Evolutionary Algorithms for Hard Quantum Control

Ehsan Zahedinejad,[1] Sophie Schirmer,[2, 3] and Barry C. Sanders[1, 3, 4, 5, *]

[1]*Institute for Quantum Science and Technology, University of Calgary, Alberta, Canada T2N 1N4*
[2]*College of Science, Swansea University, Singleton Park, Swansea SA2 8PP, Wales, United Kingdom*
[3]*Kavli Institute for Theoretical Physics, University of California at Santa Barbara, California 93106-4030, USA*
[4]*Program in Quantum Information Science, Canadian Institute for Advanced Research, Toronto, Ontario M5G 1Z8, Canada*
[5]*Hefei National Laboratory for Physical Sciences at the Microscale and Department of Modern Physics, University of Science and Technology of China, Anhui 230026, China*

(Dated: August 23, 2014)

Although quantum control typically relies on greedy (local) optimization, traps (irregular critical points) in the control landscape can make optimization hard by foiling local search strategies. We demonstrate the failure of greedy algorithms as well as the (non-greedy) genetic-algorithm method to realize two fast quantum computing gates: a qutrit phase gate and a controlled-not gate. We show that our evolutionary algorithm circumvents the trap to deliver effective quantum control in both instances. Even when greedy algorithms succeed, our evolutionary algorithm can deliver a superior control procedure, for example reducing the need for high time resolution.

PACS numbers: 03.67.Lx,03.67.Ac, 42.50.Ex

I. INTRODUCTION

Quantum control aims to steer quantum dynamics towards closely realizing specific quantum states or operations [1, 2] with applications to femtosecond lasers [3, 4], nuclear magnetic resonance [5, 6] and other resonators [7–9], laser-driven molecular reactions [10, 11], and quantum gate synthesis for quantum computing [12]. Control is achieved by varying the strengths of different contributing processes (external fields) over time such that the resultant evolution closely approximates the desired evolution. The quality of a given quantum control procedure is quantified by its fitness [13] such as fidelity or distance for the approximated quantum state [14] or quantum gate [15] and the target state or gate.

A key goal in quantum control is to reach the fittest procedure possible within the target time T subject to certain resource limits such as limiting the number of independent control parameters K and therefore the time resolution T/K for time-domain quantum control. Practical considerations usually tightly constrain the maximum allowable values for T, and lower bounds for T are central to questions about fundamental "quantum speed limits" to operations in quantum computing, quantum metrology and quantum chemistry [16–19]

Choosing control parameters to maximize the procedure fitness is an optimization problem. Early quantum control employed non-greedy approaches, v.g., the genetic algorithm (GA) [20, 21]. Today greedy algorithms dominate the methodology as local optimization strategies usually have lower computational cost than global search algorithms and the fitness landscape (plot of fitness vs. control parameters) typically appears to be tractable [22]. Unfortunately, greedy algorithms can fail even for low-dimensional quantum control with simple Hamiltonians if T must be short. This seemingly innocuous constraint eliminates any guarantee of global optimality for local extrema.

Although it is tempting to attribute failure to find a satisfactory control procedure to infeasibility of the constrained problem, we show that this failure can instead be due to restricting strategies to greedy algorithms. To make our case, we present examples of control problems involving simple systems for which greedy algorithms overwhelmingly fail. These two control problems are especially contrived to be hard to solve using common quantum-control techniques, but the problems are physically meaningful as discussed in Subsecs. V A and V B, respectively. We use the term 'hard' to refer to problems that defy existing methods in the sense that the probability that they produce a satisfactory solution is small. A key element of these problems are that the time required for the unitary operation is short, which could make many quantum control problems hard. We show that these hard quantum control problems can be solved using global optimization techniques based on the differential evolution (DE) algorithm [23], which succeeds in finding effective controls up to the computational-power limits (machine error) even for very short T and very few controls.

We compare greedy vs non-greedy algorithms for realizing two different quantum computing gates: the original qutrit phase gate [24, 25] and the two-qubit controlled-not (CNot) gate [26], which are key elements of standard quantum computing instructions sets for qutrits and for qubits, respectively. We show that, for each gate and given our selected drift and control Hamiltonians, the greedy algorithm fails to find a high-fitness quantum control procedure for short target time T while our non-greedy DE algorithm succeeds. Moreover, for larger T where greedy algorithms work, DE is able to find solutions requiring fewer independent control param-

*Electronic address: sandersb@ucalgary.ca

eters K than the greedy algorithms tested. Interestingly, the common non-greedy GA also strongly fails for our test problems.

II. QUANTUM CONTROL

In any quantum control problem, the goal is to decompose the system's Hamiltonian into a controllable and an uncontrollable part, and steering the dynamics towards a desired evolution through varying the controllable part of the system. Here we first explain the system Hamiltonian in the context of control theory and then discuss our choice of the fitness function serving as the objective function for the purpose of optimization.

A. Quantum control Hamiltonian

For a closed system, the Hamiltonian

$$\hat{H}\left[\varepsilon(t)\right] = \hat{H}^{\mathrm{dr}} + \boldsymbol{\varepsilon}(t)\cdot\hat{\boldsymbol{H}}^{\mathrm{c}} = \hat{H}^{\mathrm{dr}} + \sum_{\ell=1}^{L}\varepsilon_\ell(t)\hat{H}^{\mathrm{c}}_\ell, \quad (1)$$

acts on Hilbert space $\mathscr{H}$ [27] with drift Hamiltonian $\hat{H}^{\mathrm{dr}}$ describing free (uncontrolled) evolution, which we treat as being time-independent here. The control Hamiltonians, represented by the vector operator $\hat{\boldsymbol{H}}^{\mathrm{c}}(t) = (\hat{H}^{\mathrm{c}}_\ell)$ (for $\{\ell\}$ the control field labels) should steer the system towards the desired evolution with time-varying (here piecewise constant) control amplitudes contained in the vector $\boldsymbol{\varepsilon}(t) := \{\varepsilon_\ell(t)\}$.

The resultant unitary evolution operator is formally

$$U\left[\boldsymbol{\varepsilon}(\boldsymbol{t});T\right] = \mathcal{T}\exp\left\{-\mathrm{i}\int_0^T \hat{H}(\boldsymbol{\varepsilon}(t))\mathrm{d}t\right\} \quad (2)$$

with $\mathcal{T}$ the time-ordering operator [28]. We aim to approximate a target unitary evolution operator U within duration T by a unitary operator $\tilde{U}[\boldsymbol{\varepsilon}(\boldsymbol{t});T]$ with minimum distance

$$\|U - \tilde{U}[\boldsymbol{\varepsilon}(t);T]\|. \quad (3)$$

between the realized evolution and the target.

B. Fitness functional

The quality of a candidate quantum control procedure is quantified by the fitness functional

$$\mathscr{F}[\boldsymbol{\varepsilon}(t)] = \mathscr{F}(U[\boldsymbol{\varepsilon}(t);T]) = 1 - \|U - \tilde{U}[\boldsymbol{\varepsilon}(t);T]\| \quad (4)$$

with $\|\bullet\|$ the operator norm and the final term in (4) the trace distance [29] between the target and actual evolution operators. The optimization problem is to maximize $\mathscr{F}[\boldsymbol{\varepsilon}(t)]$, i.e., to reduce the distance (3). For numerical simulation we use the explicit form

$$\mathscr{F}(t) = \frac{1}{N}\mathrm{Re}\left(\mathrm{Tr}\left(U[\boldsymbol{\varepsilon}(t),T]\tilde{U}[\boldsymbol{\varepsilon}(t);T]\right)\right) \quad (5)$$

of the fidelity function [12] between the target $U[\boldsymbol{\varepsilon}(t);T]$ and the approximate unitary operator $\tilde{U}[\boldsymbol{\varepsilon}(t);T]$ with N the Hilbert-space dimension.

III. CRITERIA TO EVALUATE ALGORITHM PERFORMANCE

Evaluating and comparing algorithms for optimization should be conducted fairly and clearly. Using run-time directly as a cost criterion obscures fundamental issues in comparing the intrinsic differences. Therefore, we evaluate and compare algorithms based on whether the algorithm yields a sufficiently optimal solution over many attempts, here called "runs". Each run is allowed to iterate until it succeeds or fails in which case the run aborts.

The iteration number of run r is $\imath$, and the total number of iterations for run r is denoted I_r with I_R the maximum iteration number over all R runs. For R runs, we determine and tabulate the best and worst fitness value obtained over these runs, and we characterize the statistics of error values according to the median error and the probability $\wp$, or percentage, of runs whose error is less than some threshold value.

We compare the performance of the optimization algorithms based on the bounds and statistics of the statistics of runs, and these statistics are analyzed in plots that depict the fidelity vs $\imath$ for each of the many runs. A plot of $\mathscr{F}$ vs $\imath$ is overly crowded to reveal key features clearly. Therefore, we stretch the plots by presenting the monotonically-related "logarithmic infidelity"

$$L := \log_{10}(1 - \mathrm{Re}\,(\mathscr{F})) \quad (6)$$

vs. $\imath$ for each run. Logarithmic infidelity is zero for perfect infidelity

$$\mathscr{F} = 0, \quad (7)$$

hence approximately bounded by $L = -16$ for double precision, and ideally $-\infty$ for perfect fidelity ($\mathscr{F} = 1$).

The algorithm for run r is deemed successful if the final $\mathscr{F}_r$ exceeds a minimum threshold L^{t}, which we set to $L_{\mathrm{t}} = -4$ commensurate with the widely accepted gate fidelity required for scalable quantum computing [30]. Our algorithm aborts a run after I_r iterations only if the change in $\mathscr{F}_r$ is within machine error or an infidelity within machine precision is reached. The percentage of runs that beat L^{t} is denoted $\wp^{\mathrm{t}}$.

A fair comparison of greedy vs evolutionary algorithms would consider an equal number of trials in each case. In order to make a stronger case that our evolutionary algorithm is superior, we are giving the greedy algorithms an advantage by allowing them twice as many runs as for the evolutionary case. This allowance is feasible because greedy algorithms typically run much quicker so allotting additional time to double the number of greedy runs is not onerous in terms of computational time. Specifically we run the greedy algorithms over 80 trials and the evolutionary algorithms over 40 trials. Our choice of 80 and 40 trials, respectively, comes from our experience in testing these different algorithms, and these numbers correspond to balancing achieving sufficient success probability against excessive computational cost.

IV. METHODS

We begin this section with explaining the choice of control function that we use for the purpose of optimizing the external field. We then discuss the details of how we use the external field to numerically approximate the unitary operation in (2). The last part of this section discusses the optimization routines that we have tested on two quantum control cases. We discuss two class of optimization routines namely: local (greedy) and global (evolutionary) algorithms. Our focus is on the evolutionary algorithms for which we provide a detailed explanation of each algorithm in the Appendix section.

A. Type of control function and control parameters

Now we discuss how the computation works. Numerically the fitness functional (5) is evaluated by discretizing the control function vector $\boldsymbol{\varepsilon}$ by expressing it as a sum of K orthonormal functions over the time domain $[0, T]$ as follow:

$$\boldsymbol{\varepsilon} := \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_K \end{pmatrix} \tag{8}$$

such that each vector element ε_l is constant over sequential time steps of equal duration $\Delta t = T/\mathrm{K}$.

The K control parameters refer to choosing various weightings of these control functions. For our analysis, these K orthonormal functions are non-overlapping rectangular functions with identical durations T/K [31]; i.e., the control functions are expressed as a weighted series of time bins. Each control element is randomly generated from the interval $[-1, 1]$ and evolves through the optimization process toward its best optimal value.

This time-bin discretization is commonly used and justified by the fact that control pulses on experimental hardware are often limited to this form although there are alternatives such as decomposition into K monochromatic functions to be solved in the frequency domain. The rectangular time bins also have computational advantages in that the time-ordered integral (2) is straightforward to evaluate.

B. Optimizing the control function

For any optimization problem, greedy algorithms are the primary choice if they can provide a satisfactory result. Greedy algorithms locally explore the landscape so their convergence rate toward a local optimum is much faster than for global optimization algorithms. If the landscape includes many traps, most trials striving to find global optima become ensconced within local traps in which case greedy algorithms fail to find the best solution. Evolutionary algorithms are specifically designed for non-convex optimization problems, which typically arise when the landscape includes many traps and a global optimum is the goal. Here we find that greedy algorithms are preferred when there is enough time to approximate the unitary operation and enough control parameters to search the landscape. Otherwise evolutionary algorithms are the choice to solve the problems. We discuss these two class of optimization algorithms here.

Greedy algorithms include the Nelder-Mead technique [32], Krotov [33–35], the quasi-Newton method, which employs the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approximation of the Hessian [36–40]. The Nelder-Mead technique uses the fitness functional only, whereas the Krotov algorithm uses both the fitness functional and its gradient ($\nabla\mathscr{F}$) with respect to control elements, and quasi-Newton uses the fitness functional as well as its gradient ($\nabla\mathscr{F}$) and Hessian ($\nabla^2\mathscr{F}$) to find local optima over many iterations. Lie group techniques can help to determine the gradient analytically [41, 42], and numerical techniques work generally.

Greedy algorithms are especially successful if both T and K are sufficiently large. For constrained T, second-order traps such that the Hessian is negative semidefinite arise [43–45] and numerical evidence arguably exists for the presence of other traps [13, 46, 47].

As an alternative to greedy algorithms, we consider evolutionary algorithms for quantum control. Evolutionary algorithms are stochastic optimization algorithms, inspired by the process whereby biological organisms adapt and survive [48]. These algorithms only require the fitness functional and not its gradient or Hessian.

The large class of evolutionary algorithms includes simulated annealing [49], ant-colony systems [50], memetic algorithms [51], DE [23], and particle swarm optimization (PSO) [52], GA [21] inter alia [48], but we choose to test just the three most common or promising evolutionary algorithms, namely traditional GA (a commonly used algorithm) and the modern PSO and DE algorithms (promising for this type of problem). The promising nature of PSO and DE is based on many studies [53–55] that have shown the superiority of DE and PSO over other evolutionary algorithms. All PSO, DE and GA employ the initial condition of multiple guesses, called particles in PSO and chromosomes in DE and GA Each test function evolves iteratively along trajectories in parameter space and experience different fitness values.

In GA, “parent” chromosomes go through three steps: selection, crossover and mutation to generate a new generation (“offspring”) of chromosomes. We test all Matlab® (version R12118) GA options and found that the Wheel-Roulette, Two-Point and Uniform methods perform best. We test GA fairly by optimizing population number N independently for each GA variant and fix the run time to equal those for PSO and DE runs.

In PSO the particle evolves according to a Langevin equation that includes a random kick, an attractive force to its previous best fitness and a force pulling to the particle to the fittest particle in its neighborhood (where the size of the neighborhood is logarithmic in the number of particles). Neighborhoods overlap such that they do not partition into distinct sets. Specifically we employ three PSO variants labeled here as PSO1, PSO2 [56] and PSO3 [57].

In DE each chromosome breeds with three other randomly chosen chromosomes from the same generation to produce a “daughter”, and the fittest of the original vs the bred “daughter” survives to the next generation. We use a DE variant that incorporates mutation scaling factor $\mu \in [0,2]$ and cross-over rate ξ [23]. In each generation the difference between two randomly chosen target vectors is weighted by μ then added to a third randomly selected target vector to generate the new set of vectors called donors. This quantity μ determines the DE step size for exploring the control landscape. Donor vector elements are incorporated into target vectors with probability ξ to generate trial vectors, and the fittest of the target and trial vectors survive to the next generation.

Details of DE, PSO and GA and a comparison between these three algorithm can be found in the Appendices.

C. Evaluating the objective function

We use the following decomposition approach to construct the gate:

$$U[\boldsymbol{\varepsilon}(\boldsymbol{t});T] = U_K U_{K-1} U_{K-2} \dots U_3 U_2 U_1 \qquad (9)$$

with $U_K = \exp(iH(\varepsilon_l)\Delta t)$ and T the fixed target time for the unitary operation. The next step is to optimize $\mathscr{F}$ over $\boldsymbol{\varepsilon}(t)$ within target time T keeping the number K of time bins small.

V. TWO QUANTUM-CONTROL CASES

Now we proceed to the two quantum-control cases of a qutrit phase gate and a CNot gate. For each individual problem we first test conventional greedy algorithms. For sufficiently large T for effecting the unitary operation, we show that greedy algorithms can rapidly converge to a local optimum, but not necessarily a global optimum, which we characterize here as a local optimum that meets the infidelity condition $L := -4$. We also show that reducing the time and the number of control parameters transforms the problem into a hard optimization problem for which we employ evolutionary algorithms as an alternative approach.

A. Qutrit phase gate

For a qutrit phase gate the Hilbert space is $\mathscr{H} = \operatorname{span}\{|0\rangle, |1\rangle, |2\rangle\}$, and the target gate is:

$$U = \mathrm{e}^{-\mathrm{i}T\hat{H}^{\mathrm{dr}}}(\mathrm{e}^{-\mathrm{i}\phi}|0\rangle\langle 0| - \mathrm{i}\mathrm{e}^{-\mathrm{i}\gamma}|1\rangle\langle 1| - \mathrm{i}\mathrm{e}^{\mathrm{i}\gamma}|2\rangle\langle 2|) \qquad (10)$$

with objective parameters corresponding to the phases ϕ and γ and $\hat{H}^{\mathrm{dr}}$ is the drift Hamiltonian defined in (11). As our interest is in hard quantum control problems, we choose a challenging T-dependent drift Hamiltonian and a single control given by [13]

$$\hat{H}^{\mathrm{dr}} = \begin{pmatrix} 1+\frac{\pi}{T} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \hat{H}^{\mathrm{c}} = \begin{pmatrix} a & 1 & 0 \\ 1 & b & 1 \\ 0 & 1 & c \end{pmatrix}, \qquad (11)$$

respectively.

This choice of control and drift Hamiltonian (11) provides a rich lode for studying hard quantum control because, for any target time T, many choices of a, b, c, ϕ, and γ lead to $\varepsilon(t) \equiv 0$ being a critical point. The resultant criticality results in $\operatorname{Re}[\mathscr{F}[\boldsymbol{\varepsilon}(t)] < 1$ for which the Hessian becomes strictly negative definite. We consider the specific choice

$$a = 2,\ b = 2,\ c = 1. \qquad (12)$$

Then the phase choices

$$\gamma = \frac{5\pi}{3},\ \sin\phi = -\frac{b+c}{a}\cos\gamma \qquad (13)$$

ensure that

$$\varepsilon(t) \equiv 0 \qquad (14)$$

is a critical point i.e. a point in which

$$\nabla\mathscr{F}(\boldsymbol{\varepsilon}(t)) = 0,\ \nabla^2\mathscr{F}(\boldsymbol{\varepsilon}(t)) < 0. \qquad (15)$$

Therefore, a strong trap in the fitness landscape is deliberately set [13]. The resultant Hamiltonian is obtained by inserting (11) into (1). The external field, or control parameters, are adjusted to realize (10) according to expression (2). In this way, we can realize the qutrit phase gate by quantum control using the control and drift Hamitlonians (11).

While this problem is deliberately contrived to illustrate the existence of traps, the drift Hamiltonian (11), for a fixed value of T, effectively describes the free evolution of a three-level system such as encountered in a spin-one system or in a single atom with three pertinent electronic levels. The diagonal terms correspond to the electronic energy levels of the atom.

The first and second energy levels are non-degenerate in the absence of the driving field, but the states approach degeneracy in the limit of long time T. A system whose energy levels depend on the control time T does not appear in nature but is a legitimate system for mathematically exploring the limitations of greedy algorithms and power of evolutionary algorithms. Therefore, we employ this model in the qutrit case to compare these different optimization strategies.

The control Hamiltonian (11) can represent the interaction of an atom with a driving field. The diagonal terms of the control Hamiltonian represent level shifts due to the effect of the field. The off-diagonal terms are the Rabi frequencies between the corresponding pairs of levels, in this case between the first and second levels and between the second and third levels. Here we have scaled the Rabi frequencies to unity.

B. CNot gate

The second example concerns the two-qubit CNOT gate [26]. Inspired by the one-dimensional linear Ising-ZZ model [35], the drift and control Hamiltonians are

$$\hat{H}^{\mathrm{dr}} = \frac{J}{2} Z \otimes Z,\ \hat{\boldsymbol{H}}^{\mathrm{c}} = \begin{pmatrix} X \otimes \mathbb{1} \\ \mathbb{1} \otimes X \\ Y \otimes \mathbb{1} \\ \mathbb{1} \otimes Y \end{pmatrix}, \tag{16}$$

respectively, for

$$X = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \frac{1}{2}\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, Z = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{17}$$

the non-identity Pauli matrices and $\mathbb{1}$ the 2×2 identity matrix. We normalize time by setting $J \equiv 1$. The time-dependent four-dimensional control vector $\boldsymbol{\varepsilon}(t)$ in Eq. (1) is optimized so that the resultant evolution (2) approximates CNOT with high $\mathscr{F}$.

Physically the Ising-ZZ model corresponds to a one-dimensional spin chain, which was originally studied in the context of explaining ferromagnetism. The weak interaction is described by a tensor product of Pauli Z operators for nearest neighbours. The spin chain interacts with an external field, for example a magnetic field, and this interaction involves only single non-identity Pauli operators as seen in the control Hamiltonian (16).

VI. RESULTS

In this section we first discuss the performance of the quasi-Newton method on two specific problems where there is enough time for unitary operation or a sufficient number of control parameters. Then we numerically show that reducing the time T and control parameters K transform the problem into a hard optimization problem and results in runs of the greedy algorithms getting trapped. In the next part of this section we evaluate the performance of evolutionary algorithms when the time is shortened and K is reduced and compare the performance of different algorithms in terms of their median and best plots and finally tabulate the resultant data.

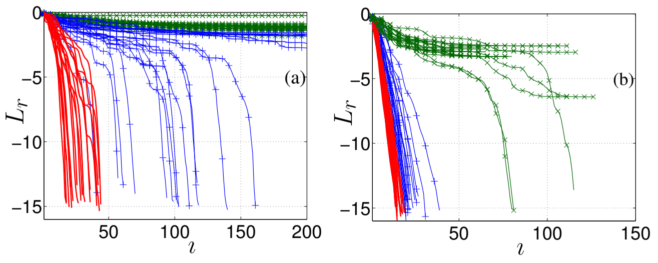

FIG. 1: (color online) Logarithmic-infidelity L vs iteration number $\imath$ for (a) the qutrit gate and (b) the CNOT gate using the quasi-Newton method with (a) $T = 10\pi$ (red, solid lines), $T = 4\pi$ (blue lines with '+' markers), and $T = 3\pi$ (green lines with '×' markers) such that $K = 50$ in all cases and with (b) $T = 30$ and $K = 30$ ((red, solid lines), $T = 10$ and $K = 10$ (blue lines with '+' markers) and $T = 4$ and $K = 4$ (green lines with '×' markers).

L	GA	DE	PSO	PSO1	PSO2	PSO3	Newton	simplex	Krotov
median	-0.6	-15.9	-1.7	-2.5	-2.4	-1.1	-0.7	-0.7	-0.6
best-case	-1.2	-15.9	-2.4	-4.4	-4.1	-1.5	-1.4	-1.3	-1.16
worst-case	-0.4	-2.2	-1.3	-1.6	-1.4	-0.9	-0.4	-0.4	-0.4
$\wp^{\mathrm{t}}$	0	72.5	0	12.5	7.5	0	0	0	0

TABLE I: Median, best-case, worst-case, and $\wp^{\mathrm{t}}$ for logarithmic-infidelity L for the qutrit phase gate with $T = 2.5\pi$, $K = 10$, dfs and $R = 80$ ($R = 40$) for greedy (evolutionary) algorithms.

We choose to begin with a plot of the fast-convergent quasi-Newton method because this approach should be the preferred choice for when it succeeds to deliver a satisfactory results. Figure 1 depicts the logarithmic infidelity $\{L_r\}$ as a function of $\imath$ for the qutrit phase gate and for the CNOT gate using the quasi-Newton method.

In Fig. 2, we compare the greedy simplex, Krotov and quasi-Newton methods against GA, DE, Common PSO, PSO1, PSO2, and PSO3 algorithms. Specifically we depict best-run performance and median-run performance in terms of final L. These plots are indicative only. Careful comparisons are summarized in Table I for the qutrit case and in Table II for median, best-case, and worst-case performance as well as for the percentage of runs $\wp^{\mathrm{t}}$ that exceed L^{t} over $R = 40$ runs for evolutionary algorithms and over $R = 80$ runs for greedy algorithms.

VII. DISCUSSION

This section begins with a discussion about the greedy algorithm used for two quantum control examples namely the qutrit phase gate and the CNOT gate. As we are de-

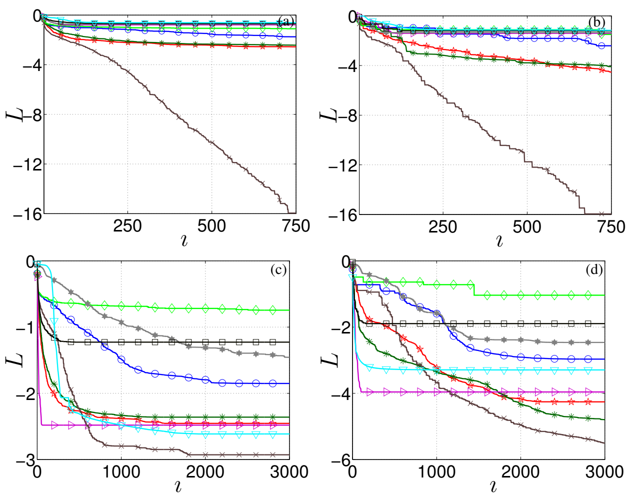

FIG. 2: (color online) Logarithmic-infidelity L vs iteration number $\imath$ for (□) GA, (×) DE, (○) Common PSO, (☆) PSO1, (∗) PSO2, (◇) PSO3, (▷) quasi-Newton, (✶) simplex and (▽) Krotov with $R = 80$ ($R = 40$) for greedy (evolutionary) algorithms. Median-run performance is depicted in (a) and (c); best run performance is depicted in (b) and (d). The qutrit phase gate is the target ($T = 2.5\pi$ and $K = 10$) in (a) and (b) CNOT is the target ($T = 3.2$ and $K = 4$) in (c) and (d).

L	GA	DE	PSO	PSO1	PSO2	PSO3	Newton	simplex	Krotov
median	-1.2	-2.9	-1.8	-2.4	-2.3	-0.7	-2.4	-1.45	-2.6
best	-1.8	-5.5	-2.9	-4.2	-4.7	-1.0	-3.9	-2.4	-3.2
worst	-0.7	-2.0	-1.3	-1.9	-1.3	-0.6	-2.0	-0.8	-1.9
$\wp^{t}$	0	15.0	0	2.5	10.0	0	0	0	0

TABLE II: Median, best-case, worst-case, and $\wp^{t}$ for logarithmic-infidelity L for the CNOT gate with $T = 3.2$, $K = 4$, and $R = 80$ ($R = 40$) for greedy (evolutionary) algorithms.

liberately making the problem harder by reducing the time and control resources, we resort to evolutionary algorithms and discuss their performances on these specific cases. We discuss the numerical evidence of local traps later on in this section and compare the performance of evolutionary algorithms with greedy algorithms when local traps dominates the landscape. In the last part we compare evolutionary algorithms performance and discuss why DE outperforms its ancestor GA.

As we explained in earlier sections, greedy algorithms converge faster than evolutionary algorithms and they should be the first choice for quantum control optimization if they can provide a satisfactory results. We show the greedy-algorithm performance in Fig. 1 where, in both cases (qutrit phase gate and CNOT gate), most quasi-Newton runs converge rapidly within machine precision ($L = -15.65$) to the target gate for large T and for small time resolution T/K. For small T and K, a majority of runs becoming trapped at low fitness (high L) values. Evidently the quasi-Newton method fails (freen plots in Fig. 1) for short-time and fine–time-resolution constraints.

Our results show that greedy algorithms perform poorly for the highly-constrained-T, low-K problems as do PSO and GA. Figure 2 compares the performance of different algorithms for two cases by providing the median and best plots for each algorithm. For the qutrit phase gate, optimization performance is shown in Figs. 2(a) and 2(b). Evidently, all quasi-Newton, Krotov and simplex runs become trapped at very low fidelity. On the other hand, DE and PSO1 and PSO2 are the only algorithms that successfully achieve the infidelity target of $L = -4$ 72.5%, 12.5% and 7.5% of the time, respectively (c.f., Table I) and DE achieves the best performance in terms of the best and median infidelity among all algorithms.

In the qutrit-phase-gate example we are searching the landscape around a critical point $\varepsilon(t) = 0$ by sampling each trial $\varepsilon(t)$ randomly from $[-1, 1]$ and evolving them toward their optimal values. As there are other studies that numerically provide the proof of local traps in the quantum control landscape [13, 46, 47], here our results show many local traps in the landscape as many runs from greedy algorithms get trapped at low fidelities. Using an efficient global-optimization routine like DE is necessary to avoid these local traps and to find a global optimal.

For the CNOT gate, whose performance is shown in Figs. 2(a, b), all runs become trapped at poor fidelities for the greedy algorithm case, and the GA and various PSO algorithms are also poor. In contrast the DE performance is vastly superior for the qutrit phase gate and significantly better for the CNOT gate under the extreme conditions of $T = 3.2$ and $K = 4$. Naturally the greedy and PSO algorithms can be improved by increasing K, and this strategy is common in the quantum control literature, but our aim is to constrain T and limit the number of control parameters K, and DE is the superior tool for doing so in that it works when the greedy and GA algorithms fail.

In the CNOT case, DE succeeds in providing a satisfactory result (See Table II) 15% of the time whereas this success rate is 2.5% and 10% for the PSO1 and PSO2 cases and zero for other cases. Therefore, this result shows that for short T and small K there are many local traps in the landscape causing the greedy algortihms to fail.

In all evolutionary algorithms discussed here, DE always outperforms its algorithmic ancestor GA. One might ask why DE performs better than GA [53] on these two specific quantum control problems and the answer lies in the mechanism of generating the new population from the old population. In GA, parents are selected based on probabilities that lead to individuals with better fitness. The crossover operation combines partial parts of two parents to generate a new offspring. As the new offspring comes from a combination of two parents, in this sense GA explores the optimal solution around some good solution candidates. GA must perform the mutation operation on the individual with a low mutation probability constant; otherwise it turns into a searching algorithm and becomes inefficient. This low mutation probability limits the GA's ability in searching the whole domain of landscape and thus might cause GA to fail with locating the global optimum.

Unlike GA, which converts candidate solutions into a binary format, DE constructs candidate solutions that are represented by real numbers. The crossover operation in DE generates offspring individuals from the entire set of populations so that newly generated offspring are always different from parent individuals. The higher mutation probability in DE, compared to GA, enables DE to explore the search space more efficiently while reducing the chance of getting trapped in local minima hence outperforms GA in term of the quality of the results (See Fig 2)

Optimization strategies can be compared in various ways. The most important criterion is whether the optimization approach delivers a satisfactory result. A secondary consideration is the rate of convergence, which is relevant to the run-time. Of course use of computational space is another consideration. In our case we are most concerned with the primary consideration of whether the optimization works, as determined by whether the threshold infidelity reaches $L = -4$.

As shown in Fig. 2, the quasi-Newton method converges faster than all other approaches but fails to achieve $L = -4$. Our message is that the most efficient, fastest optimization strategy should be used as long as it delivers a satisfactory result. If the fastest routine failed, then our analysis of two tightly time-constrained control problems is that differential evolution is an excellent alternative that appears to deliver a satisfactory result even when the other approaches fail.

The fast convergence of quasi-Newton runs in Fig. 2 raises the tantalizing possibility of whether increasing the number of quasi-Newton runs would result in a small but non-zero success probability $\wp^{\mathrm{t}}$. A fast algorithm like quasi-Newton with a low probability of success could make it superior to the slow DE approach with a high success probability. To test this hypothesis, we did 500 repetitions of 100 quasi-Newton iterations applied to the (CNot) gate control problem. We chose 100 iterations of quasi-Newton runs as the average "wall time" (the true run time on the given computer) for 100 iterations approximates the average wall time for 40 iterations of the DE method. Our numerical study showed that the quasi-Newton runs never reached $L \leq -4$. In principle the quasi-Newton method would work with a sufficient number of trials simply because the global search would be achieved by a huge number of local searches, but replacing a good global search by extremely many local searches is not feasible in practice.

Finally we emphasize that, when greedy algorithms work, the quantum control strategy should be to employ current practice and use the best available greedy algorithm. When greedy algorithms fail, though, evolutionary algorithms could work and differential evolution is the best amongst these according to our investigation. This is particularly relevant when exploring quantum speed limits numerically. In view of our results, quantum speed limits found using greedy algorithms reflect the limitations of these algorithms rather than intrinsic speeds limits for quantum control.

VIII. CONCLUSION

In conclusion we have shown that evolutionary algorithms such as DE and PSO are essential alternatives to greedy algorithms for hard quantum control problems with strong constraints. Greedy algorithms are often used because fitness landscapes are assumed to be well-behaved [22], and traps presumed to be negligible if T can be long and K can be increased without paying a significant price. In such cases greedy algorithms work because most local optima are globally optimal or close enough. However, when resources are limited, even straightforward control problems for simple systems can become hard due to a proliferation of traps in the landscape and non-convexity, thereby causing greedy algorithms to fail.

We have considered two quantum gates relevant to quantum information and used drift and control Hamiltonians that illustrate our point. These examples show that differential evolution is effective for hard quantum control problems. The superiority of differential evolution over greedy algorithms is unsurprising because the fitness landscape is no longer well behaved for hard quantum control. On the other hand, the superiority of differential evolution over GA and PSO and its variants is due to the greater efficacy of DE for optimization over higher-dimensional search spaces, which is the case for hard quantum control.

Acknowledgments

EZ acknowledges a Murray Fraser Memorial Graduate Scholarship and Eyes High International Doctoral Scholarship and support from NSERC. SS acknowledges support from EPSRC and EU network QUAINT. BCS is supported by NSERC, CIFAR, USARO and AITF and acknowledges hospitality and financial support from Macquarie University in Sydney and from the Raman Research Institute in Bangalore where some of this research was performed. SS and BCS acknowledge valuable discussions with A. Pechen and Y. R. Sanders during the early stages of this work. This project was initiated at a Kavli Institute for Theoretical Physics Workshop and thus supported in part by the National Science Foundation under Grant no. NSF PHY11-25915.

Appendix A: Genetic Algorithm

Genetic algorithms (GA) [58] are well known for global optimization. A candidate solution is first coded in a binary representation, called a parent vector. These parents evolve through several algorithmic steps, namely selection, crossover, and mutation. These steps lead to the generation of new candidates, known as children or offspring, for subsequent generations. These children become parents for the next generation.

Several variants of algorithmic steps exist for GA [21]. These steps evolve the fitness function towards its optimal state. We choose the following GA variant that leads to the best performance for our problem.

1. **Selection:**– This step specifies how the GA chooses the next-generation parent for subsequent breeding. We use the Roulette Wheel method, which assigns a selection probability to each individual parent vector according to

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \tag{A1}$$

for N the total population number and f_i the fitness level of each individual parent vector. This probability distribution is used to select parent vectors for the crossover step.

2. **Crossover:**– This step, which is considered to be the heart of GA, specifies how the two parents unite to generate the new offspring. We use the two-point selection method to choose two random integers m and n between one and the number of variables in each parent vector. Offspring elements are constructed from the element of the first parent vector P_1, whose indices are less than n or greater than m, and those elements of the second parent P_2, whose elements share equal indices or are between n and m.

3. **Mutation:**– The purpose of mutation is to introduce small changes in an individual selected from the population, which leads to the creation of mutant offspring. We mutate uniformly on each individual offspring. The mutation algorithm thus selects vector elements that are be mutated according to a small rate of 0.001. Then those selected elements are replaced by a random number selected uniformly from the set of all elements of the corresponding offspring vector.

For all problem instances, we set $N = 70$. This choice of population number ensures the same computational time for GA as for other evolutionary algorithms here, namely differential evolution (DE) and particle swarm optimization (PSO).

Appendix B: Particle Swarm Optimization

PSO optimizes by enabling exploration of the fitness landscape using a swarm of particles with position-velocity pairs $\{(x_n, v_n)\}$. These pairs are updated in each iteration of the algorithm based on the rules

$$v_{n+1} = \chi(wv_n + c_1r_1(x_{n,*} - x_n) + c_2r_2(x_* - x_n)), \tag{B1a}$$

$$x_{n+1} = x_n + v_n \tag{B1b}$$

where $x_{n,*}$ is the n^{th} particle's previous personal best and x_* the global best position so far. We employ χ as a constriction factor, and w is an inertial weight. The coefficients c_1 and c_2 are deterministic weights, and r_1 and r_2 are uniformly distributed random numbers in $[-1, 1]$.

For the common PSO algorithm, the inertial weight decreases linearly starting from $w_{\max} = 0.9$ to $w_{\min} = 0.4$ over N iterations according to $w_n = w_{\max} - (n - 1)(w_{\max} - w_{\min})/N$ and the standard parameters are $c_1 = c_2 = 2$ and $\chi = 1$. Clerc's and Trelea's variants use constant inertial weights and different parameter values. Clerc uses $w = 1$ (PSO1) whereas Trelea uses $w = 0.6$ (PSO2) and $c_1 = c_2 = 1.7$ (PSO3) and $w = 0.729$ and $c_1 = c_2 = 1.492$ (variant 2).

Appendix C: Differential Evolution

Individuals in DE are represented by a D-dimensional vector (X_i), $i \in \{1, \ldots, N_{\text{P}}\}$, where D is the number of control parameters and N_{P} is the population size. The classical DE algorithm can be summarized as follow [23]:

1. **Mutation:**– The update step is

$$V_i = X_{i_1} + \mu \left(X_{i_2} - X_{i_3}\right) \tag{C1}$$

with $i,\ i_1,\ i_2,\ i_3 \in [1, N_P]$ being integers and mutually different. Here μ is the mutation factor controlling the differential variation $d_i := X_{i_2} - X_{i_3}$.

2. **Crossover:**–

$$C_i(j) = \begin{cases} V_i(j) & \text{if } C_j(0,1) < \xi \\ X_i(j) & \text{otherwise} \end{cases} \tag{C2}$$

with $C_j(0,1)$ representing the uniform random between 0 and 1, and $\xi \in (0,1)$ is the crossover rate.

3. **Selection:**– The final step is the assignment

$$X_i' = \begin{cases} C_i & \text{if } f(C_i) < f(X_i) \\ X_i & \text{otherwise} \end{cases} \tag{C3}$$

with X_i' the offspring of X_i for the next generation and $f(X_i)$ the objective function, which, in our case, is the measured fidelity.

For all instances we choose $\mu = 0.5$, $\xi = 0.9$ and $N_{\text{P}} = 15K$, with K being the number of control parameters, for all problems.

[1] M. Shapiro and P. W. Brumer, *Principles of the Quantum Control of Molecular Processes*, vol. 1 (Wiley, Hoboken, 2003).

[2] D. Dong and I. R. Petersen, IET, Contr. Theor. Ap. **4**, 2651 (2010).

[3] A. Assion, T. Baumert, M. Bergt, T. Brixner, B. Kiefer, V. Seyfried, M. Strehle, and G. Gerber, Science **282**, 919 (1998).

[4] D. Meshulach and Y. Silberberg, Nature **396**, 239 (1998).

[5] C. Ryan, M. Laforest, J. Boileau, and R. Laflamme, Phys. Rev. A **72**, 062317 (2005).

[6] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, J. Magn. Reson **172**, 296 (2005).

[7] R. Ruskov, K. Schwab, and A. N. Korotkov, Phys. Rev. B **71**, 235407 (2005).

[8] S. Mancini, D. Vitali, and P. Tombesi, arXiv preprint quant-ph/9802034 (1998).

[9] A. Hopkins, K. Jacobs, S. Habib, and K. Schwab, Phys. Rev. B **68**, 235328 (2003).

[10] P. Brumer and M. Shapiro, Annu. Rev. Phys. Chem. **43**, 257 (1992).

[11] D. J. Tannor and S. A. Rice, J. Chem. Phys. **83**, 5013 (1985).

[12] T. Schulte-Herbrüggen, A. Spörl, N. Khaneja, and S. J. Glaser, Phys. Rev. A **72**, 042331 (2005), URL `http://link.aps.org/doi/10.1103/PhysRevA.72.042331`.

[13] P. De Fouquieres and S. G. Schirmer, Infin. Dimens. Anal. Quantum. Probab. Relat. Top. **16**, 1350021 (2013).

[14] D. Egger and F. Wilhelm, Supercond. Sci. Technol. **27**, 014001 (2014).

[15] M. Murphy, S. Montangero, V. Giovannetti, and T. Calarco, Phys. Rev. A **82**, 022318 (2010).

[16] L. B. Levitin and T. Toffoli, Phys. Rev. Lett. **103**, 160502 (2009), URL `http://link.aps.org/doi/10.1103/PhysRevLett.103.160502`.

[17] M. M. Taddei, B. M. Escher, L. Davidovich, and R. L. de Matos Filho, Phys. Rev. Lett. **110**, 050402 (2013), URL `http://link.aps.org/doi/10.1103/PhysRevLett.110.050402`.

[18] A. del Campo, I. L. Egusquiza, M. B. Plenio, and S. F. Huelga, Phys. Rev. Lett. **110**, 050403 (2013), URL `http://link.aps.org/doi/10.1103/PhysRevLett.110.050403`.

[19] G. C. Hegerfeldt, Phys. Rev. Lett. **111**, 260501 (2013), URL `http://link.aps.org/doi/10.1103/PhysRevLett.111.260501`.

[20] C. J. Bardeen, V. V. Yakovlev, K. R. Wilson, S. D. Carpenter, P. M. Weber, and W. S. Warren, Chem. Phys. Lett. **280**, 151 (1997).

[21] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Longman, Boston, 1989), 1st ed.

[22] H. A. Rabitz, M. M. Hsieh, and C. M. Rosenthal, Science **303**, 1998 (2004).

[23] R. Storn and K. Price, J. Global Optim. **11**, 341 (1997).

[24] D. Gottesman, A. Kitaev, and J. Preskill, Phys. Rev. A **64**, 012310 (2001), URL `http://link.aps.org/doi/10.1103/PhysRevA.64.012310`.

[25] S. D. Bartlett, H. de Guise, and B. C. Sanders, Phys. Rev. A **65**, 052316 (2002), URL `http://link.aps.org/doi/10.1103/PhysRevA.65.052316`.

[26] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995), URL `http://link.aps.org/doi/10.1103/PhysRevA.52.3457`.

[27] D. d'Alessandro, *Introduction to Quantum Control and Dynamics*, Chapman & Hall/CRC Applied Mathematics & Nonlinear Science (CRC, New York, 2007).

[28] G. Dattoli, J. Gallardo, and A. Torre, J. Math. Phys. **27**, 772 (1986).

[29] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 4 of *Johns Hopkins Studies in the Mathematical Sciences (Book 3)* (JHU Press, Baltimore, 2012).

[30] A. M. Steane, Phys. Rev. A **68**, 042322 (2003), URL `http://link.aps.org/doi/10.1103/PhysRevA.68.042322`.

[31] G. P. Rao, ed., *Piecewise Constant Orthogonal Functions and Their Application to Systems and Control*, vol. 55 of *Lecture Notes in Control and Information Sciences* (Springer, Berlin, 1983).

[32] D. M. Olsson and L. S. Nelson, Technometrics **17**, 45 (1975).

[33] A. Konnov and V. Krotov, Automation and Remote Control **60**, 1427 (1999).

[34] S. E. Sklarz and D. J. Tannor, Phys. Rev. A **66**, 053619 (2002), URL `http://link.aps.org/doi/10.1103/PhysRevA.66.053619`.

[35] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Phys. Rev. A **84**, 022305 (2011), URL `http://link.aps.org/doi/10.1103/PhysRevA.84.022305`.

[36] C. G. Broyden, IMA J. Appl. Math. **6**, 76 (1970).

[37] R. Fletcher, Comput. J. **13**, 317 (1970).

[38] R. Fletcher, *Practical Methods of Optimization*, vol. 2 (Wiley, Padstow, 2013).

[39] D. Goldfarb, Math. Comput. **24**, 23 (1970).

[40] D. F. Shanno, Math. Comput. **24**, 647 (1970).

[41] S. G. Schirmer and P. de Fouquieres, New J. Phys. **13**, 073029 (2011).

[42] F. F. Floether, P. de Fouquieres, and S. G. Schirmer, New J. Phys. **14**, 073023 (2012).

[43] A. N. Pechen and D. J. Tannor, Phys. Rev. Lett. **106**, 120402 (2011), URL `http://link.aps.org/doi/10.1103/PhysRevLett.106.120402`.

[44] H. Rabitz, T.-S. Ho, R. Long, R. Wu, and C. Brif, Phys. Rev. Lett. **108**, 198901 (2012), URL `http://link.aps.org/doi/10.1103/PhysRevLett.108.198901`.

[45] A. N. Pechen and D. J. Tannor, Phys. Rev. Lett. **108**, 229901 (2012), URL `http://link.aps.org/doi/10.1103/PhysRevLett.108.229901`.

[46] A. N. Pechen and D. J. Tannor, Isr. J. Chem. **52**, 467 (2012), URL `http://http://onlinelibrary.wiley.com/doi/10.1002/ijch.201100165/abstract`.

[47] A. N. Pechen and D. J. Tannor, Phys. Rev. Lett. **108**, 198902 (2012), URL `http://link.aps.org/doi/10.1103/PhysRevLett.108.198902`.

[48] K. A. De Jong, *Evolutionary Computation: A Unified Approach*, vol. 262041944 of *Bradford Book* (MIT press Cambridge, Cambridge, 2006), URL `http://books.google.co.in/books?id=OIRQAAAAMAAJ`.

[49] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983), ISSN 00368075, URL `http://www.`

`jstor.org/stable/1690046`.

[50] M. Dorigo, M. Birattari, and T. Stutzle, Computational Intelligence Magazine, IEEE **1**, 28 (2006), ISSN 1556-603X.

[51] P. Moscato et al., Caltech concurrent computation program, C3P Report **826**, 1989 (1989).

[52] J. Kennedy, in *Encyclopedia of Machine Learning* (Springer, 2010), pp. 760–766.

[53] M. A. Panduro, C. A. Brizuela, L. I. Balderas, and D. A. Acosta, Prog. Electromagn. Res. B **13**, 171 (2009).

[54] E. Elbeltagi, T. Hegazy, and D. Grierson, Advanced Engineering Informatics **19**, 43 (2005), ISSN 1474-0346, URL `http://www.sciencedirect.com/science/article/pii/S1474034605000091`.

[55] S. A. Ethni, B. Zahawi, D. Giaouris, and P. Acarnley, in *Conference on Industrial Informatics, 2009. INDIN 2009. 7th IEEE International* (2009), pp. 470–474, ISSN 1935-4576.

[56] I. C. Trelea, Inf. Proc. Lett. **85**, 317 (2003).

[57] M. Clerc and J. Kennedy, Evol. Comput., IEEE Trans. on **6**, 58 (2002).

[58] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (University of Michigan Press, Ann Arbor, MI, 1975).