INF421 PI 2020: Nash Equilibrium in a Discrete Colonel Blotto Game

Patrick Loiseau

patrick.loiseau@inria.fr

Final version – November 30, 2020

In this project, we implement a number of naive and refined solutions to compute best responses and Nash equilibrium strategies in a discrete Colonel Blotto game; the final goal being to find the Nash equilibrium in polynomial time. This builds on different techniques, in particular dynamic programming, linear programming, and graph algorithms.

Preambule on the project's rules: The project can be done in Python or Java. If you wish to use another language, please consult with me first. Students must do at least one of Sections 3.2 and 4 but can choose which one; doing both is not mandatory but will be appreciated. All other sections are mandatory. Note that throughout the project, different questions may carry unequal amounts of work.

The Colonel Blotto game is a famous resource allocation game. It was first introduced by Borel in 1921 [1] and spurred a lot of research since then. In the Colonel Blotto game, two players allocate a finite budget of resources to several battlefields and each battlefield is won by the player who allocated more resources to it. Each player seeks to maximize the gains from the battlefields they win (see a rigorous description below). This game has been used to model a variety of strategic resource allocation problems such as allocation of advertisement expenditures or lobbying resources, R&D contests, election and political contests, or security resource allocation.

In this project, we consider a discrete version of the Colonel Blotto game where resources are indivisible troops, which imposes that allocations must be integer. The game is therefore a finite game (i.e., each player has a finite number of possible strategies) and it is known since 1951 that such games always have a Nash equilibrium [3]. Moreover, the Colonel Blotto game is a zero-sum game (at least in some formulations) and it is known that for such games, finding a Nash equilibrium is equivalent to the minmax property [4], which can be expressed by a linear program and solved in time polynomial in the size of the strategy space. The Colonel Blotto game, however, has a strategy space which is itself of size exponential in the natural parameters of the game. The goal of the project is to explore algorithms to compute best-response and Nash equilibrium strategies of this game, from naive solutions to refined solutions that achieve polynomial complexity in the natural parameters of the game.

1 Introduction and Definitions

The Colonel Blotto game is a game between two players, A and B. There are n battlefields, each endowed with a value $v_i, i \in \{1, \cdots, n\}$. Each player has a budget of troops denoted $X \in \mathbb{N}$ and $Y \in \mathbb{N}$ respectively for Player A and B. A pure strategy x of Player A is an allocation of their X troops to the n battlefields, that is $x \in \mathbb{N}^n$ is an integer vector such that $\sum_{i=1}^n x_i = X$. We denote the set of all pure strategies of Player A by $\mathcal{X} = \{x \in \mathbb{N}^n : \sum_{i=1}^n x_i = X\}$. Similarly, we denote by y pure strategies of Player B, and by $\mathcal{Y} = \{y \in \mathbb{N}^n : \sum_{i=1}^n y_i = Y\}$ their set.

We now define the payoffs[1] of the players as a function of the strategy of both.[2] In any given battlefield $i \in \{1, \cdots, n\}$, Player A wins the battlefields value v_i if they allocated more resource to it than Player B, and loses the value v_i if they allocated less resource than Player B; in case of a tie, they win 0. Denoting by $u_i(x_i, y_i)$ the payoff of Player A in battlefield i we then have:

$$u_i(x_i, y_i) = \begin{cases} v_i & \text{if } x_i > y_i, \\ -v_i & \text{if } x_i < y_i, \\ 0 & \text{if } x_i = y_i. \end{cases}$$

The payoff of Player B is simply the opposite of that of Player A in each battlefield, that is $-u_i(x_i, y_i)$. Finally, the total payoffs of Players A and B are the sum of their payoff in each battlefield; that is respectively $U(x, y) = \sum_{i=1}^n u_i(x_i, y_i)$ and $-U(x, y)$.

A mixed strategy for a given player is a probability distribution over their set of pure strategies. We denote by p and q respectively mixed strategies of Player A and B, that is p_x will denote the probability assigned to pure strategy $x \in \mathcal{X}$. We denote by $\mathcal{P} = \Delta(\mathcal{X})$ the set of probability distributions over $\mathcal{X}$, that is the set of mixed strategies of Player A, and by $\mathcal{Q}$ the set of mixed strategies of Player B. The players' payoffs under mixed strategies are simply the bilinear extension of their payoff under pure strategies, that is, for all $p \in \mathcal{P}$ and $q \in \mathcal{Q}$,

$$U(p, q) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p_x q_y U(x, y).$$

2 Best responses in the Colonel Blotto game

Question 0: What is the number of pure strategies of each player? Show that, for any profile of mixed strategies (p, q), the payoffs only depend on the marginal allocation of each player on each battlefield.

In this section, we explore algorithms to compute the best response of Player A to a fixed strategy of Player B. A strategy p^* of Player A is a best response to strategy q^* of Player B if it maximizes Player A's utility:

$$p^* \in \arg\max_p U(p, q^*).$$

2.1 Best response through brute force

Question 1: Show that there always exists a best response that is a pure strategy.

Question 2: Propose a brute-force algorithm to compute a best response. Give the pseudo-code and implement the algorithm. What is its complexity?

[1] also equivalently called utility, or objective function.

[2] Note that this is the distinctive feature of a *game* compared to an optimization problem: the payoff of a player depends not only on their action but also on the action of the other player(s).

2.2 Best response through dynamic programming

In fact, due to the specific structure of our problem, it is possible to find better algorithms to find a best response. One approach is through dynamic programming (DP), by considering subproblems that correspond to the allocation to a subset of battlefields.

Question 3: Propose a DP algorithm for computing the best response. Give the pseudo-code and implement the algorithm. What is its complexity?

2.3 Best response through shortest path

Another approach is to find a graph that encodes the possible strategies of a player.

Question 4: Find a graph representation of the strategies such that allocations to individual battlefields correspond to edges and strategies correspond to paths from a source vertex to a destination vertex. Show that, under this graph representation, the best response problem is equivalent to finding a shortest path from source to destination.

Question 5: Implement an algorithm to solve the shortest path problem of Question 4. What is its complexity? Compare to the algorithm of Question 3.

2.4 Tests

Question 6: Test the algorithms of Questions 2, 3, 5 on the following game instances:

Instance 1: $n = 3$; $X = 5$; $Y = 5$; $v_i = 1, \forall i$.

Instance 2: $n = 3$; $X = 1$; $Y = 5$; $v_i = 1, \forall i$.

Instance 3: $n = 3$; $X = 5$; $Y = 1$; $v_i = 1, \forall i$.

Instance 4: $n = 6$; $X = 12$; $Y = 12$; $v_i = 1, \forall i$.

Instance 5: $n = 6$; $X = 12$; $Y = 12$; $v_i = i, \forall i$.

In all cases, compute the best response to a strategy q_1^* of Player B that is the uniform distribution on $\mathcal{Y}$, and to a strategy q_1^* of Player B that assigns all Y troops on one battlefield chosen uniformly at random. Plot the result.

3 Nash equilibrium in the Colonel Blotto game

In this section, we focus on finding a Nash equilibrium of the Colonel Blotto game. A Nash equilibrium (NE) is a couple of strategies (p^*, q^*) such that each one is a best response to the other. Nash's famous theorem [3] shows that there exists a NE in mixed strategy (but not necessarily in pure strategy). Moreover, for zerosum games, Von Neuman's famous minmax theorem [4] implies that (p^*, q^*) is a NE if and only if p^* and q^* are minmax strategies, that is

$$p^* \in \arg\max_p \min_q U(p, q), \tag{1}$$

and

$$q^* \in \arg\max_q \min_p -U(p, q).$$

In the rest of the project, we will focus on finding a maxmin solution p^* for Player A.

Question 7: Identify some parameter values for which the solution is trivial. [*Hint: think of special cases of the games parameters for which there is an obviously winning strategy.*]

3.1 Naive linear program

Question 8: Formulate the maxmin problem (1) as a linear program (LP). How many variables and constraints does this LP have? Implement the LP and its solution (you can use an existing library to solve LPs). Check that it gives the correct solution on cases identified in Question 7. Compute the NE on the game instances from Question 6.

3.2 A better LP and the ellipsoid method

Note: This section is more difficult. Solving it is not necessary for the next sections (flow representation and extensions).

To improve the NE computation, it is possible to transfer the space of strategies to a new space as follows. For every pure strategy $x \in \mathcal{X}$, we map it to a point $\hat{x} \in \{0,1\}^{n\times(X+1)}$ such that $\hat{x}_{i,j} = 1$ if and only if strategy x allocates j troops to battlefield i, i.e., $x_i = j$. Let $\hat{\mathcal{X}} = \{\hat{x} \in \{0,1\}^{n\times(X+1)} : \exists x \in \mathcal{X} \text{ s.t. } x \text{ maps to } \hat{x}\}$, that is $\hat{\mathcal{X}}$ is the set of $\hat{x}$ such that there exists a valid pure strategy mapping to it. Similarly, we map any mixed strategy p to $\hat{p} \in [0,1]^{n\times(X+1)}$, where $\hat{p}_{i,j}$ represents the probability of allocating j troops to battlefield i under the mixed strategy p. Note that $\hat{p}_{i,\cdot}$ is the marginal distribution of troops allocated to battlefield i. We denote by $\hat{\mathcal{P}}$ the set of $\hat{p}$ such that there is at least one valid mixed strategy of Player A that maps to it: $\hat{\mathcal{P}} = \{\hat{p} \in [0,1]^{n\times(X+1)} : \exists p \in \mathcal{P} \text{ s.t. } p \text{ maps to } \hat{p}\}$. Finally, we define similarly the quantities $\hat{y}$, $\hat{q}$ and the sets $\hat{\mathcal{Y}}$, $\hat{\mathcal{Q}}$ for Player B.

In the space defined above, the maxmin property can be written as the following LP:

$$\begin{aligned} \max_{\hat{p},z} \quad & z \\ \text{s.t.} \quad & \hat{p} \in \hat{\mathcal{P}} \\ & U(\hat{p},\hat{y}) \geq z, \quad \forall \hat{y} \in \hat{\mathcal{Y}}, \end{aligned} \tag{2}$$

where $U(\hat{p},\hat{y}) = \sum_{i=1}^{n}\sum_{j_a=1}^{X}\sum_{j_b=1}^{Y} \hat{p}_{i,j_a}\hat{y}_{i,j_b}u_i(j_a,j_b)$ is the expected payoff of Player A. We admit that $\hat{\mathcal{P}}$ is a convex polyhedron that can be expressed with $O(2^{\text{poly}(n)})$ constraints.

LP (2) has a polynomial number of variables, but an exponential number of constraints. In this subsection, we show that it can be solved in polynomial time via the ellipsoid method. The ellipsoid method is a fundamental method in linear programming. It was introduced by [2].[3] This method can solve an LP in polynomial time if we can find a polynomial time separation oracle, that is an oracle that, in polynomial time, either returns that $\hat{p}$ is in the constraint polyhedron, or finds a separating constraint, that is a constraint that is not satisfied by $\hat{p}$. We will do that for the two sets of constraints separately.

We first focus on the constraint $\hat{p} \in \hat{\mathcal{P}}$. Here we need to find a polynomial-time algorithm that takes as an input $\hat{p}$ and either returns that $\hat{p} \in \hat{\mathcal{P}}$ or finds a hyperplane separating $\hat{p}$ from $\hat{\mathcal{P}}$. This is equivalent to solving the following other LP that searches for such a hyperplane:

$$\begin{aligned} \max_{\alpha_0,(\alpha_{i,j})_{i=1,\cdots,n;j=1,\cdots X}} \quad & 0 \\ \text{s.t.} \quad & \alpha_0 + \sum_{i=1}^{n}\sum_{j=1}^{X} \alpha_{i,j}\hat{p}_{i,j} < 0 \\ & \alpha_0 + \sum_{i=1}^{n}\sum_{j=1}^{X} \alpha_{i,j}\hat{x}_{i,j} \geq 0, \quad \forall \hat{x} \in \hat{\mathcal{X}}. \end{aligned} \tag{3}$$

[3] See a more accessible introduction, e.g., at http://www-math.mit.edu/~goemans/18433S09/ellipsoid.pdf.

This LP will itself be solved via the ellipsoid method; that is that we now need to find a separation oracle for this LP (3).

Question 9: Propose a DP algorithm that finds the pure strategy $x \in \mathcal{X}$ that minimizes $\alpha_0 + \sum_{i=1}^{n}\sum_{j=1}^{X}\alpha_{i,j}\hat{x}_{i,j}$, where $\hat{x}$ is mapped from x. [*Hint: follow a reasoning similar to the response to Question 3.*]

Question 10: Based on the answer to the previous question, find a separation oracle for the constraint of LP (3).

Question 11: Implement the ellipsoid method to solve LP (3).

We now move to the second set of constraints of LP (2), that is

$$\sum_{i=1}^{n}\sum_{j_a=1}^{X}\sum_{j_b=1}^{Y}\hat{p}_{i,j_a}\hat{y}_{i,j_b}u_i(j_a,j_b) \geq z, \forall \hat{y} \in \hat{\mathcal{Y}}.$$

Question 12: Propose and implement a separation oracle for this constraint. [*Hint: use a DP algorithm as in Question 9.*]

Question 13: Implement the ellipsoid method to solve LP (2). Compare the solution to the solution of the naive LP in Question 8.

The above method gives a solution $\hat{p}$ in the new space. It is possible to recover from that a solution p in the original space, in polynomial time. We prefer, however, to explore a simpler solution in the next subsection.

3.3 An polynomial-size LP via a flow representation

In this subsection, we explore another, simpler, solution through a flow representation. The key idea is that a mixed strategy can be represented as a flow of size 1 from the source node to the destination node in the graph from Question 4.

Question 14: Formulate the minmax problem (1) as an LP in the new space of flows. How many variables and constraints does the LP have?

Question 15: Propose an algorithm, given a flow of size 1, to compute a mixed strategy of the corresponding flow. Give a bound on the number of pure strategies contained in the support of this mixed strategy.

Question 16: Implement the LP from Question 14 (you can use an existing solver) and the algorithm from Question 15.

Question 17: Check that it gives the correct solution on cases identified in Question 7. Compute the NE on the game instances from Question 6. Compare the solution to the solution of the naive LP from Question 8 and verify the NE property using the best response algorithms from Section 2.

4 Extensions to multiple resource types

A Colonel Blotto game with multiple resource types is the same game as above with n battlefields, but where each player has $k \in \mathbb{N}$ different types of resources, each with its own budget denoted $X_1, \cdots, X_k$ and $Y_1, \cdots, Y_k$ for Player A and B respectively. We assume that each player can allocate any type of resources to any battlefield.

Question 18: The model above is not fully specified as it does not say, for a given allocation profile, which player will win each battlefield. Propose a utility function and discuss what are the important characteristics of the utility function to be able to solve the game.

Question 19: Redo Questions 4–5 and 14–17 for this extended model.

References

[1] E Borel. La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes rendus de l'Académie des Sciences*, 173(1304-1308):58, 1921.

[2] Leonid G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[3] John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[4] John Von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.