Using a Quantum Computer to cheat at Diplomacy

Louis Tessler, Sam Sutherland

May 2021

1 Introduction

Diplomacy is a game famous for betrayal and distrust. Thus it is appropriate that I have already lied to you. This is not a paper about Diplomacy. However we will use Diplomacy as a framing device to give a concrete example of the type of problem solved in the paper "Quantum Algorithms for Zero-Sum Games" [1]. The reason we have chosen this motif is because it is a real example (in the sense of not being invented for the purposes of game theory itself) but at the same time the rules are simple enough that it may be regarded as a "toy problem".

This document is structured as follows: First we give a basic primer on Game Theory. Then we explain the classical algorithm for addressing this problem. Then we explain how this was generalised into a quantum algorithm. Finally we render our remarks on the suitability of the algorithm for our chosen benchmark.

2 A Diplomatic introduction to the Nash Equilibrium.

In the game of Diplomacy the objective is to capture certain territories known as Supply Centers (SC). The game is played by writing down orders for each unit (either an army or a fleet). Thereafter the orders of all players are revealed and then executed simultaneously. Units can be ordered to either stay in place (hold), move to an adjacent territory (move), or help a different unit into to a territory which it could otherwise move (support). Whenever two or more units have orders which bring them into conflict, whichever one has the most support wins and all others stay in place. If no invader has the most support then the existing occupant (if any) keeps the territory. A non-moving unit with insufficient support is said to be dislodged and its fate is resolved later. When a supporting unit is attacked from outside the territory it supports to, or when it is dislodged, the support is invalidated. This brief explanation of the rules of Diplomacy has omitted some details for brevity. The full rule set can be found at [2]. I now present a small example in Figure 1 and enumerate possible moves in Figure 2.

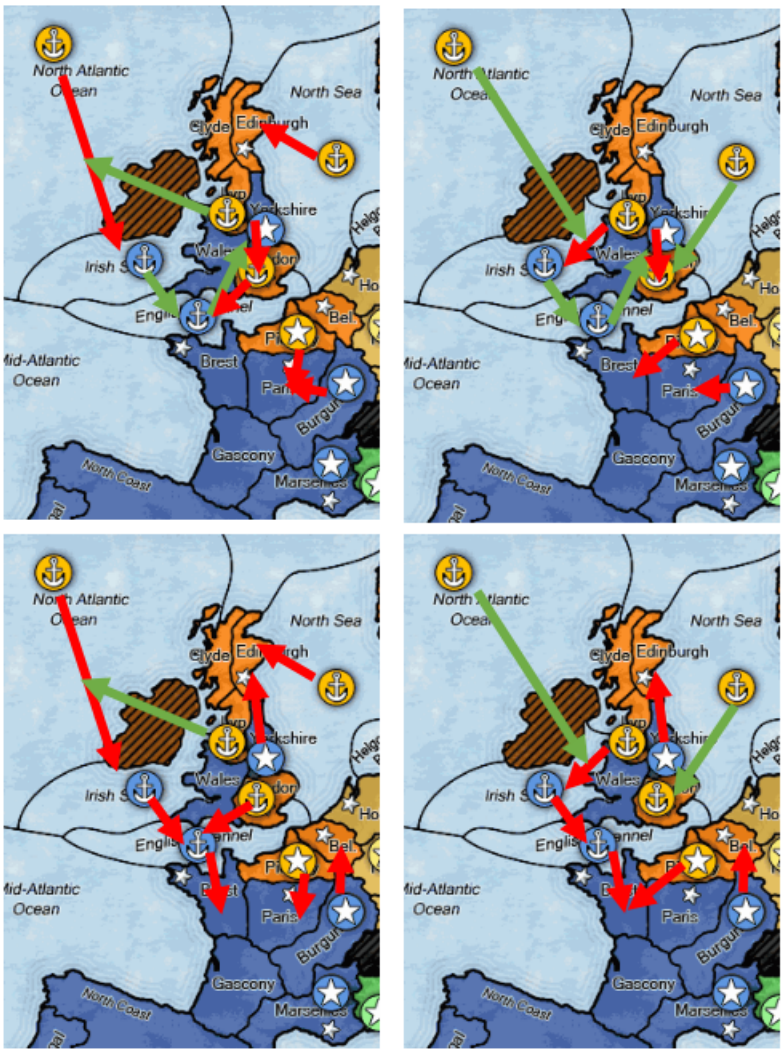


Figure 1: England (Orange) and France (Blue) are at war. Each unit either holds, moves (red arrow) or supports a different unit (green arrow). In the left column is the English plan to move North Sea to Edinburgh, London to English Channel, Picardy to Paris, and North Sea to Irish sea with support from Liverpool. In the right column is the English plan to to support London to hold, support Liverpool to the Irish sea, and move Picardy to Brest. In the top row is the French plan to move Burgundy to London and have Yorkshire move to London with support from the English Channel. In the bottom row is the French plan to move English Channel to Brest, Burgundy to Belgium, and Yorkshire to Edinburgh. Both players stand to loose SC if they incorrectly surmise which plan the other is going to choose.

Possible French orders irish sea	english channel	yorkshire	burgundy
moves liverpool	moves london	supports english channel	moves belgium
moves liverpool	moves london	supports english channel	moves paris
moves liverpool	moves london	moves edinburghh	moves belgium
...	...	...	...
supports yorkshire	moves north sea	moves liverpool	moves paris
supports english channel	supports yorkshire	moves london	moves belgium
supports english channel	supports yorkshire	moves london	moves paris

Possible English orders north sea	london	north atlantic	liverpool	picardy
supports london	holds	supports liverpool	holds	moves brest
supports london	holds	supports liverpool	holds	moves paris
supports london	holds	supports liverpool	holds	moves belgium
...	...	...	...	...
moves edinburgh	holds	moves irish sea	supports north atlantic	moves brest
moves edinburgh	holds	moves irish sea	supports north atlantic	moves paris
moves edinburgh	holds	moves irish sea	supports north atlantic	moves belgium

Figure 2: A list of possible moves for England and France. The full list of orders can be found in the supplementary material.

In Game Theory terms this is a zero-sum, two player game. The reason it is called zero sum is because every SC gained by one player was lost by another. In other words the net sum of all payoffs is zero. This fact allows us to represent the game as a matrix A where each row index of A represents the strategy chosen by France and the column index of A represents the strategy chosen by England (indeed the arrangement of figure 1 foreshadowed this revelation). The value $A_{x,y}$ is the number of SC gained by France when he chooses strategy x and England chooses strategy y. By the zero-sum property, the payoff matrix from England's perspective is just $-A^{\dagger}$. See Figure 3 for an explicit construction of A for our toy case.

A game is in a Nash Equilibrium when neither player can increase their payoff by unilaterally switching to a different strategy. France is looking for the column of A with the highest payoff under the assumption that England has done the same with $-A^{\dagger}$. Of course England finding the maximum of the negative is the same as finding the minimum of the positive. Thus the Nash Equilibrium can be found with the so called minmax.

$$N_{ash} = \min_{y} \max_{x} A_{x,y}. \tag{1}$$

The set of strategies represented by the rows and columns of A are known as pure strategies. A mixed strategy is a probability distribution of pure strategies. Without loss of generality one may assume that England and France are using mixed strategies because every pure strategy x is included in the set of mixed

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & -1 & 0 & -1 & -1 & 0 & -1 & -1 & 0 \\
1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
-1 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
-1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
-1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
-1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\
1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\
0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1
\end{bmatrix}$$

Figure 3: Payoff matrix for France. The (x, y) entry of this matrix corresponds to the gain/loss of SC's given that France plays strategy x and England plays strategy y. For example, the top left entry corresponds to the case in which England plays North Sea supports London to hold, the North Atlantic supports Liverpool to hold, and Picardy moves to Brest. This gains England an SC and thwarts the French attack on Liverpool (with support from Yorkshire), and London (with support from the English Channel). However the gains are short lived as France also moved Burgundy to Belgium making up for his loss in Brest. Hence there was no net change in SC and the matrix entry is 0.

strategies in the form of the trivial distribution ($P(x) = 1$). In general there is no promise that a pure strategy Nash equilibrium exists, but thanks to a theorem by Nash [3], we know that a mixed strategy equilibrium does always exist.

A mixed strategy is best represented as a vector $\vec{e}$ such that each entry e_x is the probability $P(x)$ of England choosing pure strategy x. By design the matrix multiplication $A\vec{e}$ results in a vector representing the expected payoff for each French pure strategy. In a mixed strategy Nash Equilibrium, England chooses $\vec{e}$ such that all French strategies have the same expected payoff and France chooses its mixed strategy $\vec{f}$ in a likewise manner.

Finding the mixed strategy Nash Equilibrium may be thought of as mapping the original game to a meta game and then finding the pure strategy equilibrium for that. The rows/columns of the meta game are indexed by the mixed strategy vectors $\vec{e}$ and $\vec{f}$. The meta-game payoff matrix M_A is given by

$$(M_A)_{\vec{e},\vec{f}} = \vec{f}^T A\vec{e}. \tag{2}$$

The definition of the Nash Equilibrium is unchanged but we rewrite it anyway to emphasize that the indices are now continuously valued vectors rather than integers.

$$N_{ash} = \min_{\vec{e}} \max_{\vec{f}} (M_A)_{\vec{e},\vec{f}}. \tag{3}$$

For the sake of a smaller example let's look at a 2 option sub-game shown in figure 1. The payoff matrix for this sub-game is

$$A_s = \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}. \tag{4}$$

The subscript s stands for sub-game.

Next let's look at the mixed strategies. Suppose with probability e_1 England will choose the pure strategy represented by the first column of the figure, and with probability $1 - e_1$ will choose the pure strategy represented by the last column. France chooses the strategy represented by the first row of the figure with probability f_1 and the second row with probability $1 - f_1$. By way of equation 2, the payoffs for the meta game are given by

$$M_{A_s} = 4e_1 f_1 - 2e_1 – 3f_1 + 2. \tag{5}$$

Recall that at the equilibrium all English strategies have the same payoff and vice versa. Therefore the derivative with respect to e_1 and f_1 is 0. By elementary calculus the Nash Equilibrium for this sub-game is $f_1 = .5$ and $e_1 = .75$.

The payoffs of the meta game M_{A_s} are plotted in Figure 4. In this two dimensional example we can see that the Nash equilibrium is a saddle point.

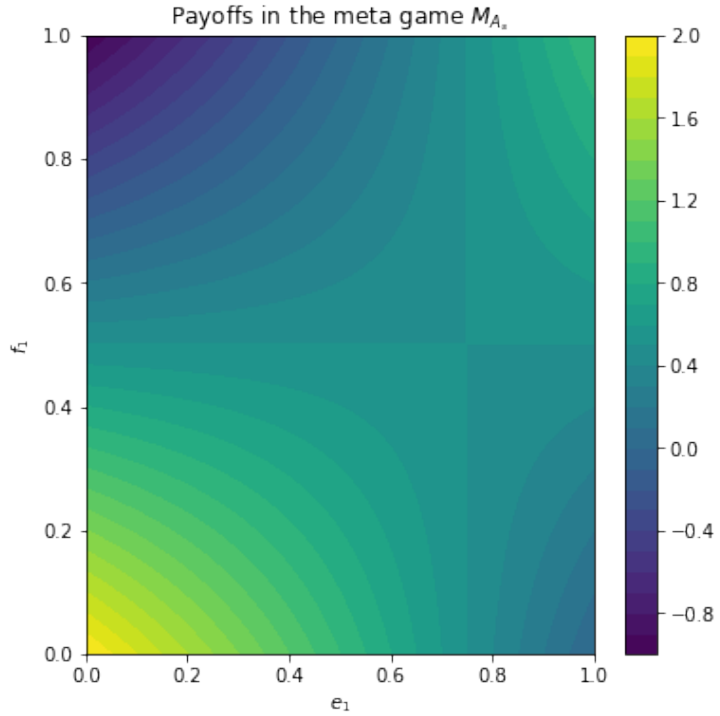


Figure 4: The payoffs for France and England as a function of the probability f_1 and e_1. The Nash Equilibrium is the saddle point in this plot.

In the full matrix A presented earlier, the Nash Equilibrium is the mixed strategy $\begin{bmatrix} 0.323 \\ 0.019 \\ 0.012 \\ \dots \\ 0.005 \\ 0.092 \\ 0.007 \end{bmatrix}$ $\begin{bmatrix} 0.006 \\ 0.016 \\ 0.318 \\ \dots \\ 0.005 \\ 0.006 \\ 0.035 \end{bmatrix}$ for France and England respectively. Once again we have truncated for brevity and moved the information to the supplementary material. In the next section we will explain the classical algorithm for finding the mixed strategy equilibrium in such intimidatingly large matrices.

3 Classical algorithm for finding the Nash Equilibrium.

We have taken the liberty of translating the classical algorithm provided in the paper into real functioning code. A less terse version of this code with more

insightful variable names is included in the supplementary material.

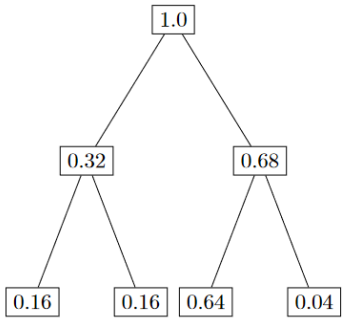
```
def NashE(A, iters = 1000, epsilon = 0):
    x, y = np.zeros(A.shape[0]), np.zeros(A.shape[1])
    if epsilon == 0: eta = 1/(2*np.sqrt(i))
    for i in range(1,iters+1):
        eta = 1/(2*np.sqrt(i))
        u, v = -A.T@x, A@y
        P, Q = np.exp(u), np.exp(v)
        p, q = P/norm(P, ord=1), Q/norm(Q, ord=1)
        x, y = x + eta*sample(q), y + eta*sample(p)
    return x/norm(x, ord=1), y/norm(y, ord=1)
```

Two versions of the algorithm are given in the paper, one with fixed accuracy ϵ and one which is adaptive. The only difference is the variable η which I will ignore for now [1]. The quantities x and y are vectors that represent mixed strategy of England and France respectively. In each iteration the expected payoff for England and France is computed and stored in u and v. The next step is to map these expected payoffs to a probability distribution such that higher payoffs are more likely to get selected. The mapping is accomplished through the exponential function (appropriately normalised). The exponential distribution is a natural choice for reasons that will be explained momentarily. Instead of normalising these distributions per se, one is free to use rejection sampling instead. In any case the resultant distributions are stored as $p = e^u$ and $q = e^v$ respectively. Finally, p and q are sampled from, and the resultant values are added into x and y.

To understand the reasoning of this algorithm it is imperative to realise that the x distribution represents England's current belief about France's mixed strategy, and the y vector represents France's current belief about England's mixed strategy. Furthermore these belief vectors are arrived at by simply averaging together the moves that were actually made over the course of the simulated turns. In this context the algorithm can be understood as repeatedly simulating England and France playing each other and then updating their beliefs based on observation. The magnitude of the change in belief is controlled by η. This method of finding the Nash equilibrium is called fictitious play [4].

Suppose England observed that the expected payoff for strategy x is greater than for x'. Should England always pick x? The answer depends on how much uncertainty there is in the observed expected payoff and how close together the expected payoffs are together. Assuming the observed plays are an i.i.d random variable, the probability of erroneously observing x to have a higher payoff than x' is given by $e^{-\Delta}$ where Δ is the observed difference in payoff. Hence why the exponential distribution works in this setting.

[1] in the provided code, leaving $\epsilon = 0$ defaults to the adaptive version. If only one version of the algorithm is desired, the if statement may be omitted.



Let $|\phi\rangle = 0.4\,|00\rangle + 0.4\,|01\rangle + 0.8\,|10\rangle + 0.2|11\rangle$.

- Rotation on qubit 1:
  $|0\rangle\,|0\rangle \to (\sqrt{0.32}\,|0\rangle + \sqrt{0.68}\,|1\rangle)\,|0\rangle$
- Rotation on qubit 2 conditioned on qubit 1:

$$(\sqrt{0.32}\,|0\rangle + \sqrt{0.68}\,|1\rangle)\,|0\rangle \to$$
$$\sqrt{0.32}\,|0\rangle\,\frac{1}{\sqrt{0.32}}(0.4\,|0\rangle + 0.4\,|1\rangle) +$$
$$\sqrt{0.68}\,|1\rangle\,\frac{1}{\sqrt{0.68}}(0.8\,|0\rangle + 0.2\,|1\rangle)$$

Figure 5: A diagram explaining how QCRAM functions. Each node (stored classically) determines a controlled rotation on each qubit. The tree can be updated in a logarithmic number of steps starting from the root. Figure taken from [5]

3.1 Trees

Since only one element of x and y are updated per iteration, it is beneficial to store P and Q using a tree structure as in Figure 5, where each leaf of the tree stores the sum of its branches. Updating a single value in such a structure is efficient as one can start at the lowest level of the tree and proceed upwards, updating values as you go. This takes time $O(\log m)$, where m is the length of the vector. For a sparse $n \times m$ game matrix A with a maximum of s entries in any row and a maximum of d entries in any column, the change in u and v in one iteration can be found in time $O(s \log m)$ and $O(d \log n)$ respectively. P can now be updated in time $O(s \log m)$ since s elements will change, each requiring $O(\log m)$ time to change. In order to sample from P in $O(\log m)$, one can start at the highest node then take a random path down the tree, with left/right probabilities dependent on the values of the nodes in the layer below.

The quantum implementation of this algorithm is dependent on storing x (and y) in such a tree-like structure in QCRAM (quantum-read/classical-write RAM). Given this capability, one can both query x and sample from the probability distribution defined by the components of the x in time $O(\log m)$

4 Quantum generalisation of the classical algorithm for NE.

The bulk of the paper describes the conversion of the classical algorithm to the quantum algorithm. Since x and y are stored in QCRAM, they can both be queried, written to, and sampled from efficiently. The quantum algorithm then becomes an exercise in implementing efficient Gibbs sampling.

This is achieved using block-encoded unitaries. A block-encoded unitary U

utilises a register of ancilla qubits to implement the map

$$U\left|0\right\rangle\left|\psi\right\rangle = \left|0\right\rangle A\left|\psi\right\rangle + \left|1\right\rangle\left|garbage\right\rangle. \tag{6}$$

That is, with some probability you get the state $|0\rangle$ in the ancilla register, which heralds that you have successfully implemented your (possibly non-unitary) operation A on the computational register. If you measure $|1\rangle$ in the ancilla qubit register, the implementation has failed and must be repeated until success is heralded. The aim is then to construct a block-encoded unitary so that its action on the state is as follows:

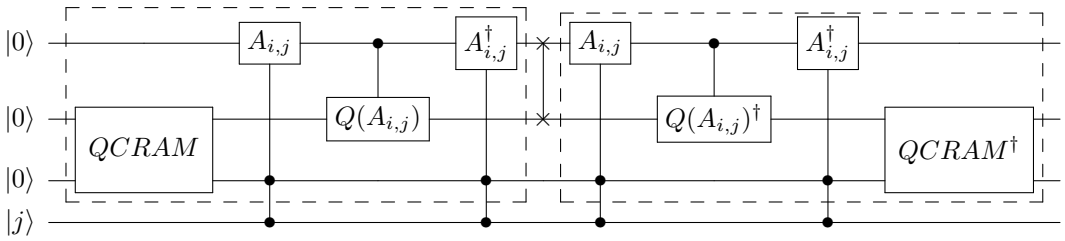
$$\left|0\right\rangle\left|j\right\rangle \rightarrow \left|0\right\rangle e^{(u_j - u_{max})/2}\left|j\right\rangle + \left|1\right\rangle\left|garbage\right\rangle, \tag{7}$$

where u_j is the j^{th} component of $x^T A$, and u_{max} is the maximum u_j. This is subtracted to ensure that all the coefficients are ≤ 1, which is necessary to ensure the validity of the polynomial approximation results introduced shortly. After a successful implementation, the second register can be measured to obtain a j with probability $\propto e^{(x^t A)_j}$ and the algorithm can proceed as in the classical case.

In order to implement this unitary map, the aforementioned QCRAM data structure containing x and y is heavily utilised. The details of QCRAM are explained in 5. Define U to be

$$U := V^\dagger (SWAP_{12} \otimes \mathbb{I}_{34}) V.$$

The quantum circuit for U is shown below with $V^\dagger$ and V in the dotted lines.



where

$$Q(s) = \begin{bmatrix} \sqrt{s} & \sqrt{1-s} \\ \sqrt{1-s} & -\sqrt{s} \end{bmatrix}$$

and

$$QCRAM\left|0\right\rangle\left|\bar{0}\right\rangle \rightarrow \sum_{i\in n} \sqrt{x_i/||x||_1}\left|i\right\rangle + \left|1\right\rangle\left|garbage\right\rangle.$$

Therefore it is easily checked that

$$V\left|0\right\rangle\left|0\right\rangle\left|\bar{0}\right\rangle\left|j\right\rangle \rightarrow \left|0\right\rangle\left(\left|0\right\rangle \sum_{i\in[n]} \sqrt{x_i A_{ij}/||x||_1}\left|i\right\rangle + \left|1\right\rangle\left|garbage\right\rangle\right)\left|j\right\rangle$$

and so

$$(\langle 00\bar{0}| \otimes \mathbb{I})U(|00\bar{0}\rangle \otimes \mathbb{I}) = diag(x^T A/||x||_1)$$

is the desired block encoding of A. We will now explicitly show that the desired transform $diag(x^T A/||x||_1)$ is affected by this procedure:

$$\begin{aligned}
&(\langle 00\bar{0}| \otimes \mathbb{I})U(|00\bar{0}\rangle \otimes \mathbb{I}) \\
&= (\langle 00\bar{0}| \otimes \mathbb{I})V^\dagger (SWAP_{12} \otimes \mathbb{I}_{34}) |0\rangle \left(|0\rangle \sum_{i\in[n]} \sqrt{x_i A_{ij}/||x||_1}\, |i\rangle + |1\rangle |garbage\rangle \right) \mathbb{I} \\
&= (\langle 00\bar{0}| \otimes \mathbb{I})V^\dagger \left(|0\rangle |0\rangle \sum_{i\in[n]} \sqrt{x_i A_{ij}/||x||_1}\, |i\rangle + |1\rangle |0\rangle |garbage\rangle \right) \mathbb{I} \\
&= \left(\langle 0| \langle 0| \sum_{j\in[n]} \left(\sqrt{x_i A_{ij}/||x||_1} \right)^* \langle j| + \langle 0| \langle 1| \langle garbage| \right) \mathbb{I} \\
&\qquad \left(|0\rangle |0\rangle \sum_{i\in[n]} \sqrt{x_i A_{ij}/||x||_1}\, |i\rangle + |1\rangle |0\rangle |garbage\rangle \right) \mathbb{I} \\
&= \langle 0|0\rangle \langle 0|0\rangle \sum_{j\in[n]} \sum_{i\in[n]} \left(\sqrt{x_i A_{ij}/||x||_1} \right)^* \sqrt{x_i A_{ij}/||x||_1}\, \langle j|i\rangle \\
&\qquad + \langle 0|1\rangle \langle 1|0\rangle \langle garbage|garbage\rangle \\
&= \sum_{j\in[n]} \sum_{i\in[n]} x_i A_{ij}/||x||_1 \delta_{ij} \\
&= \sum_{i\in[n]} x_i A_{ii}/||x||_1 \\
&= diag(x^T A/||x||_1)
\end{aligned}$$

Hence we have implemented the Hermitian matrix $diag(x^T A/||x||_1)$. Now that we have this map, we can use the amplitude estimation procedure from [6] to estimate a single u_j and the maximum finding algorithm from [7] to estimate u_{max}. This results in the ability to create the block-encoded map $M := diag(x^T A - u_{max}\mathbf{1}^m)/(2||x||_1)$ in constant time.

The last step is to use M to sample from the exponential distribution. To do this, several results about polynomial approximations are utilised. It can be shown that for $\beta > 1$, there is a δ-accurate approximating polynomial $P(z)$ for the function $e^{\beta z}$ with degree $||x||_1 \log 1/\delta$. Thus, one can approximate the map given in equation 7 as such:

$$|0\rangle |j\rangle \to |0\rangle P(M) |j\rangle + |1\rangle |garbage\rangle . \tag{8}$$

The probability of obtaining the $|0\rangle$ heralded state can be improved with oblivious amplitude amplification. This state can then finally be measured to obtain an approximate sample from the Gibbs distribution of the state.

In summary, x and y are first initialised in QCRAM. The procedure outlined above is then used to sample from the distributions $e^{-x^T A}/||e^{x^T A}||_1$ and $e^{Ay}/||e^{Ay}||_1$, utilising the efficient reading and sampling of x and y due to their storage in QCRAM. Finally, x and y are updated with the sampled strategy, again using the efficiency of the data format to perform the update. After many iterations, x and y will store a close approximation of the Nash Equilibrium strategies for the two players.

4.1 Reduction of general LP-solving to zero-sum games

The final part of this paper describes how one can use the quantum algorithm proposed to also solve general linear programming problems. They do this by showing how to reduce an arbitrary LP problem to a zero sum game. I will briefly outline the four steps required to do this.

First the LP is reduced to a feasibility problem. This is done by introducing additional constraints $\sum_{i=1}^{m} y_i \leq R$ and $\sum_{i=1}^{n} x_i \leq r$ where r and R are constants chosen to be large enough to not affect the final solution of the problem. Because of strong duality, the optimal values for the two problems are equal. Now since all the multiplicative constraints can be assumed to be in $[-1, 1]$ it follows that the optimal value OPT must be in the range $-R \leq \text{OPT} \leq R$ and the problem becomes finding identifying if there is a feasible y such that all the constraints are satisfied and OPT is in this range.

Secondly, to scale y to the range $[0, 1]$, an auxiliary variable z is appended to y to form y' and the optimisation matrix is modified using the extra degree of freedom so that y' is in the correct range. y is not simply divided by R as this would change the behaviour of the additive error.

Thirdly, the right hand side of all constraints should equal zero. This is achieved by appending another new variable, h, to y' giving y''. You can use this extra degree of freedom as in the last step to modify the existing constraints such that they are equal to 0 on the right hand side.

Finally, the problem becomes a zero sum game by noticing that the final LP has a feasible solution of value λ iff the value of the equivalent zero sum game is less than λ.

5 What happens when you ask a Quantum Computer to play Diplomacy?

It prepares a Nation State.

At the start I stated that part of our reason for the Diplomacy motif is to check the usefulness of this quantum algorithm by demonstrating that it can be applied to a problem of our choosing. It is time to actually test the suitability of a two player zero-sum game solver for the problem of cheating at Diplomacy.

The task at hand is in two parts. First figure out how to procedurally turn a Diplomacy game state into a payoff matrix, and second figure out how to translate this matrix to an oracle.

5.1 Turning a Diplomacy game into a payoff matrix.

In our introduction to game theory we have already peeked at how to construct the payoff matrix. I reiterate the process briefly in order to highlight complications which were ignored before.

1. Enumerate the possible actions of each player. These will index the rows and columns of the payoff matrix.
2. For each possible row and column, apply the rules of the game to get the resultant game state.
3. For each game state, calculate how much payoff was realised by each player.

A germane issue to point out with enumerating the possible moves is just how large the resultant payoff matrix will be. The median territory on the Diplomacy board is adjacent to 6 others and at the start every player has 3 units (except Russia which has 4). Thus an opening position has a lower bound of 216 possible directions for a move/support and there are 2^3 possible combinations of moving/supporting, which leads to an estimate of $1,728$ legal order sets (note: the payoff matrix presented earlier was much smaller because I manually excluded irrelevant territories and futile options). As this estimate applies to all players, constructing the payoff matrix would require adjudicating over a million possible game states. The tyranny of size only gets worse as the game progresses and the number of units changes.

Just because a move is legal does not mean it is worthy of consideration. At first glance one might try to come up with some better method for generating possible moves in a way that skips over the obviously poor choices. For example, a naive approach might be to add a rule that orders should never contradict each other in certain ways, such as two units being ordered to move to the same place, or one unit supporting and another unit attacking the same place. However both of these are real tactics which have been used in real games [8]. The problem of pruning away pointless-but-legal moves is clearly possible since human players do it, but an automatic pruning method is as yet unknown.

5.2 Making an oracle to represent a Diplomacy payoff matrix.

In the paper there are two types of oracle access that are suitable for this quantum algorithm. First the dense case, given x and y the oracle should tell us the value of the payoff $A_{x,y}$. This is nothing more than the classical computation of the game adjudication program. The only complication in our case is the enumeration of the possible moves. What is needed is a way to map integer indices into valid order sets. We quickly outline a procedure to do this.

For each territory, label the neighbouring territories in a clockwise fashion with an integer. The most connected territory on the map has 12 connections, so no more than 4 bits are required for these labels. For each unit a player posses,

generate two 4-bit numbers. The first number points to the destination territory (with 0 representing a hold) and the second number points to the supported unit. A number which goes above the available number of neighbouring territories can simply be wrapped back to zero. Thus all 8 bit integers can be mapped (albeit somewhat redundantly) to a valid order. While perhaps not the most efficient way to represent things, this method at least promises that every integer in a certain range will be a valid row / column of the payoff matrix.

The main quantum algorithm promises lower resource requirements when provided an oracle in a sparse format. In the papers own words "When A is given by a sparse oracle which also allows querying for any j the location of the $j-th$ non-zero entry in each row and column, then a further speedup is possible." In the context of our example, such a sparse oracle would be a program that, for a given English plan, finds the French plans with a non-zero payoff. In principle such a program is possible, as one could match English support orders with French units that could break those supports, and match move orders other move orders. However, just like with the generation of legal orders, the actual construction of such a "Diplomacy oracle" is surprisingly difficult. A simple order-matcher would erroneously dismiss too many cases.

5.3 Conclusion: does it pass the test? Can a quantum computer be used to cheat at Diplomacy?

First, some minor caveats. The Nash Equilbrium isn't promised to exist for infinite games [9]. However this issue is easily rectified by adding an end date rule to force turn it into a finite game (Eg, the game ends by turn 200 if no one has won). Additionally, Diplomacy is not a two player game. It is a seven player game. However it is often the case that all the players are in one of two coalitions, so the same ideas can be applied. It is also not the case that Diplomacy is always zero-sum. In the beginning there are neutral SC that can be gained at no other players loss. This fact is not a major concern because it becomes a zero sum game after those are captured (which usually happens early on).

Next, a non-trivial caveat. The Nash Equilibrium assumes rational play but this assumption is frequently violated in practice. Claiming that finding the Nash Equilibrium let's you cheat at Diplomacy is yet another lie in my recurring theme of telling lies. By definition, if someone is not playing the Nash Equilibrium strategy then it is possible to find a response strategy with a higher payoff. Even if one had a predictive model of an irrational opponent, the quantum algorithm presented here does not give us a way to leverage this information into finding the optimal strategy.

Furthermore, ignoring irrationality per se, it is not always the case that everyone agrees on the payoffs. The objective is to win the game overall, not to win the particular battles of a given turn. Since it is impossible to compute the game tree all the way to the bottom, players might be using a heuristic for 'tactical value' in lieu of calculating the 'point value' payoff. However there is no universally agreed upon meaning to tactical value. Everyone makes up their

own beliefs about which places are more valuable than others. So in addition to France and England having different payoffs (making it non-zero sum), a priori France doesn't know what England's payoff matrix looks like and vice versa.

In the game theory literature, this situation is known as a Bayesian game and the relevant thing to compute is the "Bayesian Nash Equilibrium" [10]. To give a brief summary, imagine France starts the game with the knowledge that England can be categorized as one of several archetypes (meaning with probability P_k his payoff matrix is A_k). Eg perhaps type 1 has an unusual obsession with Picardy, type 2 mostly makes defensive moves etc. In this type of game each player starts with the same common priors and on each turn, based on the strategies played, update these priors via Bayesian inference. The quantum algorithm in the paper is not suitable for Bayesian games. I speculate that a generalisation of this quantum algorithm to bayesian games might be easily attained because we might be able to treat a Bayesian game as a superposition of ordinary games.

The most important difference between the algorithm as designed and the Diplomacy use case is the fact that Diplomacy consists of more than a single turn. In the language of game theorists this is known as an "extensive form" game and its states are represented by a tree (contrasted with the matrix representation which is known as "normal form"). In principle it is always possible to turn an extensive form game into a normal form one by enumerating all the different branches one might go down in the game tree. However the payoff matrix for such game trees grows very quickly, far worse than the single-turn payoff matrix. The pertinent question to ask is if the quantum advantage on finding the NE for large matrices is enough to overcome the disadvantage of not pruning bad moves early on.

The authors claim an algorithmic complexity of $O(\frac{\sqrt{n+m}}{\epsilon^3})$ queries to the payoff matrix where ϵ is the error in the computed value (note: a polylog factor has been neglected). For the sake of simplicity let's assume that England and France will have about the same number of options ($n = m$). Additionally let us suppose that the possible moves per turn is constant, so the payoff matrix for all k-step strategies is a square matrix of size n^k. Plugging these assumptions into the complexity statement gives us $O(\frac{n^{k/2}}{\epsilon^3})$ queries to the payoff matrix. By similar reasoning, the classical version of this algorithm has complexity of $O(\frac{n^k}{\epsilon^2})$ queries. For the sake of argument, let's suppose that $\epsilon = .1$ and that the classical algorithm becomes impractical after $k = 2$. Plugging in my earlier estimate of $n = 1,728$ legal moves per player, the impractical point happens when the big-O statement is somewhere between $3 * 10^8$ and $5 * 10^{11}$. Working backwards to find the value of k where the quantum algorithm would become impractical. The answer I arrive at is between 3 to 5 turns. This is a fairly underwhelming outcome. There are human players with a higher degree of clairvoyance than that.

In conclusion, it is not yet the case that you can cheat at Diplomacy using a Quantum Computer.

References

[1] J. van Apeldoorn and A. Gilyén, "Quantum algorithms for zero-sum games," *arXiv:1904.03180 [quant-ph]*, Apr. 2019. arXiv: 1904.03180.

[2] "Diplomacy." https://avalonhill.wizards.com/games/diplomacy/info.

[3] J. Nash, "Non-cooperative games," *The Annals of Mathematics*, vol. 54, p. 286, Sept. 1951.

[4] V. Krishna and T. Sjöström, "On the convergence of fictitious play," *Mathematics of Operations Research*, vol. 23, pp. 479–511, May 1998.

[5] I. Kerenidis and A. Prakash, "Quantum recommendation systems," 2016.

[6] S. J. Lomonaco and H. E. Brandt, *Quantum Computation and Information*, vol. 305 of *Contemporary Mathematics*. American Mathematical Society, 2002. ISSN: 1098-3627, 0271-4132.

[7] J. Van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, "Quantum SDP-Solvers: Better Upper and Lower Bounds," in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 403–414, Oct. 2017. ISSN: 0272-5428.

[8] "Guest post: Advanced diplomacy maneuvers – brotherbored." https://brotherbored.com/guest-post-advanced-diplomacy-maneuvers/.

[9] E. Maskin and P. Dasgupta, "The existence of equilibrium in discontinuous economic games, part i (theory)," *Review of Economic Studies*, vol. 53, no. 1, pp. 1–26, 1986. Reprinted in K. Binmore and P. Dasgupta (eds.), Economic Organizations as Games, Oxford: Basil Blackwell, 1986, pp. 48-82.

[10] T. Sadzik, "Beliefs revealed in bayesian-nash equilibrium," 04 2010.

6 Supplementary

Full table of orders represented in our payoff matrix:

Possible French orders irish sea	english channel	yorkshire	burgundy
moves liverpool	moves london	supports english channel	moves belgium
moves liverpool	moves london	supports english channel	moves paris
moves liverpool	moves london	moves edinburghh	moves belgium
moves liverpool	moves london	moves edinburghh	moves paris
moves liverpool	moves north sea	moves edinburghh	moves belgium
moves liverpool	moves north sea	moves edinburghh	moves paris
moves liverpool	moves north sea	supports irish sea	moves belgium
moves liverpool	moves north sea	supports irish sea	moves paris
moves liverpool	moves london	supports irish sea	moves belgium
moves liverpool	moves london	supports irish sea	moves paris
moves liverpool	moves north sea	supports irish sea	moves belgium
moves liverpool	moves north sea	supports irish sea	moves paris
supports yorkshire	moves london	moves liverpool	moves belgium
supports yorkshire	moves london	moves liverpool	moves paris
supports yorkshire	moves north sea	moves liverpool	moves belgium
supports yorkshire	moves north sea	moves liverpool	moves paris
supports english channel	supports yorkshire	moves london	moves belgium
supports english channel	supports yorkshire	moves london	moves paris

Possible English orders north sea	london	north atlantic	liverpool	picardy
supports london	holds	supports liverpool	holds	moves brest
supports london	holds	supports liverpool	holds	moves paris
supports london	holds	supports liverpool	holds	moves belgium
supports london	holds	supports liverpool	moves irish sea	moves brest
supports london	holds	supports liverpool	moves irish sea	moves paris
supports london	holds	supports liverpool	moves irish sea	moves belgium
supports london	holds	moves irish sea	supports north atlantic	moves brest
supports london	holds	moves irish sea	supports north atlantic	moves paris
supports london	holds	moves irish sea	supports north atlantic	moves belgium
moves english channel	supports north sea	supports liverpool	holds	moves brest
moves english channel	supports north sea	supports liverpool	holds	moves paris
moves english channel	supports north sea	supports liverpool	holds	moves belgium
moves english channel	supports north sea	supports liverpool	moves irish sea	moves brest
moves english channel	supports north sea	supports liverpool	moves irish sea	moves paris
moves english channel	supports north sea	supports liverpool	moves irish sea	moves belgium
moves english channel	supports north sea	moves irish sea	supports north atlantic	moves brest
moves english channel	supports north sea	moves irish sea	supports north atlantic	moves paris
moves english channel	supports north sea	moves irish sea	supports north atlantic	moves belgium
moves edinburgh	holds	supports liverpool	holds	moves brest
moves edinburgh	holds	supports liverpool	holds	moves paris
moves edinburgh	holds	supports liverpool	holds	moves belgium
moves edinburgh	holds	supports liverpool	moves irish sea	moves brest
moves edinburgh	holds	supports liverpool	moves irish sea	moves paris
moves edinburgh	holds	supports liverpool	moves irish sea	moves belgium
moves edinburgh	holds	moves irish sea	supports north atlantic	moves brest
moves edinburgh	holds	moves irish sea	supports north atlantic	moves paris
moves edinburgh	holds	moves irish sea	supports north atlantic	moves belgium

The Nash Equilibrium mixed strategies for

$$\text{France} \begin{bmatrix} 0.323 \\ 0.019 \\ 0.012 \\ 0.001 \\ 0.199 \\ 0.014 \\ 0.003 \\ 0.252 \\ 0.005 \\ 0.003 \\ 0.000 \\ 0.006 \\ 0.002 \\ 0.000 \\ 0.000 \\ 0.007 \\ 0.014 \\ 0.002 \\ 0.003 \\ 0.006 \\ 0.001 \\ 0.000 \\ 0.011 \\ 0.000 \\ 0.009 \\ 0.004 \\ 0.000 \\ 0.005 \\ 0.092 \\ 0.007 \end{bmatrix} \text{ and England } \begin{bmatrix} 0.006 \\ 0.016 \\ 0.318 \\ 0.004 \\ 0.005 \\ 0.072 \\ 0.004 \\ 0.004 \\ 0.033 \\ 0.003 \\ 0.002 \\ 0.006 \\ 0.002 \\ 0.003 \\ 0.007 \\ 0.000 \\ 0.000 \\ 0.001 \\ 0.025 \\ 0.024 \\ 0.325 \\ 0.010 \\ 0.007 \\ 0.078 \\ 0.005 \\ 0.006 \\ 0.035 \end{bmatrix} \text{ respectively.}$$

The classical algorithm rewritten for improved readability.

```
def NashE(A, iters = 1000, epsilon = 0):
    #at first there are no observed moves, any pure strat is equally likely
    Fra_observed_P_strat_x = np.zeros(A.shape[0])
    Eng_observed_P_strat_y = np.zeros(A.shape[1])

    eta = epsilon/4
    for i in range(1,iters+1):
        eta = 1/(2*np.sqrt(i))

        #payoff matrix times strategy vector gives expected payoffs
        Fra_strat_x_expected_payoff = A @ Eng_observed_P_strat_y
        Eng_strat_y_expected_payoff = -A.T @ Fra_observed_P_strat_x

        #given observed diff in payoff delta, France prefers x over x' with prob e^(delta)
        Fra_strat_x_dist = np.exp(Fra_strat_x_expected_payoff)
        Eng_strat_y_dist = np.exp(Eng_strat_y_expected_payoff)

        #renormalise the distributions
        Fra_P_strat_x = Fra_strat_x_dist/norm(Fra_strat_x_dist, ord=1)
        Eng_P_strat_y = Eng_strat_y_dist/norm(Eng_strat_y_dist, ord=1)

        #sample and update observed strategy probs
        Fra_observed_P_strat_x += eta * reject_sample(Fra_P_strat_x)
        Eng_observed_P_strat_y += eta * reject_sample(Eng_P_strat_y)

    #normalise then quit
    Fra_observed_P_strat_x = Fra_observed_P_strat_x/norm(Fra_observed_P_strat_x, ord=1)
    Eng_observed_P_strat_y = Eng_observed_P_strat_y/norm(Eng_observed_P_strat_y, ord=1)

    return Fra_observed_P_strat_x, Eng_observed_P_strat_y
```