Solving Games

Linear programming applied

This Python code shows how to use library SciPy, module optimize in order to solve zero sum game with payoff matrix

$$A = \begin{pmatrix} 1 & -1 & 2 \\ -1 & 2 & 0 \\ 1 & 0 & -1 \end{pmatrix}.$$

```
from scipy.optimize import linprog

# Coeficients of the function that we want minimize
# that is v=0y_1+0y_2+0y_3+1v
c = [0,0,0,1]

# Inequality constraint constants (as many rows as A)
b_ub = [0,0,0]

# Payoff matrix extended where first 3 collumns is payoff and remaining is -
A_ub = [[1, -1, 2,-1],
        [-1, 2, 0,-1],
        [1, 0, -1,-1]]

SumIsOne=[[1,1,1,0]]
One=[1]

# Variable bounds - all variables are positive except the last one,
# which is v
bounds = [(0, None), (0, None),(0, None), (None, None)]

# Solve the linear program that minimizes v
result2=linprog(c,A_ub=A_ub,b_ub=b_ub,A_eq=SumIsOne,b_eq=One,bounds=bounds,
                                                    method='highs')
print("Value:", result2.fun)
print("Optimal␣strategy␣of␣the␣second␣player:", result2.x[:-1])
```

- **Objective Coefficients** `c`: The vector `c = [0, 0, 0, 1]` represents the coefficients of the function we want to minimize, which is

$$v = 0y_1 + 0y_2 + 0y_3 + 1v$$

.

- **Inequality Constraints**: We minimize v subject to the constrains

$$y_1 - y_2 + 2y_3 - v \leq 0, -y_1 + 2y_2 - v \leq 0, y_1 - y_3 - v \leq 0.$$

From here we know that `A_ub` is

$$\begin{pmatrix} 1 & -1 & 2 & -1 \\ -1 & 2 & 0 & -1 \\ 1 & 0 & -1 & -1 \end{pmatrix}.$$

and the vector `b_ub` is $[0, 0, 0]$.

- **Equality Constraint**: There is only one equality constraint

$$y_1 + y_2 + y_3 = 1$$

which means that the corresponding coefficient matrix `SumIsOne` is $[1, 1, 1]$ and vector `One` responsible for right hand side coefficients is $[1]$.

- **Variable Bounds** `bounds`: This list sets the bounds for each variable. Here, only y_1, y_2, and y_3 are constrained to be non-negative, while v has no restrictions, hence

```
bounds = [(0, None),(0, None),(0, None),(None, None)].
```

- **Solving the Linear Program**: The function `linprog` is used with the method `'highs'` to solve the linear program, minimizing v subject to the given constraints. It is the most versatile method. You can use also `'simplex'` or `'interior-point`.

- **Output**: If the optimization is successful, the program prints the game value v (stored in `result2.fun`) and the optimal strategy of the second player (all elements of `result2.x` except the last one). As far as I am concerned, it is impossible to obtain using `linprog` the solution of the dual problem, hence in order to obtain optimal strategy for the first player one has to formulate the problem as dual explicitly.