(Conic) Linear Optimization: Problem Instances II

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

http://www.stanford.edu/~yyye

LY 5th, Chapter 1, Chapter 2.1-2.2
# Prediction Market I: World Cup Information Market

<table>
<thead>
<tr>
<th>Order:</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Brazil</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Italy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Germany</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>France</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bidding Prize: $\pi$</th>
<th>0.75</th>
<th>0.35</th>
<th>0.4</th>
<th>0.95</th>
<th>0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity limit: $q$</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Order fill: $x$</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
</tr>
</tbody>
</table>
Prediction Market II: Call Auction Mechanism

Given $m$ potential states that are mutually exclusive and exactly one of them will be realized at the maturity.

An order is a bet on one or a combination of states, with a price limit (the maximum price the participant is willing to pay for one unit of the order) and a quantity limit (the maximum number of units or shares the participant is willing to accept).

A contract on an order is a paper agreement so that on maturity it is worth a notional $1$ dollar if the order includes the winning state and worth $0$ otherwise.

There are $n$ orders submitted now.
The $i$th order is given as $(\mathbf{a}_i, \pi_i, q_i) \in R_+^m \times R_+ \times R_+): \mathbf{a}_i$ is the betting indication row vector where each component is either 1 or 0

$$\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{im})$$

where 1 is winning state and 0 is non-winning state; $\pi_i$ is the price limit for one unit of such a contract, and $q_i$ is the maximum number of contract units the better like to buy.
Let $x_i$ be the number of units or shares awarded to the $i$th order. Then, the $i$th bidder will pay the amount $\pi_i \cdot x_i$ and the total amount collected would be $\pi^T x = \sum_i \pi_i \cdot x_i$.

If the $j$th state is the winning state, then the auction organizer needs to pay the winning bidders

$$\left( \sum_{i=1}^{n} a_{ij} x_i \right) = a_{.j}^T x$$

where column vector

$$a_{.j} = (a_{1j}; a_{2j}; \ldots; a_{nj})$$

The question is, how to decide $x \in \mathbb{R}^n$, that is, how to fill the orders.
Prediction Market V: Worst-Case Profit Maximization

\[
\begin{align*}
& \text{max } \pi^T x - \max_j \{a_j^T x\} \\
& \text{s.t. } \quad \quad x \leq q, \\
& \quad \quad x \geq 0.
\end{align*}
\]

\[
\begin{align*}
& \text{max } \pi^T x - \max( A^T x ) \\
& \text{s.t. } \quad \quad x \leq q, \\
& \quad \quad x \geq 0.
\end{align*}
\]

This is \textbf{NOT} a linear program.
However, the problem can be rewritten as

$$\max \quad \pi^T x - y$$

s.t. \quad A^T x - e \cdot y \leq 0,

\quad x \leq q,

\quad x \geq 0,$$

where $e$ is the vector of all ones. This is a linear program.

$$\max \quad \pi^T x - y$$

s.t. \quad A^T x - e \cdot y + s_0 = 0,$$

\quad x + s = q,$$

\quad (x, s_0, s) \geq 0, \quad y \text{ free},$$
This is the Max-Cut problem on an undirected graph $G = (V, E)$ with non-negative weights $w_{ij}$ for each edge in $E$ (and $w_{ij} = 0$ if $(i, j) \notin E$), which is the problem of partitioning the nodes of $V$ into two sets $S$ and $V \setminus S$ so that

$$w(S) := \sum_{i \in S, j \in V \setminus S} w_{ij}$$

is maximized. A problem of this type arises from many network planning, circuit design, and scheduling applications.
Figure 1: Illustration of the Max-Cut Problem
Max-Cut Formulation

\[
 w^* := \text{Maximize} \quad w(x) := \frac{1}{4} \sum_{i,j} w_{ij} (1 - x_i x_j) \\
(MC)
\]

Subject to \( (x_j)^2 = 1, \ j = 1, \ldots, n. \)
Semidefinite Relaxation for (MC)

\[
\begin{align*}
    z^{SDP} := & \quad \text{Maximize} \quad \frac{1}{4} \sum_{i,j} w_{ij}(1 - X_{ij}) \\
    \text{Subject to} & \quad X_{ii} = 1, \quad i = 1, \ldots, n, \\
    & \quad X \succeq 0.
\end{align*}
\]

When \( X \) constrained to be rank-one or \( X = xx^T \), the SDP formulation is equivalent to the original problem.

Let \( \bar{X} \) be an optimal solution for (SDP). Then, generate a random vector \( u \in N(0, \bar{X}) \):

\[
\hat{x} = \text{Sign}(u), \quad E[\hat{x}_i \hat{x}_j] = \arcsin(\bar{X}_{ij})
\]

**Theorem 1** *(Goemans and Williamson)*

\[
E[w(\hat{x})] \geq 0.878z^{SDP} \geq 0.878w^*.
\]
Max-Bisection Formulation

\[ w^* := \text{Maximize} \quad w(x) := \frac{1}{4} \sum_{i,j} w_{ij} (1 - x_i x_j) \]

\begin{align*}
\text{(MB)} \\
\text{Subject to} \quad (x_i)^2 &= 1, \quad i = 1, \ldots, n, \\
\sum_{i=1}^{n} x_i &= 0.
\end{align*}

What complicates matters in Max-Bisection, comparing to Max-Cut, is that two objectives are actually sought—the objective value of \( w(x) \) and the size balance \( \sum_i x_i \). Therefore, in any (randomized) rounding method at the beginning, we need to balance the (expected) quality of \( w(\hat{x}) \) and the (expected) size balance of \( \sum_i \hat{x}_i \).
Semidefinite Relaxation for (MB)

\[ z^{SDP} := \text{Maximize } \frac{1}{4} \sum_{i,j} w_{ij} (1 - X_{ij}) \]

Subject to

\[ X_{ii} = 1, \quad i = 1, \ldots, n, \]

\[ \sum_{i,j} X_{ij} = 0, \]

\[ X \succeq 0. \]

**Theorem 2** (Y 1994) There is a randomized algorithm that generates a bisection solution \( \hat{x} \) from the SDP relaxation such that

\[ E[w(\hat{x})] \geq 0.699 z^{SDP} \geq 0.699 w^*. \]
Given a graph \( G = (V, E) \) and sets of non-negative weights, say \( \{d_{ij} : (i, j) \in E\} \), the goal is to compute a realization of \( G \) in the Euclidean space \( \mathbb{R}^d \) for a given low dimension \( d \), where the distance information is preserved.

More precisely: given anchors \( a_k \in \mathbb{R}^d \), \( d_{ij} \in N_x \), and \( \hat{d}_{kj} \in N_a \), find \( x_i \in \mathbb{R}^d \) such that

\[
\|x_i - x_j\|^2 = d_{ij}^2, \quad \forall (i, j) \in N_x, \quad i < j,
\]

\[
\|a_k - x_j\|^2 = \hat{d}_{kj}^2, \quad \forall (k, j) \in N_a.
\]

This is a set of Quadratic Equations, which can be represented as an optimization problem:

\[
\min_{x_i \forall i} \sum_{(i, j) \in N_x} (\|x_i - x_j\|^2 - d_{ij}^2)^2 + \sum_{(k, j) \in N_a} (\|a_k - x_j\|^2 - \hat{d}_{kj}^2)^2.
\]

Does the system have a localization or realization of all \( x_j \)'s? Is the localization unique? Is there a certification for the solution to make it reliable or trustworthy? Is the system partially localizable with a certification?

It can be relaxed to SOCP (change “=” to “\( \leq \)”) or SDP.
Figure 2: 50-node 2-D Sensor Localization.
Matrix Representation of SNL and SDP Relaxation

Let \( X = [x_1 \ x_2 \ldots \ x_n] \) be the \( d \times n \) matrix that needs to be determined and \( e_j \) be the vector of all zero except 1 at the \( j \)th position. Then

\[
x_i - x_j = X(e_i - e_j) \quad \text{and} \quad a_k - x_j = [I \ X](a_k; -e_j)
\]

so that

\[
\|x_i - x_j\|^2 = (e_i - e_j)^TX^TX(e_i - e_j)
\]

\[
\|a_k - x_j\|^2 = (a_k; -e_j)^T[I \ X]^T[I \ X](a_k; -e_j) =
\]

\[
(a_k; -e_j)^T \begin{pmatrix}
I & X \\
X^T & X^TX
\end{pmatrix} (a_k; -e_j).
\]
Or, equivalently,

\[
(e_i - e_j)^T Y (e_i - e_j) = d_{ij}^2, \forall i, j \in N_x, i < j,
\]

\[
(a_k; -e_j)^T \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} (a_k; -e_j) = \hat{d}_{kj}^2, \forall k, j \in N_a,
\]

\[Y = X^T X.\]

Relax \(Y = X^T X\) to \(Y \succeq X^T X\), which is equivalent to matrix inequality:

\[
\begin{pmatrix} I & X \\ X^T & Y \end{pmatrix} \succeq 0.
\]

This matrix has rank at least \(d\); if it’s \(d\), then \(Y = X^T X\), and the converse is also true.

The problem is now an SDP problem: when the SDP relaxation is exact?

Algorithm: Convex relaxation first and steepest-descent-search second strategy?
Reinforcement Learning: Markov Decision/Game Process

- RL/MDPs provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.

- Markov Game Processes (MGPs) provide a mathematical framework for modeling sequential decision-making of two-person turn-based zero-sum game.

- MDGPs are useful for studying a wide range of optimization/game problems solved via dynamic programming, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).

- Modern applications include dynamic planning under uncertainty, reinforcement learning, social networking, and almost all other stochastic dynamic/sequential decision/game problems in Mathematical, Physical, Management and Social Sciences.
MDP Stationary Policy and Cost-to-Go Value

- An MDP problem is defined by a given number of states, indexed by $i$, where each state has a set of actions, denoted by $\mathcal{A}_i$, to take. Each action, say $j \in \mathcal{A}_i$, is associated with an (immediate) cost $c_j$ of taking, and a probability distribution $p_j$ to transfer to all possible states at the next time period.

- A stationary policy for the decision maker is a function $\pi = \{\pi_1, \pi_2, \cdots, \pi_m\}$ that specifies an action in each state, $\pi_i \in \mathcal{A}_i$, that the decision maker will take at any time period; which also lead to an expected cost-to-go value for each state: the total expected cost over all time periods if the process starts from state $i$ and follows the policy.

- The MDP is to find a stationary policy to minimize/maximize the expected (discounted) sum over the infinite horizon with a discount factor $0 \leq \gamma < 1$:

$$
\sum_{t=0}^{\infty} \gamma^t E[c^{\pi_i(t)}(i^{t}, i^{t+1})].
$$

- If the states are partitioned into two sets, one is to minimize and the other is to maximize the discounted sum, then the process becomes a two-person turn-based zero-sum stochastic game.
Actions are in red, blue and black; and all actions have zero cost except the state 4 to the exit/termination state 5. Which actions to take from every state to minimize the total cost (called optimal policy)?
States \( \{0, 1, 2, 5\} \) minimize, while States \( \{3, 4\} \) maximize.
Cost-to-go values on each state when actions in red are taken: the current policy is not optimal since there are better actions to choose to minimize the cost.
The Cost-to-Go Value in General

\[ y_i = c_j + p^T_j y; \text{ when } j \in A_i \text{ action is taken.} \]
The Optimal Cost-to-Go Value Vector

Let \( y \in \mathbb{R}^m \) represent the cost-to-go values of the \( m \) states, \( i \)th entry for \( i \)th state, of a given policy.

The MDP problem entails choosing an optimal policy where the corresponding cost-to-go value vector \( y^* \) satisfying:

\[
y^*_i = \min \{ c_j + \gamma p_j^T y^*, \forall j \in A_i \}, \forall i,
\]

with optimal policy

\[
\pi^*_i = \arg \min \{ c_j + \gamma p_j^T y^*, \forall j \in A_i \}, \forall i.
\]

In the Game setting, the conditions becomes:

\[
y^*_i = \min \{ c_j + \gamma p_j^T y^*, \forall j \in A_i \}, \forall i \in I^-,
\]

and

\[
y^*_i = \max \{ c_j + \gamma p_j^T y^*, \forall j \in A_i \}, \forall i \in I^+.
\]

They both are fix-point or saddle-point optimization problems. The MDP problem can be cast as a linear program; see next page.
The Fixed-Point formulation:

\[ y_0 = \min\{0 + \gamma y_1, 0 + \gamma(0.5y_2 + 0.25y_3 + 0.125y_4 + 0.125y_5)\} \]
\[ y_1 = \min\{0 + \gamma y_2, 0 + \gamma(0.5y_3 + 0.25y_4 + 0.25y_5)\} \]
\[ y_2 = \min\{0 + \gamma y_3, 0 + \gamma(0.5y_4 + 0.5y_5)\} \]
\[ y_3 = \min\{0 + \gamma y_4, 0 + \gamma y_5\} \]
\[ y_4 = 1 + \gamma y_5 \]
\[ y_5 = 0 \text{ (or } y_5 = 0 + \gamma y_5) \]

The LP formulation:

\[
\text{maximize } y_0 + y_1 + y_2 + y_3 + y_4 + y_5
\]

subject to change each equality above into inequality
In general, the fixed-point model can be reformulated as an LP:

\[
\text{maximize}_y \quad \sum_{i=1}^{m} y_i \\
\text{subject to} \quad y_1 - \gamma p_j^T y \leq c_j, \ j \in A_1 \\
\quad \vdots \\
\quad y_i - \gamma p_j^T y \leq c_j, \ j \in A_i \\
\quad \vdots \\
\quad y_m - \gamma p_j^T y \leq c_j, \ j \in A_m.
\]

**Theorem 3** When \( y \) is maximized, there must be at least one inequality constraint in \( A_i \) that becomes equal for every state \( i \), that is, maximal \( y \) is a fixed point solution.
The LP variables \( y \in \mathbb{R}^m \) represent the expected present cost-to-go values of the \( m \) states, respectively, for a given policy.

The LP problem entails choosing variables in \( y \), one for each state \( i \), that maximize \( e^T y \) so that it is the fixed point

\[
y_i^* = \min_{j \in A_i} \{ c_{ji} + \gamma p_{ji}^T y \}, \forall i,
\]

with an optimal policy

\[
\pi_i^* = \arg \min \{ c_j + \gamma p_{ji}^T y, j \in A_i \}, \forall i.
\]

It is well known that there exist a unique optimal stationary policy value vector \( y^* \) where, for each state \( i \), \( y_i^* \) is the minimum expected present cost that an individual in state \( i \) and its progeny can incur.
States/Actions in the Tic-Tac-Toe Game

The diagram illustrates various states and actions in a Tic-Tac-Toe game. Each state is represented by a grid and is connected to another state through arrows, indicating possible moves. The probabilities of transitioning from one state to another are marked as 1/3. The diagram shows a sequence of states and actions, highlighting the game's possible paths and outcomes.
Action Costs in the Tic-Tac-Toe Game

Any action leading to win has cost -1
Any action leading to lose has cost 1