets.
- Let $\mathcal{S} = \{x \in \mathbb{Q}^n \mid P(x)\}$, where $P : \mathbb{Q}^n \to \{TRUE, FALSE\}$.
- The simplest question we can ask is whether $\mathcal{S}$ is non-empty

$$\mathcal{S} \stackrel{?}{=} \emptyset.$$

- Given function f and constant K, the related question of

$$\mathcal{S}(f, K) := \{x \in \mathcal{S} \mid f(x) < K\} \stackrel{?}{=} \emptyset$$

is the *decision version* of the optimization problem

$$\min_{x \in \mathcal{S}} f(x) \qquad \text{(OPT)}$$

Constructing Proofs

- What do proofs of theorems about sets look like?
  - Certifying $\mathcal{S} \neq \emptyset$ is easy: produce a point in the set.
  - Certifying $\mathcal{S} = \emptyset$ is more difficult in general.
- The difficulty of proving a set is empty is most easily seen by re-stating the theorems we are trying to prove/disprove, as follows.

$$\mathcal{S} \neq \emptyset \Leftrightarrow \exists x \in \mathcal{S}$$

$$\mathcal{S} = \emptyset \Leftrightarrow \forall x \in \mathbb{Q}^n \; x \notin \mathcal{S} \Leftrightarrow \forall x \in \mathbb{Q}^n \; x \in \bar{\mathcal{S}}$$

- the statement that that a set is non-empty is *existentially quantified*, whereas the statement that a set is empty is *universally quantified*.
- Universally quantified statements are intuitively more difficult to prove than existentially quantified ones.

De Morgan Duality

- There is a duality between existential and universal quantifiers that can be seen as one of a number of generalized forms of De Morgan's Laws.

DeMorgan's Laws

- The complement of the union is the intersection of the complements.
- The complement of the intersection is the union of the complements.

- These laws can be used to equivalently formulate logical statements in different dual forms to aid in constructing proofs.

$$P(x)\ \forall x \in \mathcal{S} \Leftrightarrow \neg[\exists x \in \mathcal{S}\ \neg P(x)] \Leftrightarrow \neg \bigvee_{x \in \mathcal{S}} \neg P(x) \Leftrightarrow \bigwedge_{x \in \mathcal{S}} P(x)$$

$$\exists x \in \mathcal{S} : P(x) \Leftrightarrow \neg[\forall x \in \mathcal{S}\ P(x)] \Leftrightarrow \neg \bigwedge_{x \in \mathcal{S}} \neg P(x) \Leftrightarrow \bigvee_{x \in \mathcal{S}} P(x)$$

- Note also the duality between conjunction and disjunction.

Convexity and Nonconvexity

- Related dualities exist between between conjunction and disjunction, which are reflected in the way convex and nonconvex sets are described.
  - Convex sets are described by conjunctive logic: the *intersection* of convex sets is convex.
  - Nonconvex sets are described using disjunctive logic: the *union* of convex sets is nonconvex (in general).
- This is why there is a short proof that a point is *not* in a convex set.
  - The Farkas Lemma and the separating hyperplane theorem in convex analysis provide methods for generating such proofs.
  - There is a short proof of emptiness for any set described as the intersection of simple convex sets, e.g., half-spaces.
- Proving a point is not in a nonconvex set is hard, which is why we can't expect short proofs of emptiness for disjunctive unions of convex sets.

Short Proofs of Emptiness

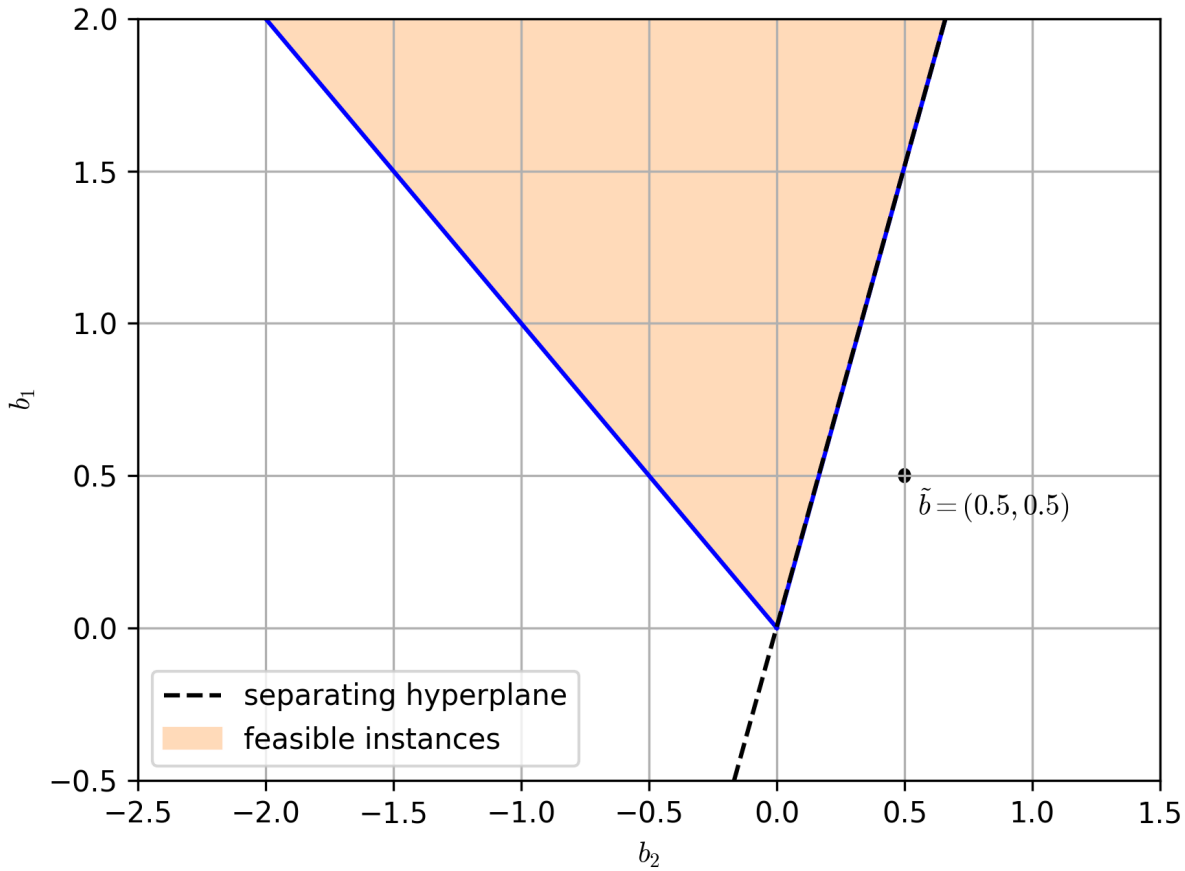
- In the case of convex sets, we can use a duality argument to obtain short proofs of emptiness.
- Consider the case of a polyhedron.

$$\mathcal{P} = \{x \in \mathbb{Q}^n_+ \mid Ax = \tilde{b}\}$$

- **Farkas Lemma**: $\mathcal{P} = \emptyset \Leftrightarrow \exists u \in \mathbb{Q}^m \; A^\top u \leq 0, \tilde{b}^\top u > 0$
- Equivalently, $S = \emptyset$ if and only if we can separate $\tilde{b}$ from the convex cone $\mathcal{C} = \{b \in \mathbb{Q}^m \mid \exists x \in \mathbb{Q}^n_+, Ax = b\} = \{b \in \mathbb{Q}^m : b^\top u \leq 0 \; \forall u \in \mathcal{C}^*\}$, where $\mathcal{C}^* = \{u \in \mathbb{Q}^m : A^\top u \leq 0\}$ (the *polar* of $\mathcal{C}$).
- One way to interpret this procedure is as follows.
  - We first lift the problem into a higher dimensional space by making b a vector of variables to obtain a related *non-empty* set.
  - Then project out the original variables and apply the separating hyperplane theorem.

Example

$$6y_1 + 7y_2 + 5y_3 = 1/2$$
$$2y_1 - 7y_2 + y_3 = 1/2$$
$$y_1, y_2, y_3 \in \mathbb{R}_+$$



Meaning from Duality

- On one level, this is a "trick" for recasting a question of emptiness as one of non-emptiness (universal $\rightarrow$ existential), but there's a bigger picture.
- We are embedding a single theorem into a *parametric class* containing both TRUE and FALSE theorems.
- The questions we are asking is being re-cast as a question of where this theorem lies relative to the set of all TRUE theorems (in the class).
- To prove the theorem is FALSE, we separate it from the set of theorems that are TRUE—this is a"dual" proof based on a separation argument.
- In the terminology of complexity theory, the set of true theorems is called a *language*.

Proofs of Optimality

- The problem (OPT) is *not* a decision problem as stated.
- We can nevertheless build a proof that the optimal solution value is K using proofs for two related theorems.

1. $\exists x \in \mathcal{S} : f(x) = K$
2. $\nexists x \in \mathcal{S} : f(x) < K \Leftrightarrow \forall x \in \mathcal{S} : f(x) \geq K$

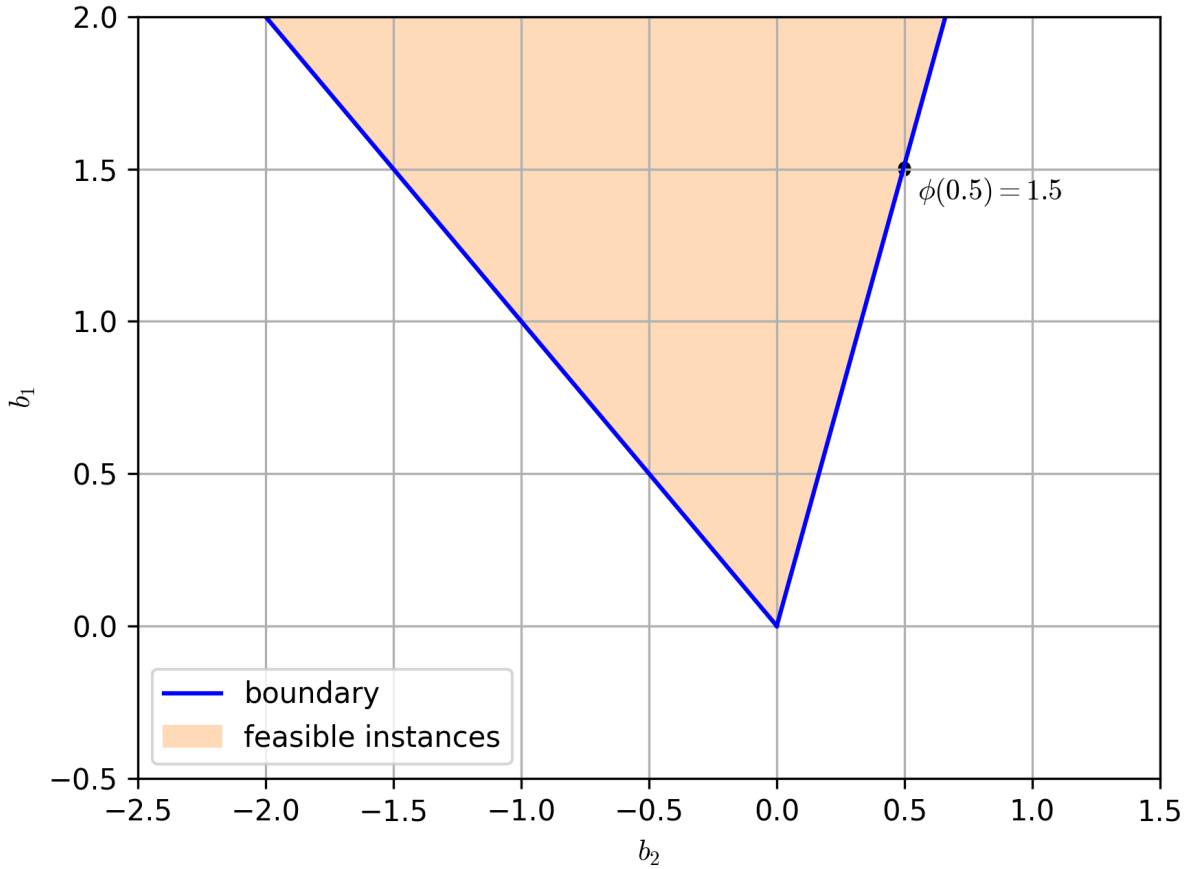
- The fact that one of these statements is universally quantified is the reason why short proofs of optimality cannot be expected in general.

Short Proofs of Optimality

- We consider the case of a linear optimization problem (LP).
- We can get an LP as follows.
  - Convert the first row of A from a constraint to the objective function.
  - Let $N = \{2, \ldots, m\}$ and $\tilde{b}_N \in \mathbb{Q}^{m-1}$ be all but the first element of $\tilde{b}$.
- The problem of finding the optimal value can then be recast as $b^* = \min\{b_1 \in \mathbb{Q} \mid b \in \mathcal{C}\}$.
- To prove optimality, we need to show that $(b^*, \tilde{b}_N)$ is not only a member of $\mathcal{C}$, but on its *boundary*.
- The proof is only slightly modified: $\exists u \in \mathbb{Q}^m, A^\top u \leq 0, (b^*, \tilde{b}_N)^\top u = 0, u_1 < 0$.
  - Assume u is scaled so that $u_1 = -1$.
  - Then we have $A_N^\top u_N \leq A_1^\top, (\tilde{b}_N)^\top u_N = b^*$.
  - This is equivalent to the usual LP optimality conditions, but also proves that $(b^*, \tilde{b}_N)$ is on the boundary of $\mathcal{C}$.
- The vector u is a solution to the usual LP dual problem.

Example

$$\min 6y_1 + 7y_2 + 5y_3$$
$$\text{s.t. } 2y_1 - 7y_2 + y_3 = 1/2$$
$$y_1, y_2, y_3 \in \mathbb{R}_+$$

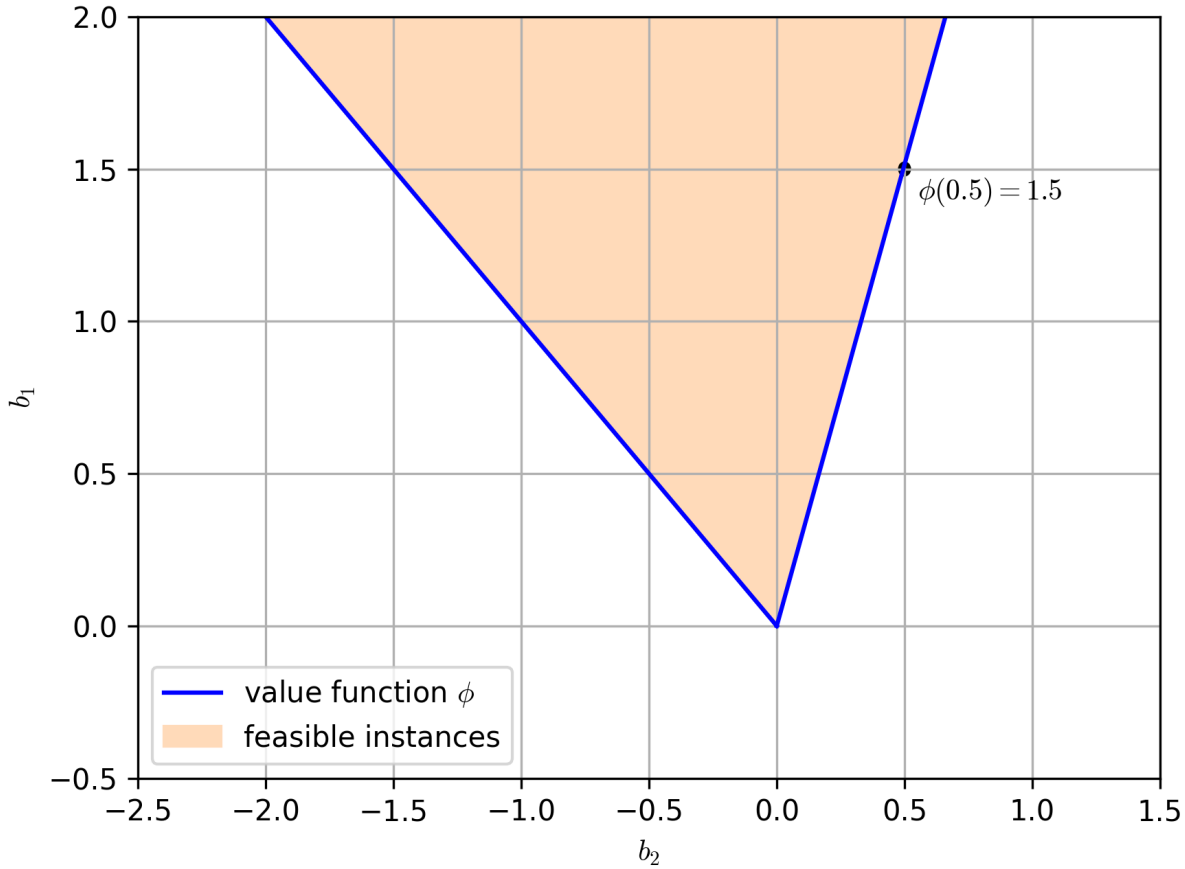


Interpreting

- It is not only the theorems in the class that are parametrically related, the proofs are themselves parametric.
- This parameterization can provide intuition to a human interpreter.
- For example, the boundary of the cone $\mathcal{C}$ describes a parametric collection of proofs and has several nice interpretations.
  - The boundary can be interpreted as specifying the *value function* of the associated optimization problem.
  - The solution to the LP dual problem is a (sub)gradient of this function.
  - Alternatively, the boundary also encodes the way constraints can be traded off against each other (the *Pareto frontier*).
- Both of these interpretations provide a human-interpretable meaning to the result.
- The "dual price" of a given constraint has an economic interpretation when the constraints are interpreted as allocating resources.
- This provides a language for describing the result of a convex optimization to a human.

Example

$$\begin{aligned} \min\ & 6y_1 + 7y_2 + 5y_3 \\ \text{s.t. } & 2y_1 - 7y_2 + y_3 = 1/2 \\ & y_1, y_2, y_3 \in \mathbb{R}_+ \end{aligned}$$

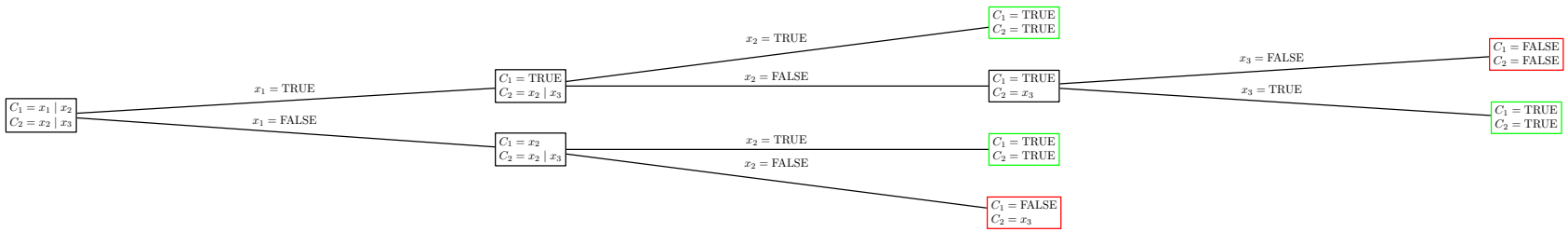


Nonconvex Sets

- We now move to nonconvex sets.
- Consider $\mathcal{S} \subseteq \mathbb{B}^n$, a collection of binary vectors.
- Such a set arises, for example, as a representation of the set of satisfying assignments in the Satisfiability Problem (SAT).
  - We're given a set of n literals $\{x_i\}_{i=1}^n$ which must be assigned value FALSE (0) or TRUE (1).
  - We're also given a set of clauses, which are disjunctions of (subsets of) the variables and their negations/complements.
  - Clause j is specified by the set C_j^0 of variables and C_j^1 of complements of variables.
  - Satisfaction of clause j means $\sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1$.
- SAT is to prove or disprove the theorem that there are values of the variables that simultaneously satisfy all clauses.
- For satisfiable (TRUE) instances, the (short) proof is a satisfying assignment.
- What about unsatisfiable instances?

Enumerating Proofs

- A naive algorithm consists of enumerating all proofs in the obvious way.
- If a satisfying assignment is found, output the assignment.
- In this case, the proof is short because we discard all the computational "dead ends."
- If no proof is found, then the set of proofs is empty and the theorem is FALSE.
- There doesn't seem to be any proof shorter than a representation of the entire search tree.
- In the worst case, such a proof has size exponential size.



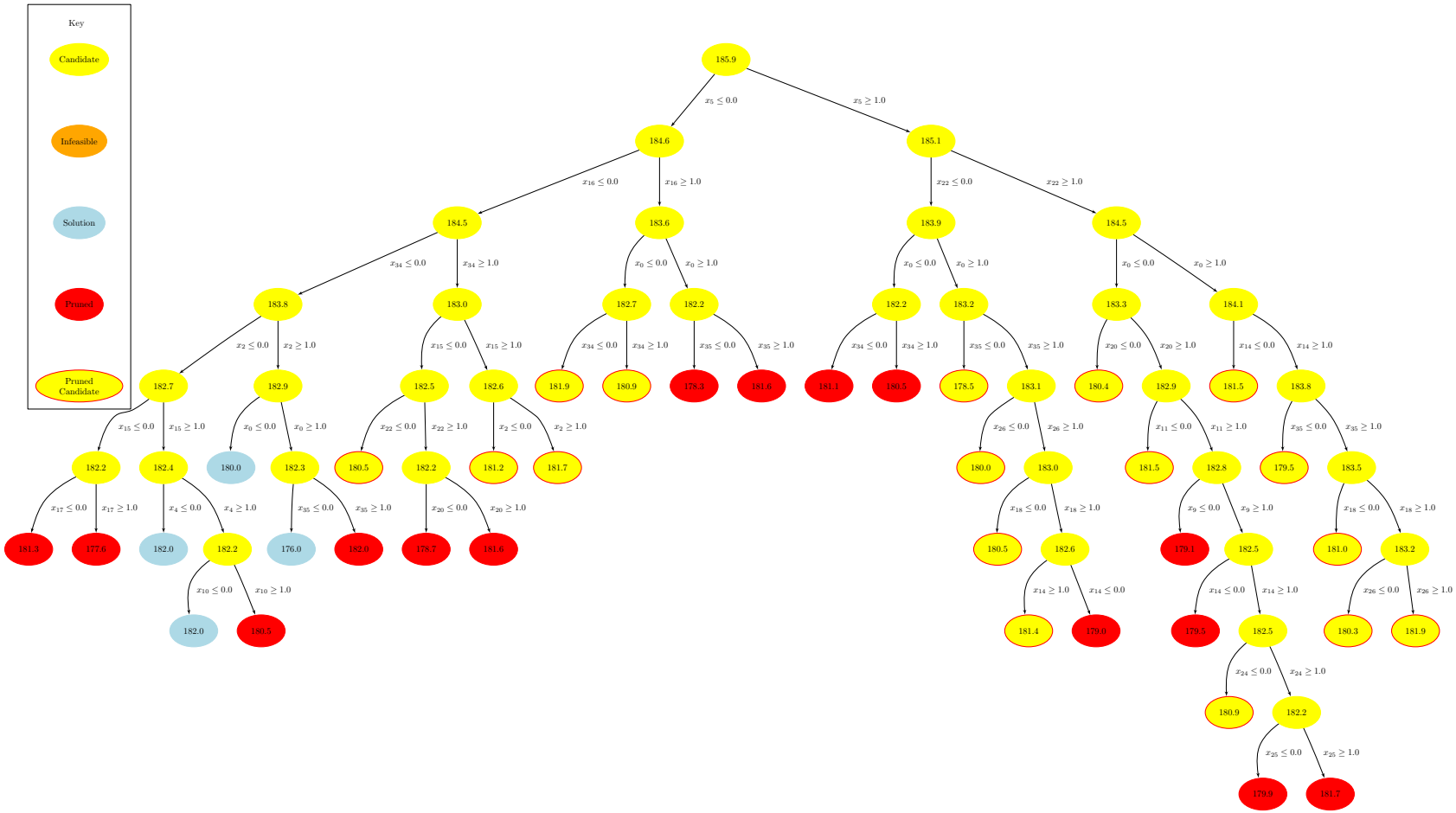
Extracting Proofs from Computations

- How do we make this more general?
- Suppose we have a program implementing an algorithm for solving some decision problem that doesn't explicitly output a proof.
- We can nevertheless think of the programming language as a formal system and the algorithm as a method of constructing a proof.
- In the Turing model, the algorithm modifies the "state" of the machine (i.e., the memory) systematically, searching for an "accepting state."
- A simple way to extract a proof is to make a record of the machine's state at each time step.
- If we're using a pure functional language, then we might simply record the input/output of (a subset of) the function calls in a *call tree*.
  - To verify a positive result, we need to show how to construct the accepting state from initial conditions (similarly eliminating dead ends).
  - As with SAT, it seems there is no short proof of the fact that we did not reach an accepting state.

Building a Better Proof

- How can we leverage the tools of duality to build a shorter, more interpretable proof?
- The proof of the Cook-Levin Theorem shows how to interpret the state of a Turing machine as a solution to a mathematical optimization problem.
- In fact, Liberti used a similar construction to show that mathematical optimization (MINLP) is a Turing-complete language.
- Hence, we can arguably think of proofs as solutions to a discrete optimization problem.
- Roughly speaking, this means that for any class of theorems in the PH, the search for a proof is somehow isomorphic to a branch-and-bound.
- Branch-and-bound, in turn, can be viewed as a technique for restating theorems as a disjunctive collection of theorems about convex sets.
- This means that a proof for the original result can be obtained as a (possibly large) collection of short Farkas-type proofs.

Branch And Bound



The Value Function of an MILP

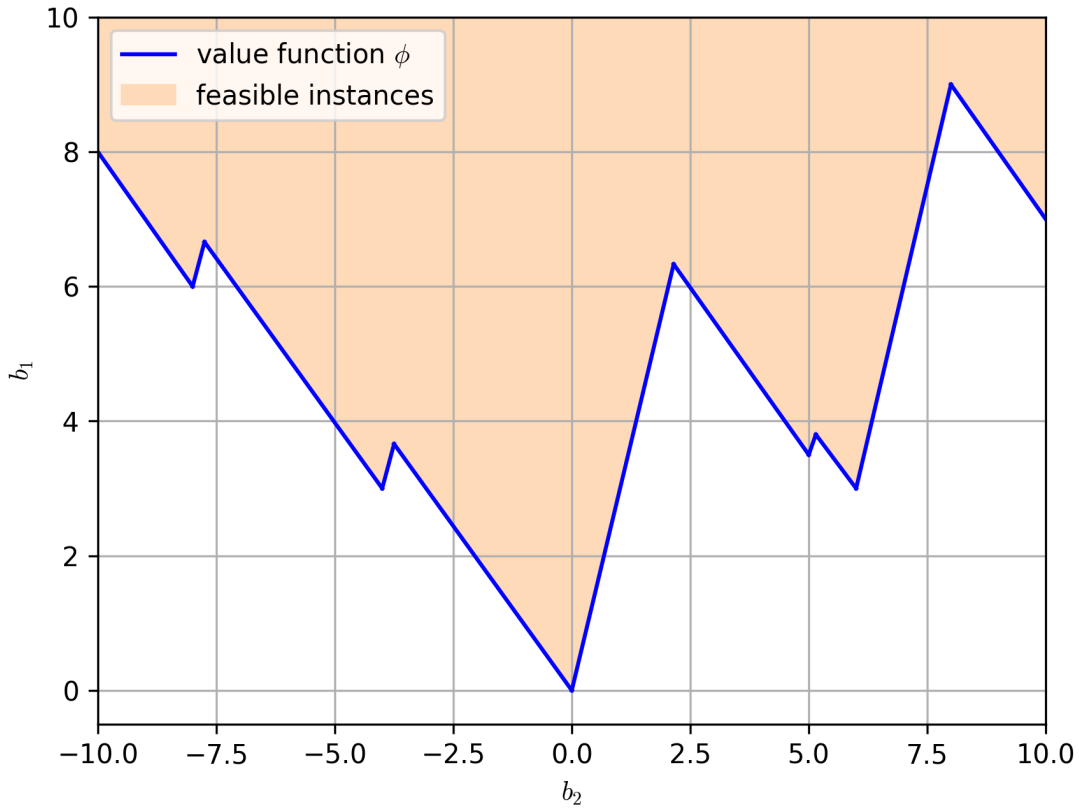
- The tree generated by an LP-based branch and bound encodes a kind of dual proof that generalizes the one seen earlier for LPs.
- Let us define the *value function* of an MILP as

$$\phi(\beta) = \min_{x \in \mathcal{S}(\beta)} c^\top x \tag{VF}$$

  for $\beta \in \mathbb{R}^m$, where $\mathcal{S}(\beta) = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = \beta\}$.
- As before, this function can be viewed as encoding the boundary between between feasible and infeasible sets of instances.
- This boundary is the value function.

Example

$$\phi(\beta) = \min\ 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 + 5x_6$$
$$\text{s.t. } 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 + x_6 = \beta$$
$$x_1, x_2, x_3 \in \mathbb{Z}_+,\ x_4, x_5, x_6 \in \mathbb{R}_+$$



Dual Proofs From Branch and Bound

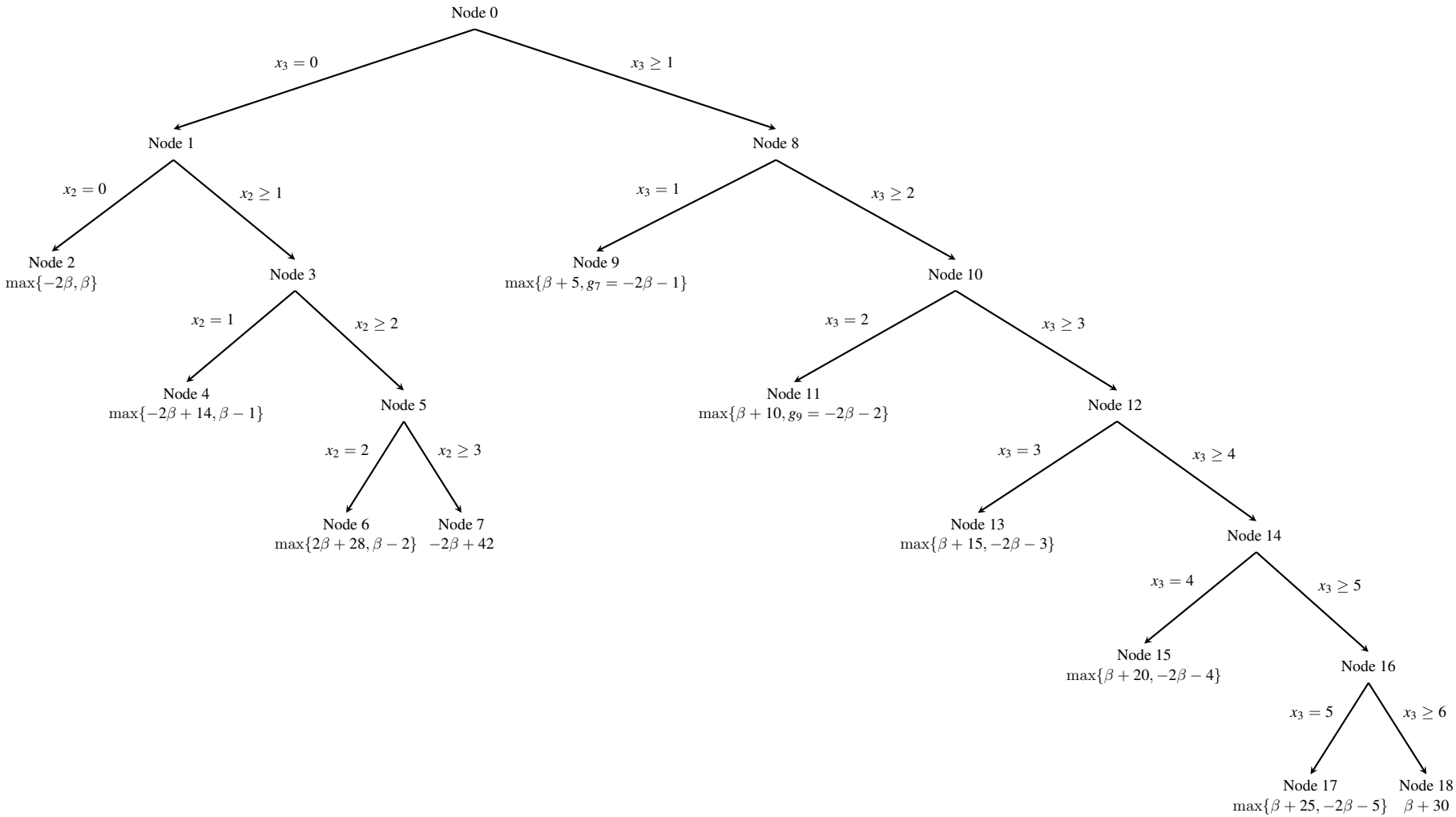
- The proof generated by a branch-and-bound tree can be seen as a separating hypersurface that bounds the value function from below.

$$\underline{\phi}^*(\beta) = \min_{t \in T} c_{I_t}^\top x_{I_t}^t + \phi_{N \setminus I_t}^t (\beta - A_{I_t} x_{I_t}^t), \tag{1}$$

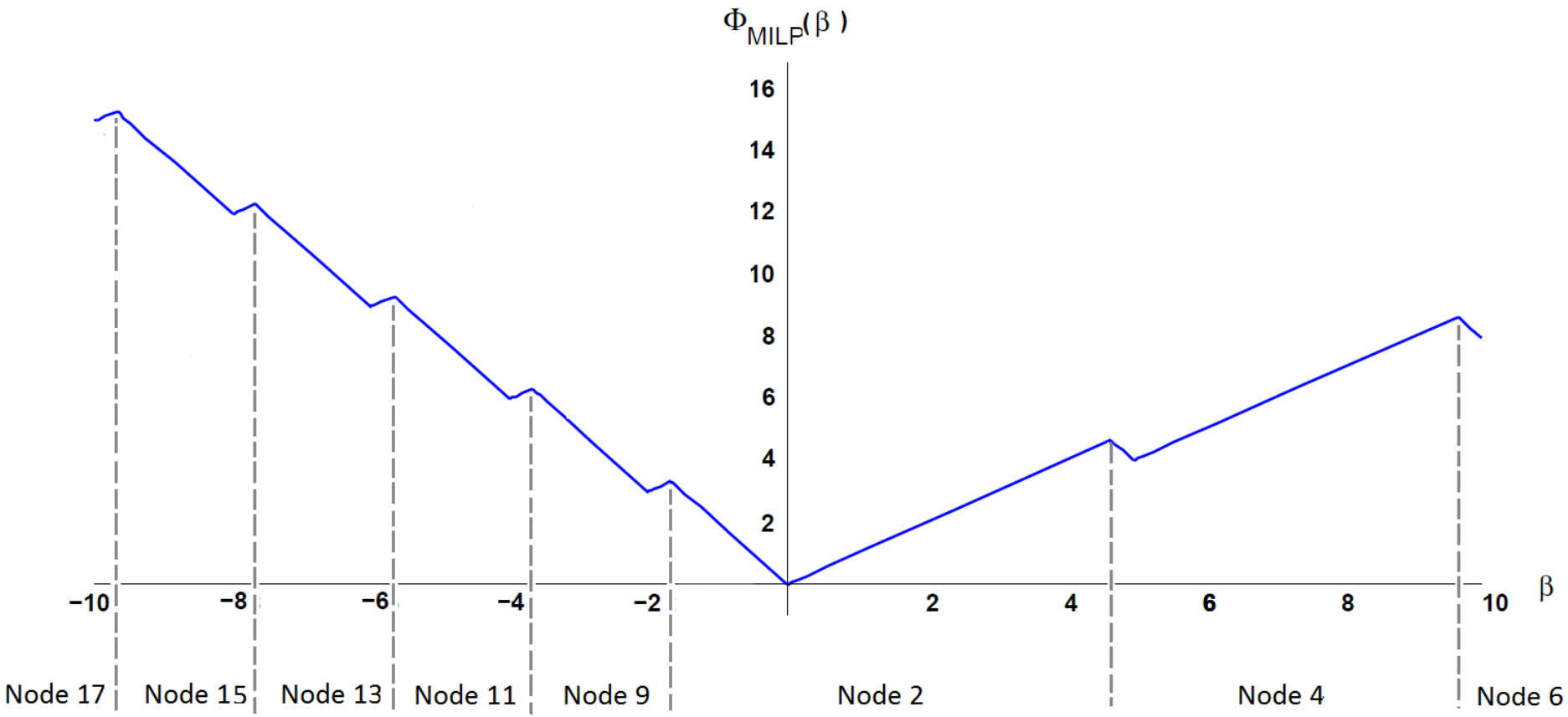
- I_t is the set of indices of fixed variables, $x_{I_t}^t$ are the values of the corresponding variables in node t.
- $\phi_{N \setminus I_t}^t$ is the value function of the linear optimization problem at node t, including only the unfixed variables.

Theorem 1 *Under the assumption that* $\{\beta \in \mathbb{R}^{m_2} \mid \phi_I(\beta) < \infty\}$ *is finite, there exists a branch-and-bound tree with respect to which* $\underline{\phi}^* = \phi$.

Example


Node 0
$x_3 = 0$
$x_3 \geq 1$
Node 1
Node 8
$x_2 = 0$
$x_2 \geq 1$
$x_3 = 1$
$x_3 \geq 2$
Node 2
$\max\{-2\beta, \beta\}$
Node 3
Node 9
$\max\{\beta + 5, g_7 = -2\beta - 1\}$
Node 10
$x_2 = 1$
$x_2 \geq 2$
$x_3 = 2$
$x_3 \geq 3$
Node 4
$\max\{-2\beta + 14, \beta - 1\}$
Node 5
Node 11
$\max\{\beta + 10, g_9 = -2\beta - 2\}$
Node 12
$x_2 = 2$
$x_2 \geq 3$
$x_3 = 3$
$x_3 \geq 4$
Node 6
$\max\{2\beta + 28, \beta - 2\}$
Node 7
$-2\beta + 42$
Node 13
$\max\{\beta + 15, -2\beta - 3\}$
Node 14
$x_3 = 4$
$x_3 \geq 5$
Node 15
$\max\{\beta + 20, -2\beta - 4\}$
Node 16
$x_3 = 5$
$x_3 \geq 6$
Node 17
$\max\{\beta + 25, -2\beta - 5\}$
Node 18
$\beta + 30$

Example


$\Phi_{MILP}(\beta)$
16
14
12
10
8
6
4
2
−10
−8
−6
−4
−2
2
4
6
8
10
β
Node 17
Node 15
Node 13
Node 11
Node 9
Node 2
Node 4
Node 6

Making It Interpretable

- Our branch-and-bound-based proof essentially consists of a collection of dual proofs for convex problems.
- I would argue that disjunctive arguments are natural for humans.
- We're used to discrete choice—I can have this or that or that.
- The main cognitive challenge seems to be that we may have a high number of terms.
- Once the problem is solved, a secondary phase could be added in which the proof is made as short as possible.
- Based on the structure of the value function, it seems intuitive that relatively short proofs should be possible with some additional work.
- It is easy to imagine generating "approximate proofs" with a smaller number of terms that are easier for humans to interpret.

Generalizing

- How do we generalize these concepts?
- The idea of dual proofs based on separating TRUE theorems from FALSE ones can be generalized to any parametric class of problems.
  - Note that multilevel integer linear optimization constitute a set of complete problems spanning every level of the PH.
  - All such problems have piecewise polyhedral, lower semi-continuous value functions.
  - This seems to indicate that separation, while certainly difficult, essentially only requires one (very complex) hypersurface.
- We can also generalize the notion of convexifying sets of possible proofs in order to more quickly prune infeasible search paths.

Inverse Problems and Machine Learning

- Given any parametric class of decision problems, we can consider the boundary between sets of parameter values that yield *TRUE* and *FALSE*.
- In the case of an LP or MILP, the parameters are right-hand side values and the value function defines the boundary.
- The optimization problem can then be interpreted as: given a right-hand side, determine the corresponding point on the boundary.
- An *inverse optimization problem* inverts this process: given one (or more) points on the boundary, we aim to construct the instance.
- Classification problems in machine learning can be thought of very roughly as inverse problems where we are given *many* points.
- The goal is still to construct a boundary that "separates" them.

Where Do we Go From Here?

- Hilbert's 24th problem, which he ended up removing from his famous list anticipated the current dialogue over explainability.
- The question he asked was about how to produce the "simplest" proof of a given mathematical result.
- When it comes to problems in NP for which many alternative proofs exist, asking whether there is a simpler one is a natural question.
- The material presented in this talk provides a framework to think about this, although the precise path may not be clear.
- I hope to have a chance to engage some of you these ideas down the road.