Minimal Variable Quantum Decision Makers for Robotic Control

Walter O. Krawec
Stevens Institute of Technology
Hoboken NJ, 07030
walter.krawec@gmail.com

ABSTRACT

In this report we describe our research involving the construction of quantum-based robotic controllers. By careful use of quantum interference as a computational resource and by utilizing only a linear number of elementary unitary transformations, we are able to construct systems which seem to provide a computational advantage even when simulated on a classical computer.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

Keywords

Quantum Computing; Evolutionary Robotics; Artificial Life

1. THE CONTROLLER DESIGN

This paper reports on our recent work involving the construction of efficient quantum-based controllers to be utilized as reactive robotic controllers. Unlike most previous work in quantum controlled robots (e.g., [5]), our controllers are meant to be operated using today's classical computers - they are quantum only in that they utilize certain nice mathematical properties of quantum computation. Note that designing "quantum inspired" but classical algorithms has seen interest and success in other fields including information retrieval [6] and certain instances of the graph isomorphism problem [2]. Our work builds on the ideas presented in [3] which introduced the idea of studying quantum based controllers simulated by classical computers for robotics and artificial life.

Our work has been analyzing an extended version of the quantum decision maker (QDM) described in [1], originally meant to model certain human behavioral paradoxes. We view it as the quantum analogue of the following process: given input state $\mathcal{S} = (s_1, \cdots, s_n) \in \{0,1\}^n$ (e.g., each s_i a sensor input), the controller, consisting of $N \geq n$ *event states* denoted X_i and m *action states* denoted A_i, transitions from a start state, to an event state (probabilistically depending on a preprocessing function f taking as input $\mathcal{S}$ and outputting transition rules for start to event states), then to an action state (the probabilistic transition rules for event states to actions are not based on $\mathcal{S}$ or f). Whichever action state is transitioned to is the action decided on (alternatively, the output are values p_i: the probability of transitioning to A_i).

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2598409.

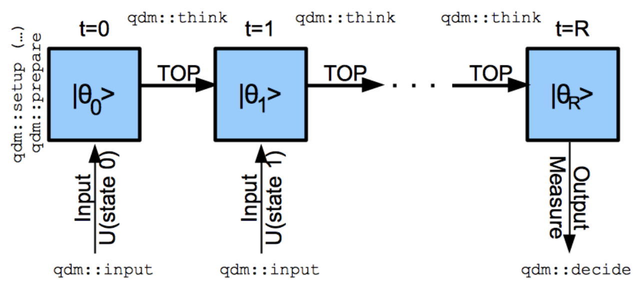

Figure 1: Process by which a QDM makes a decision (from [3]). Each $|\theta_i\rangle$ is the state of the QDM at time $t = i$ (usually represented as a vector). The $U(\text{state } i)$ are (typically unitary) operators which depend on the state of the agent at time $t = i$. TOP is a unitary matrix fixed regardless of the current state. The state at time $i+1$ is then $|\theta_{i+1}\rangle = TOP \cdot U_i\,|\theta_i\rangle$. This repeats R times (the *reaction rate*) before a decision is made via a simulated measurement. Following this, the process repeats.

We extend the model of [1] to support n event states $\{X_i\}$ (as opposed to two) where n is again as above. Though we extend the number of supported inputs, we still consider only two actions (for now). This results in a Hilbert space of dimension $2n$ - see [4] for an introduction to quantum computing. Instead of running the sensor state $(s_1, \cdots, s_n) \in \{0,1\}^n$ through f, we will simply start the QDM in an equal superposition of all input states X_i where $s_i = 1$ (we assume at least one $s_i = 1$ - otherwise we add a "dummy" sensor which is equal to one if all others are zero). A unitary matrix TOP (notation from [3]) is then applied and a (simulated) measurement is made resulting in a decided action. We call this QDM the *Event Based QDM* (ebQDM). The most general TOP matrix however requires $4n^2$ variables to encode and optimize over for learning.

As described in [7], an arbitrary $M \times M$ unitary matrix ($M = 2n$ in our case) may be decomposed as the product of

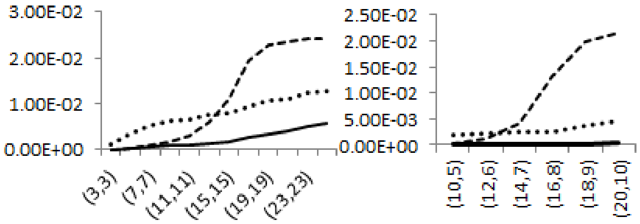

Figure 2: Average squared error for data sets (n, P). Each set (n, P) was evaluated and averaged using 400 independent trials. Solid line: mvebQDM; Dotted line: CDM-Fixed; Dashed line: CDM-Var.

$O(M^2)$, elementary unitary transformations. Changing the basis used in [1] from $|X_a, A_b\rangle$ to the standard computational basis $|(b-1)n+a\rangle$ for $a = 1, \cdots, n$; $b = 1, 2$, let $E^{(i,j)}(\phi)$ be the unitary operator sending $|i\rangle$ to $\cos\phi\,|i\rangle + \sin\phi\,|j\rangle$ and sending $|j\rangle$ to $-\sin\phi\,|i\rangle + \cos\phi\,|j\rangle$ (see Equation 3.1 of [7]). Then we set TOP to be the (unitary) matrix:

$$TOP = \prod_{i=1}^{n-1} E^{(n-i,2n-i)}(\phi_i) \prod_{i=0}^{n-1} E^{(n-i,2n)}(\phi_{2,i}).$$

This requires $2n-1$ variables to describe (the "ϕ" values). We call the ebQDM utilizing this particular TOP, the *Minimal Variable Event Based QDM* or mvebQDM. Note this is just one possible "minimal" TOP - others might perform better or worse for some problems.

To operate this QDM, given input $\mathcal{S} = (s_1, \cdots, s_n)$, we create the $2n$ vector: $\frac{1}{\sqrt{p}}(s_1, \cdots, s_n, 0, \cdots, 0)$ where p is the number of non-zero s_i. Considering the abstract interface of [3] (see Figure 1), this is handled by the `qdm::input` command (not that it greatly matters, but this input process may be done via a unitary operator). Next we multiply by the TOP matrix, described above (`qdm::think`). Finally, we compute p_1, the probability that action A_1 is chosen (`qdm::decide`). This is simply the sum of the squares of the first n entries of the resulting vector after multiplication. For robotic control, this value p_1 is either used directly by some other process or is used to "pick" from A_1 or A_2 (the decided A_i is then translated to an appropriate action). The reaction rate for this QDM is $R = 1$. Training is accomplished by adjusting the $2n-1$ ϕ_i and $\phi_{2,i}$ variables.

To evaluate the mvebQDM with this TOP, we first test its ability to model random data points. We compare with two different versions of classical devices: the first uses $2n-1$ event states and a fixed f. This is denoted CDM-Fixed (Classical Decision Maker). The second has n event states and a variable function f which could be trained to map $n-1$ distinct states $\mathcal{S}$ (out of $2^n - 1$ possible non-zero states) to any one of the first $n-1$ event states; all others being mapped to the n'th input state X_n. This we denote CDM-Var (note it is equivalent to a list of n `IF-THEN-ELSE` statements). Transition rules for events to actions are specified by a single value q_i for each X_i. Thus, both these classical processes require $2n-1$ variables to encode and learn.

Our tests were performed on data sets denoted (n, P). For each $i = 1, \cdots, P$, we chose a random (non-zero) input state $\mathcal{S}_i \in \{0,1\}^n$ and a random probability $p_i \in [0.2, 0.8]$ (this interval chosen to help avoid certain range limits in the QDM - this bound also benefits the CDM). For each model, a genetic algorithm was run for 200 iterations (regardless of n and P; also population size of 200 was used) to find a solution whereby action A_1 was chosen with probability p_i given input $\mathcal{S}_i$ ($\forall\ i = 1, \cdots, P$). Note that, given sufficient time, the CDM-Var should be able to model exactly those points with $n \geq P$, however the search space grows exponentially in n. Also note that, if we knew exactly which sensor states $\mathcal{S}_i$ were to be picked, we could construct preprocessing function f manually and the CDM-Fixed would model the data perfectly (if $2n-1 \geq P$). The mvebQDM however, using only a standard GA, outperforms both for large enough n. See Figure 2. This leads us to conjecture that the QDM could be very useful as a reactionary controller for evolutionary robotics if only a linear number of states are important, if it is unknown what these states actually are, and if the input space is large. Furthermore, this advantage is gained on a classical computer without any quantum hardware required.

We also tested the mvebQDM on various evolutionary robotics tasks (here the value p_1 of choosing A_1 is used directly for motor control: $p < .3$ implying "move left", $.3 \leq p \leq .6$ is "forward", else right - we can get away with this as this is a simulated quantum process). We are currently collecting additional evidence, however we have observed that, when the agent's input space is large and unstructured (generally over 200 different possible states), however if only $O(n)$ of them are actually "important" (there are only a few "distinguished" states that arise during operation), then we've observed the QDM can greatly outperform a similar classical device. However when the input space is small, or well structured, the classical device performs just as well or better. More experimentation, however, is required. For more details, see:
`walterkrawec.org/robots/gecco14_paper.html`.

2. REFERENCES

[1] J. R. Busemeyer and Z. Wang. Quantum information processing explanation for interactions between inferences and decisions. *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pages 91–97, March 2007.

[2] Brendan L Douglas and Jingbo B Wang. A classical approach to the graph isomorphism problem using quantum walks. *Journal of Physics A: Mathematical and Theoretical*, 41(7):075303, 2008.

[3] W.O. Krawec. On the application of quantum decision making to artificial life. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3323–3330, 2013.

[4] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, MA, 2000.

[5] A. Raghuvanshi, Y. Fan, M. Woyke, and M. Perkowski. Quantum robots for teenagers. In *Multiple-Valued Logic, 2007. ISMVL 2007. 37th International Symposium on*, May 2007.

[6] Alessandro Sordoni, Jian-Yun Nie, and Yoshua Bengio. Modeling term dependencies with quantum language models for ir. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 653–662. ACM, 2013.

[7] Karol Zyczkowski and Marek Kus. Random unitary matrices. *Journal of Physics A: Mathematical and General*, 27(12):4235, 1994.