Algorithms Clearly Beat Gamers at Quantum Moves
A Verification

Allan Grønlund
Computer Science, Aarhus University, jallan@cs.au.dk

Abstract

The paper [1] presents and discusses results for a Quantum Moves game, BringHomeWater, where players have attempted to move a quantum state from one position to another in a simulated optical tweezer and atoms setup. The paper compares the player solutions to numerical methods that the authors discuss. In particular, [1] show evidence that the so-called Krotov optimization method [5] performs worse than solutions that the human players have come up with. Given the assumption that the Krotov method has been correctly applied, the evidence points to the fact that human players can outperform the Krotov method. This is the find and claim of [1] and it features prominent in the abstract of the paper. It leads the authors to conclude that 'players succeed where purely numerical optimization fails, and analysis of their solutions provide insights into the problem of optimization of a more profound and general nature.'

While it seems clear from the presented data that human players have indeed outperformed a particular implementation of the Krotov algorithm (and, according to [1], as a consequence also algorithms like CRAB [6] and evolutionary approaches [8] which perform worse than Krotov), there is no reason to believe that this finding is of any particular significance. In fact, as has been discussed first by [4], a very simple approach using classical arguments can capture the BringHomeWater Quantum Moves game far better than the player approach. Furthermore, as also shown by [4], one of the simplest optimization algorithms available, Stochastic Ascent, can outperform all of the above. This clearly demonstrates that the problem solved in [1] is not hard at all and efficiently solvable using well-known numerical methods. The authors of [1] note in the abstract that 'Using player strategies, we have thus developed a few-parameter heuristic optimization method that efficiently outperforms the most prominent established numerical methods.' This must be regarded as the major outcome of [1]. It is, however, completely misleading to conclude such things when a simple Stochastic Ascent algorithm can be used to solve the problem better than any of the methods discussed in [1]. Here we elaborate on the method discussed by [4] and show that Stochastic Ascent solves the problem very fast, also around the so-called quantum speed limit (QSL, defined as the minimum duration required to get fidelity of $F \geq 0.999$). For extra perspective we include a comparison with the classic GRAPE algorithm from Quantum Control and shows that this algorithm also outperforms all gamers.

1 Introduction

The Quantum Moves ‘BringHomeWater’ game is designed to have human players move atoms around using optical tweezers. The claim by [1] is that humans can outperform widely used algorithms. However, as initially discussed by [4] and elaborated on below, a simple discretized approach with standard Stochastic Ascent outperforms all players for which data is presented in [1] by a very large margin. Albeit that [4] only compares to player solutions, the simple algorithm also outperforms the numerical optimization based on player solutions and Krotov discussed by [1]. The Stochastic Ascent algorithm is a very simple and basic approach, and it is also very fast. Given a duration time for the game and using the parameters in [4] it takes a few minutes on a laptop to run the algorithm and get a high fidelity solution. This should be compared to the so-called KASS implementation of Krotov discussed in [1] where the reported calculation time per run was 6(!) hours which was tried 2400(!) times (a combined calculation time of 600 days!). Each sweep computed the fidelity achievable for many durations but this is still a massive amount of computing
time compared to the simple Stochastic Ascent algorithm. This inevitably leads to two conclusions. First, it is clear that the Krotov algorithm [5] is seriously challenged by the problem at hand. Second, given the simplicity of the Stochastic Ascent, it must be concluded that the BringHomeWater problem cannot be considered difficult for numerical optimization. Furthermore, the Stochastic Ascent used below is equivalent to applying constant Hamiltonians in small time intervals to the system. This is the same tactics as in the well-known GRAPE algorithm [7], although the present algorithm is considerably simpler as it is discretized and local. It is extremely surprising that GRAPE is not discussed in [1] and that no comparison to GRAPE appears to have been made.

To make the point clear that no elaborate algorithms or initialization is required for BringHomeWater, the Stochastic Ascent algorithm that solves BringHomeWater is given in Figure 1.

**Stochastic Ascent Algorithm:**

- let \(x_1, \ldots, x_N\) be the tweezer positions of the **controlled** tweezer initialized randomly (the parameters)
- while not in local minima
  - iterate over \(x_i\), for \(i = 1, \ldots, N\) in uniform random order
  - fix all parameters except \(x_i\) and optimize over \(x_i\) i.e.
    \[x_i \leftarrow \arg \max_{x_i} \text{fidelity}(x_1, \ldots, x_{i-1}, \hat{x}_i, x_{i+1}, \ldots, x_N)\]
- return \(x_1, \ldots, x_N\)

Figure 1: Optimization Algorithm for maximizing fidelity in BringHomeWater by [4].

This is a simple and straightforward Coordinate Ascent [3] optimization algorithm from mathematical optimization. The only thing that may be considered complicated is the computation of the fidelity but that has nothing to do with the algorithm itself. In words, the algorithm is as follows. Let us say we are given 40 steps to move the atom. Then we have 40 parameters, the position of the controllable tweezer after each of these steps. The task is now to find a set of parameters (positions) that makes the fidelity large. First, initialize randomly the parameters. That will almost surely give a bad fidelity score. Now, repeatedly iterate over the parameters in random order. For each parameter, fixing the value of the remaining parameters, set the value such that in combination with the fixed parameters the fidelity is maximized. Repeat these random scans of the parameters until the fidelity stops improving. You could visualize this as having 40 knobs you can turn up and down, and every time you turn one the score changes. The algorithm then simply randomly selects knobs to turn and turns the knob to the position maximizing the score with the remaining knobs fixed.

In [4] the author simply imposes a uniformly spaced fixed grid of allowed tweezer positions and then the algorithm simply tests all possibilities, instead of elaborately trying to find the exact best position. This is a classic and simple way of trying to solve a one-dimensional problem, and as it turns out nothing more is required. Computing the fidelity for a given set of positions requires multiplication of \(N\) unitary matrices (with end and start state), one for each position parameter in the defined order. This allows the maximization to be optimized since the product of the matrices corresponding to \(x_1, \ldots, x_{i-1}\) and \(x_{i+1}, \ldots, x_N\) can be precomputed making the iteration over all allowed tweezer positions rather efficient. The algorithm takes mere minutes to find high fidelity solution from random initialization.

There is actually another degree of freedom in BringHomeWater, namely the amplitude of the controllable tweezer. In most of the discussion below we keep this amplitude fixed at its maximal value, the reason why is explained later.

## 2 The Game

The BringHomeWater game is defined by a setup with two tweezers and an atom, and the job is to use one of the tweezers to move the atom, initially caught by the other tweezer, to some fixed position. The tweezer that has initially caught the atom cannot be controlled and does not change i.e. it does not move and is
turned on at a fixed amplitude. The other tweezer is controllable by the user/algorithm, this tweezer can move around and its amplitude is controllable as well.

The job of an algorithm is to use the controllable tweezer to move the atom, intuitively by moving the tweezer to the atom and then back to the target position while ensuring that the atom does not escape the tweezer (spill over in BringHomeWater analogy) using as little time as possible. See [2] for more details.

We note that, surprisingly, the numerical optimization problem that is solved in BringHomeWater is not really defined in the paper [1], neither in the main text, methods, or supplementary materials. In particular several important parameter values are not given. A description and the parameters can be found in [4], and we follow the description given there.

First, given a parameter $L$ the atom and the fixed tweezer is placed at position $x_{\text{start}} = L/2$. The task is to move the atom to $x_{\text{end}} = -L/2$ using the controllable tweezer, that in the game is initially placed at $-L/2$ as well. It seems that it is not important where the controllable tweezer is initially placed since we can move it while it is turned off. It is, however, not stated anywhere in [1] if an amplitude of zero is allowed or if there is some lower bound. For completeness we test both possibilities in the numerical experiment section below and show it does not really make any difference.

The game is usually considered by giving a fixed duration of time to move the atom. The larger the duration the easier the problem. Given a fixed time duration, a protocol for the controllable tweezer is a list of positions and corresponding amplitudes of the controllable tweezer for a set of time-steps. The quality of a protocol is determined by the fidelity under the condition that we have zero velocity of the atom at the start and end of the protocol. Denote the desired target state of the atom by $\langle \phi \|$ and the initial state as $\| \psi \rangle$. For a given protocol $P$, the state of the atom after the protocol is given by the time evolution operator $U_P$ applied to the initial state. The fidelity of the protocol $P$ is formally defined as

$$F(P) = | \langle \phi | U_P | \psi \rangle |^2$$

which is what we want to maximize over choice of protocol $P$. To be able to compute the fidelity we need to compute the start and end state vectors and the required evolution operator matrices that make up $U_P$. This is specified in Appendix A.

3 Experiments

In this section we experimentally show that the Stochastic Ascent algorithm as presented in [4] is far superior to human players and also outperforms the algorithms presented in [1] that use player solutions as initial seeds for other algorithms.

In all experiments, the amplitude of the controllable tweezer is set to 160 if nothing else is stated following [4]. Extended Data Figure 3 of [1] indicates that an amplitude of $A \sim 150$ may be a maximal value for the amplitude of the controllable tweezer since this is the maximal value used by the algorithms in [1], that states in regards to KASS that “Optimizations showed that $A = -150$ was the optimal value”. This seems to mean that the 150 is the best value to use for the algorithms used in [1]. Here we use the values provided by [4] taken from the BringHomeWater game itself, i.e. we use the maximal allowable amplitude as provided to the gamers. As mentioned, it is also not clear from [1] whether an amplitude of zero for the controllable tweezer is allowed in BringHomeWater or there is some lower bound for the amplitude of the controllable tweezer.

The reason for fixing the amplitude at the maximal value is based on two observations. First, the analytic results in [4] shows consistently better results when the amplitude is larger. Second, since it is a linear control field in the Hamiltonian, Pontryagin theorem tells us that the solution is (almost) always on the boundary. This means that $A = 160$ or $A = 0$ at any time and you can switch between these two. But since $A = 0$ means that the atom is not being confined the most likely optimal solution is that $A = 160$ all the time (besides maybe for the first step).

To cover the different possibilities we add restricted versions of the Stochastic Ascent algorithm from [4] where the first position of the controllable tweezer is always $-L/2$, and a version where additionally the amplitude of the controllable tweezer is set to 150.
3.1 Full Comparison

Figure 2: Fidelity for different BringHomeWater solutions as a function of duration $T$. Each algorithm Quantum Speed Limit (min duration $T$ with fidelity above 0.999) is shown in parentheses. Fixed start means that the controllable tweezers first position is fixed to $-L/2$ for the Stochastic Ascent. For all Stochastic Ascent Durations $N$ is defined such that $T/N = 0.0025$ as in Figure 6 in [4]. We use 201 possible position for the controllable tweezer which allow putting the controllable tweezer at $x_{\text{start}} = -0.55$ as required.

In Figure 2 we have shown a comparison of the Stochastic Ascent algorithms with the players and the derived algorithm from [1]. This plot clearly shows the superiority of the simple Stochastic Ascent algorithm over all players by a massive margin. The gamers are not even close to the performance of the simple algorithm. This clearly contradicts the assertion in [1], that “players succeed where purely numerical optimization fails”.

Comparing with the algorithms derived by combining player solutions with different complicated quantum control algorithms, the plot shows that these are also clearly beaten by the simple algorithm albeit the difference is smaller. So while [1] claims that they harness player strategies and “outperform the most prominent numerical methods” the solutions provided are outperformed by a much simpler and actually quite non-optimized standard algorithm.

The difference between fixing the first position of the controllable tweezer at $-L/2$ and allowing the algorithm to optimize this step is very small and makes essentially no difference for the result. If we reduce the amplitude of the controllable tweezer to 150 the results are also slightly worse. This reduction in QSL is predicted by [4] that says QSL should follow $1/\sqrt{A}$ and the reduction we see in the experiments fits well with that. If we allow a higher $T/N$ ratio and tune other parameters like the number of possible tweezer positions the QSL may possibly be reduced for all versions of the Stochastic Ascent algorithm. Since the results of the Stochastic Ascent are already clearly superior, and thus verifying [4], we do not follow this direction further in this note.

To experimentally verify that the best algorithm found indeed starts by a fast move from the starting position at $-L/2$ to where the atom is caught by the fixed tweezer we have visualized a protocol for $T = 0.16$ that achieves a fidelity above 0.999, the goal set in [1], in Figure 3.
Figure 3: Tweezer positions for a protocol with fidelity > 0.999 at time $T = 0.16$, $N = 64$ with fixed starting position at $-L/2$.

### 3.2 Fidelity Traces

We have also created the fidelity trace plot from ([4] Figure 6), that zooms in on one the most volatile durations at $T = 0.1$. We have created the plot for the two invocations of the algorithm as above, one invocation where the first position is optimized in the algorithm and one invocation where the first position is fixed at $-L/2$, see the figures in Table 1. A simple visual comparison shows that our implementation gets results that are essentially the same as in [4]. As noted in [4] the algorithm converges after very few iterations over the parameters, each round taking only a few seconds on a laptop.

The simple conclusion is that the algorithm essentially gets the same fidelity all the time with different but very similar protocols having found local minima (in the sense that one parameter change cannot increase the score) with almost the same fidelity.

Fixing the first position of the tweezer does not change this, only reduces the fidelity achieved slightly for this duration as was also apparent from Figure 2. It is not surprising that fixing the first position of the controllable tweezer does not change much, since fixing the first position essentially just corresponds to a slight perturbation of the starting state from $|\phi\rangle$ to $U_{-L/2} |\psi\rangle$, where $U_{-L/2}$ is the evolution operator the first step with the controllable tweezer at $-L/2$, and then running the algorithm from this configuration instead. The simple algorithm does not care about the starting and ending state, and if we add the fact that we could have used the smallest possible amplitude allowed for the controllable tweezer for that position of the protocol the change in the starting state would be even smaller. In the extreme case of the amplitude of the controllable tweezer being zero at the first step the only force acting on the atom is the fixed tweezer that already has the atom caught. The difference in fidelity between the two algorithms in Table 1 is around 0.04. Running Stochastic Ascent while not fixing the first position for a duration $T = 0.0975$ and $N = 39$ (maintain the same $T/N$ tradeoff) the fidelity achieved hovers closely around 0.492, essentially the same as for Stochastic Ascent with fixed first position with one additional step and time unit. As the fidelity
Fidelity Trace, $N = 40, T = 0.1$, 128 allowed positions for the controllable tweezer, 100 traces as in [4]. Fidelity range of experiment, 0.533 - 0.537. A single protocol has been outlined in purple.

Table 1: The fidelity trace and protocol plot as shown [4] (note that the protocol plot is in the arXiv version [9] and that the amplitude of the controllable tweezer is 160). The fidelity achievable varies greatly around $T = 0.1$ losing a step means losing $T/N$ time which matches the difference between the two versions. Since we have explained the difference we saw no reason to include an experiment with a fixed starting point for the controllable tweezer at $-L/2$ while allowing amplitude of zero.

Note that the protocols that are found by both version of the algorithms are the simple intuitive ones as above: Move the tweezer to the atom, move the tweezer back slowly, and shake it on the way to avoid spilling.

3.3 Superposition states

As mentioned in [4], the BringHomeWater (Quantum Moves) is not really a good representation of a truly quantum state since it is basically a Gaussian wave packet moving around, and such states are well-understood in a classical sense. To add as much quantum complexity as possible to the BringHomeWater challenge, [4] suggests replacing the target state $\langle \phi |$ with a superposition of the target state and end the initial state, i.e. $\langle \phi | q_1, q_2 = q_1 \langle \phi | + q_2 \langle \psi |$. In Table 2 we have the fidelity achieved for different durations and a representative high fidelity protocol for a specific superposition. Now this protocol, looks much more complex than in the "vanilla" BringHomeWater and it would be fun to see if human players can get high fidelity here (as the simple stochastic ascent algorithm can). The Stochastic Ascent algorithm is able to solve the superposition
version of the problem as well, albeit as noted in [4] the variance of the result of a run is higher for this problem than it was for the “vanilla” BringHomeWater, and more runs of the algorithm may be required to achieve high enough fidelity. This indicates that this is more complex optimization problem.

Together with the previous experiments this shows that the simple algorithm efficiently solves the BringHomeWater game in any configuration without any prior information with better performance than all results provided in [1].

3.4 Gradient Based Algorithms

Seeing that a basic standard algorithm from mathematical optimization completely outperforms all players and the best results in [1] that has used more than ten thousand computer hours raises some questions. How bad is this KASS algorithm which is claimed to be the best algorithm for BringHomeWater in [1]. Since we already know how to compute the fidelity of a given protocol (list of positions) we can straightforward implement a gradient ascent algorithm, which in quantum control is known as the GRAPE algorithm. For this case, GRAPE is no more than gradient ascent on the tweezer positions (the amplitude is still fixed to the maximal value). So instead of fixing a grid of allowed positions and iteratively improving the cost by picking one variable to optimize, GRAPE instead computes the gradient of the fidelity of all positions and takes a step along that direction. As mentioned earlier, this algorithm is surprisingly not mentioned [1] despite that it appears that it is a standard algorithm in quantum control. The implementation details can be found in Appendix A in short we implement the fidelity computation in TensorFlow and use automatic differentiation.

We have not elaborately tested gradient ascent, since there are so many variants and tools to improve on the quality and convergence of the methods. We just want to check how a basic implementation compare with the other solutions. The main question we need to consider is how to initialize the protocol before starting the iterative algorithm. Gradient ascent requires that we give the algorithm a reasonable starting point. Finding a bad one is not hard, for instance fixing all the initial tweezer positions at the left end or something similar as a starting position gives essentially zero gradient to work with and the gradient ascent algorithm fails to do anything. Gradient ascent does not search the space of solutions in some clever way,
the algorithms moves uphill until it dies down in a flat area, so if we start it at a poor place the results will be poor. No surprise there. To hinder the algorithm as much as possible, without killing it, we start by testing GRAPE using the classic and very poor strategy of assuming that we do not know anything about the problem and hence do uniform sampling of parameters in their range.

The results for the uniform initialization out-of-the-box GRAPE implementation is shown in Figure 4. From this figure it is obvious that GRAPE also clearly outperforms all players and drastically outperforms the KASS algorithm as well. The solutions from [1] where player solutions are used as seeds on the numerical optimization are still often better than GRAPE, but this is not surprising as they are given a better starting point which is usually required for achieving good results from gradient ascent.

It is not hard to make GRAPE with uniform initialization perform poorly for BringHomeWater. Just make the domain large enough such that random initialization in the domain puts the algorithm in a useless position. In this case gradient ascent has essentially no gradient to work with and cannot escape the low fidelity area where it started. If you happen to be in this regime, and cannot come up with any better initialization, one could argue that the obvious solution is to not use gradient based algorithms. Note that the gamers are shown a visualization of what to do and what may go wrong that works very differently where the state relative to the goal is always visualized for the gamer. So it is neither surprising nor interesting in this sense that it is possible for a gamer to beat some poorly initialized gradient based algorithm.

To gain a little more perspective into gradient ascent we try to remedy GRAPE to perform better. Obviously, we can if we find a good initial protocol. Let us use the theoretically derived adiabatic protocol for a single tweezer problem proposed by [4] as the initial protocol. As described by D. Sels in [4], the simple line gives the smallest magnitude of the acceleration. However, the line has the issue that the velocity does not cancel out at the beginning and the end. This is handled by using a third order polynomial instead in [4]. However, a gradient ascent algorithm may be able to take care of that problem, which it seemingly does. Starting with this simple initialization for gradient ascent by picking points on a line from end point to starting point gets a better QSL than the best hybrid algorithm from [1] in one try! The results are also shown in Figure 4.

If we think that is helping too much, maybe it is okay to just initialize with the zero vector. This corresponds to initializing the tweezer to sit in the middle between the atoms starting position and the target position. With this trivial initialization we can get a QSL of 0.18, pretty close to the best QSL of 0.176 reported in [1]. Sampling from a Gaussian with zero mean and small standard deviation leads to similar results.
These results are simple perspectives on the nature of BringHomeWater. They show how little is needed from the players to get good results when combined with some gradient based algorithm. Furthermore as pointed out by D. Sels in [4], the player solutions are really the intuitive ones that move the tweezer to the atom and then slowly back again with a few shakes to avoid spilling. Using such protocols (while ignoring the shakes) gives essentially the initializations that seems to be the main result achieved in [1], albeit they use a different algorithm that is seemingly worse than standard gradient ascent on top of it. From this perspective this hardly seems a big revelation. Thousands and thousands of gamers have played the game and have not come up with anything that, when combined with the best algorithms by [1], is as good as the basic Stochastic Ascent algorithm or the basic gradient ascent algorithm with an almost trivial initialization of maintaining constant acceleration (linear motion), and this linear motion is actually much better than all players for small durations. We also note that we only tested two basic algorithms from mathematical optimization and that there are many others that may outperform both gamers and the hybrid algorithms as well.

With these results in hand it is clear that human players and human intuition with the help provided by the game designers is clearly outperformed by basic theory and algorithms and as such has no relevance for BringHomeWater and in extension other more complicated problems for instance the superposition version of BringHomeWater as introduced by D. Sels. Note that this is essentially the opposite conclusion to the one put forward in [1].

4 Final Comments

We have experimentally verified the results of the simple Stochastic Ascent algorithm from [4]. Since the BringHomeWater is not fully specified in [1], we have tested different possibilities and shown that this does not change the performance of the simple Stochastic Ascent algorithm. We conclude that the results, discussion and conclusions in [4] are verified. The BringHomeWater challenge is not hard for numerical algorithms and numerical algorithms easily outperform any human player at any version of the game. It is also clear that KASS and Krotov are indeed a very poor choices for BringHomeWater. Even simple gradient ascent beats it with uniform initialization which as argued above is not a good choice.

KASS was claimed to best numerical algorithm for Bring Home Water in [1], which seemingly lead to the conclusion that gamers beat algorithms. As shown by D. Sels in [4] and verified here, this is clearly not true, gamers do not even come close to compare to basic algorithms. Stated differently, the conclusion from [1] that “Players succeed where purely numerical optimization fails” is false.

The other main result from [1] that “Using player strategies, we have thus developed a few-parameter heuristic optimization method that efficiently outperforms the most prominent established numerical methods”. As argued this can only be true if you define the most prominent numerical algorithms as KASS and CRAB, this is completely misleading since these algorithms are in fact poor for BringHomeWater as we have already argued. Again as shown by D. Sels in [4] and verified here the completely standard algorithm of Stochastic Ascent clearly falsifies this statement as well. Stochastic Ascent is simpler, it is more efficient, it does not require hundreds of trials to get the a good solution and it achieves better results. The same is true for a reasonably initialized gradient ascent. So this second main conclusion from [1] is also clearly wrong.

On a side note the Stochastic Ascent algorithm also contradicts the argument from [1], that “The choice of an initial seed is a central challenge in such complex optimizations” as BringHomeWater, since no fancy initialization, using human players or otherwise, is needed as shown by the results discussed here using Stochastic Ascent. If you use gradient ascent algorithms it is a trivial observation that you need to initialize properly.

We end with the final discussion in [4] that we believe we have verified in this note.

[4] “We conclude that Quantum Moves gamers did not devise strategies that go beyond the classical laws of physics [Maniscalco, Nature 532, 184 (2016)]; they also did not need to. Gamers simply came up with the expected intuitive strategy where you try to move the tweezer without spilling the “water.” They do not outperform any reasonable numerical method and at short times perform much worse than a theoretical designed counterdiabatic protocol.
This does not only falsify the thesis of Sørensen and co-workers [Sørensen et al., Nature 532, 210 (2016)], but it also poses serious questions about the generalizability of the result. Both the initial and target state in the BringHomeWater challenge are localized states in position space, which is also the space in which the tweezer can move. It thus allows to display everything in position space. It has been argued that this makes the game cleverly designed [Gibney, Nature 532, 160 (2016)], but it also makes the problem easy. Quantum mechanics itself is basis invariant and any physics can only be established by the trinity of the Hamiltonian, observable, and quantum states, in this case, the control, target, and initial states. To make the problem quantum mechanically interesting, at least one of them must be local in a basis that is sufficiently different from the others. In that respect, the only quantum mechanically interesting target state in Quantum Moves is the superposition of the atom in the tweezer on the left and on the right, with a well-defined phase difference between the two. So here is the real challenge: Can gamers even come up with any reasonable strategy (see Fig. 5) to this problem?

A  Stochastic Ascent Algorithm

The hamiltonians defining the system in BringHomeWater is defined as follows. Let $B$ be the amplitude of the fixed tweezer, then the initial state of the atom is the ground state of the Hamiltonian

$$ H = \frac{p^2}{2m} - Be^{-(x-x_{\text{start}})^2/(2\sigma)} $$

and the final state is the ground state of the Hamiltonian

$$ H = \frac{p^2}{2m} - Be^{-(x-x_{\text{end}})^2/(2\sigma)} $$

Let $A_t$ be the amplitude of the controllable tweezer, and $x_t$ be the position of the controllable tweezer at time $t$ then the time evolution operator Hamiltonian is

$$ H_t = \frac{p^2}{2m} - A_t e^{-(x-x_t)^2/2\sigma} - Be^{(x-x_{\text{start}})^2/2\sigma} $$

Following [4], we implement a Stochastic Coordinate Ascent algorithm for BringHomeWater as shown in Figure [1]. The algorithm does not optimize over the allowed amplitude of the controllable tweezer, keeping the amplitude fixed at the maximum allowed value all the time which is 160 as discussed earlier.

Secondly, the algorithm uses a reasonably small discrete set of positions for the controllable tweezer which is simply a uniformly spaced grid of some fixed size $s$. Note that the same grid is used for all steps of the protocol. This allows computing all the needed unitary matrices only once before the iteration starts.

Now consider the configuration space of the problem. Let $s$ denote the number of possible positions for the controllable tweezer. In any protocol we have to specify a number of steps that we can take, denote this by $N$. The configuration space is therefore $s^N$. For any reasonable values of $s$ and $N$, $s^N$ will be an astronomical number, and testing all protocols is not feasible. Of course, this is the case for any interesting optimization problem.

A.1  Numerical Computations

To compute the fidelity of a protocol, we use a mesh of $h_g$ of uniformly spaced points to discretize the Schrodinger equation and Finite Difference approximation for the Laplacian. This requires computing matrices for the three parts in the Schrödinger Equation (Equation [4]). First, define a uniformly spaced grid $gh$ over $[-1, 1]$ with $h_g$ uniformly separated points (including $-1$, +1) at distance $d_h = 2/(h_g - 1)$. For the Laplacian, generate a finite difference matrix as $\frac{0.5}{d_h^2}$ times a tridiagonal matrix with two on diagonal and -1
above and below the diagonal.

\[ t_{\text{mat}} = \frac{0.5}{d_h^2} \begin{bmatrix}
2 & -1 & 0 & 0 & 0 \\
-1 & \ddots & \ddots & \ddots & 0 \\
0 & \ddots & \ddots & \ddots & 0 \\
0 & \ddots & \ddots & \ddots & -1 \\
0 & 0 & 0 & -1 & 2
\end{bmatrix} \]

For the part of the Schrodinger Equation for the fixed tweezer, \( B e^{-(x-x_{\text{start}})^2/(2\sigma^2)} \), generate a diagonal matrix \( M_B \) where the \( i \)th entry is \( B e^{-(h_g[i]-x_{\text{start}})^2/(2\sigma^2)} \). In vectorized notation this can be written as

\[ \text{diag}(\exp((-((g_h - x_{\text{start}})^2)/(2\sigma^2)))) \]

where \( g_h \) is the list of grid points used for discretization.

For the final part of the equation for the controllable tweezer, \( A e^{-(x-x_{\text{tweezer}})^2/(2\sigma^2)} \), compute a matrix for each allowed position of the tweezer and for each such position, construct a matrix as for the fixed tweezer. Thus, for each tweezer position \( t_p \), make a diagonal matrix where the \( i \)'th entry is \( A e^{-(t_p-g_h[i])^2/(2\sigma^2)} \). For position \( t_p \) this can be written in vectorized notation as

\[ \text{diag}(A \exp(-(g_h - t_p)^2/(2\sigma^2))) \]

where again \( g_h \) is the list of grid points used for discretization.

Finally, for every potential tweezer position \( t_p \) with associated matrix \( M_p \) compute the Hamiltonian matrix

\[ h_p = t_{\text{mat}} - M_b - M_p \]

and to get the unitaries compute

\[ U_p = \exp(h_p \cdot (T/N)(-i)) \]

This defines all the required unitary time evolution matrices, and for a given protocol the time evolution operator is the a product of an the corresponding multiset of these ordered by the protocol. For the initial and target state compute the eigenvector with the smallest eigenvalue of the appropriate corresponding Hamiltonian matrix. In particular the starting state \( |\psi\rangle \) is the eigenvector corresponding to the smallest eigenvalue of \( t_{\text{mat}} - M_b \). For the target state, create the matrix \( \tilde{M}_b \) symmetric to \( M_b \) just for \( x_{\text{end}} \) instead of \( x_{\text{start}} \), i.e. a diagonal matrix with the \( i \)'th entry equal to \( B e^{-(h_g[i]-x_{\text{end}})^2/(2\sigma^2)} \). Now \( \langle \phi | \) is the eigenvector corresponding to the smallest eigenvalue of the matrix \( t_{\text{mat}} - \tilde{M}_b \). Notice that the controllable tweezer is turned off for both the start and end state.

### A.2 Implementing Algorithm

With the math behind us the the Stochastic Ascent algorithm can now be implemented with a fairly short python program using numpy and scipy (or any other language if preferred). We always use \( m = 1, B = 130, L = 1.1, \sigma = \frac{1}{8} \) as defined in [4]. The remaining parameters depends on the experiment. A description of the algorithm is also contained in [4] but for completeness we include (a shorter) one here in Figure 5.

In [4], \( s \), the number of positions available for the controllable tweezer is set to 128. With 128 as the number of tweezer positions we cannot even place the controllable tweezer at the atom or at the target position! For the general result of showing that the Stochastic Ascent algorithm easily and clearly beats human gamers 128 is more than enough. If it is important to get fidelity above 0.999 as the goal set in [1] it helps to increase this parameter for instance doubling it or setting it to 201 to ensure the controllable tweezer can be placed on \( x_{\text{start}} \) and \( x_{\text{end}} \) if needed. This will also of course make the algorithm slower by a constant factor. The grid size \( h_g \) for discretizing the Hamiltonians is a numerical precision parameter...
Stochastic Ascent Algorithm

Input Parameters $A, B, \sigma, T, N, s$

1. Compute target state, $\langle \phi |$, the initial state $| \psi \rangle$, and unitary matrices $U_0, \ldots, U_{s-1}$ (Appendix A.1).
2. Randomly initialize a protocol of $N$ random integers $[x_N, \ldots, x_1] \in \{0, 1, \ldots, s - 1\}^N$.
3. Let $I$ be random permutation of $\{1, \ldots, N\}$
4. for $i \in I$
   4a. Compute prefix and suffix of fidelity computation, $a = \langle \phi | \prod_{j=i+1}^{N} U_{x_j} | \psi \rangle$ and $b = \prod_{j=1}^{i-1} U_{x_i} | \psi \rangle$.
   4b. $x_i = \arg \max_{k \in \{0, \ldots, s-1\}} |a U_k b|^2$
5. If fidelity does not improve in step 4 then return current protocol, else goto step 3

Figure 5: More detailed Stochastic Ascent Algorithm for BringHomeWater (Quantum Moves) from [4]

which we set to 512, albeit for fast tests using a smaller value gives almost the same results. We checked that the discretization indeed changes very little to the results at that point. To fix the first position of the controllable tweezer in the protocol, initialize $x_1$ to $-L/2$ (or closest grid point) and then do not optimize that variable in the algorithm by not including it in the random permutation $I$. To allow more amplitudes than $A = 160$, just add the necessary unitary matrices as described above. We have only tested this with a few parameter settings. The results from these experiments indicate that the algorithm naturally slows down but still finds solutions of the same quality, and as the theory suggest only using the maximal value works.

A.3 GRAPE

To implement the gradient ascent algorithm we implemented the fidelity computation as described above in TensorFlow and used automatic differentiation. Only the positions of the tweezer are optimized, the amplitudes are always fixed to the maximal value of 160 as in Stochastic Ascent. The number of steps set as for stochastic ascent such that $T/N = 0.0025$. For each duration $T$ tested we ran several GRAPE invocations from random uniform random starting points. Compared to Stochastic Ascent, GRAPE is quite time consuming and in order to save time, we ran the main bulk of experiments using imprecise computation by simply setting a low value for the hamiltonian grid size $h_g$ and running each experiment only a limited amount of iterations. We then used the best protocol found like this, increased the grid size back by doubling and restarting the iteration, until we hit a grid size of 512 again as in Stochastic Ascent. Since ADAM [10] is the standard recommended optimizer in TensorFlow (albeit for deep learning) we use that for the experiments and played only with the learning rate parameter. For a full test one should really test different optimization methods and very their hyperparameters. Since such a large experimental evaluation is not the point of that note, we leave that to the interested reader. Since we only tested a few things we believe that even better results can be achieved for GRAPE with more experiments and tuning.

To ensure the tweezer starts as in the game we always fix the first of the $N$ position available to $-L/2 = -0.55$ and never update it in the algorithm. For initialization with a line with $N$ allowed points we arrange the remaining $N-1$ points from $L/2$ to $-L/2$ using numpy.arange.

References

[1] J. J. W. H. Sørensen et al., Nature 532, 210 (2016).
[2] https://www.scienceathome.org/games/quantum-moves
[3] https://en.wikipedia.org/wiki/Coordinate_descent
[4] D. Sels, Phys. Rev. A 97, 040302(R) (2018).
[5] V. F. Krotov, *Global Methods in Optimal Control Theory*, (Marcel Dekker, New York, 1996).

[6] Tommaso Caneva, Tommaso Calarco, and Simone Montangero, Phys. Rev. A **84**, 022326 (2011).

[7] Khaneja, Reiss, Kehlet, Schulte-Herbrüggen, Glaser, J. Magn. Reson. **172**, 296-305 (2005)

[8] Zahedinejad, Schirmer, Sanders, Phys. Rev. A **90**, 032310 (2014).

[9] D. Sels, https://arxiv.org/abs/1709.08766

[10] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization*, ICLR 2015