Ted Ralphs\textsuperscript{1}

Joint work with Aykut Bulut\textsuperscript{1}, Andrea Lodi\textsuperscript{2}, Gerhard Woeginger\textsuperscript{3}

\textsuperscript{1}COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University
\textsuperscript{2}DEI, University of Bologna
\textsuperscript{3}Eindhoven University of Technology

INFORMS Annual Meeting, Minneapolis, MN, 7 October 2013
Outline

1 Introduction
2 Complexity
3 Algorithms
4 Final Remarks
Our goal is to analyze certain *finite extensive-form games*, which are sequential games involving $n$ players.

**Loose Definition**

- The game is specified on a tree with each node corresponding to a move and the outgoing arcs specifying possible choices.
- The leaves of the tree have associated payoffs.
- Each player’s goal is to maximize payoff.
- There may be *chance* players who play randomly according to a probability distribution and do not have payoffs (*stochastic games*).

- All players are rational and have perfect information.
- The problem faced by a player in determining the next move is a *multilevel/multistage* optimization problem.
- The move must be determined by taking into account the *responses of the other players*. 
Example Game Tree
Multilevel and Multistage Games

- We use the term *multilevel* for competitive games in which there is no chance player.
- We use the term *multistage* for cooperative games in which all players receive the same payoff, but there are chance players.
- A *subgame* is the part of a game that remains after some moves have been made.

**Stackelberg Game**
- A Stackelberg game is a game with two players who make one move each.
- The goal is to find a *subgame perfect Nash equilibrium*, i.e., the move by each player that ensures that player’s best outcome.

**Recourse Game**
- A cooperative game in which play alternates between cooperating players and chance players.
- The goal is to find a *subgame perfect Markov equilibrium*, i.e., the move that ensures the best outcome in a probabilistic sense.
A standard mathematical program models a (set of) decision(s) to be made *simultaneously* by a *single* decision-maker (i.e., with a *single* objective).

Decision problems arising in sequential games and other real-world applications involve

- multiple, independent decision-makers (DMs),
- sequential/multi-stage decision processes, and/or
- multiple, possibly conflicting objectives.

Modeling frameworks

- Multiobjective Programming $\iff$ multiple objectives, single DM
- Mathematical Programming with Recourse $\iff$ multiple stages, single DM
- Multilevel Programming $\iff$ multiple stages, multiple objectives, multiple DMs

*Multilevel programming* generalizes standard mathematical programming by modeling hierarchical decision problems, such as finite extensive-form games.

Such models arises in a *remarkably wide array of applications.*
A canonical extensive-form game that illustrates many of the basic principles 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.
A Simple Example

- This diagram illustrates the search for solutions to the problem as a tree.
- The nodes in green represent settings of the truth values that satisfy all the given clauses; red represents non-satisfying truth values.
  - With one player, the solution is any path to one of the green nodes.
  - With two players, the solution is a subtree in which there are no red nodes.
- The latter requires knowledge of *all* leaf nodes (important!).

\[
\begin{align*}
& C_1 = x_1 | x_2 \\
& C_2 = x_2 | x_3 \\
& C_1 = \text{TRUE} \\
& C_2 = x_2 \\
& C_1 = \text{TRUE} \\
& C_2 = x_3 \\
& C_1 = \text{TRUE} \\
& C_2 = \text{TRUE} \\
& C_1 = \text{TRUE} \\
& C_2 = \text{FALSE} \\
& C_1 = \text{FALSE} \\
& C_2 = x_3 \\
& C_1 = \text{FALSE} \\
& C_2 = x_3 \\
& C_1 = \text{TRUE} \\
& C_2 = \text{FALSE} \\
& C_1 = \text{FALSE} \\
& C_2 = \text{TRUE} \\
& C_1 = \text{TRUE} \\
& C_2 = \text{TRUE} \\
\end{align*}
\]
More Formally

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

For $k$ odd, the SAT game can be formulated as

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

(1)

For even $k$, we have

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

(2)

A more general form of this problem, known as the quantified Boolean formula problem (QBF) allows an arbitrary sequence of quantifiers.
For $k = 1$, SAT can be formulated as the (feasibility) integer program

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

(SAT) can be formulated as the optimization problem

$$\max_{x \in \{0,1\}^n} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i)$$

subject to

$$\sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \ \forall j \in J \setminus \{0\}$$

For $k = 2$, we then have

$$\max_{x_1 \in \{0,1\}^{l_1}} \min_{x_2 \in \{0,1\}^{l_2}} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i)$$

subject to

$$\sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \ \forall j \in J \setminus \{0\}$$
Consider the earlier example of the SAT game, now as an optimization problem.

In the one player version, the goal is simply to maximize payoff.

The two player game is zero-sum with the first player attempting to maximize while the second player attempts to minimize.

The complexity of the two-player game comes from the requirement to account for the payoff at all leaf nodes.
How Difficult is the SAT Game?

- Fundamentally, we would like to know how difficult it is to solve player one’s decision problem.
- It is well-known that the (single player) satisfiability problem is in the complexity class $NP$-complete.
- It is perhaps to be expected that the $k$-player satisfiability game is in a different class.
  - The $k^{th}$ player to move is faced with a satisfiability problem.
  - The $(k - 1)^{th}$ player is faced with a 2-player subgame in which she must take into account the move of the $k^{th}$ player.
  - And so on . . .
- Each player’s decision problem appears to be exponentially more difficult than the succeeding player’s problem.
- This complexity is captured formally in the hierarchy of complexity classes known as the *polynomial time hierarchy*. 
The formal complexity framework traditionally employed in discrete optimization applies to decision problems (Garey and Johnson, 1979).

The formal model of computation is a deterministic Turing machine (DTM).

The possible execution paths of a DTM can be thought of as forming a tree.

For problems that are efficiently solvable, we know how to construct an execution path that is guaranteed to end in an accepting state.

For more difficult problems, some enumeration is needed.

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.
Back to SAT

\[
\begin{align*}
\text{max } & x_1 + x_2 \geq 1 \\
\text{s.t. } & x_2 + x_3 \geq 1
\end{align*}
\]

\[
\begin{align*}
\text{max } & 1 + x_2 \\
\text{s.t. } & x_2 + x_3 \geq 1
\end{align*}
\]
The polynomial hierarchy is a scheme for classifying multi-level and multi-stage decision problems. We have

$$\Delta^p_0 := \Sigma^p_0 := \Pi^p_0 := P,$$

(3)

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

$$\Delta^p_{k+1} := P^{\Sigma^p_k},$$

$$\Sigma^p_{k+1} := NP^{\Sigma^p_k}, \text{ and}$$

$$\Pi^p_{k+1} := coNP^{\Sigma^p_k}.$$

$PH$ is the union of all levels of the hierarchy.
The satisfiability games with $k$ players is complete for $\Sigma^p_k$.

For the corresponding $k$-level optimization problem, the optimal value is one 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 $\geq 1$?

Thus, the (the decision version of) $k$-level mixed integer programming is also complete for $\Sigma^p_k$.

By swapping the “min” and the “max,” we can get a similar decision problem that is complete for $\Pi^p_k$.

$$\min_{x_{N_1} \in \{0,1\}^{N_1}} \max_{x_{N_2} \in \{0,1\}^{N_2}} \sum_{i \in C_0^0} x_i + \sum_{i \in C_1^l} (1 - x_i)$$

s.t. $\sum_{i \in C_0^0} x_i + \sum_{i \in C_1^l} (1 - x_i) \geq 1 \forall j \in J \setminus \{0\}$

The question remains whether the optimal value is $\geq 1$, but now we are asking it with respect to a minimization problem.
Outline

1 Introduction

2 Complexity

3 Algorithms

4 Final Remarks
Formally, a *bilevel linear program* is described as follows.

- \( x \in X \subseteq \mathbb{R}^{n_1} \) are the *upper-level variables*
- \( y \in Y \subseteq \mathbb{R}^{n_2} \) are the *lower-level variables*

### Bilevel (Integer) Linear Program

\[
\max \{ c^1 x + d^1 y \mid x \in \mathcal{P}_U \cap X, \ y \in \arg\min \{ d^2 y \mid y \in \mathcal{P}_L(x) \cap Y \} \} \quad \text{(MIBLP)}
\]

The *upper- and lower-level feasible regions* are:

\[
\mathcal{P}_U = \{ x \in \mathbb{R}_+ \mid A^1 x \leq b^1 \} \quad \text{and} \\
\mathcal{P}_L(x) = \{ y \in \mathbb{R}_+ \mid G^2 y \geq b^2 - A^2 x \}.
\]

We consider the general case in which \( X = \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1-p_1} \) and \( Y = \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2-p_2} \). This basic model can be extended to multiple levels.
If $d^1 = -d^2$, we can view this as a *mathematical program with recourse*. We can reformulate the bilevel program as follows.

\[
\min \{-c^1 x + Q(x) \mid x \in \mathcal{P}_U \cap X\},
\]

(4)

where

\[
Q(x) = \min \{d^1 y \mid y \in \mathcal{P}_L(x) \cap Y\}.
\]

(5)

The function $Q$ is known as the *value function* of the recourse problem.
Two-Stage Stochastic Programs with Recourse

For most of the remainder of the talk, we consider the two-stage stochastic mixed integer program

$$\min \{ c^1 x + \mathbb{E}_\xi Q_\xi(x) \mid x \in \mathcal{P}_U \cap \mathcal{X} \}, \quad (6)$$

where

$$Q_\xi(x) = \min \{ d^2 y \mid y \in \mathcal{Y}, G^2 y \geq \omega(\xi) - A^2 x \}, \quad (7)$$

$\xi$ is a random variable from a probability space $(\Xi, \mathcal{F}, \mathbb{P})$, and for each $\xi \in \Xi$, $\omega(\xi) \in \mathbb{R}^{m_2}$.

If the distribution of $\xi$ is discrete and has finite support, then (6) is a bilevel program.
Benders’ Principle (Linear Programming)

\[ z_{LP} = \min_{(x,y) \in \mathbb{R}^n} \{ c'x + c''y \mid A'x + A''y \geq b \} \]

\[ = \min_{x \in \mathbb{R}^n} \{ c'x + \phi(b - A'x) \}, \]

where

\[ \phi(d) = \min c''y \]

\[ \text{s.t. } A''y \geq d \]

\[ y \in \mathbb{R}^{n''} \]

**Basic Strategy:**

- The function \( \phi \) is the *value function* of a linear program.
- The value function is piecewise linear and convex.
- We iteratively generate a lower approximation by sampling the domain.
Benders’ Principle (Integer Programming)

\[ z_{IP} = \min_{(x,y) \in \mathbb{Z}^n} \{ c'x + c''y \mid A'x + A''y \geq b \} \]

\[ = \min_{x \in \mathbb{R}^n} \{ c'x + \phi(b - A'x) \}, \]

where
\[ \phi(d) = \min c''y \quad \text{s.t.} \quad A''y \geq d \]
\[ y \in \mathbb{Z}^{n''} \]

**Basic Strategy:**

- Here, \( \phi \) is the value function of an integer program.
- In the general case, the function \( \phi \) is piecewise linear but not convex.
- Here, we also iteratively generate a lower approximation by evaluating \( \phi \).
Example: MILP Value Function

The value function of a MILP is non-convex and discontinuous piecewise polyhedral.

Example

\[ \phi(d) = \min 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 + 5x_6 \]

s.t. \[ 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 + x_6 = d \]

\[ x_1, x_2, x_3 \in \mathbb{Z}_+, \quad x_4, x_5, x_6 \in \mathbb{R}_+ \]
Example: MILP Value Function

Example

\[ \phi(b) = \min x_1 - \frac{3}{4} x_2 + \frac{3}{4} x_3 \]
\[ \text{s.t. } \frac{5}{4} x_1 - x_2 + \frac{1}{2} x_3 = b \]
\[ x_1, x_2 \in \mathbb{Z}^+, x_3 \in \mathbb{R}^+ \]

(Ex2.MILP)
Approximating the Value Function

- In general, it is difficult to construct the value function explicitly.
- We therefore propose to approximate the value function by either upper or lower bounding functions.

**Lower bounds**
Derived by considering the value function of *relaxations* of the original problem or by constructing *dual functions* $\Rightarrow$ Relax constraints.

**Upper bounds**
Derived by considering the value function of *restrictions* of the original problem $\Rightarrow$ Fix variables.
A dual function $\varphi : \mathbb{R}^m \rightarrow \mathbb{R} \cup \{\pm \infty\}$ is a function such that

$$\varphi(b) \leq \phi(b) \quad \forall b \in \Lambda$$

For a particular value of $\hat{b}$, the dual problem is

$$\phi_D = \max \{ \varphi(\hat{b}) : \varphi(b) \leq \phi(b) \quad \forall b \in \mathbb{R}^m, \varphi : \mathbb{R}^m \rightarrow \mathbb{R} \cup \{\pm \infty\} \}$$
The algorithmic framework we utilize builds on a number of previous works.

- **Modification to the L-shaped framework** (Laporte and Louveaux, 1993; Carøe and Tind, 1998; Sen and Higle, 2005)
  - Linear cuts in first stage for binary first stage
  - Optimality cuts from B&B and cutting plane, applied to pure integer second stage
  - Disjunctive programming approaches and cuts in the second stage

- **Value function approaches**: Pure integer case (Ahmed et al., 2004; Kong et al., 2006)

- **Scenario decomposition** (Carøe and Schultz, 1998)

- **Enumeration/Gröbner basis reduction** (Schultz et al., 1998)
MILP Duals from Branch-and-Bound

Figure: Dual Functions from B&B for right hand sides 1, 2.125, 3.5
Example
The general case is much more difficult because we need the *solution* to the lower-level problem, not just the *value*.

Algorithms must involve some kind of relaxation of the problem.

Relaxations are inherently weak.

Some progress has been made, but incorporating knowledge of the value function into the relaxation has proven exceptionally challenging.
Conclusions

- This has been a high level overview of the complexity of this broad class of problems.
- The complexity of these problems is closely related to the complexity of more traditional types of problems.
- Understanding these issues can provide important insight and lead to practical algorithms.
- This a wide open area and there is much more to be done.
- See talks by Anahita Hassanzadeh and Aykut for more details.


