Evolutionary Algorithms for Quantum Computers

Daniel Johannsen[*]
School of Mathematical Sciences
Tel Aviv University
Tel Aviv, Israel
johannse@tau.ac.il

Piyush P Kurur[†]
Dept of Comp. Sci. and Engg
Indian Institute of Technology Kanpur
Kanpur UP, India 208016
ppk@cse.iitk.ac.in

Johannes Lengler
Department of Theoretical Computer Science
Eidgenössische Technische Hochschule ETH
Zürich, Switzerland
johannes.lengler@inf.ethz.ch

May 10, 2013

[*]Daniel Johannsen is supported by a Postdoc Scholarship of the German Academic Exchange Service (DAAD).

[†]Work done on a visit to the Max Planck Institute for Informatics (MPII) funded by MPII and Research I project (NRNM/CS/20030163)

Abstract

In this article, we formulate and study quantum analogues of randomized search heuristics, which make use of Grover search [15] to accelerate the search for improved offsprings. We then specialize the above formulation to two specific search heuristics: Random Local Search and the (1+1) Evolutionary Algorithm. We call the resulting quantum versions of these search heuristics *Quantum Local Search* and the (1+1) *Quantum Evolutionary Algorithm*.

We conduct a rigorous runtime analysis of these quantum search heuristics in the computation model of quantum algorithms, which, besides classical computation steps, also permits those unique to quantum computing devices. To this end, we study the six elementary pseudo-Boolean optimization problems OneMax, LeadingOnes, Discrepancy, Needle, Jump, and TinyTrap.

It turns out that the advantage of the respective quantum search heuristic over its classical counterpart varies with the problem structure and ranges from no speedup at all for the problem Discrepancy to exponential speedup for the problem TinyTrap. We show that these runtime behaviors are closely linked to the probabilities of performing successful mutations in the classical algorithms.

1 Introduction

Quantum algorithms are algorithms that can be executed on a quantum computing device. One of the prominent computational problems which a quantum algorithm solves more efficiently than classical algorithms is searching in an unordered database. In his seminal work [15, 22], Grover gave an algorithm which can search in an unordered data base of N elements in time proportional to $\sqrt{N}$, whereas any classical algorithm requires time proportional to N. When the underlying search space has no structure, Grover search is known to be optimal [6, 26]. In the last decade, algorithms based on Grover search have been studied extensively. Many specialized algorithms have also been studied for problems such as searching [15, 9], Element Distinctness [23], Minimum-Finding [13] and many others (e.g., [12, 7, 27]).

Although Grover's algorithm gives a quadratic speedup[1] for search, this is not a universal phenomenon for all computational problems. For example, Grover search can be thought of as evaluating the Boolean function OR on N bits. If instead of the OR function we consider the XOR on N bits, a lower bound of $\Theta(N)$ queries is known in quantum setting [2, 5]. Thus, the actual speedup that can be achieved depends on the problem at hand.

Optimization problems, which are the topic of interest of this paper, have received much attention in the quantum setting. Using Grover's algorithm, Dürr, Heiligman, Høyer, and Mhalla [12] have shown that it is possible to find the global optimum of a black-box optimization problem on the search space $\{0,1\}^n$ in an expected number of $O(2^{n/2})$ queries, while the classical complexity is $\Theta(2^n)$. Moreover, a matching lower bound of $\Omega(2^{n/2})$ for all possible quantum algorithms exists [26]. In addition, if there is enough structure in the search space, better bounds can be shown. For example on general graph-based search spaces, Magniez, Nayak, Roland, and Santha [21, Theorem 3] have shown that if the Markov chain associated with the random walk on the space is ergodic, significant improvement in the expected query complexity is possible provided that the spectral gap is large. Furthermore, if the underlying *quantum random walk* is symmetric, better problem-specific quantum algorithms are available [25, 20].

From the view-point of complexity theory, the model of computation which allows the formulation of quantum algorithms is a generalization of the model of computation which allows the formulation of classical randomized algorithms (just as this model is a generalization of that which allows the formulation of deterministic algorithms). If we consider the complexity of a computational problem, then (i) upper bounds on the classical complexity are also upper bounds on the quantum complexity, (ii) lower bounds on the quantum complexity are also lower bounds on the classical complexity,

[1]By *quadratic speedup* we mean that the order of the runtime of the classical version of an algorithm is quadratic in the runtime of the quantum version.

and, most importantly, (iii) in certain settings lower bounds on the *quantum complexity* can imply even stricter lower bounds on the *classical complexity*.

For example, in [1] Aaronson has derived a lower bound of $\Omega(2^{n/2}/n^2)$ on the randomized complexity of the problem of finding a local optimum on the n-dimensional hypercube from an $\Omega(2^{n/4}/n)$ bound on its quantum complexity using the *quantum adversary method* (cf. [3]). Only later has this result been improved to $\Theta(2^{n/2}n^{1/2})$ using classical methods of analysis [28].

Another example is given in [19], where the authors show lower bounds on the length of locally decodable codes with quantum arguments. They show that the classical algorithm which is allowed to look at two bits (in order to recover the desired bit of information) is at most as powerful as the quantum algorithm which is allowed to look only at one bit.

In this light, theoretical work on quantum complexity and runtime analyses of quantum algorithms serve not only the aim of preparing for a bright (but admittedly as of now only hypothetical) future sporting actual quantum computers, but also furthers the understanding of the classical complexity of a problem.

In this work, we consider quantum versions of *elitist (1+1) randomized search heuristics*, which, for convenience, we simply abbreviate as *RSH*s. RSHs are heuristics that successively generate candidate solutions according to some distribution depending only on the best solution found so far[2]. In the language of evolutionary algorithms, this sampling procedure is called *mutation*. In a subsequent *selection* step, the algorithm decides whether to replace the current candidate solution by the sampled one.

The optimization problems that we are interested in are pseudo-Boolean optimization problems: Given a finite search space $\mathcal{S}$, in our case always the set of all n-bit strings (where n is a positive integer), and an objective function f from $\mathcal{S}$ to $\mathbb{R}$, we want to compute a global optimum (that is, either a maximum or a minimum) of f. A RSH starts with a candidate solution $\mathbf{x}^{(0)}$ and repeatedly improves the objective value (or *fitness*) of the solution by performing the following two steps:

(1) Mutation Generate a new solution $\mathbf{y}$ according to a distribution $\textsc{mut}(\mathbf{x})$ depending on the current solution $\mathbf{x}$;

(2) Selection If the new solution $\mathbf{y}$ has better (or possibly equal) fitness than $\mathbf{x}$ then set $\mathbf{x} := \mathbf{y}$, otherwise discard $\mathbf{y}$.

Thus, different RSHs differ in the *mutation operator*, that is, the nature of the distribution $\textsc{mut}(\mathbf{x})$, and the *selection strategy*, that is, a partial order relation on $\mathcal{S}$.

[2]Actually, the distribution of the candidate solutions might as well depend on the number of steps already performed by the algorithm. However, in this work we only regard algorithms with time-homogeneous distributions.

For example, the mutation operator of Randomized Local Search (RLS) selects an index i at random and flips the bit x_i to get the new candidate solution whereas the mutation operator of the (1+1) Evolutionary Algorithm (EA) flips each bit x_i with probability $1/n$. Orthogonally, we can choose one of the following two selection strategies for each of these algorithms, which we name *progressive* or *conservative* selection. Progressive selection accepts new search points of equal or better objective value, whereas conservative selection only accepts new search points of strictly better objective value (where *better* means larger or smaller, depending on whether we consider a maximization or, respectively, a minimization problem). Whenever two algorithms are the same except for the selection strategy, we denote the conservative algorithms by a superscript " * ", e.g., RLS and RLS* for the progressive and conservative versions of Randomized Local Search, respectively.

When transferring the concept of RSHs to the setting of quantum computing, we encounter one main difficulty. Clearly, an RSH for a maximization problem can never move from a solution of higher objective value to a solution of smaller objective value. Hence the Markov processes underlying these algorithms are not ergodic and far from symmetric. Therefore, the setting of quantum random walks as in [25, 20, 21] does not apply. More seriously, all quantum operations apart from *measurements* are required to be reversible. This rules out a direct adaptation of the RSH to the quantum world because it would force us to make a measurement after every mutation step and perform the selection step based on the outcome of this measurement. Performing a measurement after each mutation amounts to sampling from the classical distribution associated with the mutation. Hence, such a version is the restatement of the same classical randomized search heuristic in terms of quantum operators and measurements and therefore uninteresting. We need to defer the measurements long enough so as to allow quantum mechanical interference to have an effect on the sampling.

The main idea of our paper is to use *quantum probability amplification* [10] to speed up the process of generating a new candidate solution in Step (1) that is accepted by the selection strategy in Step (2). Instead of picking a new candidate solution directly from the distribution given by the mutation operator, which is what is done classically, we amplify the probability of getting a better solution to at least a constant (say 1/2) using quantum probability amplification (see Section 2). We call this quantum variant of a RSH a *Quantum Search Heuristic* (QSH). In particular, we call the quantum variants of RLS and EAs *Quantum Local Search* (QLS) and *Quantum Evolutionary Algorithms* (QEAs). These QSHs can only run on a quantum computer.

We measure the runtime of a RSH on a given problem by the expected number of function calls (*queries*) to f until a global optimum is found. In the quantum setting, we cannot evaluate the function f for mixed (nonclassical) quantum states. Instead, for each search point $\mathbf{x}$ we construct a membership oracle associated with f that distinguishes between search points of higher and of lower fitness than $\mathbf{x}$. We define the runtime of a quantum algorithm to be the expected number of calls to such membership oracles for f until a global optimum is found (see Section 4 for details).

For comparing the runtime of a RSH and its quantum counterpart, the *progress probability* $p_{\mathrm{RSH}}(\mathbf{x})$ plays a central role. This is the probability of obtaining an acceptable new solution from a candidate solution $\mathbf{x}$ by mutation in the classical setting. That is, assuming the RSH maximizes the function f, if $\mathbf{y}$ is the random variable generated by $\textsc{mut}(\mathbf{x})$ then

$$p_{\mathrm{RSH}}(\mathbf{x}) = \Pr(f(\mathbf{y}) \geq f(\mathbf{x}) \wedge \mathbf{y} \neq \mathbf{x})$$

for the progressive selection strategy and

$$p_{\mathrm{RSH}}(\mathbf{x}) = \Pr(f(\mathbf{y}) > f(\mathbf{x}))$$

for the conservative selection strategy. In the classical setting, a RSH requires in expectation $1/p_{\mathrm{RSH}}(\mathbf{x})$ queries to f in order to make progress in $\mathbf{x}$, that is, to move from the current solution $\mathbf{x}$ to a new solution $\mathbf{y}$. Using quantum probability amplification, the expected number of queries reduces to $\Theta\big(1/\sqrt{p_{\mathrm{RSH}}(\mathbf{x})}\big)$.

The search heuristic Quantum Local Search which we define here is a restricted version of the quantum algorithm by Aaronson [1] which first chooses $\Theta(n^{1/3}2^{2n/3})$ search points uniformly at random and then uses Grover search to determine the optimal initial search point among them. The algorithm of Aaronson then proceeds exactly like ours. However, our algorithm does not attempt to optimize on the starting point, because (i) the runtime of such an optimization would dominate the runtimes of our algorithms by orders of magnitude for problems with polynomial runtime and (ii) the classical RSHs we compare to do not attempt to do so either.

To avoid confusion, we want to point out two other streams of work which might be mistakenly associated with our work but are in fact not related.

First, our results on QSHs and QEAs are significantly different from that of Quantum Inspired Evolutionary Algorithms (QIEAs) as introduced in [16]. QIEAs are classical algorithms where the mutation and selection steps, though classical, are inspired from quantum operations. In contrast, our mutation process is genuinely quantum and cannot be implemented on a classical computer.

Second, our algorithms are no attempts to apply genetic programming techniques to better design quantum algorithms unlike for example the work of Spector *et. al.* [24] where the "code" of an ordinary quantum algorithm is optimized by an evolutionary algorithm. To the best of our knowledge, the (1+1) QEA investigated here is the first attempt to generalize evolutionary algorithms to the quantum setting.

	(1+1) EA / RLS	(1+1) QEA / QLS	(1+1) EA* / RLS*	(1+1) QEA* / QLS*
OneMax	$\Theta(n \log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n)$
LeadingOnes	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^{3/2})$
Discrepancy	$\Theta(\sqrt{n})$	$\Theta(\sqrt{n})$	$\Theta(\sqrt{n})$	$\Theta(\sqrt{n})$
Needle	$\Theta(2^n)$	$\Theta(2^n)$	$\Theta(\frac{1}{2^n} n^n)$ / ∞	$\Theta(\frac{e^{\sqrt{n}}}{2^n} n^{n/2})$ / ∞
Jump$_m$	$\Theta(n^m)$ / ∞	$\Theta(n^{m-1/2})$ / ∞	$\Theta(n^m)$ / ∞	$\Theta(n^{m/2})$/ ∞
TinyTrap	$\Omega(2^{n/4})$ / ∞	$\Theta(1)$ / ∞	$\Omega(2^{n/4})$ / ∞	$\Theta(1)$/ ∞

Table 1: A runtime comparison between progressive and conservative (*) RSHs and QSHs on different objective functions.

The theorem "No free lunch" states that no RSH can be good on all pseudo-Boolean problems. Similarly, the general bound of time $\Theta(2^{n/2})$ for optimizing an arbitrary pseudo-Boolean function in the black-box model also applies to QSHs. However, we may ask whether QSHs still experience a speedup over ordinary RSHs on particular problems. In order to answer this question, we follow the approaches of [8] and [11] and study the behavior of QLS and the (1+1) QEA on specific pseudo-Boolean optimization problems.

1.1 Our Results

We give asymptotically tight bounds on the runtimes of QLS and the (1+1) QEA. For both QSHs, we investigate the progressive and the conservative selection strategy. We consider the objective functions OneMax, LeadingOnes, Discrepancy, Needle, Jump$_m$, and TinyTrap, which are defined in Section 6.

Our results are summarized in Table 1. We see that in some cases the speedup is quadratic or almost quadratic (e.g., conservative (1+1) QEA* on Needle and on Jump$_m$), for other functions there is only a smaller speedup (polynomial for the conservative algorithms on LeadingOnes, logarithmic for all algorithms on OneMax), or no asymptotic speedup at all (Discrepancy; progressive algorithms on Needle and Jump$_m$). We also see that the selection strategy may be crucial for the speedup (Jump$_m$ and LeadingOnes) even if one of the strategies does not change the runtime of the classical algorithms (LeadingOnes). We even find an example (TinyTrap) where the runtime decreases from $\Omega(2^{n/4})$ to $\Theta(1)$.

We now give a broad reason for the differences in speedup. The quantum acceleration does not differ from its classical counterpart in the statistical nature of the candidate solutions picked on its way to the optimal solution. In other words, for a fixed trajectory of different search points the probabilities to take this trajectory coincide for the classical and the quantum algorithm.

However, the quantum version is faster because it reduces the expected time required for a successful mutation.

For example, for RLS* on LeadingOnes, it is comparatively hard to find the next search point (that is, the right bit to flip), so there is a substantial speedup. On the other hand, for Discrepancy it is very easy to find a better search point: the expected time for improving the objective value is bounded from above by a constant, and so there is no asymptotic speedup. In general, the speedup is higher when it is difficult to find a search point that is accepted by the selection strategy.

Therefore, the gap between the RSHs and the QSHs tends to be larger for the conservative selection strategy. Again for LeadingOnes, the progress probability of RLS* is $1/n$, namely to flip the first zero-bit. Consequently, the runtime decreases by a factor of $\Theta(\sqrt{n})$. On the other hand, in early steps of a run of RLS (that is, the progressive version) on LeadingOnes, the progress probability is $\Theta(1)$, since flipping any bit after the first zero-bit results in an accepted candidate solution in this setting. Consequently, there is no asymptotic speedup at all.

However, this does not necessarily imply that conservative algorithms are superior. For some problems like Needle and Jump$_m$, the conservative selection strategy leads to high or even infinite (the global optimum is never found) runtimes for both RSHs and QSHs. In this case, the speedup due to quantum computing is negligible compared to the speedup due to progressive selection.

Finally, let us briefly discuss the example of TinyTrap where both, the conservative and the progressive (1+1) QEA, have runtimes bounded from above by a constant instead of the exponential runtime of their classical counterparts. This result may seem most impressive. However, it is quite artificial since the results hold only in expectation but with an exponentially small probability. In a nutshell, it is based on the following observation about quantum probability amplification. Consider the search space consisting only of two elements a and b, where b is defined to be the "optimum". With probability p, the initial search point is chosen as a and with probability $1-p$ as b. The probability that a mutation in a samples b is set to p^2. Then the corresponding RSH has runtime $1/p$ (starting in a with probability p and moving to b after $1/p^2$ steps in expectation), while the corresponding QSH has runtime $\Theta(1)$ (starting in a with probability p and moving to b after only $1/p$ steps in expectation). Thus, the speedup $1/p$ can be arbitrary large if we choose p small enough. The function TinyTrap models this situation on the search space $\{0,1\}^n$.

Summing up, we see that quantum search may speed up evolutionary algorithms in some cases and that there are problems which are substantially accelerated by quantum search. However, it depends on the specific problem how much is really gained, and for some problems there is no improvement in the runtime at all.

This paper is the sequel of a conference paper by the same authors [18]. The prequel paper only considered the problems OneMax, Discrepancy and LeadingOnes. By including Needle, Jump$_m$, and TinyTrap we are able to demonstrate the impact of different selection strategies, while the conference version only considered the progressive variant. Also, it did not include rigorous proofs and formulations, but merely stated informally that the optimization time of a quantum algorithm should equal $\sum_{t \leq T} p_t^{-1/2}$, where p_t is the progress probability in time step t and T is the number of steps the algorithm needs. Although this formula captures the intuition, it is difficult to turn it into a rigid definition. In particular, p_t is not a random variable because it is not always defined. Also, the formula is not well-suited for computations, and the analyses used in fact slightly different formulas. Theorem 5.11 of this paper removes this defect.

1.2 Outline

The paper is structured as follows. In Section 2, we review very briefly the quantum algorithms we need. For the reader not familiar with quantum computations, it suffices to use Theorem 2.2 as a black box. In Section 3 and Section 4, RSHs and QSHs, respectively, are introduced formally.

In Section 5, we provide tools for analyzing the runtimes of QSHs. In the standard framework for evolutionary algorithms one query is performed for each candidate search point $\mathbf{x}^{(t)}$, so the runtime is simply the expectation of the minimal t for which $\mathbf{x}^{(t)}$ is a global optimum. Unfortunately, this framework collapses in the quantum setting, since the number of queries (calls to the oracle function) needed to produce the next search point is not constant. Instead, we introduce in Definition 5.4 the notion of *progress times*, which is compatible with both the classical and the quantum setting.

In Theorem 5.11, we describe the runtimes of QSHs purely in non-quantum terms, and in the remainder of Section 5, we prove some lemmas that are useful for comparing the complexity of a RSH and its corresponding QSH. Finally, in Section 6 we apply these tools to determine the runtimes of the introduced QSHs for the problems OneMax, LeadingOnes, Discrepancy, Needle, Jump$_m$, and TinyTrap.

2 Quantum Probability Amplification

In this section, we describe the basics of quantum computation, the Grover search algorithm, and its reformulation as quantum probability amplification in a form that is suitable for our purpose. For a detailed presentation, we refer the reader to any standard text book on quantum computation, e.g. [22].

The most basic unit of information in the quantum setting is a qubit, which is a unit vector in a 2 dimensional vector space $\mathcal{H} = \mathbb{C}^2$. To see the analogy with classical bits, we fix an orthonormal basis $\{|0\rangle, |1\rangle\}$. The first basis vector $|0\rangle$ stands for the classical bit 0 and the second basis vector $|1\rangle$ stands for the classical bit 1. However, a qubit can also be in the superposed state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ where α and β are complex numbers satisfying the relation $|\alpha|^2 + |\beta|^2 = 1$. An n-qubit system is captured by a unit vector in the n-fold tensor product $\mathcal{H}^{\otimes^n} = \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$.

The vector space $\mathcal{H}^{\otimes^n}$ is a 2^n-dimensional vector space with orthonormal basis $\{|\mathbf{x}\rangle \mid \mathbf{x} \in \{0,1\}^n\}$. More generally, if we have a sample space Ω of cardinality N in the classical setting, the corresponding object in the quantum setting is an N-dimensional complex Hilbert space $\mathcal{H}_\Omega$. A set of orthonormal vectors of $\mathcal{H}_\Omega$ is fixed and is denoted by $\{|\omega\rangle \mid \omega \in \Omega\}$. A *state* of the system is then a unit vector in $\mathcal{H}_\Omega$.

Any quantum operation is either an application of a *unitary operator* or a *measurement*. Unitary operations preserve inner products between vectors and are reversible. The measurement is an irreversible process. We mention the von Neumann measurement postulate for qubits. If a set of n-qubits $\sum \alpha_{\mathbf{x}} |\mathbf{x}\rangle$ is measured, we obtain the outcome $|\mathbf{x}_0\rangle$ with probability $|\alpha_{\mathbf{x}_0}|^2$. Furthermore, the state of the original system then collapses to $|\mathbf{x}_0\rangle$, so the original state is irretrievably lost. So, as far as measurement is considered, a state is like a distribution in the classical setting. What makes quantum probability different and thus more powerful than classical computation is that by combining certain unitary operations with measurement of a latter state, we can perform a "constructive interference" of good probabilistic paths. It is the correct use of unitary maps together with the correct timing of measurement that gives quantum computation its power. Due to lack of space we leave the details to any standard text book on quantum computation (c.f. [22]).

We now describe the Grover search algorithm. Let $\mathcal{S}_0$ be a subset of $\mathcal{S}$ for which we are given a membership oracle, that is, we are given an oracle M from $\mathcal{S}$ to $\{0,1\}$ such that $\mathcal{S}_0 = \{\mathbf{x} \mid M(\mathbf{x}) = 1\}$. Our task is to *search* for a string in $\mathcal{S}_0$ using queries to M. In this setting, we are interested in minimizing the number of queries made to M. In an important breakthrough, Grover [15] gave a quantum algorithm to search for such an element $\mathbf{x}_0 \in \mathcal{S}_0$ that makes only $\sqrt{|\mathcal{S}|/|\mathcal{S}_0|}$ queries to the oracle M. One needs to, however, make the oracle M work for quantum states. The standard approach, which we describe briefly for completeness, is to consider the membership oracle as unitary operator U_M on the n-qubit Hilbert space $\mathcal{H} = \mathbb{C}^{2^{\otimes^n}}$ defined as

$$U_M|\mathbf{x}\rangle = (-1)^{M(\mathbf{x})}|\mathbf{x}\rangle .$$

One application of this unitary operator U_M is considered as a single query to the membership oracle.

Let N denote the cardinality of the search space $\mathcal{S}$, and let the cardinality of the set $\mathcal{S}_0$ be N_0. During initialization, Grover's quantum search algorithm prepares the uniform superposition $|\psi_0\rangle = 1/\sqrt{N} \sum_{\mathbf{x}\in\mathcal{S}} |\mathbf{x}\rangle$. The algorithm iteratively applies the Grover step, a unitary operator which we denote by G, to $|\psi_0\rangle$. Let $|\psi_t\rangle$ denote the state after t applications of G, that is, $|\psi_t\rangle = G^t |\psi_0\rangle$. If we choose some appropriate t in $O(\sqrt{N/N_0})$ then on measuring the state $|\psi_t\rangle$ we obtain an element $\mathbf{x} \in \mathcal{S}_0$ with probability bounded from below by a positive constant. More precisely, if we write the state as $|\psi_t\rangle = \sum_{\mathbf{x}\in\mathcal{S}} \alpha_{\mathbf{x}}(t) |\mathbf{x}\rangle$, then for $t = O(\sqrt{N/N_0})$ we have $\sum_{\mathbf{x}\in\mathcal{S}_0} |\alpha_{\mathbf{x}(t)}|^2$ is a constant (say 1/2). The exact form of the Grover step G is not relevant (for details see the text book of Nielsen and Chuang [22, Chapter 6]) but the crucial point is that G can be constructed using one application of the unitary operator U_M. Hence the Grover search makes $\sqrt{N/N_0}$ queries to the oracle.

Grover search starts with the uniform superposition as a priori there is no specific reason to prefer one bit string over the other. Instead, if we start the search algorithm with the state $|\psi_0\rangle = \sum \alpha_{\mathbf{x}} |\mathbf{x}\rangle$, then the runtime will be $\sqrt{1/p}$ where $p = \sum_{\mathbf{x}\in\mathcal{S}_0} |\alpha_{\mathbf{x}}|^2$ is the probability of picking $\mathbf{x} \in \mathcal{S}_0$ if we would measure the initial state $|\psi_0\rangle$ directly. This reformulation due to Brassard *et al* [10] is often called the *quantum probability amplification* or *quantum amplitude amplification* as a quantum algorithm is able to amplify the probability by performing just $\sqrt{1/p}$ queries in expectation as opposed to $1/p$ required by a classical algorithm.

There is a caveat to the Grover search algorithm. One needs to stop the Grover iteration after $\Theta(\sqrt{N/N_0})$ steps, for otherwise the probability of getting a favorable $\mathbf{x}_0 \in \mathcal{S}_0$ actually deteriorates. Thus it appears as if without knowing the count $|\mathcal{S}_0|$, or in the case of probability amplification, the probability p of sampling an $x \in \mathcal{S}_0$ under the given distribution, one cannot use Grover search. However, using phase estimation, Brassard *et al* [10] gave a way to overcome this difficulty with essentially no change in the overall runtime. From now on, by quantum probability amplification we mean this generalized version where we do not need to know the probabilities.

We now explain an important invariant of the Grover search algorithm. This property is crucial for our results. Even though each Grover iteration amplifies the probability of finding a solution in $\mathcal{S}_0$, for any $\mathbf{x}_0 \in \mathcal{S}_0$, the conditional probability of obtaining $\mathbf{x}_0$ given the event that an element of $\mathcal{S}_0$ has been obtained remains unchanged by the algorithm. We sketch the reason for this. Let $\mathcal{H}$ be the Hilbert space of n-qubits, which has as basis $\{|\mathbf{x}\rangle \mid \mathbf{x} \in \mathcal{S}\}$. Let $\mathcal{H}_A$ denote the space spanned by $\{|\mathbf{x}\rangle \mid \mathbf{x} \in \mathcal{S}_0\}$, and let $\mathcal{H}_B = \mathcal{H}_A^{\perp}$ be its orthogonal complement. Consider any vector $|\psi\rangle = \sum_{\mathbf{x}} a_{\mathbf{x}} |\mathbf{x}\rangle$ in $\mathcal{H}$. Then $|\psi\rangle = \alpha |\psi\rangle_A + \beta |\psi\rangle_B$ where $|\psi\rangle_A$ and $|\psi\rangle_B$ are the projections of $|\psi\rangle$ to $\mathcal{H}_A$ and $\mathcal{H}_B$ with their norms normalized to 1, respectively. It is easy to verify that the normalized projection $|\psi\rangle_A$ is given by $|\psi\rangle_A = \frac{1}{\sqrt{|\alpha|^2}} \sum_{\mathbf{x}\in\mathcal{S}_0} a_{\mathbf{x}} |\mathbf{x}\rangle$ and the amplitudes α and β are given by $|\alpha|^2 = \sum_{\mathbf{x}\in\mathcal{S}_0} |a_{\mathbf{x}}|^2$ and $|\beta|^2 = 1 - |\alpha|^2$. The following proposition then follows from the measurement postulate.

Proposition 2.1. *Consider the probability distribution D_ψ on $\mathcal{S}$ obtained by measuring the state $|\psi\rangle$. Then*

1. *the probability to obtain an element from $\mathcal{S}_0$, that is,* $\Pr[\mathcal{S}_0]$, *is* $|\alpha|^2$,
2. *for $|\psi\rangle_A = \sum_{\mathbf{x}\in\mathcal{S}_0} \gamma_\mathbf{x} |\mathbf{x}\rangle$, the conditional probability* $\Pr[\mathbf{x}|\mathcal{S}_0]$ *is* $|\gamma_\mathbf{x}|^2$,

where all probabilities and conditional probabilities are with respect to the distribution D_ψ.

Proof. By the measurement postulate, the probability to obtain $\mathbf{x} \in \mathcal{S}$ is $|a_\mathbf{x}|^2$. Therefore,

$$\Pr[\mathcal{S}_0] = \sum_{\mathbf{x}\in\mathcal{S}_0} \Pr[\mathbf{x}] = \sum_{\mathbf{x}\in\mathcal{S}_0} |a_\mathbf{x}|^2 = |\alpha|^2 .$$

Similarly, the second statement follows from

$$\Pr[\mathbf{x}|\mathcal{S}_0] = \frac{\Pr[\mathbf{x}]}{\Pr[\mathcal{S}_0]} = \frac{|a_\mathbf{x}|^2}{|\alpha|^2} = |\gamma_\mathbf{x}|^2 .$$

□

Let G be the Grover iteration associated with the solution space $\mathcal{S}_0$. For any vector $|\psi\rangle = \alpha |\psi\rangle_A + \beta |\psi\rangle_B$ the vector $G|\psi\rangle$ is a linear combination $\alpha' |\psi\rangle_A + \beta' |\psi\rangle_B$ for some other constants α' and β' such that $|\alpha'|^2 + |\beta'|^2 = 1$ (see the analysis of Grover search in Section 6.1.3 of Nielsen's and Chuang's book [22]). It follows from Proposition 2.1 that the conditional probability $\Pr[\mathbf{x}|\mathcal{S}_0]$ does not change after the application of G.

When quantum probability amplification is applied to an initial state that has success probability p, then it achieves at least a constant success probability $c > 0$ with a runtime in $\Theta(\sqrt{1/p})$ that is known *a priori*. Now we take this algorithm as a black box B. If we consider the algorithm that repeats B until it finds a solution, then its success probability is 1 by definition. The runtime is no longer known *a priori* since we do not know how often we need to call the black box B. However, the *expected* runtime is $1/c$ times the runtime of B, which is still in $\Theta(\sqrt{1/p})$. Summarizing, we get the following theorem.

Theorem 2.2 (Probability Amplification). *There exists two positive absolute constants c and C such that the following statement is true.*

Let $\mathcal{S}$ be a finite search space, $\mathcal{S}_0$ be any non-empty subset of $\mathcal{S}$ for which there is a membership oracle M, and a sampling procedure A that produces a distribution D_A on $\mathcal{S}$. Let p be the probability $\Pr_{D_A}[\mathbf{x} \in \mathcal{S}_0]$ *of obtaining an element in $\mathcal{S}_0$ on running A. Then there exists a quantum algorithm that makes in expectation at least $cp^{-1/2}$ and at most $Cp^{-1/2}$ queries to the* *membership oracle M and samples an element $\mathbf{x}_0$ in $\mathcal{S}_0$ with a distribution D_ψ on $\mathcal{S}_0$ given by*

$$\mathrm{Pr}_{D_\psi}[\mathbf{x} = \mathbf{x}_0] = \mathrm{Pr}_{D_A}[\mathbf{x} = \mathbf{x}_0 \mid \mathbf{x} \in \mathcal{S}_0].$$

The statement regarding the conditional probability comes from the fact that the final state of the algorithm is $G^t \left|\psi_0\right\rangle$ and that the Grover operator G preserves the relevant conditional probability as discussed before.

3 Randomized Search Heuristics

In this section, we look at elitist (1+1) Randomized Search Heuristics, RSHs for short. The RSHs we study in this article are Random Local Search (RLS) and the (1+1) Evolutionary Algorithm (EA). Let $\mathcal{S}$ be the search space and let f be a function from $\mathcal{S}$ to $\mathbb{R}$ that we want to maximize. A RSH like Random Local Search or the (1+1) EA can be formalized by defining what is known as its *mutation operator*.

Definition 3.1 (Mutation Operator MUT)**.** *Let $\mathcal{S}$ be a finite search space. A* mutation operator MUT *over $\mathcal{S}$ is a function from $\mathcal{S}$ to the space of probability distributions on $\mathcal{S}$.*

The mutation operator MUT is essentially the search strategy of the corresponding RSH. With a slight abuse of notation we write MUT$(\mathbf{x})$ to denote a sample from $\mathcal{S}$ according to the distribution MUT$(\mathbf{x})$.

To any mutation operator MUT, we associate a RSH, which starts from an initial solution $\mathbf{x}^{(0)}$, and successively improves by mutating the current solution according to the probability distribution given by MUT. If the new solution is better than the current one, we discard the current solution and keep the new solution for further improvement. On the other hand, if the new solution is worse we discard it and retain the current solution. If the new solution is of the same fitness as the current solution, we can either choose to retain the current solution or move to the newly generated solution. We call the former strategy *conservative* and the latter *progressive*. Which of these two variant is better depends very much on the problem at hand. We now formalize these two algorithms.

Algorithm 3.2 (RSH)**.** *The* elitist (1+1) randomized search heuristic *(RSH) over the finite search space $\mathcal{S}$ with mutation operator* MUT *that maximizes the objective function $f : \mathcal{S} \to \mathbb{R}$ is the following iterative algorithm.*

1. *Start with $\mathbf{x}^{(0)} \in \mathcal{S}$ uniformly at random.*
2. *For each $t \in \mathbb{N}$, iteratively assume that $\mathbf{x}^{(t)}$ has been chosen.*
    (a) Pick $\mathbf{y}^{(t)} \in \mathcal{S}$ according to the distribution MUT$(\mathbf{x}^{(t)})$.

(b) Set $\mathbf{x}^{(t+1)} = \mathbf{y}^{(t)}$ *if*

- $f(\mathbf{y}^{(t)}) > f(\mathbf{x}^{(t)})$ *for the* conservative *selection rule,*
- $f(\mathbf{y}^{(t)}) \geq f(\mathbf{x}^{(t)})$ *for the* progressive *selection rule.*

Otherwise, set $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$.

One can define a RSH for *minimizing* f by changing Step 2b of Algorithm 3.2. For the conservative selection strategy, we set $\mathbf{x}^{(t+1)} = \mathbf{y}^{(t)}$ if $f(\mathbf{y}^{(t)}) < f(\mathbf{x}^{(t)})$. For the progressive selection strategy, we set $\mathbf{x}^{(t+1)} = \mathbf{y}^{(t)}$ if $f(\mathbf{y}^{(t)}) \leq f(\mathbf{x}^{(t)})$.

Based on this general scheme of a RSH, we define Randomized Local Search and the (1+1) Evolutionary Algorithm by their corresponding mutation operators.

Algorithm 3.3 (RLS and RLS*)**.** Randomized Local Search *is the RSH on the search space* $\{0,1\}^n$ *for which the mutation operator* $\textsc{mut}_{\text{RLS}}$ *assigns to the search point* $\mathbf{x}$ *the probability distribution on* $\{0,1\}^n$ *obtained by picking an index* $1 \leq i \leq n$ *uniformly at random and flipping the i-th bit of* $\mathbf{x}$*. We denote the progressive variant of Randomized Local Search by RLS and the conservative variant by RLS*.*

Algorithm 3.4 ((1+1) EA and (1+1) EA*)**.** *The* (1+1) Evolutionary Algorithm *is the RSH on the search space* $\{0,1\}^n$ *for which the mutation operator* $\textsc{mut}_{\text{EA}}$ *assigns to the search point* $\mathbf{x}$ *the probability distribution on* $\{0,1\}^n$ *obtained by flipping each bit of* $\mathbf{x}$ *independently with probability* $\frac{1}{n}$*. We denote the progressive variant of the (1+1) Evolutionary Algorithm by (1+1) EA and the conservative variant by (1+1) EA*.*

4 Quantum Search Heuristics

We now study quantum versions of RSHs. As in the previous section, we have a search space $\mathcal{S}$ and an objective function $f\colon \mathcal{S} \to \mathbb{R}$ that we want to maximize. Consider a mutation operator $\textsc{mut}$. Recall that, in the classical randomized search heuristic, we successively improve the current solution by sampling a new solution according to the mutation operator $\textsc{mut}$ and retaining the new solution if it is better than the previous one. In the quantum version, all the mutation and selection operations needed to find an improved solution are considered as a single search, and we use quantum probability amplification to speed up this search: In step k, if $\mathbf{x}^{(k)}$ denotes the current solution, we generate the distribution $\textsc{mut}(\mathbf{x}^{(k)})$, amplify the probability of getting a better solution using quantum probability amplification and measure the amplified distribution to obtain a new solution.

The quantum probability amplification requires a membership oracle. Given the objective function f, we define, for each $\mathbf{x} \in \mathcal{S}$, membership oracles $M_{f,\mathbf{x}}$ (progressive version) and $M^*_{f,\mathbf{x}}$ (conservative version) as follows.

$$M_{f,\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 & \text{if } f(\mathbf{y}) \geq f(\mathbf{x}) \text{ and } \mathbf{y} \neq \mathbf{x}, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$M^*_{f,\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 & \text{if } f(\mathbf{y}) > f(\mathbf{x}) \\ 0 & \text{otherwise.} \end{cases}$$

We now define (progressive and conservative) elitist (1+1) Quantum Search Heuristics (QSHs) associated with a mutation operator MUT.

Algorithm 4.1 (QSH)**.** *The* elitist (1+1) quantum search heuristic *(QSH) over the finite search space $\mathcal{S}$ with mutation operator* MUT *that maximizes the objective function $f : \mathcal{S} \to \mathbb{R}$ is the following iterative algorithm.*

1. *Start with $\mathbf{x}^{(0)} \in \mathcal{S}$ uniformly at random.*
2. *For each $k \in \mathbb{N}$, iteratively assume that $\mathbf{x}^{(k)}$ has been picked. Sample $\mathbf{x}^{(k+1)}$ according to sampling procedure for Theorem 2.2 with search space $\mathcal{S}$, membership oracle $M_{f,\mathbf{x}^{(k)}}$ for progressive and $M^*_{f,\mathbf{x}^{(k)}}$ for conservative selection, and sampling procedure* MUT$(\mathbf{x}^{(k)})$*. In the case that the set of possible samples for $\mathbf{x}^{(k+1)}$ is empty (that is, the mutation operator cannot reach a better search point) then set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$.*

It seems that the condition $\mathbf{y} \neq \mathbf{x}$ in the oracle function $M_{f,\mathbf{x}}$ does not have any influence on the algorithm. This is true in so far as the algorithm would not visit different search points if this condition was dropped. However, from the runtime analysis in Section 5 it will become clear that there would be a huge difference in the runtime. In particular, the (1+1) EA has at least a constant positive probability to sample the current search point $\mathbf{x}$ again. Therefore, the algorithm (1+1) QEA defined below would have asymptotically exactly the same runtime as the classical (1+1) EA for every objective function f by Corollary 5.13, so there would be no gain in speed at all. For the classical version, on the other hand, this changes the runtime of the algorithm by at most a constant factor which does not influence our runtime analysis.

We conclude this section by defining the quantum versions of Randomized Local Search and the (1+1) Evolutionary Algorithm. Note that these definitions arise directly from the corresponding mutation operators.

Algorithm 4.2 (QLS and QLS*)**.** Quantum Local Search *is the QSH on the search space $\{0,1\}^n$ defined by the mutation operator* MUT$_{\text{RLS}}$ *of Randomized Local Search (Algorithm 3.3). We denote the progressive variant of Quantum Local Search by QLS and the conservative variant by QLS*.*

Algorithm 4.3 ((1+1) QEA and (1+1) QEA*)**.** *The* (1+1) Quantum Evolutionary Algorithm *is the QSH on the search space* $\{0,1\}^n$ *defined by the mutation operator* $\textsc{mut}_{\text{EA}}$ *of the (1+1) Evolutionary Algorithm (Algorithm 3.4). We denote the progressive variant of the (1+1) Quantum Evolutionary Algorithm by (1+1) QEA and the conservative variant by (1+1) QEA*.*

5 Runtime Analysis of Quantum Search Heuristics

In this section, we introduce a selection of methods which allow us to link the runtime of a QSH to the optimization behavior of the corresponding RSH. In the first two sub-sections, we develop the basic terminology and formulas. Afterwards, we present the main theorem (Theorem 5.11), which expresses the runtime of a QSH by a purely non-quantum parameter of the corresponding RSH. Moreover, we give tools to relate the runtimes of the RSH and the QSH if the probability of moving to a new search point in the next step is bounded (Corollary 5.13), or if it is bounded in certain regions of the search space, where the time spent in these regions is known for the classical algorithm (Lemma 5.12). Finally, we derive an alternative characterization of the runtime of a QSH by scaling the transition probabilities of the Markov chain associated to the corresponding RSH.

For the results of this section we fix a positive integer n and consider the optimisation of an objective function f on the domain $\mathcal{S} = \{0,1\}^n$ of n-bit strings. In this section, we fix a mutation operator $\textsc{mut}$ on $\mathcal{S}$ and a selection strategy (either progressive or conservative). By fixing these, recall that the RSH (Algorithm 3.2) and its associated QSH (Algorithm 4.1) that optimises f is completely determined.

In the following, whenever a definition or statement applies to both heuristics, we simply refer to both as the *considered search heuristic.* In particular, we use a common mathematical notation for both heuristics and signify the distinction by the subscripts "RSH" and "QSH" only if needed. For example, in Definition 5.1 we define the "optimization time T of the considered search heuristic". By this, we implicitly define T_{RSH} for the RSH and T_{QSH} for the QSH.

Our aim is to compare the performance of different search heuristics. To this end, we assume the query complexity model: The considered search heuristics are only charged for the number of queries it makes to the objective function, all other operations are free of cost. Recall that in the case of the RSH, a *query* is an evaluation of the objective function f. For the QSH, a *query* is an evaluation of the associated membership oracle. To ease the following calculations (and since we are only interested in asymptotic results anyhow), we do not charge the RSH or QSH for querying the first search point $\mathbf{x}^{(0)}$.

Since we use the query complexity model, we define the *runtime* of the search heuristic as its expected optimization time.

Definition 5.1 (Optimization Time)**.** *The* optimization time *of the considered search heuristic is the random variable T that denotes the number of queries performed by the search heuristic until it has found the first (globally) optimal search point. The* runtime *of the considered search heuristic is its expected optimization time.*

In general, the runtime of the considered search heuristic is unbounded. For example, if the objective function has a local optimum (Jump, Needle, TinyTrap), then the runtime of RLS is unbounded (with positive probability the local optimum is the initial search point of RLS). We treat the special case of unbounded runtimes separately in our analysis of specific objective functions in Section 6. In this section, we provide tools to bound the runtime of a QSH on problems for which the corresponding RSH has finite runtime. Therefore, for the rest of this section, we always assume that the runtime of the considered RSH (on the considered objective function) is finite.

For the RSH, we need exactly one query to move from the search point $\mathbf{x}^{(t)}$ to the search point $\mathbf{x}^{(t+1)}$. Therefore, the optimization time T_{RSH} is the first point in time $t \in \mathbb{N}$ such that $\mathbf{x}^{(t)}$ is optimal. Unfortunately, there is no analogous description for QSHs, as the number of queries needed to move from a search point $\mathbf{x}^{(t)}$ to its successor $\mathbf{x}^{(t+1)}$ is a random variable. In order to overcome this difficulty, we develop the framework of progress times and trajectories, which turns out to be equally suited for RSHs and QSHs.

5.1 Transition Probabilities and Progress Times

We now introduce the notions of transition probabilities, progress probabilities, and progress times for the considered search heuristics. Let $\sigma := (\mathbf{x}^{(t)})_{t\in\mathbb{N}}$ be the random sequence of search points in $\mathcal{S}$ generated by the considered search heuristic. We call σ a *run* of the search heuristic. It follows from the definitions of Algorithm 3.2 and Algorithm 4.1 that the sequence σ forms a Markov chain. Moreover, since $\mathbf{x}^{(0)}$ is chosen uniformly at random from the finite space $\mathcal{S}$, the event "$\mathbf{x}^{(0)} = \mathbf{x}$" has a positive probability for every $\mathbf{x} \in \mathcal{S}$. We may therefore define the transition probabilities for the search heuristic as follows.

Definition 5.2 (Transition Probability $p(\mathbf{x}, \mathbf{y})$)**.** *For two search points* $\mathbf{x}$ *and* $\mathbf{y}$ *in* $\mathcal{S}$*, the* transition probability *from* $\mathbf{x}$ *to* $\mathbf{y}$ *of the considered search heuristic is*

$$p(\mathbf{x}, \mathbf{y}) := \Pr\left[\mathbf{x}^{(1)} = \mathbf{y} \,\middle|\, \mathbf{x}^{(0)} = \mathbf{x}\right].$$

Note that, since σ forms a Markov chain, we actually have

$$p(\mathbf{x}, \mathbf{y}) = \Pr\left[\mathbf{x}^{(t+1)} = \mathbf{y} \,\middle|\, \mathbf{x}^{(t)} = \mathbf{x}\right]$$

for all $t \in \mathbb{N}$ for which the event "$\mathbf{x}^{(t)} = \mathbf{x}$" has positive probability.

In our analysis, the probability $p(\mathbf{x}, \mathbf{x})$ that the considered search heuristic stays at the current search point and does not move to a better solution plays a major role. For example, if $p(\mathbf{x}, \mathbf{x}) = 1$, then the search heuristic will never leave the search point $\mathbf{x}$. Following the terminology of Markov chains, we call such a search point *absorbing* and all other search points *non-absorbing*.

As stated above, we only consider RSHs with finite runtimes in this section. Since each search point appears with positive probability as the initial search point of the RSH, this implies that in our case all absorbing search points must be optimal. However, the inverse is not necessarily true since the considered RSH might move between different optima.

We will be particularly interested in the probability that the RSH does *not* remain in the same search point. We call this probability the progress probability.

Definition 5.3 (Progress Probability $p_{\mathrm{RSH}}(\mathbf{x})$). *For a search point $\mathbf{x} \in \mathcal{S}$, the* progress probability $p_{\mathrm{RSH}}(\mathbf{x})$ *at $\mathbf{x}$ of the considered RSH is given by*

$$p_{\mathrm{RSH}}(\mathbf{x}) := 1 - p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{x}).$$

If $\mathbf{x}$ is the current search point of the RSH and $\mathbf{x}$ is non-absorbing, then the expected time the RSH remains in $\mathbf{x}$ is the reciprocal of the progress probability $p_{\mathrm{RSH}}(\mathbf{x})$. In other words, the progress probability determines the expected number of queries the RSH needs to leave $\mathbf{x}$. (In case $\mathbf{x}$ is absorbing, the future behavior of the RSH is fixed since it will never leave $\mathbf{x}$.)

We might as well define the quantity $1 - p_{\mathrm{QSH}}(\mathbf{x}, \mathbf{x})$ as the progress probability of the QSH. However, we deliberately refrain from doing so for the following reasons. First, $p_{\mathrm{QSH}}(\mathbf{x}, \mathbf{x})$ takes only the two values zero and one, since the QSH always progresses if possible. Second, even if $p_{\mathrm{QSH}}(\mathbf{x}, \mathbf{x})$ equals zero, unlike to the RSH the expected numbers of queries needed for the QSH to progress is usually not one, since the QSH applies in each step the sampling procedure from Theorem 2.2 which takes several steps. Third, we will soon see that it is much more comfortable to link the expected number of queries the QSH needs to leave the search point $\mathbf{x}$ to the progress probability $p_{\mathrm{RSH}}(\mathbf{x})$ of the RSH.

In the light of these facts, we focus on the number of queries needed for both search heuristics, the RSH and the QSH, to progress from the current search point, rather than on a definition of progress probabilities for the QSH. Since we only count in the optimization time queries of the considered search heuristic which happen until an optimum is found, we do not charge the search heuristics for queries if the current search point is already optimal.

Definition 5.4 (Progress Time $R(\mathbf{x})$ with Expectation $r(\mathbf{x})$). *For all $\mathbf{x} \in \mathcal{S}$, the* progress time $R(\mathbf{x})$ at $\mathbf{x}$ *is the random variable defined as follows. If $\mathbf{x}$*

is optimal, then $R(\mathbf{x})$ takes the value zero. Otherwise, $R(\mathbf{x})$ denotes the number of queries needed by the considered search heuristic starting at $\mathbf{x}$ to find a search point different from $\mathbf{x}$. We denote the expected progress time by $r(\mathbf{x})$.

Let $\mathbf{x} \in \mathcal{S}$ be a non-optimal search point. Then it is well-known that the expected progress time $r_{\mathrm{RSH}}(\mathbf{x})$ and the progress probability $p_{\mathrm{RSH}}(\mathbf{x})$ of the considered RSH satisfy

$$r_{\mathrm{RSH}}(\mathbf{x}) = \frac{1}{p_{\mathrm{RSH}}(\mathbf{x})}. \tag{5.1}$$

Recall, that we only consider RSHs with finite runtime, that is, non-optimal implies non-absorbing.

As discussed above, there is no such direct relation between the expected progress time of the considered QSH and the probability for it to leave the current search point. However, Theorem 2.2 allows us to express the expected progress time of the QSH in terms of the expected progress time of the RSH.

Lemma 5.5. *Let c and C be the two positive absolute constants from Theorem 2.2. For every $\mathbf{x} \in \mathcal{S}$, the expected progress time $r_{\mathrm{RSH}}(\mathbf{x})$ of the considered RSH and the expected progress time $r_{\mathrm{QSH}}(\mathbf{x})$ of the associated QSH satisfy*

$$c \cdot r_{\mathrm{RSH}}(\mathbf{x})^{1/2} \leq r_{\mathrm{QSH}}(\mathbf{x}) \leq C \cdot r_{\mathrm{RSH}}(\mathbf{x})^{1/2}. \tag{5.2}$$

Proof. The relations (5.2) follow directly from the definition of Algorithm 4.1 and the first part of Theorem 2.2 if we set $p := p_{\mathrm{RSH}}(\mathbf{x}) = r_{\mathrm{RSH}}(\mathbf{x})^{-1}$. □

Apart from relating the expected progress times of the RSH and the QSH, Theorem 2.2 also allows us to relate the transition probabilities of the two search heuristics.

Lemma 5.6. *Let $\mathbf{x} \in \mathcal{S}$ be non-optimal and let $\mathbf{y} \in \mathcal{S}$ with $\mathbf{x} \neq \mathbf{y}$. Then the transition probability $p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{y})$ of the considered RSH, the progress probability $p_{\mathrm{RSH}}(\mathbf{x})$ of the considered RSH and the transition probability $p_{\mathrm{QSH}}(\mathbf{x}, \mathbf{y})$ of the associated QSH satisfy*

$$p_{\mathrm{QSH}}(\mathbf{x}, \mathbf{y}) = \frac{p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{y})}{p_{\mathrm{RSH}}(\mathbf{x})}. \tag{5.3}$$

Proof. Recall, that we only consider RSHs with finite runtime. That is, since $\mathbf{x}$ is non-optimal, it is also non-absorbing. Thus, we have $p_{\mathrm{RSH}}(\mathbf{x}) > 0$ and the above fraction is well defined.

Equation (5.3) follows directly from Algorithm 4.1 and from Theorem 2.2 if we set $\mathcal{S}_0 := \{\mathbf{y} \in \mathcal{S} \colon \mathbf{y} \neq \mathbf{x}\}$, and let the distribution D_A be given by the probabilities $p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{y})$ for all $\mathbf{y} \in \mathcal{S}$. Then, we have by the second part of Theorem 2.2 that

$$\mathrm{Pr}_{\mathrm{QSH}}\big[\mathbf{x}^{(1)} = \mathbf{y} \,\big|\, \mathbf{x}^{(0)} = \mathbf{x}\big] = \mathrm{Pr}_{D_A}\big[\mathbf{x}^{(1)} = \mathbf{y} \,\big|\, \mathbf{x}^{(0)} = \mathbf{x} \wedge \mathbf{x}^{(1)} \neq \mathbf{x}\big].$$

Thus, by the law of conditional probability, we have

$$p_{\text{QSH}}(\mathbf{x},\mathbf{y}) = \frac{\Pr_{D_A}\left[\mathbf{x}^{(1)} = \mathbf{y} \,\middle|\, \mathbf{x}^{(0)} = \mathbf{x}\right]}{\Pr_{D_A}\left[\mathbf{x}^{(1)} \neq \mathbf{x} \,\middle|\, \mathbf{x}^{(0)} = \mathbf{x}\right]} = \frac{p_{\text{RSH}}(\mathbf{x},\mathbf{y})}{p_{\text{RSH}}(\mathbf{x})}$$

which shows equation (5.3). □

The previous two lemmas, Lemma 5.5 and Lemma 5.6, capture the consequence of Theorem 2.2 to the setting of QSHs and serve as the (only) link between the runtime analysis of QSHs and the results and observations in Section 2. Together, these two lemmas formulate the central observations that allow us to analyze the runtime behavior of the QSH. Lemma 5.5 tells us that using the considered QSH gives us a quadratic speedup over the associated RSH in the expected number of queries necessary to move from a non-optimal search point to the next one. Lemma 5.6 tells us that, conditioned on the event that both search heuristics indeed move to a new search point (which happens with certainty for the QSH), the distributions of these new (random) search points are the same for both search heuristics.

In the next section, we see how these observations extend from a single step of the considered search heuristics to the whole run.

5.2 The Trajectory of a Run

The run of a QSH never reproduces the same search point in consecutive steps except for the last search point. For RSHs, this does not need to be the case. At the t-th sampling step a search heuristic might sample a point which is worse than the current solution in which case it discards it. In this case, both $\mathbf{x}^{(t)}$ and $\mathbf{x}^{(t+1)}$ are the same point in the search space. Thus, in a run σ_{RSH} of the RSH, many of the consecutive sample points may be repetitions. To overcome this difference and to compare the optimization behavior of the considered QSH with that of the associated RSH, we introduce the notion of the trajectory of a run of the RSH. It is obtained from the sequence of search points generated by the RSH if we keep only one element in each consecutive repetition of the same search point, with the potential exception of the last point which then repeats itself forever.

Definition 5.7 (Trajectory τ of a Run σ). *Let $\sigma_{\text{RSH}} := (\mathbf{x}^{(t)})_{t\in\mathbb{N}}$ be a run of the considered RSH. Then the* trajectory *of σ_{RSH}, denoted by τ_{RSH}, is the sub-sequence $(\mathbf{x}^{(t_k)})_{k\in\mathbb{N}}$ of σ_{RSH} such that $t_0 = 0$ and, for all $k \in \mathbb{N}$,*

$$t_{k+1} := \min\left\{t \in \mathbb{N} \colon t > t_k \,\wedge\, \mathbf{x}^{(t)} \neq \mathbf{x}^{(t_k)}\right\}$$

if this minimum exists, and

$$t_{k+1} := t_k + 1$$

otherwise.

According to this definition, if $\tau_{\mathrm{RSH}} = (\mathbf{y}^{(k)})_{k\in\mathbb{N}}$ is the trajectory of a run σ_{RSH} of the RSH, we have either

$$\underbrace{\mathbf{x}^{(0)},\ldots,\mathbf{x}^{(t_1-1)}}_{=\mathbf{y}^{(0)}},\underbrace{\mathbf{x}^{(t_1)},\ldots,\mathbf{x}^{(t_2-1)}}_{=\mathbf{y}^{(1)}},\underbrace{\mathbf{x}^{(t_2)},\ldots,\mathbf{x}^{(t_3-1)}}_{=\mathbf{y}^{(2)}},\ldots$$

or

$$\underbrace{\mathbf{x}^{(0)},\ldots,\mathbf{x}^{(t_1-1)}}_{=\mathbf{y}^{(0)}},\underbrace{\mathbf{x}^{(t_1)},\ldots,\mathbf{x}^{(t_2-1)}}_{=\mathbf{y}^{(1)}},\ldots,\underbrace{\mathbf{x}^{(t_\ell)}}_{=\mathbf{y}^{(\ell)}},\underbrace{\mathbf{x}^{(t_\ell)}}_{=\mathbf{y}^{(\ell+1)}},\underbrace{\mathbf{x}^{(t_\ell)}}_{=\mathbf{y}^{(\ell+2)}},\ldots$$

in the special case that $\mathbf{x}^{(t_k)} = \mathbf{x}^{(t_{k+1})}$ for all $k \geq \ell$. Note that a run σ_{RSH} is a random sequence of search points and so is its trajectory τ_{RSH}.

As mentioned above, a run σ_{QSH} of a QSH never reproduces the same search point in consecutive steps except for the last search point. Therefore, we refrain from defining the trajectory of σ_{QSH} as we did for RSHs in Definition 5.7, since this trajectory would be equal to σ_{QSH}.

Now, the crucial observation is that for a fixed sequence σ, the probability that a run of the associated QSH coincides with σ is exactly the same as the probability that a run of the considered RSH has trajectory σ. In other words, the runs of the QSH and the trajectories of the runs of the RSH share the same distribution[3]. This is a direct consequence Lemma 5.6.

Lemma 5.8. *Let $\sigma_{\mathrm{QSH}} := (\mathbf{x}^{(k)}_{\mathrm{QSH}})_{k\in\mathbb{N}}$ be a run of the considered QSH and let $\tau_{\mathrm{RSH}} := (\mathbf{x}^{(t_k)}_{\mathrm{RSH}})_{k\in\mathbb{N}}$ be the trajectory of a run $\sigma_{\mathrm{RSH}} := (\mathbf{x}^{(t)}_{\mathrm{RSH}})_{t\in\mathbb{N}}$ of the associated RSH. Then,*

$$\Pr\left[\forall k \in \{0,\ldots,\ell\}\colon \mathbf{x}^{(k)}_{\mathrm{QSH}} = \mathbf{y}^{(k)}\right] = \Pr\left[\forall k \in \{1,\ldots,\ell\}\colon \mathbf{x}^{(t_k)}_{\mathrm{RSH}} = \mathbf{y}^{(k)}\right]$$

holds for every $\ell \in \mathbb{N}\cup\{\infty\}$ and for every infinite sequence $\sigma := (\mathbf{y}^{(k)})_{t\in\mathbb{N}}$ of search points in $\mathcal{S}$.

Proof. On the one hand, for $P_{\mathrm{QSH}} := \Pr\left[\forall k \in \{0,\ldots,\ell\}\colon \mathbf{x}^{(k)}_{\mathrm{QSH}} = \mathbf{y}^{(k)}\right]$, we have

$$P_{\mathrm{QSH}} = \Pr\left[\mathbf{x}^{(0)}_{\mathrm{QSH}} = \mathbf{y}^{(0)}\right] \cdot \prod_{k=1}^{\ell} \Pr\left[\mathbf{x}^{(k)}_{\mathrm{QSH}} = \mathbf{y}^{(k)} \,\middle|\, \mathbf{x}^{(k-1)}_{\mathrm{QSH}} = \mathbf{y}^{(k-1)}\right]$$

and on the other hand, for $P_{\mathrm{RSH}} := \Pr\left[\forall k \in \{0,\ldots,\ell\}\colon \mathbf{x}^{(t_k)}_{\mathrm{RSH}} = \mathbf{y}^{(k)}\right]$, we have

$$P_{\mathrm{RSH}} = \Pr\left[\mathbf{x}^{(t_0)}_{\mathrm{RSH}} = \mathbf{y}^{(0)}\right] \cdot \prod_{k=1}^{\ell} \Pr\left[\mathbf{x}^{(t_k)}_{\mathrm{RSH}} = \mathbf{y}^{(k)} \,\middle|\, \mathbf{x}^{(t_{k-1})}_{\mathrm{RSH}} = \mathbf{y}^{(k-1)}\right].$$

[3]Note that the set of infinite sequences of search points in not countable. Thus, technically, we consider the probability space over the sigma-algebra generated by all sets of sequences starting with the same first elements. Lemma 5.8 reflects this notion. For reasons of simplicity, we assume this argument implicitly for the remainder of this section.

Thus, in order to show Lemma 5.8, it suffices to show that

$$\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(t_0)} = \mathbf{x}\right] = \Pr\left[\mathbf{x}_{\mathrm{QSH}}^{(0)} = \mathbf{x}\right] \tag{5.4}$$

holds for all $k \in \mathbb{N}$ and $\mathbf{x} \in \mathcal{S}$ and that

$$\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(t_{k+1})} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(t_k)} = \mathbf{x}\right] = \Pr\left[\mathbf{x}_{\mathrm{QSH}}^{(k+1)} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{QSH}}^{(k)} = \mathbf{x}\right] \tag{5.5}$$

holds for all $\mathbf{x}, \mathbf{y} \in \mathcal{S}$ with $\mathbf{x} \neq \mathbf{y}$. In this, we assume all conditional probabilities are defined and non-zero, since otherwise the result holds trivially with both probabilities equal to zero.

Equation (5.4) holds since $t_0 = 0$, and both the considered RSH and the associated QSH generate the initial search point uniformly at random.

To show equation (5.5), we recall that by the definition of the t_k's, we have that $\mathbf{x}_{\mathrm{RSH}}^{(t_{k+1}-1)} = \mathbf{x}_{\mathrm{RSH}}^{(t_k)}$ and that $\mathbf{x}_{\mathrm{RSH}}^{(t_{k+1})} \neq \mathbf{x}_{\mathrm{RSH}}^{(t_k)}$. Since σ_{RSH} forms a Markov chain, we have for all $\mathbf{x} \neq \mathbf{y}$ that

$$\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(t_{k+1})} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(t_k)} = \mathbf{x}\right] = \Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(1)} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(1)} \neq \mathbf{x} \wedge \mathbf{x}_{\mathrm{RSH}}^{(0)} = \mathbf{x}\right].$$

Applying the laws of conditional probability, we get

$$\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(1)} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(1)} \neq \mathbf{x} \wedge \mathbf{x}_{\mathrm{RSH}}^{(0)} = \mathbf{x}\right] = \frac{\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(1)} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(0)} = \mathbf{x}\right]}{\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(1)} \neq \mathbf{x} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(0)} = \mathbf{x}\right]}.$$

Therefore,

$$\Pr\left[\mathbf{x}_{\mathrm{RSH}}^{(t_{k+1})} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{RSH}}^{(t_k)} = \mathbf{x}\right] = \frac{p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{y})}{p_{\mathrm{RSH}}(\mathbf{x})}.$$

Finally, since σ_{QSH} also forms a Markov chain, we have

$$\Pr\left[\mathbf{x}_{\mathrm{QSH}}^{(k+1)} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{QSH}}^{(k)} = \mathbf{x}\right] = \Pr\left[\mathbf{x}_{\mathrm{QSH}}^{(1)} = \mathbf{y} \,\middle|\, \mathbf{x}_{\mathrm{QSH}}^{(0)} = \mathbf{x}\right] = p_{\mathrm{QSH}}(\mathbf{x}, \mathbf{y}).$$

Then the equation (5.5) follows from Lemma 5.6. $\square$

A central notion in the main lemma of this section (Lemma 5.10) is the notion of *frequency* of a search point. Lemma 5.8 assures that the following definition of the *expected frequency* is well-defined.

Definition 5.9 (Frequency $M(\mathbf{x})$ with Expectation $m(\mathbf{x})$)**.** *Let the* frequency $M_{\mathrm{QSH}}(\mathbf{x})$ *and* $M_{\mathrm{RSH}}(\mathbf{x})$ *of a non-optimal search point* $\mathbf{x}$ *be the random variable that denotes the number of occurrences of* $\mathbf{x}$ *in the run* σ_{QSH} *of the considered QSH and in the trajectory* τ_{RSH} *of the run of associated RSH, respectively. If* $\mathbf{x}$ *is optimal, we set* $M_{\mathrm{RSH}}(\mathbf{x})$ *and* $M_{\mathrm{QSH}}(\mathbf{x})$ *to be the random variable that is* 0 *with probability* 1*. Then, for every* $\mathbf{x} \in \mathcal{S}$*, we call the value*

$$m(\mathbf{x}) := \mathrm{E}\left[M_{\mathrm{RSH}}(\mathbf{x})\right] = \mathrm{E}\left[M_{\mathrm{QSH}}(\mathbf{x})\right]$$

the expected frequency *of* $\mathbf{x}$.

Note that for the conservative selection rule, the fitness is strictly monotonically increasing along the trajectory. Therefore, the random variable $M(\mathbf{x})$ takes only values in $\{0,1\}$, and its expectation $m(\mathbf{x})$ (for non-optimal $\mathbf{x}$) is just the probability that $\mathbf{x}$ occurs in the trajectory. We now give the main lemma of this section which connects the runtime of the considered search heuristic to its expected frequencies and expected progress times.

Lemma 5.10. *Let $r(\mathbf{x})$ be the expected progress time of the considered search heuristic and let its optimization time T be finite in expectation. For a search point $\mathbf{x} \in \mathcal{S}$, let $m(\mathbf{x})$ be the expected frequency of $\mathbf{x}$ as defined in Definition 5.9. Then, we have*

$$\mathrm{E}[T] = \sum_{\mathbf{x}\in\mathcal{S}} m(\mathbf{x}) \cdot r(\mathbf{x}).$$

Note, that in this lemma the quantity $r(\mathbf{x})$ depends on whether we consider the classical or the quantum search heuristic but the quantity $m(\mathbf{x})$ does not.

Proof of Lemma 5.10. Let $\sigma = (\mathbf{x}^{(t)})_{t\in\mathbb{N}}$ be a run of the considered search heuristic and let $(\mathbf{y}^{(k)})_{k\in\mathbb{N}} := (\mathbf{x}^{(t_k)})_{k\in\mathbb{N}}$ be its trajectory in the case where we consider the RSH and $(\mathbf{y}^{(k)})_{k\in\mathbb{N}} := \sigma$ in the case where we consider the QSH.

For a non-optimal search point $\mathbf{x} \in \mathcal{S}$ and $k \in \mathbb{N}$, let $M_k(\mathbf{x})$ be the random indicator variable that takes the value 1 if $\mathbf{y}^{(k)} = \mathbf{x}$, and the value 0 otherwise. Moreover, let $R_k := R(\mathbf{y}^{(k)})$ be the random variable that denotes the number of queries the considered search heuristic needs to move from $\mathbf{y}^{(k)}$ to $\mathbf{y}^{(k+1)}$. We set $R_k(\mathbf{x}) := 0$ and $M_k(\mathbf{x}) := 0$ if $\mathbf{x}$ is optimal.

Then we have

$$T = \sum_{k\in\mathbb{N}} R_k$$

and

$$M(\mathbf{x}) = \sum_{k\in\mathbb{N}} M_k(\mathbf{x}).$$

Let $k \in \mathbb{N}$. We want to determine $\mathrm{E}[R_k]$. By the law of total expectation, we have that

$$\mathrm{E}[R_k] = \sum_{\mathbf{x}\in\mathcal{S}} r(\mathbf{x}) \Pr[\mathbf{y}^{(k)} = \mathbf{x}],$$

since, for both the RSH and the QSH, $r(\mathbf{x})$ is the expected number of queries needed to leave the search point $\mathbf{x}$. Next, we have

$$\Pr[\mathbf{y}^{(k)} = \mathbf{x}] = \mathrm{E}[M_k(\mathbf{x})],$$

and therefore

$$\mathrm{E}[R_k] = \sum_{\mathbf{x}\in\mathcal{S}} r(\mathbf{x}) \, \mathrm{E}[M_k(\mathbf{x})].$$

Finally, we can determine $\mathrm{E}[T]$. By the linearity of expectation, we get

$$\mathrm{E}[T] = \sum_{k\in\mathbb{N}} \mathrm{E}[R_k] = \sum_{k\in\mathbb{N}} \sum_{\mathbf{x}\in\mathcal{S}} r(\mathbf{x})\,\mathrm{E}[M_k(\mathbf{x})] = \sum_{\mathbf{x}\in\mathcal{S}} r(\mathbf{x})\,\mathrm{E}[M(\mathbf{x})],$$

which concludes the proof of the statement. Note that in the last step we could reorder the sum since we assumed that T has finite expectation. □

5.3 Approximate Runtimes

In this section, we introduce the tools we will apply later to approximate the runtime of the considered QSH by studying the optimization behavior of the associated RSH. We start with our central theorem, which allows us to bound the runtime of the considered QSH in terms of the progress times and transition probabilities of the associated RSH.

Theorem 5.11. *Let c and C be the two positive absolute constants from Theorem 2.2. For all search points $\mathbf{x} \in \mathcal{S}$, let $r_{\mathrm{RSH}}(\mathbf{x})$ be the expected progress time of the considered RSH (see Definition 5.4), and let $m(\mathbf{x})$ be the expected frequency of $\mathbf{x}$ in the trajectory of its run (see Definition 5.9).*

The optimization time of the considered RSH is finite if and only if the optimization time T_{QSH} of the associated QSH is finite. In this case, T_{QSH} satisfies

$$c\sum_{\mathbf{x}\in\mathcal{S}} m(\mathbf{x})\cdot\big(r_{\mathrm{RSH}}(\mathbf{x})\big)^{1/2} \le \mathrm{E}[T_{\mathrm{QSH}}] \le C\sum_{\mathbf{x}\in\mathcal{S}} m(\mathbf{x})\cdot\big(r_{\mathrm{RSH}}(\mathbf{x})\big)^{1/2}.$$

Proof. The RSH and QSH are equally likely to take any fixed trajectory through the search space, only with different speed. Recall from Lemma 5.10 that

$$\mathrm{E}[T] = \sum_{\mathbf{x}\in\mathcal{S}} m(\mathbf{x})\cdot r(\mathbf{x}) \tag{5.6}$$

and that the quantity $r(\mathbf{x})$ depends on whether we consider the classical or the quantum search heuristic but the quantity $m(\mathbf{x})$ does not.

Since the search space is finite, the expected progress times $r_{\mathrm{RSH}}(\mathbf{x})$ and $r_{\mathrm{QSH}}(\mathbf{x})$ differ at most by a constant factor depending on the search space $\mathcal{S}$, the mutation operator MUT, and the objective function f. Thus if Equation (5.6) yields a finite value for the RSH, then it also does so for the QSH, and vice versa.

So the RSH has finite expected optimization time if and only if the QSH has. Hence, Theorem 5.11 is a direct consequence of Lemma 5.10 and Lemma 5.5. □

Next, we prove a lemma which is tailored for analyzing the problems in Section 6. It is useful if we can partition the search space into regions where the progress probability of the RSH behaves similarly. This turns out to be very convenient for the analysis of a QSH when the associated RSH is already understood. A good example for the situation is the fitness level based analysis of the function OneMax in Section 6.1. (However, in general the regions do not need to correspond to fitness levels.) In this case, the runtimes of QSH and RSH are strongly related.

Lemma 5.12. *Let c and C be the two positive absolute constants from Theorem 2.2. Let $\ell \in \mathbb{N}$ and let $\mathcal{S}$ be partitioned into ℓ parts $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$.*

Suppose, for every $j \in \{1, \ldots, \ell\}$, there exist two real values p_j and P_j with $0 < p_j \leq P_j \leq 1$ such that for all non-optimal search points $\mathbf{x} \in \mathcal{S}_j$, the progress probability $p_{\text{RSH}}(\mathbf{x})$ of the considered RSH satisfies the inequalities

$$p_j \leq p_{\text{RSH}}(\mathbf{x}) \leq P_j.$$

For $j \in \{1, \ldots, \ell\}$, let T_j denote the number of queries spend by the RSH on leaving search points in $\mathcal{S}_j$. Then the optimization time T_{QSH} of the associated QSH satisfies

$$c \sum_{j=1}^{\ell} p_j^{1/2} \, \mathrm{E}[T_j] \leq \mathrm{E}[T_{\text{QSH}}] \leq C \sum_{j=1}^{\ell} P_j^{1/2} \, \mathrm{E}[T_j].$$

Before we proof this lemma, we restate it for the case that we only consider one region. This proves useful for bounding the runtime of the considered QSH in case that there is an easy way to bound the progress probability as well as the runtime of the associated classical variant.

Corollary 5.13. *Let c and C be the two positive absolute constants from Theorem 2.2. Suppose there exist two values $p_{\min}$ and $p_{\max}$ in $\mathbb{R}$ with $0 < p_{\min} \leq p_{\max} \leq 1$ such that, for all non-optimal $\mathbf{x} \in \mathcal{S}$, the progress probability $p_{\text{RSH}}(\mathbf{x})$ of the considered RSH satisfies the inequality*

$$p_{\min} \leq p_{\text{RSH}}(\mathbf{x}) \leq p_{\max}.$$

Then the optimization time T_{RSH} of the considered RSH and the optimization time T_{QSH} of the associated QSH satisfy

$$c p_{\min}^{1/2} \, \mathrm{E}[T_{\text{RSH}}] \leq \mathrm{E}[T_{\text{QSH}}] \leq C p_{\max}^{1/2} \, \mathrm{E}[T_{\text{RSH}}].$$

Note that Corollary 5.13 implies that $\mathrm{E}[T_{\text{QSH}}] \in O(\mathrm{E}[T_{\text{RSH}}])$ since we may always choose $p_{\max} = 1$. We now turn to the proof of Lemma 5.12.

Proof of Lemma 5.12. For the beginning, let $j \in \{1, \ldots, \ell\}$ and $\mathbf{x} \in \mathcal{S}$ be fixed. Starting with Lemma 5.5, we have

$$c\big(r_{\text{RSH}}(\mathbf{x})\big)^{1/2} \leq r_{\text{QSH}}(\mathbf{x}) \leq C\big(r_{\text{RSH}}(\mathbf{x})\big)^{1/2}$$

Thus, by equation (5.1), we get

$$c\big(p_{\mathrm{RSH}}(\mathbf{x})\big)^{-1/2} \leq r_{\mathrm{QSH}}(\mathbf{x}) \leq C\big(p_{\mathrm{RSH}}(\mathbf{x})\big)^{-1/2}.$$

We multiply the three parts of the two inequalities by $m(\mathbf{x})$. This gives us

$$c\,m(\mathbf{x})\big(p_{\mathrm{RSH}}(\mathbf{x})\big)^{-1/2} \leq m(\mathbf{x})r_{\mathrm{QSH}}(\mathbf{x}) \leq Cm(\mathbf{x})\big(p_{\mathrm{RSH}}(\mathbf{x})\big)^{-1/2}.$$

Since we assumed that $p_j \leq p_{\mathrm{RSH}}(\mathbf{x}) \leq P_j$ for all non-optimal points $\mathbf{x} \in \mathcal{S}_j$ (and since $m(\mathbf{x}) = 0$ for all optimal $\mathbf{x}$), this implies

$$cp_j^{1/2}m(\mathbf{x})\big(p_{\mathrm{RSH}}(\mathbf{x})\big)^{-1} \leq m(\mathbf{x})r_{\mathrm{QSH}}(\mathbf{x}) \leq CP_j^{1/2}m(\mathbf{x})\big(p_{\mathrm{RSH}}(\mathbf{x})\big)^{-1}.$$

We substitute equation (5.1) again and obtain

$$cp_j^{1/2}m(\mathbf{x})r_{\mathrm{RSH}}(\mathbf{x}) \leq m(\mathbf{x})r_{\mathrm{QSH}}(\mathbf{x}) \leq CP_j^{1/2}m(\mathbf{x})r_{\mathrm{RSH}}(\mathbf{x}).$$

Finally, we sum over all $\mathbf{x} \in \mathcal{S}_j$ and all $j \in \{1, \ldots, \ell\}$ which results in

$$c\sum_{j=1}^{\ell} p_j^{1/2} \sum_{\mathbf{x}\in\mathcal{S}_j} m(\mathbf{x})r_{\mathrm{RSH}}(\mathbf{x}) \leq \sum_{\mathbf{x}\in\mathcal{S}} m(\mathbf{x})r_{\mathrm{QSH}}(\mathbf{x}) \leq C\sum_{j=1}^{\ell} P_j^{1/2} \sum_{\mathbf{x}\in\mathcal{S}_j} m(\mathbf{x})r_{\mathrm{RSH}}(\mathbf{x}).$$

Then Lemma 5.12 follows directly from Lemma 5.10. □

5.4 The Adapted Markov Chain

Recall that the considered RSH can be seen as a Markov chain that performs one query each step and has transition probabilities $p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{y})$ as defined before. The situation for the associated QSH is slightly more complicated. At each sampling step, the QSH has probability 1 of sampling a new solution (unless it has found an absorbing optimum) because the probability has been quantum mechanically amplified. However, if the current solution is $\mathbf{x}$ then to make the one sampling step costs the algorithm in expectation a total of $r_{\mathrm{QSH}}(\mathbf{x}) \in \Theta\big(p_{\mathrm{RSH}}(\mathbf{x})^{-1/2}\big)$ queries where $p_{\mathrm{RSH}}(\mathbf{x})$ is the progress probability of the RSH. Nevertheless, we can still model the QSH as a Markov chain which charges in expectation one query per step by scaling the transition probabilities of the corresponding RSH appropriately. It turns out that, with respect to the runtime, the QSH behaves asymptotically as if it was a RSH with these adapted transition probabilities.

Theorem 5.14. *Let T_{QSH} be the optimization time of the considered QSH. For the corresponding RSH, let $p_{\mathrm{RSH}}(\mathbf{x}, \mathbf{y})$ be the transition probability between the points $\mathbf{x}$ and $\mathbf{y}$ and let $p_{\mathrm{RSH}}(\mathbf{x})$ be the progress probability at the point $\mathbf{x}$. Then*

$$\mathrm{E}[T_{\mathrm{QSH}}] \in \Theta\big(\,\mathrm{E}[T_{\widetilde{\mathrm{RSH}}}]\big),$$

where $T_{\widetilde{\mathrm{RSH}}}$ is the optimization time of the (adapted) RSH corresponding to the adapted transition probabilities

$$p_{\widetilde{\mathrm{RSH}}}(\mathbf{x},\mathbf{y}) = \begin{cases} 1 - p_{\mathrm{RSH}}(\mathbf{x})^{1/2} & \text{if } \mathbf{x} = \mathbf{y}, \\ p_{\mathrm{RSH}}(\mathbf{x},\mathbf{y})/p_{\mathrm{RSH}}(\mathbf{x})^{1/2} & \text{if } \mathbf{x} \neq \mathbf{y} \text{ and } p_{\mathrm{RSH}}(\mathbf{x}) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. First note that $p_{\widetilde{\mathrm{RSH}}}(\mathbf{x},\mathbf{y})$ is well-defined, since, for $p_{\mathrm{RSH}}(\mathbf{x}) > 0$, we have

$$\sum_{\mathbf{y}\in\mathcal{S}} p_{\widetilde{\mathrm{RSH}}}(\mathbf{x},\mathbf{y}) = 1 - p_{\mathrm{RSH}}(\mathbf{x})^{1/2} + \frac{\sum_{\mathbf{y}\in\mathcal{S}\setminus\{\mathbf{x}\}} p_{\mathrm{RSH}}(\mathbf{x},\mathbf{y})}{p_{\mathrm{RSH}}(\mathbf{x})^{1/2}} = 1.$$

Next, let $p_{\widetilde{\mathrm{RSH}}}(\mathbf{x}) = 1 - p_{\widetilde{\mathrm{RSH}}}(\mathbf{x},\mathbf{x})$ be the progress probability at $\mathbf{x}$ of the adapted RSH. Then we have, for every non-absorbing point, that

$$r_{\mathrm{RSH}}(\mathbf{x})^{1/2} = p_{\mathrm{RSH}}(\mathbf{x})^{-1/2} = p_{\widetilde{\mathrm{RSH}}}(\mathbf{x})^{-1} = r_{\widetilde{\mathrm{RSH}}}(\mathbf{x}).$$

For every non-absorbing points $\mathbf{x}$ and every other point $\mathbf{y}$ we have

$$\frac{p_{\widetilde{\mathrm{RSH}}}(\mathbf{x},\mathbf{y})}{p_{\widetilde{\mathrm{RSH}}}(\mathbf{x})} = \frac{p_{\mathrm{RSH}}(\mathbf{x},\mathbf{y})}{p_{\mathrm{RSH}}(\mathbf{x})},$$

that is, the probabilities to move from $\mathbf{x}$ to $\mathbf{y}$ conditioned on the event that the process moves at all coincide for the adapted RSH and the RSH corresponding to the considered QSH. Thus, these probabilities also coincide for the adapted RSH and the QSH itself. Therefore, the run of the considered QSH, the trajectory of the corresponding RSH, and the trajectory of the adapted RSH all have the same distribution and

$$m_{\mathrm{QSH}}(\mathbf{x}) = m_{\mathrm{RSH}}(\mathbf{x}) = m_{\widetilde{\mathrm{RSH}}}(\mathbf{x}).$$

Together, this implies that

$$\sum_{\mathbf{x}\in\mathcal{S}} m_{\mathrm{QSH}}(\mathbf{x}) \cdot r_{\mathrm{RSH}}(\mathbf{x})^{1/2} = \sum_{\mathbf{x}\in\mathcal{S}} m_{\widetilde{\mathrm{RSH}}}(\mathbf{x}) \cdot r_{\widetilde{\mathrm{RSH}}}(\mathbf{x})$$

and $\mathrm{E}[T_{\mathrm{QSH}}] \in \Theta(\mathrm{E}[T_{\widetilde{\mathrm{RSH}}}])$ follows from Lemma 5.10 and Theorem 5.11. $\square$

6 Runtime Analysis of Basic Fitness Functions

In this section we present asymptotically tight runtime bounds for the progressive and conservative variants of QLS and the (1+1) QEA on the pseudo-Boolean optimization problems OneMax, LeadingOnes, Discrepancy, Needle, and Jump$_m$ and compare them to their classical counterparts. The results of this section are summarized in Table 1 in the introduction.

Throughout this section, let $n \in \mathbb{N}$ be the length of the bit-strings that are the search points. Since we are only interested in asymptotic results we may assume that n is sufficiently large. In particular, in order to make the following proofs more readable, we suppress rounding signs. For example, without further notice we assume that $n/2$ is an integer.

6.1 OneMax

The pseudo-Boolean function OneMax returns the Hamming weight $|\cdot|_1$ of a bit-string $\mathbf{x} \in \{0,1\}^n$, that is, it counts the number of one-bits in $\mathbf{x}$,

$$\textsc{OneMax}(\mathbf{x}) := |\mathbf{x}|_1 = \sum_{i=1}^{n} x_i \,. \tag{6.1}$$

We start with the well-known result on the runtimes of the considered classical search heuristics on the objective function OneMax (compare [11]).

Theorem 6.1. *The runtimes of the (1+1) EA, RLS, the (1+1) EA*, and RLS* for minimizing* OneMax *are in* $\Theta(n \log n)$.

We show that the expected query time in the quantum version decreases only by a logarithmic factor.

Theorem 6.2. *The runtimes of the (1+1) QEA, QLS, the (1+1) QEA*, and QLS* for minimizing* OneMax *are in* $\Theta(n)$.

In this subsection, our particular focus is to demonstrate how the proof of Theorem 6.2 can be derived from the ingredients of the proof of Theorem 6.1. To this end, we retrace the steps necessary to prove Theorem 6.1. The core of this proof is a bound on the progress probabilities of the considered RSHs.

Proposition 6.3. *Let* $k \in \{1, \ldots, n\}$ *and let* $\mathbf{x} \in \{0,1\}^n$ *be a search point of Hamming weight* k. *Then for each, the (1+1) EA, RLS, the (1+1) EA*, and RLS*, the progress probability is in* $\Theta(k/n)$ *and consequently the expected numbers of queries needed to find a search point of Hamming weight at most* $k-1$ *when starting in* $\mathbf{x}$ *is in* $\Theta(n/k)$.

We omit the proof to this proposition. For RLS and RLS*, the proof is straight-forward. For the (1+1) EA and the (1+1) EA*, the proof is well-known and can be easily deduced from the results and proofs in [11].

The previous proposition already allows us to derive an upper bound on the runtimes of the four classical search heuristics. The search space may be subdivided into regions of equal fitness, which we call *fitness levels*. In the worst case, a run visits search points for all fitness levels, giving us the upper runtime bound of $O(n \log n)$ in Theorem 6.1.

In order to use the same approach to show the lower bound in Theorem 6.1, we have to be slightly more careful, since the search heuristic may skip some fitness values. However, the following statement ascertains that with sufficiently large probability, still linearly many fitness levels are visited.

Proposition 6.4. *Consider a run of the (1+1) EA, RLS, the (1+1) EA*, or RLS* that finds the optimum. With probability at least* $1/6$*, this run visits at least* $n/24$ *many non-optimal search points with distinct Hamming-weights.*

Proof. First, we show that with probability at least $1/3$, the Hamming weight of the initial search point $\mathbf{x}^{(0)}$ (which is uniformly distributed for all four search heuristics) is at least $n/4$. This is a direct consequence of the Markov Inequality (see, e.g., Chapter 1 in [4]). The random variable $X := n - |\mathbf{x}^{(0)}|_1$ has expectation $\mathrm{E}[X] = n/2$ (each bit is a one-bit with probability $1/2$). Thus,

$$\Pr\left[|\mathbf{x}^{(0)}|_1 < \frac{n}{4}\right] = \Pr\left[X > \frac{3}{2}\,\mathrm{E}[X]\right] \leq \frac{2}{3}.$$

Next, suppose that the initial search point has indeed Hamming weight at least $n/4$. For RLS and RLS*, this directly implies Proposition 6.4, since both search heuristics then necessarily need to visit search points of Hamming weights $1, \ldots, n/4$ in order to reach the optimum. For the (1+1) EA and the (1+1) EA*, this does not need to be true. However, we may bound the expected number of one-bits flipped each time the algorithms improve their current search point. Because of the symmetry of the OneMax function, the same analysis applies to both search heuristics.

Let $\mathbf{x} \in \{0,1\}^n$ be the current search point of Hamming weight $k \in \{1, \ldots, n\}$ of the (1+1) EA or the (1+1) EA* and let $\mathbf{y}$ be the next random search point selected by the respective search heuristic. We are interested in the progress $\Delta := |\mathbf{x}|_1 - |\mathbf{y}|_1$, in particular we want to give a constant upper bound on the expectation of Δ conditioned on the event that $\Delta \geq 1$. By the law of total expectation, we have that

$$\mathrm{E}\left[\Delta\right] = \mathrm{E}\left[\Delta \mid \Delta \geq 1\right] \Pr\left[\Delta \geq 1\right]$$

since $\Delta \geq 0$ (worse search points are never selected). We already know from Proposition 6.3 that

$$\Pr\left[\Delta \geq 1\right] \geq \frac{k}{en},$$

both for the (1+1) EA and the (1+1) EA*. Moreover, we get an upper bound of k/n on $\mathrm{E}[\Delta]$ if we condition on the event that none of the zero-bits in $\mathbf{x}$ flips. Thus,

$$\mathrm{E}\left[\Delta \mid \Delta \geq 1\right] \leq \mathrm{e} \leq 3$$

We conclude the proof of Proposition 6.4 by applying the same argument using the Markov inequality as above. We have just seen that, in expectation, the progress made by the search heuristic in the first $n/24$ improvements is at most $n/8$. Thus, the probability that it exceed $n/4$, the Hamming weight of the first search point, is at most $1/2$. In other words, with probability at least $1/6 = (1/3) \cdot (1/2)$, the run visits at least $n/24$ search points of distinct Hamming weight before reaching the optimum. $\square$

Proposition 6.4, together with Proposition 6.3, now allows us to prove Theorem 6.2 based on the following idea. Suppose the run of the considered search heuristic indeed visits at least $n/24$ fitness levels before it finds the optimum. Then we get a lower bound on the runtime if we sum over the lower bounds on the times to leave these fitness levels given in Proposition 6.4. In a worst case scenario, the fitness levels visited before finding the optimum are the levels with values $(23/24)n, \ldots, n$. However, even then the runtime is at least $\Omega(n)$ as stated in Theorem 6.2.

Proof of Theorem 6.2. The following proof holds for all four QSHs considered in Theorem 6.2. Thus, we consider one of these QSHs and its corresponding RSH.

Let T_{QSH} be the optimization time of the considered QSH. For all $k \in \{1, \ldots, n\}$, let $\mathcal{S}_k$ be the set of all search points in $\{0,1\}^n$ of Hamming weight k and let T_k be is the number of queries the RSH spends on leaving the search points in $\mathcal{S}_k$.

We start with the upper bound on $\mathrm{E}[T_{\text{QSH}}]$. By Lemma 5.12 and Proposition 6.3 we have that

$$\mathrm{E}[T_{\text{QSH}}] \in O\left(\sum_{k=1}^{n} \left(\frac{k}{n}\right)^{1/2} \mathrm{E}[T_k]\right)$$

Moreover, Proposition 6.3 gives us that $\mathrm{E}[T_k] \in O(n/k)$, where we pessimistically assume that the RSH visits all fitness levels. Together, this yields

$$\mathrm{E}[T_{\text{QSH}}] \in O\left(n^{1/2} \sum_{k=1}^{n} \frac{1}{k^{1/2}}\right).$$

Thus, since

$$\sum_{k=1}^{n} k^{-1/2} \leq 1 + \int_{1}^{n} x^{-1/2} dx = 2n^{1/2} - 1$$

we get $\mathrm{E}[T_{\text{QSH}}] \in O(n)$.

We now turn to the lower bound on $\mathrm{E}[T_{\text{QSH}}]$. Here, we have to be more careful since the typical run does not visit all fitness levels. Let $I \subseteq \{1, \ldots, n\}$ be the random set of Hamming weights of non-optimal search points visited by the RSH. Then we apply Proposition 6.4 and condition on the event that the run of the RSH visits at least $n/24$ fitness levels, that is,

$$\mathrm{E}[T_{\text{QSH}}] \geq \frac{\mathrm{E}\left[T_{\text{QSH}} \mid |I| \geq n/24\right]}{6}$$

We again invoke Lemma 5.12 and Proposition 6.3 and get

$$\mathrm{E}[T_{\mathrm{QSH}}] \in \Omega\left(\sum_{k=1}^{n}\left(\frac{k}{n}\right)^{1/2} \mathrm{E}\left[T_k \,\middle|\, |I| \geq n/24\right]\right).$$

Then, Proposition 6.3 gives us that

$$\mathrm{E}[T_{\mathrm{QSH}}] \in \Omega\left(n^{1/2}\,\mathrm{E}\left[\sum_{k\in I}\frac{1}{k^{1/2}} \,\middle|\, |I| \geq n/24\right]\right).$$

Note that the random sum $\sum_{k\in I} k^{-1/2}$ strongly depends on the random choice of I. However, for all $|I| \geq n/24$, this sum is bounded from below by

$$\sum_{k=(23/24)n}^{n}\frac{1}{k^{1/2}} \geq \int_{(23/24)n}^{n} x^{-1/2}dx = 2\big(1-(23/24)^{1/2}\big)n^{1/2} \in \Omega\big(n^{1/2}\big).$$

Therefore, we have $\mathrm{E}[T_{\mathrm{QSH}}] \in \Omega(n)$ which concludes the proof of Theorem 6.1. □

6.2 LeadingOnes

The pseudo-Boolean function LeadingOnes counts the number of one-bits preceding the first zero-bit in a bit-string $\mathbf{x} \in \{0,1\}^n$, that is, let

$$\textsc{LeadingOnes}(\mathbf{x}) := \sum_{k=1}^{n}\prod_{i=1}^{k} x_i\,. \tag{6.2}$$

The following theorem can be deduced from [11].

Theorem 6.5. *The runtimes of the (1+1) EA, RLS, the (1+1) EA*, and RLS* maximizing* LeadingOnes *are in* $\Theta(n^2)$.

For the progressive selection rule, quantum acceleration does not yield a substantial speedup. In contrast, for the conservative selection rule the runtime decreases considerably.

Theorem 6.6. *The runtimes of the (1+1) QEA* and of QLS* maximizing* LeadingOnes *are in* $\Theta(n^{3/2})$. *The runtimes of the (1+1) QEA or QLS maximizing* LeadingOnes *are in* $\Theta(n^2)$.

Proof. We start with the conservative selection strategy. Let us first consider QLS*. For any non-optimal search point $\mathbf{x}$, the mutation step yields a better search point if and only if the first zero-bit is flipped. So the progress probability of RLS* is $p_{\mathrm{RSH}}(\mathbf{x}) = 1/n$ for all non-optimal search points. For the (1+1) QEA* let $\mathbf{x}$ be any non-optimal search point. Assume that x_i is the first zero-bit in $\mathbf{x}$. Then the mutation step yields a better search point if and only if x_i is flipped, and all preceding bits are unchanged. Therefore, the progress probability is

$$p_{\mathrm{RSH}}(\mathbf{x}) = \frac{1}{n}\Big(1-\frac{1}{n}\Big)^{i-1}.$$

Thus, since $(1-1/n)^{n-1} \geq \mathrm{e}^{-1}$ and $0 \leq i \leq n$, we may bound the progress probability by $1/(\mathrm{e}\,n) \leq p_{\mathrm{RSH}}(\mathbf{x}) \leq 1/n$.

So for both QLS* and the (1+1) QEA*, we have shown that

$$\frac{1}{\mathrm{e}\,n} \leq p_{\mathrm{RSH}}(\mathbf{x}) \leq \frac{1}{n}.$$

Therefore, by Corollary 5.13,

$$\mathrm{E}[T_{\mathrm{QSH}}] \in \Theta(n^{-1/2}\,\mathrm{E}[T_{\mathrm{RSH}}]) = \Theta(n^{3/2}).$$

Now consider the progressive selection strategy. The upper bound is trivial, Lemma 5.12 implies that QSH is always asymptotically at least as fast in expectation as the corresponding RSH.

For the lower bound, we partition the search space into two sets

$$\mathcal{S}_1 := \{\mathbf{x} \mid \textsc{LeadingOnes}(\mathbf{x}) < n/2\},$$

$$\mathcal{S}_2 := \{\mathbf{x} \mid \textsc{LeadingOnes}(\mathbf{x}) \geq n/2\},$$

and give a lower bound for the time spent in $\mathcal{S}_1$.

We want to apply Lemma 5.12. For this, we have to give a lower bound on the progress probability $p_{\mathrm{RSH}}(\mathbf{x})$ of RLS and the (1+1) EA for all $\mathbf{x} \in \mathcal{S}_1$. Thus, let $\mathbf{x}$ be any search point in $\mathcal{S}_1$. First consider RLS. The progressive selection strategy will accept any search point of equal fitness. Thus if the mutation operator flips any bit in the second half of $\mathbf{x}$, then the offspring is accepted. Therefore,

$$p_{\mathrm{RSH}}(\mathbf{x}) \geq 1/2.$$

Now we turn to the (1+1) EA. If the mutation operator flips exactly one bit in the second half, and no bit in the first half, then the offspring will be accepted since it has at least the same fitness and differs from $\mathbf{x}$ (which is required in the quantum case). There are many other ways to produce offsprings that are accepted, but this particular way will suffice. Hence, the progress probability is at least

$$p_{\mathrm{RSH}}(\mathbf{x}) \geq \frac{n}{2}\cdot\frac{1}{n}\cdot\Big(1-\frac{1}{n}\Big)^{n-1} \geq \frac{1}{2\mathrm{e}}.$$

In both cases, for the QSH and the (1+1) QEA, we have $p_{\mathrm{RSH}}(\mathbf{x}) \geq 1/(2\mathrm{e})$. Therefore, by Corollary 5.13, we get

$$\mathrm{E}[T_{\mathrm{QLS}}] \in \Omega(\mathrm{E}[T^{(1)}_{\mathrm{RLS}}])$$

and

$$\mathrm{E}[T_{\text{QEA}}] \in \Omega(\mathrm{E}[T^{(1)}_{\text{EA}}]),$$

where $\mathrm{E}[T^{(1)}_{\text{RLS}}]$ and $\mathrm{E}[T^{(1)}_{\text{EA}}]$ are the expected times needed by RLS and the (1+1) EA to leave $\mathcal{S}_1$. These expected times are at least of the same order as the expected times to solve the LeadingOnes problem on $n/2$ bits. (Just consider the first $n/2$ bits of $\mathbf{x}$). Thus $\mathrm{E}[T^{(1)}_{\text{RLS}}] \in \Omega(n^2)$ and $\mathrm{E}[T^{(1)}_{\text{EA}}] \in \Omega(n^2)$. (Another way to see this is to recapture the proof of Theorem 6.5). Therefore, Theorem 6.6 follows. □

6.3 Discrepancy

The pseudo-Boolean function Discrepancy denotes half the difference in the number of one-bits and zero-bits in a bit-string $\mathbf{x} \in \{0,1\}^n$ of even length n, that is, let

$$\textsc{Discrepancy}(\mathbf{x}) := \left| \frac{n}{2} - \textsc{OneMax}(\mathbf{x}) \right| . \tag{6.3}$$

This function is not a standard test function for evolutionary algorithms, because it is too easy to optimize. However, it demonstrates that for easy problems (with high progress probabilities) a QSH is not necessarily strictly faster than the corresponding RSH.

Lemma 6.7. *Let $n \in \mathbb{N}$ be even and let $\mathbf{x} \in \{0,1\}^n$ be chosen uniformly at random. Then,*

$$\mathrm{E}[\textsc{Discrepancy}(\mathbf{x})] \in \Theta(n^{1/2}) .$$

Proof. Let $n = 2k$. Then, the lemma follows from

$$\begin{aligned}
\mathrm{E}[\textsc{Discrepancy}(\mathbf{x})] &= 2 \sum_{i=0}^{k} (k-i)\tbinom{2k}{i} 2^{-2k} \\
&= \sum_{i=0}^{k} \Big((2k-i)\tbinom{2k}{i} - i\tbinom{2k}{i} \Big)\, 2^{-2k} \\
&= \sum_{i=0}^{k} 2k \Big(\tbinom{2k-1}{i} - \tbinom{2k-1}{i-1} \Big)\, 2^{-2k} \\
&= 2k \tbinom{2k-1}{k}\, 2^{-2k} \\
&= k \tbinom{2k}{k}\, 2^{-2k},
\end{aligned}$$

and from $\binom{2k}{k} \sim 2^{2k}/\sqrt{\pi\, k}$ due to Stirling's formula. □

For the function Discrepancy, we show that the runtimes of the RSHs and QSHs we consider are asymptotically equal.

Theorem 6.8. *For each of the algorithms (1+1) EA, RLS, (1+1) QEA, and QLS, both for the conservative and for the progressive selection strategy, the runtime for minimizing* DISCREPANCY *is in* $\Theta(\sqrt{n})$.

Proof. We first show that throughout the runs of the classical algorithms (RLS, the (1+1) EA, RLS*, and the (1+1) EA*) the progress probabilities are bounded below by positive constants. By Corollary 5.13, this implies that the runtimes of the QSHs and the corresponding RSHs are of the same order.

For every search point $\mathbf{x}$ with $\text{DISCREPANCY}(\mathbf{x}) > 0$ we have at least $n/2$ zero-bits or $n/2$ one-bits. In both cases, flipping exactly one of these bits and leaving all other bits untouched decreases the discrepancy by one. Thus, the progress probability is at least $(n/2) \cdot (1/n) \geq 1/2$ for RLS and at least $(n/2) \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq 1/(2\mathrm{e})$ for the (1+1) EA, independent of the selection strategy we use.

It remains to establish the runtimes of the classical algorithms. Since the objective function DISCREPANCY behaves symmetrically for all search points of equal value, the runtimes of the classical algorithms coincide for the progressive and the conservative selection strategy. We restrict ourselves to the conservative versions and show that both have runtimes in $\Theta(n^{1/2})$.

To this end, consider a search point $\mathbf{x}$ with $\text{DISCREPANCY}(\mathbf{x}) > 0$, and let $\mathbf{y}$ be the search point after one mutation and selection step. Then for both RLS* and the (1+1) EA*, the expected progress is bounded from above and below by two positive constants, that is

$$\frac{1}{2\mathrm{e}} \leq \mathrm{E}[\text{DISCREPANCY}(\mathbf{x}) - \text{DISCREPANCY}(\mathbf{y})] \leq 1. \tag{6.4}$$

The lower bound holds since both strategies make a progress of one with probability at least 1/2e as we have already seen above. The upper bound holds since the progress is bounded from above by the expected number of flipped bits which equals one for both algorithms.

By classical drift analysis (see [17]), the inequalities in (6.4) imply that the runtimes are of the same order as the objective value of the initial search point, which we have shown in Lemma 6.7 to be in $\Theta(n^{1/2})$. □

6.4 Needle

The pseudo-Boolean function NEEDLE has a unique optimum, and is constant elsewhere. For $\mathbf{x} \in \{0,1\}^n$,

$$\text{NEEDLE}(\mathbf{x}) := \begin{cases} 1, & \text{if } \mathbf{x} = (0, \ldots, 0) \\ 0, & \text{else.} \end{cases} \tag{6.5}$$

Theorem 6.9.

(a) *The runtime of the (1+1) EA maximizing* Needle *is in* $\Theta(2^n)$.

(b) *The runtime of RLS maximizing* Needle *is in* $\Theta(2^n)$.

(c) *The runtime of the (1+1) EA* maximizing* Needle *is in* $\Theta(\frac{1}{2^n}n^n)$.

(d) *RLS* asymptotically almost surely does not find the global maximum of* Needle *(that is, with probability tending to* 1 *as* $n \to \infty$*).*

Proof. The statement for RLS* is clear because the algorithm only finds the optimum if the starting point is either the optimum itself or adjacent to the optimum. The probability that this happens is exponentially small.

For RLS and the (1+1) EA, see [14]. For the (1+1) EA*, the algorithm starts in a point $\mathbf{x} \in \{0,1\}^n$ with probability 2^{-n} and then needs in expectation $\Theta(n^d)$ steps to find the optimum $(0,\ldots,0)$, where $d > 0$ is the Hamming weight of $\mathbf{x}$. For $\mathbf{x} = (0,\ldots,0)$, it needs 0 steps. Thus, by the Binomial formula, we have $\mathrm{E}[T_{\mathrm{EA}^*}] \in \Theta(\mu_{\mathrm{EA}^*})$ with

$$\mu_{\mathrm{EA}^*} = \frac{1}{2^n}\sum_{d=1}^{n}\binom{n}{d}n^d = \frac{(n+1)^n - 1}{2^n} = \frac{(1+1/n)^n n^n - 1}{2^n}$$

Hence, $\mathrm{E}[T_{\mathrm{EA}^*}] \in \Theta\big((n/2)^n\big)$ since $\lim_{n\to\infty}(1+1/n)^n = \mathrm{e}$. □

We show that the quantum versions perform equally bad on the Needle function. Only the $(1+1)$ *QEA** is better than the $(1+1)$ *EA**. However, since the runtime is super-exponential (growing faster than any exponential function), the improvement is small in comparison.

Theorem 6.10.

(a) *The runtime of the (1+1) QEA maximizing* Needle *is in* $\Theta(2^n)$.

(b) *The runtime of QLS maximizing* Needle *is in* $\Theta(2^n)$.

(c) *The runtime of the (1+1) QEA* maximizing* Needle *is in* $\Theta(\frac{\mathrm{e}^{\sqrt{n}}}{2^n}n^{n/2})$.

(d) *QLS* asymptotically almost surely does not find the global maximum of* Needle.

Proof. First let us look at the progressive algorithms. Since all points except the optimum are of equal fitness, the algorithms accept every sample point as a new search point. Hence, the progress probability is 1, and $\mathrm{E}[T_{\mathrm{QSH}}] \in \Theta(\mathrm{E}[T_{\mathrm{RSH}}])$ by Corollary 5.13. This proves *(a)* and *(b)*.

The statement for QLS* follows immediately from the statement for RLS* in Theorem 6.9 since both algorithms have exactly the same probability to terminate by Theorem 2.2.

So let us look at the (1+1) QEA*. The algorithm will visit at most two search points $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$, with $\mathbf{x}^{(0)}$ drawn uniformly at random, and $\mathbf{x}^{(1)} =$

$(0,\dots,0)$ the optimum. Therefore, the expected number of visits $m(\mathbf{x})$ is $\frac{1}{2^n}$ for all $\mathbf{x} \neq (0,\dots,0)$.

Assume that the Hamming weight of $\mathbf{x}^{(0)}$ is d. Then the progress probability $p_{\text{QEA}^*}(\mathbf{x}^{(0)})$ of the associated (1+1) EA* is n^{-d} and its expected progress time is in $\Theta(n^d)$ (or 0 if $\mathbf{x}^{(0)} = (0,\dots,0)$). Since there are exactly $\binom{n}{d}$ search points with Hamming weight d, we have by Theorem 5.11 that $\mathrm{E}[T_{\text{QEA}^*}] \in \Theta(\mu_{\text{QEA}^*})$ with

$$\mu_{\text{QEA}^*} := \frac{1}{2^n}\sum_{d=1}^{n}\binom{n}{d} n^{d/2}.$$

By the Binomial formula, we obtain

$$\mu_{\text{QEA}^*} = \frac{1}{2^n}\left((n^{1/2}+1)^n - 1\right) = \frac{1}{2^n}\left((1+n^{-1/2})^n n^{n/2} - 1\right)$$

and therefore, since $\mathrm{e}^{x-2x^2} \leq 1 + x \leq \mathrm{e}^x$ holds for all $x \in [-1/2, 1/2]$, we have

$$\frac{\mathrm{e}^{\sqrt{n}-2} n^{n/2} - 1}{2^n} \leq \mu_{\text{QEA}^*} \leq \frac{\mathrm{e}^{\sqrt{n}} n^{n/2} - 1}{2^n}.$$

Thus,

$$\mathrm{E}[T_{\text{QEA}^*}] \in \Theta\Big(\frac{\mathrm{e}^{\sqrt{n}} n^{n/2}}{2^n}\Big).$$

□

6.5 Jump

Let m be a positive integer constant. The pseudo-Boolean function $\textsc{Jump}_m$ is defined as follows. For $\mathbf{x} \in \{0,1\}^n$,

$$\textsc{Jump}_m(\mathbf{x}) := \begin{cases} \textsc{OneMax}(\mathbf{x}), & \text{if } 0 < \textsc{OneMax}(\mathbf{x}) < m \\ 2n - \textsc{OneMax}(\mathbf{x}), & \text{if } \textsc{OneMax}(\mathbf{x}) \geq m \\ 2n, & \text{if } x = (0,\dots,0) \end{cases} \tag{6.6}$$

The function has a unique maximum in $(0,\dots,0)$, but has small fitness in a region around this point. Therefore, typically a RSH will have to jump to the optimum. This problem has been analyzed by Droste, Jansen, and Wegener [11] in order to show that a wide variance of runtimes can occur. Compared to these authors, we have changed the definition of the function slightly so that the indices are more convenient for our analysis.

Theorem 6.11. *Let $m \in \mathbb{N}$ with $m \geq 2$ be a constant.*

(a) The runtimes of the (1+1) EA and the (1+1) EA* maximizing $\textsc{Jump}_m$ are in $\Theta(n^m)$.

(b) RLS and RLS asymptotically almost surely do not find the global maximum of* $\textsc{Jump}_m$.

Proof. For the (1+1) EA, see [11]. For the (1+1) EA*, note that for any two search points $\mathbf{x}$ and $\mathbf{y}$ with the same number of one-bits, the problem is symmetric with respect to $\mathbf{x}$ and $\mathbf{y}$. That is, there is a fitness-invariant automorphism of the space mapping $\mathbf{x}$ to $\mathbf{y}$. Therefore, the runtime is the same for $\mathbf{x}$ and $\mathbf{y}$. So the runtime is equal for the (1+1) EA and the (1+1) EA*, and the statement for the (1+1) EA* follows.

The statements for RLS and RLS* are obvious because the only fitness-increasing paths ending in the optimum start either in the optimum itself or in a search point of Hamming weight 1. Since the sequence of search points will be such a path, the algorithm can only find the optimum if it starts either in the optimum itself or in a point of Hamming weight 1. By the Chernoff bound (see, e.g., Chapter 1 in [4]), the probability for this event is exponentially small as $n \to \infty$. □

We find that for $\textsc{Jump}_m$, the conservative algorithm gains quadratic speedup (for $m > 1$) while the progressive one is hardly better than its classic version. The reason is that in the conservative setting there is one very hard step (the jump) with no easy alternatives, whereas in the progressive version the algorithm is allowed to make easy moves along the boundary of the gap. Note that for the classical algorithms, there is no difference between the conservative and the progressive selection strategy.

Theorem 6.12. *Let $m \in \mathbb{N}$ with $m \geq 2$ be a constant.*

(a) The runtime of the (1+1) QEA maximizing $\textsc{Jump}_m$ *is in* $\Theta(n^{m-1/2})$.

(b) The runtime of the (1+1) QEA maximizing* $\textsc{Jump}_m$ *is in* $\Theta(n^{m/2})$.

(c) QLS and QLS asymptotically almost surely does not find the global maximum of* $\textsc{Jump}_m$.

Before we prove the theorem, we state a lemma.

Lemma 6.13. *Let $\mathbf{x} \in \{0,1\}^n$ be of Hamming weight $k \in \{1, \ldots, n\}$. Then the probability p that the mutation operator of the (1+1) EA generates a vector $\mathbf{y} \neq \mathbf{x}$ of the same Hamming weight satisfies*

$$\frac{\min\{k, n-k\}}{2en} \leq p \leq \frac{k}{n}.$$

Proof. Since $\mathbf{y} \neq \mathbf{x}$ and both vectors have the same Hamming weight, the mutation operator has to flip at least one one-bit and at least one zero-bit.

We bound p from below by the probability that exactly one zero-bit and one one-bit are flipped in $\mathbf{x}$,

$$p \geq k(n-k) \cdot \frac{1}{n^2} \cdot \left(1 - \frac{1}{n}\right)^{n-2} \geq \frac{\min\{k, n-k\}}{2en},$$

since

$$k(n-k) = \min\{k, n-k\} \cdot \max\{k, n-k\} \geq \frac{\min\{k, n-k\}n}{2}.$$

To bound p from above, we apply the union-bound to the (not necessarily independent) events that a given pair of a zero-bit and a one-bit is flipped in $\mathbf{x}$, while we do not care whether the other bits flip. Therefore

$$p \leq k(n-k) \cdot \frac{1}{n^2} \leq \frac{k}{n}.$$

□

Proof of Theorem 6.12. The statements for QLS and QLS* follow directly from the statements for RLS and RLS*, because the classical algorithm will find the optimum if and only if the quantum algorithm does.

For the (1+1) QEA and (1+1) QEA*, we divide the run of the algorithms into three phases, some of which may be empty. In the first phase, the fitness is strictly less than m. In the second phase, the fitness is at least m, but strictly smaller than $2n-m$. In the third phase, the fitness is at least $2n-m$.

We claim that the problem of leaving the first phase is strictly easier than the problem of maximizing OneMax. In fact, consider the auxiliary problem where all search points $\mathbf{x}$ with $0 < \text{OneMax}(\mathbf{x}) < m$ have the same fitness as in Jump, but all other search points have fitness $2n$. Then the problem of leaving the first phase for Jump_m is the same as finding a global maximum of the auxiliary problem. On the other hand, the auxiliary problem is identical with the problem OneMax except that a larger sets of points are global maxima.

So the problem of leaving the first phase is indeed easier than OneMax. By Theorem 6.2, the (1+1) QEA and the (1+1) QEA* will both spend in expectation at most linear time in the first phase.

Similarly, it is easy to see that again both, the (1+1) QEA and the (1+1) QEA*, will spend in expectation at most linear time in the second phase.

For the third phase, we distinguish between the (1+1) QEA and the (1+1) QEA*. First we look at the (1+1) QEA*. Given a search point $\mathbf{x} \in \{0,1\}^n$ of fitness $2n-m$, it will accept only the optimum as its next search point. Thus, the expected optimization time for the third phase is exactly $r_{\text{QSH}}(\mathbf{x})$. Since its Hamming weight is m, the progress probability $p_{\text{RSH}}(\mathbf{x})$ of the corresponding (1+1) EA* is n^{-m} (independently of the choice of x). Thus, the runtime of the third phase is

$$r_{\text{QSH}}(\mathbf{x}) \in \Theta\big((r_{\text{RSH}}(\mathbf{x}))^{1/2}\big) = \Theta\big((p_{\text{RSH}}(\mathbf{x}))^{-1/2}\big) = \Theta\big(n^{m/2}\big).$$

Since the other phases took at most linear time, for $m > 1$ the runtime of the (1+1) QEA* is dominated by the third phase and is in $\Theta\left(n^{m/2}\right)$.

We now turn to the (1+1) QEA. Again, let $\mathbf{x} \in \{0,1\}^n$ be a search point of Hamming weight m. This time, the situation is slightly more complicated since the (1+1) QEA may accept any other search point of Hamming weight m. We therefore again consider the corresponding (1+1) EA. The probability that the mutation operator produces another search point of Hamming weight m is in $\Theta(m/n) = \Theta(1/n)$ by Lemma 6.13, since m is a constant. On the other hand, the probability that the mutation operator yields the optimum is in $\Theta(n^{-m})$. Therefore, the probability to jump to the optimum subject to the condition that we accept the search point is

$$\Pr\left(\textsc{mut}(\mathbf{x}) = (0,\ldots,0) \mid \textsc{Jump}_m(\textsc{mut}(\mathbf{x})) \le m\right) \in \Theta\bigl(n^{-(m-1)}\bigr).$$

So we expect to visit $\Theta(n^{m-1})$ search points in the third phase. Moreover, for all search points $\mathbf{x}$ of Hamming weight m, the progress probability $p_{\text{RSH}}(\mathbf{x})$ of the (1+1) EA is in $\Theta(1/n)$. Therefore, by Theorem 5.11, the optimization time $T^{(3)}_{\text{QSH}}$ for the third phase of the (1+1) QEA satisfies

$$\mathrm{E}[T^{(3)}_{\text{QSH}}] \in \Theta(\mu^{(3)}_{\text{QSH}})$$

with

$$\mu^{(3)}_{\text{QSH}} = \sum_{\mathbf{x}\colon |\mathbf{x}|_1 = m} m(\mathbf{x})(r_{\text{RSH}}(\mathbf{x}))^{1/2}.$$

We have already seen that

$$(r_{\text{RSH}}(\mathbf{x}))^{1/2} \in \Theta(n^{1/2})$$

and that

$$\sum_{\mathbf{x}\colon |\mathbf{x}|_1 = m} m(\mathbf{x}) \in \Theta(n^{m-1}).$$

Therefore, $\mathrm{E}[T^{(3)}_{\text{QSH}}]$ is in $\Theta(n^{m-1/2})$. □

6.6 TinyTrap

Let $d := \frac{3n}{2\log_2 n}$ be an integer[4] and let TinyTrap be the pseudo-Boolean function that maps $\mathbf{x} \in \{0,1\}^n$ to

$$\textsc{TinyTrap}(\mathbf{x}) := \begin{cases} \textsc{OneMax}(\mathbf{x}), & \text{if } \textsc{OneMax}(\mathbf{x}) \le d-1 \\ -1, & \text{else.} \end{cases} \tag{6.7}$$

RLS and QLS with either selection strategy have unbounded runtime minimizing TinyTrap, since the initial search point might be the local minimum $(0,\ldots,0)$. We therefore restrict ourselves to the different variants of the $(1+1)$-EA.

[4]For values of n where d is non-integral we can round d and obtain the same asymptotic results.

Theorem 6.14. *The runtime of the (1+1) EA and the (1+1) EA* minimizing* TinyTrap *is at least* $2^{n/4}$.

Proof. The following argument holds for both selection strategies. Consider the event where the (1+1) EA starts in the local minimum $(0,\ldots,0)$. To leave this point, at least d of the bits have to be flipped which (by the union bound) happens with probability at most

$$\binom{n}{d}\cdot\left(\frac{1}{n}\right)^{d} \le \left(\frac{\mathrm{e}n}{d}\right)^{d}\cdot\left(\frac{1}{n}\right)^{d} = \left(\frac{d}{\mathrm{e}}\right)^{-d} \le n^{-5d/6} = 2^{-5n/4}$$

since $d/\mathrm{e} \ge n^{5/6}$ for sufficiently large n (and thus, sufficiently large d). Therefore, conditioned on the event to start in $(0,\ldots,0)$, the runtime of the (1+1) EA is at least $2^{5n/4}$ and since the probability to start in $(0,\ldots,0)$ is 2^{-n}, the unconditional runtime is at least $2^{n/4}$ by the law of total expectation. □

Theorem 6.15. *The runtime of the (1+1) QEA and the (1+1) QEA* minimizing* TinyTrap *is in* $O(1)$.

Proof. The following argument holds for both selection strategies. With very high probability the first search point has Hamming weight at least d in which case the runtime is 1. However, since $\mathrm{e}n/d \le n^{1/12}$ for sufficiently large n, there are at most

$$\sum_{i=0}^{d-1}\binom{n}{i} \le n\binom{n}{d} \le n\left(\frac{\mathrm{e}n}{d}\right)^{d} \le n\cdot n^{d/12} = n2^{n/8}$$

search points of Hamming weight at most $d-1$. Hence, the probability that the initial search point is one of them is at most $n2^{-7n/8}$. Next, we give an upper bound on the runtime of the (1+1) QEA conditioned on the event that it is indeed initialized with one of these points. In this case, we consider two phases, where the first phase ends when the (1+1) QEA has either found the local minimum $(0,\ldots,0)$ or a global optimum. Like in the proof of Theorem 6.12, the length of this phase is dominated by the runtime of the (1+1) QEA on OneMax which is in $\Theta(n)$.

At the beginning of the second phase, the (1+1) QEA either found a global optimum (in this case we are done) or the current search point is the local minimum $(0,\ldots,0)$. For the corresponding (1+1) EA, the probability to leave the local minimum $(0,\ldots,0)$ is at least the probability to flip exactly d of the zero-bits, that is

$$\binom{n}{d}\cdot\left(\frac{1}{n}\right)^{d}(1-1/n)^{n-d} \ge \frac{1}{\mathrm{e}}\cdot\left(\frac{n}{d}\right)^{d}\cdot\left(\frac{1}{n}\right)^{d} \ge \frac{1}{\mathrm{e}}\cdot n^{-d} = \frac{1}{\mathrm{e}}\cdot 2^{-3n/2}.$$

Thus, the expected progress time $r_{\mathrm{RSH}}((0,\ldots,0))$ of the (1+1) EA is in $O(2^{3n/2})$. Now, we recall that we are actually looking at the (1+1) QEA. By

Lemma 5.5, the expected progress time $r_{\text{QSH}}((0,\dots,0))$ of the (1+1) QEA is at most $O(2^{3n/4})$. Note that this difference is the reason why the following argument does not give runtime $\Theta(1)$ for the classical $(1+1)$ EA as well. However, for the (1+1) QEA we have just seen that the runtime, conditioned on the event that the Hamming weight of the initial search point is at most $d-1$, is in $O(2^{3n/4})$ (the runtime of the second phase dominates the runtime of the first phase). Recall that the probability of the initial search point actually satisfying this condition is only $O(n2^{-7n/8})$. Hence, the total (unconditional) runtime of the (1+1) QEA is in $O(1 + n2^{-7n/8} \cdot 2^{3n/4}) = O(1)$ by the law of total expectation. □

7 Conclusion

In this paper, we have presented an approach to evolutionary algorithms on a quantum computer where we keep the mutation and selection process from the classical setting and use quantum probability amplification (Grover search) in order to find an acceptable offspring more quickly. We show that this does not affect the trajectory the algorithm takes on its way to an optimal solution, the quantum amplification only speeds it up. Furthermore our approach is universal, that is, it works for any mutation operator. We also provide tools for estimating the runtime using parameters of the classical heuristic.

For five of the six problems we investigated we encountered that using quantum search gave at most a quadratic improvement over the corresponding classical heuristic. This is similar to other general settings like unordered search [6, 26] or query complexity of local search on a graph [1], in which it is proven or conjectured that quantum computers can give at most a quadratic speedup.

On the example of the function TinyTrap we saw that an exponential runtime of a RSH may drop to polynomial, even to $\Theta(1)$ for the corresponding QSH. However, keep in mind that this is due to the occurrence of an highly unlikely event (starting in the trap region) and will hardly be observed in a typical run. It is an interesting question whether such an improvement from exponential to polynomial runtime can also occur in less artificial problems and in a typical run of a QSH.

The other analyzed examples OneMax, LeadingOnes, Discrepancy, Needle, and Jump$_m$ show that a substantial speedup is possible (as for LeadingOnes) but is not guaranteed (as for Discrepancy). The harder it is for the classical search heuristic to make progress, the better will quantum acceleration work.

We have also seen that it is important to choose the selection strategy carefully, since not only the runtime in the classical setting but also the speedup due to quantum acceleration depends on the choice of the selection strategy. The reason for the different results is that by allowing equality of the objective functions we increase the number of valid successor states and thus we increase the probability to find such a state. But quantum enhancement is more powerful if these probabilities are small, as is illustrated by Corollary 5.13. However, we believe there are ways to keep quantum enhancement powerful and still allow the algorithm to move to a successor state with unchanged objective value.

To conclude, we demonstrated a wide range of different behaviors of the progressive and conservative versions of the (1+1) QEA and QLS on a number of well-studied basic pseudo-Boolean functions. In the line of this research, the next step would be to analyze the effects of quantum acceleration on classical problems in combinatorial optimization.

References

[1] S. Aaronson. Lower bounds for local search by quantum arguments. *SIAM Journal on Computing*, 35(4):804–824, 2006.

[2] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002.

[3] A. Ambainis. Quantum random walks — new method for designing quantum algorithms. In *SOFSEM '08: Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science*, volume 4910 of *LNCS*, pages 1–4. Springer, 2008.

[4] A. Auger and B. Doerr, editors. *Theory of Randomized Search Heuristics*, volume 1 of *Series on Theoretical Computer Science.* World Scientific, 2011.

[5] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001.

[6] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[7] A. Berzina, A. Dubrovsky, R. Freivalds, L. Lace, and O. Scegulnaja. Quantum query complexity for some graph problems. In *SOFSEM '04: Proceedings of the 30th Conference on Current Trends in Theory and Practice of Computer Science*, volume 2932 of *LNCS*, pages 140–150. Springer, 2004.

[8] H.-G. Beyer, H.-P. Schwefel, and I. Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287(1):101–130, 2002.

[9] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998.

[10] G. Brassard, P. Høyer, and A. Tapp. Quantum counting. In *ICALP '98: Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *LNCS*, pages 820–831. Springer, 1998.

[11] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1–2):51–81, 2002.

[12] C. Dürr, M. Heiligman, P. Høyer, and M. Mhalla. Quantum query complexity of some graph problems. *SIAM Journal on Computing*, 35(6):1310–1328, 2006.

[13] C. Dürr and P. Høyer. A quantum algorithm for finding the minimum. *arXiv:quant-ph/9607014v2*, 1996.

[14] J. Garnier, L. Kallel, and M. Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7(2):173–203, 1999.

[15] L. K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96: Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM, 1996.

[16] K.-H. Han and J.-H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6(6):580–593, 2002.

[17] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

[18] D. Johannsen, P. P. Kurur, and J. Lengler. Can quantum search accelerate evolutionary algorithms? In *GECCO '10: Proceedings of the]12th Annual Genetic and Evolutionary Computation Conference*, pages 1433–1440. ACM, 2010.

[19] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004.

[20] F. Magniez, N. A., J. Roland, and M. Santha. Search via quantum walk. *SIAM Journal on Computing*, 40(1):142–164, 2011.

[21] F. Magniez, A. Nayak, P. C. Richter, and M. Santha. On the hitting times of quantum versus random walks. *Algorithmica*, 63(1–2):91–116, 2012.

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

[23] M. Santha. Quantum walk based search algorithms. In *TAMC '08: Proceedings of the 5th Annual Conference on Theory and Applications of Models of Computation*, volume 4978 of *LNCS*, pages 31–46. Springer, 2008.

[24] L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy. Finding a better-than-classical quantum AND/OR algorithm using genetic programming. In *CEC '99: Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, volume 3, pages 2239–2246. IEEE, 1999.

[25] M. Szegedy. Quantum speed-up of markov chain based algorithms. In *FOCS '04: Proceedings of the 45th Annual IEEE Syposium on Foundations of Computer Science*, pages 32–41. IEEE, 2004.

[26] C. Zalka. Grover's quantum searching algorithm is optimal. *Physical Review Letters*, 60(4):2746–2751, 1999.

[27] S. Zhang. *New quantum algorithms and quantum lower bounds.* PhD thesis, 2006.

[28] S. Zhang. New upper and lower bounds for randomized and quantum local search. In *STOC '06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 634–643. ACM, 2006.