Applications of Quantum Algorithms to Partially Observable Markov Decision Processes

R. D. Rosenwald*, D. A. Meyer†, and H. A. Schmitt*

* Raytheon Missile Systems
P.O. Box 11337
Tucson, AZ 85734-1337, USA
e-mail: rdrosenwald@raytheon.com

† Project in Geometry and Physics,
Department of Mathematics
University of California/San Diego
La Jolla, CA 92093-0112, USA
e-mail: dmeyer@math.ucsd.edu

Abstract

Due to the enormous processing gains that are theoretically achievable by using quantum algorithms instead of classical algorithms to solve rather generic classes of numerical problems, it makes sense that one should evaluate their potential applicability, appropriateness, and efficiency for solving virtually any computationally intensive task. Since many types of control and optimization problems may be couched in terms of partially observable Markov decision processes (POMDPs), and since solutions to these types of problems are invariably extremely difficult to obtain, the use of quantum algorithms to help solve POMDP problems is investigated here. Quantum algorithms are indeed found likely to provide significant efficiency improvements in several computationally intensive tasks associated with solving POMDPs, particularly in the areas of searching, optimization, and parameter optimization and estimation.

1 Introduction

In the last decade or so, two major new areas of study have arisen in computer science and control theory. They are the quantum approach to computing and a rather all-encompassing modeling approach that addresses how agents realistically interact with their world. The latter field of study is called "partially observable Markov decision processes." The following subsections briefly describe the background of each of these two new fields. Section 2 applies quantum algorithms to POMDPs; followed by the last two sections on future directions and conclusions.

1.1 Background on quantum algorithms

In 1994, Shor [1, 2] discovered an algorithm for factoring integer numbers that, when implemented on a hypothetical quantum computer, would produce results exponentially faster than the best algorithms on a classical (conventional) computer. This discovery produced a firestorm of controversy and activity that continues to this day.

In some quarters it produced great consternation. It was correctly noted that if a working quantum computer with a sufficiently large number (~hundreds) of quantum bits could indeed be built, then the security of modern public-key cryptographic systems would be seriously threatened. Basically, the reasoning goes as follows: cryptographic codes are hard to break because finding the prime factorization of a large composite number is an extremely difficult and time-consuming task for classical computers (despite using the latest, most efficient algorithms on massively parallel systems). With a quantum computer running Shor's quantum algorithm, however, the required factorization time would be exponentially quicker, effectively making code-breaking a rapidly accomplished, routine task. Even the remotest possibility of successfully building such a quantum computer may have already changed the modus operandi of security agencies, tipping them towards increased self-censorship of their more sensitive messages. This situation is not as farfetched as it originally seems, since the "proof of concept" experiment has already been successfully performed: Vandersypen *et al.* [3] built a 7 qubit quantum computer and used it to factor the number 15. They used an exotic molecule containing 7 spin-1/2 nuclei in liquid solution and nuclear magnetic resonance techniques to accomplish their feat. Unfortunately, this particular technology is not scalable to larger numbers of qubits.

Other researchers, however, *c.f.*, Levin [4], treated the discovery of Shor's algorithm with a healthy dose of skepticism, mainly due to the anticipated extreme physical difficulties involved with actually building a large working quantum computer. (For example, a major difficulty lies in

maintaining coherent states for a sufficiently long period of time—long enough to finish the computational task. There are two ways of doing this: reduce environmental noise that interferes with the calculation, and use techniques of quantum error correction (*c.f.*, Gottesman [5] and Steane [6]) to mitigate the bad effects. Many other researchers, however, were optimistic and hailed the new development with great enthusiasm, hoping that previously intractable problems, particularly some of the key problems in their respective fields, could be solved quickly and routinely via quantum computers. Despite the naysayers, the overall impact of the discovery of Shor's quantum algorithm was a rapid increase in the monetary resources and number of researchers that became focused on this and related areas.

Several years after Shor's breakthrough, Grover [7] discovered a quantum algorithm for searching unstructured lists; it yielded a quadratic speed-up over classical search algorithms. More recently, Farhi *et al.* [8] and Hogg [9] proposed novel quantum algorithms that use adiabatic evolution to solve optimization problems. Doherty *et al.* [10] introduced a theory of quantum feedback control, comparing and contrasting it to classical optimal control theory, focusing in particular on their key differences, such as the disturbing influences that measurements may have on quantum systems. Verstraete *et al.* [11] presented a framework for sensitivity optimization in quantum parameter estimation and Meyer [12] considered game theory from the perspective of quantum algorithms. These and the Shor and Grover quantum algorithm discoveries were anticipated decades earlier by Feynman [13, 14], who had predicted that such radical improvements in computing capabilities would result from properly applying quantum mechanical effects.

1.2 Background on partially observable Markov decision processes

Markov decision processes (MDPs) may be used to model a system if the state of the system is known at all times. POMDPs (partially observable MDPs), on the other hand, are more physically realistic generalizations of MDPs. They are designed to handle or model the more robust and frequently occurring "real world" scenarios where the state information is only partially observable. Because exact states are not known, one is forced to introduce probability distribution functions into the analysis; this is what makes POMDPs so much more difficult to analyze than MDPs. Indeed, POMDPs are capable of addressing the central problem of Artificial Intelligence: making optimal decisions under uncertain conditions. POMDPs have even been proposed as models for the human thought process. These are extremely challenging and general "Operations Research" type problems, certainly general enough to contain the tough problems from the field of optimization and control.

From Littman [15], "A POMDP…is a model, originating in the operations research literature, for describing planning tasks in which the decision maker does not have complete information as to its current state. The POMDP model is a convenient way of reasoning about tradeoffs between actions to gain reward and actions to gain information."

The probabilistic, *i.e.*, incomplete knowledge, aspect that is central to POMDPs is quite similar to and reminiscent of certain aspects of quantum mechanics. For example, the Heisenberg uncertainty principle of QM states that a pair of complementary variables (*e.g.*, a particle's position and momentum) cannot simultaneously be known to infinite precision. Similarly, probabilities are built into the very foundations of QM, with, for example, the probability density of an electron cloud being calculated as the square of the electron's quantum mechanical wave function.

Regarding the difficulty of solving POMDPs, Zhang [16] states: "It is known that finding the optimal policy for even a simplified finite horizon POMDP is PSPACE-complete. Since this is a broader problem class than NP, the result suggests that POMDP problems are even harder than NP-complete problems." NP stands for the computational complexity class "nondeterministic polynomial". It is perhaps the quintessential type of operations research problem, one whose solution time scales exponentially with the input size of the problem. The Traveling Salesman Problem is an archetypical example of a problem from the complexity class NP. (Aaronson [17] provides an excellent resource for information on these and other computational complexity classes, with his current list containing a "zoo" of about 400 different entries.)

To understand POMDPs, it is easier to start with a description of what a Markov decision process (MDP) consists of. (In contrast to the situation with POMDPs, MDPs are "completely observable.") Paraphrasing from Cassandra's website [18], the four components of an MDP model (that also happen to be components of a POMDP model) are:

A Set of States—When making a decision, you need to consider how your actions will affect things. The state is the way the system currently exists and an action will have the effect of changing the state of the system.

A Set of Actions (or Controls)—The actions are the set of possible alternatives that you can make. The problem is to know which of these actions to take, given a particular state of the system.

State Transition Probabilities (or Effects of Actions)—The transitions specify how each of the actions change the state. Since an action could have different effects, depending upon the state, one needs to specify the action's effect for each state in the MDP. The most powerful aspect of the MDP is that the effects of an action can be probabilistic. MDPs allow one to specify more complex actions by allowing one to specify a set of resulting states and the probability that each state results.

Immediate Rewards (Values of Actions)—To automate the decision making process, one must be able to have some measure of an action's value so that one can quantitatively

compare different actions. Thus, the immediate value for performing each action in each state is specified.

Baxter *et al.* [19, 20] describe the several additional components that are present in a POMDP model; they are:

Observations—The measured values of data (possibly noisy) that are obtained from the state of the system.

Observation Process or Model—The probability of obtaining certain observations, given a particular state of the system.

Stochastic Policy—The probability of implementing certain actions/controls, given particular observations and specified values for the adjustable parameters.

Adjustable Parameters—The quantities under control of the observer; for example, the size and shape of smoothing kernels.

To help clarify matters, it is useful to reiterate the eight different components of a POMDP given above in the context of a concrete example, namely, that of a target tracking problem where one has access to a time sequence of images collected by a single sensor device. In equations (1-8) below, the subscript 't' always refers to a sort of pseudo-time, the computational step number (an integer) in the analysis process. It is not to be confused with the physical time values associated with each of the snapshot images taken by the sensor device while it is collecting data.

- States

$$X_t \in s; s = \{1,2,\ldots,n\} \tag{1}$$

The set of target tracks s at computational step number t.

- Observations

$$Y_t \in y; y = \{1,2,\ldots,M\} \tag{2}$$

The data cube voxel values y at computational step number t, *i.e.*, the pixel values in the time sequence of images. These voxel values would typically be indexed by three coordinate values: row number, column number, and physical time (or image number).

- Actions/Controls

$$U_t \in u; u = \{1,2,\ldots,N\} \tag{3}$$

The action/control decision u made at computational step number t, on whether to drop/add tracks.

- Observation Process ν

$$\Pr(Y_t = y \mid X_t = i) = \nu_y(i) \tag{4}$$

The probability of data cube voxel values y, given target track information i, all at computational step number t.

- Stochastic Policy μ

$$\Pr(U_t = u \mid Y_t = y) = \mu_u(\theta, y) \tag{5}$$

The probability of dropping/adding tracks via action/control decision u, given data cube voxel values y and adjustable parameters θ, all at computational step number t.

- Rewards

$$r : s \to R \tag{6}$$

This is the figure of merit function for the target tracker; a mapping from the target track space to a real number.

- Adjustable Parameters

$$\theta \in R^K \tag{7}$$

These are parameters that describe the target tracking algorithm/mechanism. Several examples of these are the following: Gaussian smoothing values, receiver operating curve characteristics, constant false alarm rate (CFAR) settings, *etc.* Together, they are set up as elements of a K-dimensional real-valued space.

- Transition Probabilities

$$\Pr(X_{t+1} = j \mid X_t = i, U_t = u) = p_{ij}(u) \tag{8}$$

The probability of target track j at computational step number t+1, given target track i and the given drop/add control action at computational step number t.

To help envision this example of a target tracking scenario, data "cubes" consisting of voxel elements are shown in Figures 1 and 2 below. The original (raw) data "cube" (Figure 1) is composed of individual voxels (volume picture elements) assembled from a series of snapshots (standard

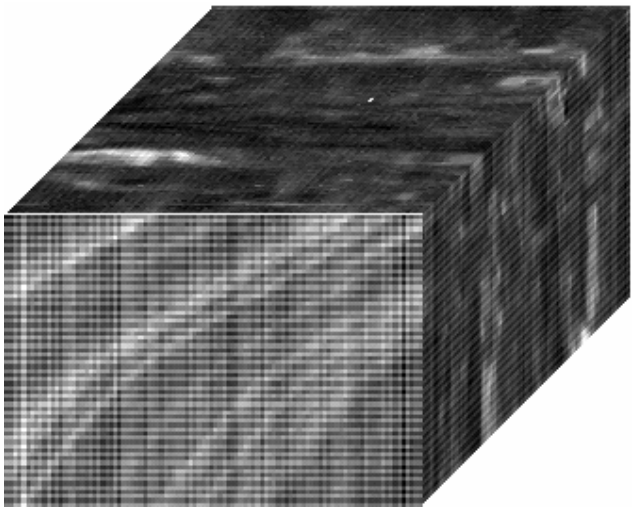

Figure 1: Original Data Cube of Voxels

row-column pixel images) collected by a single sensor. The first snapshot picture lies at the bottom plane of the cube; subsequent pictures taken at progressively later times are stacked vertically on top of it, filling out the data "cube". The signal processing of data in this form is also known as space-time adaptive processing (STAP); see the book by Guerci [21] for details.

The processed data "cube" (Figure 2) is composed of voxels that have been manipulated (processed) by a number of computational time steps. In the pictorial example shown here, Weickert's [22] anisotropic diffusion approach to image processing has been generalized from its original 2-D environment, that of processing individual images, to one higher dimension—processing time sequences of images. In its original, simpler 2-D form, pixel intensity values in an image are smoothed in the direction parallel to object boundaries and enhanced in the direction perpendicular to object boundaries. In its generalized, 3-D version, voxel intensity values in a data cube (time sequence of images) are smoothed along (*i.e.*, parallel to) object or track boundaries and are enhanced across (*i.e.*, perpendicular to) object or track boundaries. In both the 2-D and 3-D cases, diffusion tensors D are computed from the local neighbor-

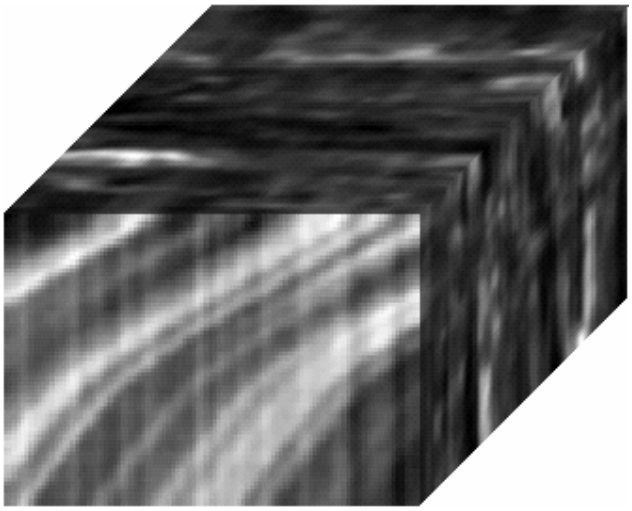

Figure 2: Processed Data Cube of Voxels

hood of intensity values I (averaging occurs in spatial dimensions alone for the 2-D case and with the temporal dimension included for the 3-D case). These D tensors are then substituted into the PDE that implements the anisotropic diffusion process, namely:

$$\partial I / \partial \tau = div(D \cdot \nabla I) \qquad (9)$$

The variable τ represents a sort of pseudo-time, during which the intensity values I are allowed to gradually diffuse according to Equation (9). For the 2-D case, the 'div' and '∇' operators involve partial derivatives along image rows and columns and the diffusion tensor D is a 2 by 2 matrix. In the 3-D case (shown in Figures 1 and 2), the 'div' and '∇' operators involve partial derivatives along not only image rows and columns but also along the vertical (time) axis, and the diffusion tensor D is a 3 by 3 matrix.

A primary goal of the processing for this target tracker example case involves forming a segmentation of the data "cube" into target tracks and everything else. Segmentation at least starts to occur in the anisotropic diffusion phase that transforms Figure 1 into Figure 2. Cottet and El Ayyadi [23] address the issue of when to stop the diffusion process so that a "hand-off" can occur to other needed processing; there are other ways of selecting a stopping point. Meandering, cylindrical-like data "tubes" that extend through the data "cube" from bottom to top (past to future) could represent the "world lines" of targets or the occurrence of ignorable items in the background clutter. Several conclusions can be drawn from the appearance of Figure 2. Looking at the front panel of the cube, one can see that the sensor was panning the scene from right to left, first at a slow rate, then at a faster rate. One can also tell that only a very poor (if any) non-uniformity correction (NUC) was applied to the image sensor, due to the systematic pixel biases that remain in the data cube, appearing as vertically oriented pattern lines in the front panel.

After the first, segmentation phase of data processing is accomplished, the selection and labeling of potential target tracks from amongst the set of candidate "tubes" lying within the data cube is the second major goal of the POMDP tracker model. Many other tricky, 'judgment-type' decisions must be made in order to complete the construction of a model tracker. For example: What restrictions should be placed on target locations, velocities, and accelerations? What conditions should be placed on creating, terminating, merging, and bifurcating tracks? What probabilities should be used as cutoff values for the above decisions? When should the next group of image snapshots be optimally added to the analysis? Then, after answering all of these types of questions, before one can have any confidence in the tracker, extensive testing must be performed, using appropriate data. None of these latter portions of the tracker analysis are carried out here, but they must be included in any decent POMDP model of a tracker.

Finally, what does it mean to "solve" a POMDP? Basically, it means finding an optimal policy (and set of adjustable parameters) for the POMDP. A policy is a means of selecting an action to take at a given time point (computational step number), based on one's belief state for the system. Policies are compared based upon their value functions, which are weighted time averages of reward functions. Algorithms for solving POMDP problems are divided into at least two classes: value iteration algorithms and policy iteration algorithms (*c.f.*, Baxter *et al.* [19, 20]).

2 Applying quantum algorithms to POMDPs

Before considering the potential applications of quantum algorithms, we first draw some comparisons between quantum and classical computing. In classical computers, the bit (or Cbit, for classical bit) is the basic unit for storing information. In quantum computers the corresponding storage unit is the quantum bit (shortened to qubit, or Qbit). An excellent tutorial on Cbits and Qbits is available in

Mermin [24], with more extensive background and detailed information available in the textbook by Nielsen and Chuang [25]. Very briefly, there are several key differences between Cbits and Qbits. Whereas Cbits may only take on the values (states) of *0* or *1* and are always explicitly available, Qbits are normalized complex linear combinations of two states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{9}$$

The complex numbers α and β are determined only probabilistically, that is, after measurement the qubit is either in the state $|0\rangle$ with probability $|\alpha|^2$ or in state $|1\rangle$ with probability $|\beta|^2$. Normalization guarantees that

$$|\alpha|^2 + |\beta|^2 = 1 \tag{10}$$

When one has n Cbits, there are 2^n special orthonormal states; the corresponding set of n Qbit states, however, resides in an enormously larger space, a Hilbert space—the set of all linear combinations of classical basis states with complex coefficients (amplitudes). This is a primary contributor to the algorithmic advantage of quantum algorithms over classical algorithms. Unitary operators are the basic tools of quantum algorithms, they are linear norm-preserving operators on the n Qbit states, mapping one such state into another.

2.1 Quantum Fourier transform

Shor's algorithm [1, 2] computes a quantum Fourier transform (QFT) of N data points in the order of $(\log N)^2$ steps, compared to the classical Fast Fourier Transform (FFT) algorithm which requires on the order of (N log N) steps. Unlike the situation with the classical algorithm, where all of the Fourier transformed (output) frequencies are available, with the QFT this is not the case—one may only sample the transform at a few points; the whole suite of N transformed values is not available. This makes sense, since one should not expect to obtain N results while only performing $(\log N)^2$ computational steps. (Shor [1, 2] cleverly worked around this "limited output" restriction and was able to factor large composite integers by building up his "answer" in a particular component of the Fourier transform.) In a similar fashion it is possible to compute the integral or summation of function values very efficiently by quantum Fourier transforming a set of values and then sampling the "DC" component of the transform. In the context of computational work on POMDPs, the QFT could be useful for efficiently evaluating path integrals or summations of value functions.

Another possible application of the QFT to POMDP processing lies in its use in pattern recognition, based on the work by Schützhold [26]. From his work, Figure 3 shows a 32 by 20 pixel binary test image with an embedded 8 by 8 checkerboard pattern located left of center and slightly higher than midpoint. Away from the regular pattern, half of the pixels are randomly selected to be 'on', the other half 'off'.

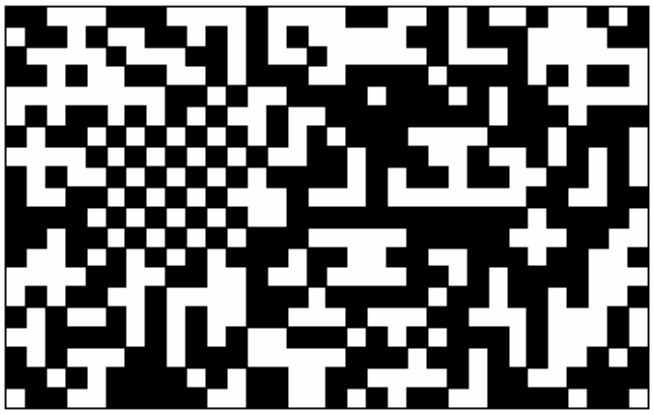

Figure 3: Binary Test Image with Embedded 8 by 8 Checkerboard Pattern

Schützhold [26] demonstrates the exponential speed-up that the QFT has over classical methods for finding and identifying the regular pattern within the test image. Generalizable to three-dimensional data sets, the same approach can, in principle, be used to search for certain patterns (tracks) within 3-D data cubes such as the one shown in Figure 2.

2.2 Quantum unstructured search algorithm

Grover's algorithm [7] is used to perform unstructured searches for particular items, finding them amongst the total of N possible listed entries in only $O(N^{1/2})$ steps; this compares to the O(N) steps that are required using the best classical search algorithms. Based on the article by Williams [27], there are many opportunities to apply this quantum search algorithmic approach when analyzing POMDPs, since such unstructured searches occur naturally in several contexts: *e.g.*, finding locations of extreme values and selecting amongst competing policies or potential reward functions. Quoting from Williams [27], "Quantum search has turned out to be remarkably versatile." and "We can also use quantum search to determine statistical properties—such as means, medians, maxima, and minima—of functions in the square root of the number of steps needed to compute these properties classically to the same precision."

2.3 Quantum adiabatic evolution algorithm

A general description of this class of quantum algorithms is given first. Start with the time-dependent Schrödinger equation (where units have been chosen so that $\hbar$ is unity):

$$i(d/dt)|\Psi(t)\rangle = H(t)|\Psi(t)\rangle \tag{11}$$

Then define the time-varying Hamiltonian by:

$$H(t) = (1 - t/T)H_0(t) + (t/T)H_F(t) \tag{12}$$

Thus,

$$H(t=0) = H_0(0) \tag{13}$$

and

$$H(t=T) = H_F(T) \qquad (14)$$

The initial Hamiltonian, H(t=0), *i.e.*, the "starting" system, and its ground state wave function, $|\Psi(0)\rangle$, the "starting" solution, are both known quantities. The final Hamiltonian, H(t=T), *i.e.*, the "target" system, is also known, but its corresponding wave function $|\Psi(T)\rangle$, the desired "target" solution, is unknown. Calculating this latter quantity, $|\Psi(T)\rangle$, is the whole purpose of this particular quantum algorithm. The algorithm derives its name from the Quantum Adiabatic Theorem, which, roughly stated, is:

Quantum Adiabatic Theorem: If H(t) varies slowly enough, then the state of the system $|\Psi(t)\rangle$, if initially started in the ground state, will remain in the instantaneous ground state of the Hamiltonian for all t values.

Adiabatic quantum algorithms use this theorem and approach from quantum mechanics to obtain the desired solution $|\Psi(T)\rangle$ by slowly evolving $|\Psi(0)\rangle$.

There are two obvious questions (pitfalls) regarding the practical physical implementation of this adiabatic solution approach, they are: (1) How slow is slow? and (2) How large should T be? Numerous research papers have addressed these and other important issues on a variety of sample problems, with varying degrees of success.

The versions of quantum adiabatic evolution algorithms proposed by Farhi *et al.* [8] and Hogg [9] may be used to solve various optimization problems more efficiently than classical algorithms. Their general conclusions, however, were contested by van Dam *et al.* [28]: "Is adiabatic quantum computing really quantum? …we give a simple example of a computational problem on which the adiabatic quantum algorithm provably takes exponential time. Although the problem is easy to solve classically, it is designed to be difficult for algorithms based on local search: its global optimum lies in a narrow basin, while there is a local optimum with a much larger basin. …the gap between the minimum and second eigenvalue of the Hamiltonian of the system is exponentially small."

So, the question still remains how well this quantum approach would work on the numerous types of optimization problems that commonly occur when analyzing POMDPs. The work by Steffen *et al.* [29] is encouraging: using NMR technology, they successfully implemented a 3 qubit adiabatic quantum optimization algorithm in an experiment that solved a simple problem.

Returning to van Dam *et al.* [28], "This paradigm [adiabatic quantum computation] bears some resemblance to simulated annealing, in the sense that the algorithm starts from an initial disordered state, and homes in on a solution (by what could be described as quantum local search) as a parameter 'λ' is smoothly varied from 0 to 1." The parameter 'λ' in simulated annealing plays a role similar to that of time 't' in the quantum adiabatic algorithm.

Following Feynman's maxim, "The same equations have the same solutions", one is led to the Chow-Yorke algorithm [30], a "homotopy continuation" method, as a promising approach to use for solving adiabatic quantum algorithms.

This leads to the point of departure that our analysis takes from the analysis in [28]. Instead of starting from an initial disordered state, the initial state that we decide to use is carefully chosen so that it exactly solves a known, simpler version of the problem. Then, using homotopy continuation techniques, the original state (point) satisfying the simpler problem conditions, is smoothly varied until a solution of the desired, more complex problem is reached. By using the more robust Chow-Yorke algorithm [30], situations when 'λ', the commonly used continuation variable, doubles back on itself may be successfully handled by using arc length in a higher dimensional space as the continuation variable. The system of equations is slightly more complicated, but this is more than compensated for by increased solution robustness.

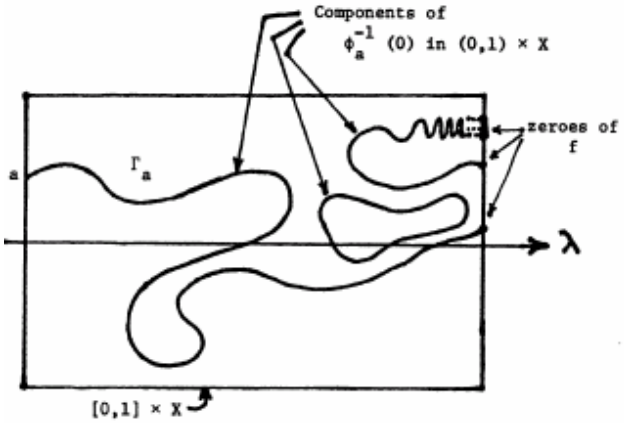

Figure 4: Chow-Yorke Continuation Scenario

This "doubling back" scenario is shown in Figure 4, taken from [30]. By using arc length as the independent variable, and integrating along the curve from λ = 0 to λ = 1, one can successfully follow the curve all the way along its entire course. If one were to use λ as the independent variable, however, after starting at the point λ = 0 the calculation would get "stuck" at the first turning point, where λ is slightly more than halfway across the rectangular region in Figure 4, and not reach the solution point at λ = 1.

3 Future Directions

DARPA's Focused Quantum Systems (FoQuS) program is anticipated to address the computationally difficult problem of factoring large composite numbers. With the great emphasis that will necessarily be placed on new and existing methods of quantum error correction, efforts in this area will likely have beneficial results for many other problem areas addressed by quantum algorithms.

Just as there is a natural fit between homotopy continuation methods (mathematics) and the quantum adiabatic theorem (physics), there appears to be another natural fit between a fairly obscure mathematical tool and the physical problem of quantum error correction.

The mathematical tool is called continuous orthonormalization and is useful for solving two-point boundary value problems and initial value problems. Davey [31] and Meyer [32] are key early references to the method. First developed to solve stiff systems of linear ODE's, the original system is converted into two non-stiff operators, one of which is nonlinear. (This doubles the size of the system.) The nonlinear system is constructed to advance an orthonormal coordinate frame. With this approach, independent solutions to the ODE automatically maintain their orthogonality at all times, a particularly useful property for problems with nonlinear or free (*i.e.*, location unknown) boundary conditions. The method is extremely reliable and especially suitable for high-order differential systems. Problems with rapid oscillations, turning points, and boundary layers are overcome. In one example, an Orr-Sommerfeld flow problem with a Reynolds number of 10^9 was easily solved. In short, the continuous orthonormalization algorithmic approach seems made to order for modeling systems governed by quantum physics. The importance of maintaining linearity properties of quantum mechanics cannot be stressed enough, for the consequences of nonlinearity in quantum computers are quite radical. Abrams and Lloyd [33] have shown that nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #P problems.

While working on these types of problems (both classical and quantum), it is useful to keep in mind that some of the really important questions do not yet have definitive answers. For example, the question of the (in)equivalence of complexity classes NP and P remains unsettled. It also has not yet been proven that determining a composite number's prime factors lies in NP. Surprises sometimes do occur; witness the recent proof by Agrawal *et al.* [34] that determining whether a number is prime or not lies in complexity class P.

4 Conclusions

This article presents several ways that quantum algorithms can be used to speed up the analysis and solution of computationally intense portions of partially observable Markov decision processes—searching, pattern matching, and parameter optimization and estimation, to name just a few. Future numerical studies are expected to confirm the theoretical scaling behavior and performance of the algorithms. With new papers coming online, such as the recent one by Romanelli *et al.* [35] that synthesizes quantum algorithms and Markov processes, the future for applying quantum algorithms to POMDPs looks very bright indeed.

This work was supported by the DARPA QuIST program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

References

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings, 35th Annual Symposium on Foundations of Computer Science*, IEEE Press, Los Alamitos, CA, pp.~124--134, 1994; or `quant-ph/9508027`.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Computing*, Vol. 26, pp.~1484--1509, 1997.

[3] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," Nature, Vol. 414, pp.~883--887, 20/27 Dec. 2001, or `quant-ph/0112176`.

[4] L. A. Levin, "The Tale of One-Way Functions," *Problems of Information Transmission*, Vol. 39, pp.~92--103, 2003; or `cs/0012013`.

[5] D. Gottesman, "An Introduction to Quantum Error Correction," in `quant-ph/0004072`.

[6] A. M. Steane, "Quantum Computing and Error Correction," in `quant-ph/0304016`.

[7] L. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Phys. Rev. Lett.*, Vol. 79, pp.~325--328, 1997, or `quant-ph/9706033`.

[8] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem," *Science*, Vol. 292, pp.~472--476, Apr. 2001.

[9] T. Hogg, "Quantum Search Heuristics," *Phys. Rev. A*, Vol. 61, Issue 5, pp.~052311, 2000.

[10] A. C. Doherty, S. Habib, K. Jacobs, H. Mabuchi, and S. M. Tan, "Quantum feedback control and classical control theory," *Phys. Rev. A*, Vol. 62, pp.~012105, 2000.

[11] F. Verstraete, A. C. Doherty, and H. Mabuchi, "Sensitivity optimization in quantum parameter estimation," *Phys. Rev. A*, Vol. 64, pp.~032111, 2001.

[12] D. A. Meyer, "Quantum Strategies," *Phys. Rev. Lett.*, Vol. 82, pp.~1052--1055, 1999.

[13] R. P. Feynman, "There's Plenty of Room at the Bottom," speech at Caltech, December 1959, www.zyvex.com/nanotech/feynman.html

[14] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.*, Vol. 21, pp.~467--488, 1982.

[15] M. L. Littman, from his web site at the following URL www.cs.duke.edu/~mlittman/topics/pomdp-page.html

[16] W. Zhang, "Algorithms for partially observable Markov decision processes," Ph.D. dissertation, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, 2001.

[17] S. Aaronson, from his website at the URL: www.cs.berkeley.edu/~aaronson/zoo.html

[18] A. R. Cassandra, from his web site at the URL: www.cs.brown.edu/research/ai/pomdp/tutorial/index.html

[19] J. Baxter and P. L. Bartlett, "Direct Gradient-Based Reinforcement Learning: I. Gradient Estimation Algorithms," Technical Report, Research School of Information Sciences and Engineering, Australian National University, July 1999.

[20] J. Baxter, L. Weaver, and P. L. Bartlett, "Direct Gradient-Based Reinforcement Learning: II. Gradient Ascent Algorithms and Experiments," Technical Report, Research School of Information Sciences and Engineering, Australian National University, Sept. 1999.

[21] J. R. Guerci, *Space-Time Adaptive Processing for Radar*, Artech House, Aug. 2003.

[22] J. Weickert, *Anisotropic Diffusion Image Processing*, Teubner-Verlag, Stuttgart, Germany, 1998. See also: www.mia.uni-saarland.de/weickert/publications.html for numerous updates.

[23] G.-H. Cottet and M. El Ayyadi, "Nonlinear PDE Operators with Memory Terms for Image Processing," *Proc. IEEE Int. Conf. on Image Processing (ICIP-96)*, Lausanne, Vol. 1, pp.~481--483, 16-19 Sep. 1996.

[24] N. David Mermin, "From Cbits to Qbits: Teaching computer scientists quantum mechanics," *Am. J. Phys.*, Vol. 71, pp.~23--30, Jan. 2003.

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

[26] R. Schützhold, "Pattern recognition on a quantum computer," `quant-ph/0208063`, 2002.

[27] C. P. Williams, "Quantum Search Algorithms in Science and Engineering," Computing in Science and Engineering, AIP/IEEE Computer Society, pp.~44--51, March/April 2001.

[28] W. van Dam, M. Mosca, and U. Vazirani, "How powerful is adiabatic quantum computation?," Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, pp.~279--287,2001.

[29] M. Steffen, W. van Dam, T. Hogg, G. Breyta, and I. Chuang, "Experimental implementation of an adiabatic quantum optimization algorithm," in `quant-ph/0302057`.

[30] S. N. Chow, J. Mallet-Paret, and J. A. Yorke, "Finding zeros of maps: Homotopy methods that are constructive with probability one," *Mathematics of Computation*, Vol. 32, pp.~887--899, 1978.

[31] A. J. Davey, "An automatic orthonormalization method for solving stiff boundary value problems," *J. Comp. Phys.*, Vol. 51, pp.~343--356, 1983.

[32] G. J. Meyer, "Continuous orthonormalization for boundary value problems," *J. Comp. Phys.*, Vol. 62, pp.~248--262, 1985.

[33] D. S. Abrams and S. Lloyd, "Nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #P problems," *Phys. Rev. Lett.*, Vol. 81, pp.~3992--3995, 1998, or `quant-ph/9801041`.

[34] M. Agrawal, N. Kayal, and N. Saxena, "Primes is in P," available at www.cse.iitk.ac.in/primality.pdf, 2002.

[35] A. Romanelli, A. C. Sicardi Schifino, R. Siri, G. Abal, A. Auyuanet, and R. Donangelo, "Quantum Random Walk on the Line as a Markovian Process," in `quant-ph/0310171`.