ETH ZÜRICH

SEMESTER THESIS

Search for Non-Interactive Entanglement Distillation Protocols Aided by Numerical Optimization

Author:
Yves SALATHÉ

Advisor:
Dejan DUKARIC
Supervisor:
Prof. Dr. Stefan WOLF

March 21, 2011

Abstract

Entanglement is a property of physical reality that cannot be explained by the classical laws of physics [1]. In quantum cryptography, entangled pairs of photons are used to establish secure channels and in quantum computing entanglement is a resource that can be used to send the state of a qubit from one location to another [2] using classical communication. An important question in the context of physical realizations of entangled systems is whether entanglement can be restored from the imperfectly entangled systems resulting from decoherence. The Schmidt projection method by Bennet et. al. [3] can concentrate entanglement from many copies of an entangled pure state into a smaller set of perfectly entangled states. Unlike many protocols that were later developed for mixed states [4, 5, 6, 7, 8], the Schmidt projection method works without communication. On the other hand, the Schmidt projection method only becomes efficient if a large number of copies of the initially entangled state are given. In the first part of this work we show that the Schmidt projection method cannot be used to increase the fully entangled fraction if used with only two copies of an entangled pure input state and under the assumption that no post selection is allowed. Conversely, we show that this task is possible if one has 3 copies of a pure input pair instead. From the protocol for 3 copies of a pure state, we derive a non-interactive protocol that works for certain mixed states. In the second part we demonstrate the possibility to numerically find entanglement distillation protocols that do not use communication by solving a high dimensional optimization problem using the evolutionary algorithm CMA-ES [9].

Contents

Chapter 1

Introduction

The field of quantum information theory deals with the implications of quantum mechanics on our ability to process information and vice-versa combines the known theoretical results from information theory and related fields with quantum theory to make predictions about the behaviour of nature. One of the most surprising features of quantum mechanics is entanglement. In 1935, Einstein et. al. [10] argued that the so called "reduction of the wave packet", i.e. the way measurements are described in quantum mechanics leads to a paradoxical situation where the probabilities of different results of a measurement that is performed on one of two or more spatially separated systems will affect the state of the combined system even though the systems are not able to interact by means of any known physical interaction. They conclude that quantum mechanics is thus unlikely to describe nature in a complete way. In 1964, John Bell [1] found a way to test whether such instantaneous effects exist in nature by formulating an inequality that limits the correlations of simultaneous local measurements on two subsystems under the assumption that the subsystems cannot interact during the time the measurements are performed. John Bell predicted that if nature acts quantum mechanically, his inequality will be violated by so called entangled states. Aspect et. al. verified the violation of the Bell inequalities by experiment. To our present knowledge exactly the opposite of Einstein's conclusion is true: we need quantum mechanics to describe effects like entanglement. Since then, a lot of research has dealt with the question whether we can use entanglement to perform useful tasks. In 1984, Bennet and Brassard [11] proposed a secure way to exchange one time pads for encrypted communication. In 1992 Bennet and Wiesner [3] showed that it is possible to send two classical bits by sending a so called qubit (a quantum mechanical two-state system), if the sender (Alice) and the receiver (Bob) initially share an entangled pair of qubits. In 1993, Bennet et. al. [2] developed a quantum protocol that uses an entangled pair of qubits to send the state of a quantum bit from one location to another by sending two classical bits. In fact, entanglement has become a resource to perform different tasks that would be impossible without it.

However, a major problem with physical implementations of these ideas is that the systems cannot be perfectly isolated. The system interacts with the environment in a way that cannot be perfectly predicted and controlled. This so called decoherence can destroy entanglement or make it weaker. So we need a way to restore it. This is where entanglement distillation comes into play. Bennet et. al. [12] were the first to develop two protocols that can concentrate entanglement from many copies of imperfectly entangled pairs of qubits into a smaller set of perfectly entangled qubits. One of the two protocols (called Schmidt projection) has the surprising property that it can be executed without the need of communication between the two distinct subsystems. While the Schmidt projection protocol was shown to be asymptotically optimal [13], there are two drawbacks: it only works for a large number of copies and only for pure states. The effect of decoherence is that the state of the system is transformed into a mixture of different states. To account for this kind of randomness, different protocols were found in various preceding publications that all deal with a certain type of mixted state [4, 5, 6, 7, 8] . Previous research on limits to the existence of non-interactive protocols for mixed states [7, 8, 14, 15] has revealed that there are many states for which entanglement distillation without communication is not possible. In the present work we further investigate the possibility of entanglement distillation without communication by summarizing the known results about the existence and non-existence of protocols in Chapter 4 and proving that the Schmidt projection method does not work without post-selection if only two copies of an imperfectly entangled state are given. In the second part of the work (Chapter 5) we demonstrate the possibility to find entanglement protocols by numerical optimization. Chapter 2 and 3 provide the foundations to introduce the notation and definitions needed to state the results of the present work.

Chapter 2

Foundations

The purpose of this chapter is to give an overview of the notation and concepts that are used in the present work.

2.1 The Qubit

The basic element of quantum information science, the quantum bit (qubit), is a physical system that acts quantum mechanical and has two distinguishable states. For such systems, the postulates of quantum mechanics (see e.g. [16, 17, 18]) imply that its state is described by a wave function, allowing it to be in any superposition of two basis states. One usually writes the state of the Qubit in Dirac's Ket notation: $|\psi\rangle \in \mathscr{H}_2$, where $\mathscr{H}_2$ is the two-dimensional Hilbert space, which means that the state is represented by a complex-valued two-dimensional vector. One defines the two vectors

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.1}$$

The set $\{|0\rangle, |1\rangle\}$ is called "computational basis". An arbitrary one-qubit pure state can then be written as:

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle, \tag{2.2}$$

where $\alpha, \beta \in \mathbb{C}^2$ and the states $|0\rangle$ and $|1\rangle$ are orthogonal i.e. $\langle 0|1\rangle = \langle 1|0\rangle = 0$. Note that the flipped Ket denotes the conjugate transpose of the corresponding vector, i.e. $\langle\psi| = (|\psi\rangle)^\dagger$ and is called "Bra".

Since the qubit is quantum mechanical, its time-evolution is described by the time-dependent Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}(t)|\psi(t)\rangle, \tag{2.3}$$

where $\hat{H}(t)$ is the Hamiltonian operator and $\hbar$ is Planks constant (see e.g. [19] for its physical value). The general solution of this equation can be written as

$$|\psi(t)\rangle = e^{-\frac{i}{\hbar}\int_0^t \hat{H}(\tilde{t})d\tilde{t}}|\psi(0)\rangle. \tag{2.4}$$

This can be verified by inserting (2.4) into (2.3). Since the Hamiltonian operator $\hat{H}$ is hermitean, the linear operator $\hat{T} := exp(-\frac{i}{\hbar}\int_0^t \hat{H}(\tilde{t})d\tilde{t})$ on the right hand side of Equation (2.4) is unitary:

$$\hat{T}^\dagger\hat{T} = \hat{T}\hat{T}^\dagger = e^{-\frac{i}{\hbar}\int_0^t \hat{H}(\tilde{t})d\tilde{t}}e^{+\frac{i}{\hbar}\int_0^t \hat{H}(\tilde{t})d\tilde{t}} = e^{\frac{i}{\hbar}(\int_0^t \hat{H}(\tilde{t})d\tilde{t}-\int_0^t \hat{H}(\tilde{t})d\tilde{t})} = e = \mathbb{1}. \tag{2.5}$$

Operations on qubits are therefore represented by unitary matrices.

The above statement leads to the so called quantum parallelism, which comes from the fact that the unitary operation $\mathbf{U}$ is linear. Thus, if it is applied on a state given by Equation (2.2) one gets

$$\mathbf{U}|\psi\rangle = \mathbf{U}(\alpha|0\rangle + \beta|1\rangle) = \alpha\mathbf{U}|0\rangle + \beta\mathbf{U}|1\rangle, \tag{2.6}$$

so the unitary operation acts on both basis states in the superposition simultaneously! This is one of the reasons why quantum algorithms can have lower complexity than the corresponding classical algorithms. Conversely, it is not so easy to design algorithms that outperform their classical counterparts since the state vector cannot be completely determined with a single measurement and because the measurement itself modifies the state vector, as will be detailed in Section 2.3.

2.2 Many Qubits: Combined Systems

To see how a system of two or more qubits can be composed from the ones of single qubits, it is best to think about how one would combine two classical bits. Such a classical combined system of two bits is described by the states of the individual bits. Since every classical bit can take on the values 0 or 1 (two possible characters), there are four possible states for the combined system: 00 01 10 and 11 (string over an alphabet of two characters). As seen in Section 2.1, this stands in contrast to quantum systems, where each qubit is described by a two-dimensional complex-valued vector, the components of which having the meaning of complex probability amplitudes for the event that the systems are in their respective two distinct basis states (usually denoted as $|0\rangle$ and $|1\rangle$). As for the classical combined system, a two qubit system is defined such that it has four different basis states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. The difference lies in the postulate that such a quantum system has to be described by a *four*-dimensional vector, each component describing the probability amplitude for the corresponding basis state. In fact, the state of the combined system is contained in the tensor product of the Hilbert spaces $\mathscr{H}_A$ and $\mathscr{H}_B$ of the individual subsystems:

$$\mathscr{H}_{AB} = \mathscr{H}_A \otimes \mathscr{H}_B, \tag{2.7}$$

where the basis vectors that span $\mathscr{H}_{AB}$ are the four different tensor products of the basis states of the single qubit systems, i.e.

$$|00\rangle = |0\rangle \otimes |0\rangle, \tag{2.8}$$
$$|01\rangle = |0\rangle \otimes |1\rangle, \tag{2.9}$$
$$|10\rangle = |1\rangle \otimes |0\rangle, \tag{2.10}$$
$$|11\rangle = |1\rangle \otimes |1\rangle. \tag{2.11}$$
$$\tag{2.12}$$

An arbitrary state $|\psi\rangle$ of the combined system is then written in terms of these basis states, i.e.

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{2.13}$$

with complex coefficients $\alpha, \beta, \gamma, \delta \in \mathbb{C}$.

This has wide ranging consequences: While each individual qubit, if in a pure state, has 3 degrees of freedom (the real and imaginary parts of the two coefficients α and β in (2.2) minus one degree of freedom due to the normalization constraint), the pure state of the combined system has 7 degrees of freedom (again the real and imaginary parts of the four coefficients α, β, γ and δ minus one degree of freedom due to the normalization constraint). So the combined system has actually more degrees of freedom than just the sum of the degrees of freedom of the individual systems! This already indicates that there are states of the combined system, that cannot be represented by just combining the vectors that describe the states of the individual system. In Section 2.4, particular examples of such states will be presented. Another important consequence can be seen from the way the number of basis states increases: For a system of n qubits, the states that form a complete basis set can be labelled by all possible n-bit-strings, of which there are exactly 2^n. So the cardinality of the complete set of basis states, and hence the dimensionality, grows exponentially! This indicates how hard it is to simulate a large-scale quantum system on a classical computer.

2.3 Measurements, Mixed States and the Partial Trace

In quantum mechanics, a measurement is described by a set of operators $\{\mathbf{M}_i \mid i = 1, \cdots, n\}$ that map the pre-measurement state to the post-measurement state that corresponds to a particular measurement result. Before a certain quantity is measured, it may in fact not yet be determined for a certain state. Mathematically, the probability to measure the result corresponding to $\mathbf{M}_i$ if the system is in a pure state $|\psi\rangle$ is

$$p_i := \langle\psi|\mathbf{M}_i|\psi\rangle \tag{2.14}$$

. The state $|\phi\rangle$ after the measurement will be

$$|\phi_i\rangle = \frac{\mathbf{M}_i|\psi\rangle}{\sqrt{p_i}}, \tag{2.15}$$

when the measurement result is known to correspond to $\mathbf{M}_i$ for a certain index i. If the measurement operators $\mathbf{M}_i$ are projection operators, the corresponding measurement is called projective measurement.

Related to the concept of measurements is the density operator. It is used to describe classical uncertainty in quantum mechanics. E.g. if one knows a certain measurement has been performed on a system that was in a known state before it was measured but one does not know the measurement result. In that case, one still knows the probability p_i with which each measurement result is obtained i.e. the probability p_i that the system is in a certain pure post-measurement state $|\phi_i\rangle$. This situation, which is called a mixed state, is described by the density operator

$$\rho = \sum_i p_i|\phi_i\rangle\langle\phi_i|. \tag{2.16}$$

Density operators are positive operators i.e. $\rho^\dagger = \rho$ and the eigenvalues are real-valued and non-negative. Furthermore the trace (the sum of diagonal elements which is equal to the sum of eigenvalues) of density operators is defined to be one, i.e. $\mathrm{tr}(\rho) = 1$. Note that for a system in a pure state $|\psi\rangle$, the density operator ρ corresponding to that state is the outer product of $|\psi\rangle$ with itself: $\rho = |\psi\rangle\langle\psi|$.

If a measurement is performed on a system in a state described by the density operator ρ, then the probabilities of the measurement outcomes are determined by

$$p_i = \mathrm{tr}(\mathbf{M}_i\rho). \tag{2.17}$$

The post-measurement state will be determined by

$$\rho_i = \frac{\mathbf{M}_i\rho\mathbf{M}_i^\dagger}{p_i}. \tag{2.18}$$

Another reason for having a mixed state may be that the system in concern interacts with another system (e.g. the environment), but the state vector for the combined system is unknown. Suppose the state vector of the combined system of two subsystems A and B is $|\psi\rangle$, then the density operator ρ of a particular subsystem A will determined by the so called partial trace over the subsystem B

$$\rho_A = \mathrm{tr}_B(|\psi\rangle\langle\psi|). \tag{2.19}$$

.

The partial trace $\mathrm{tr}_B(\rho)$ is computed by writing the density operator ρ in a basis where the basis vectors are separable, i.e. $\{|\psi_i\rangle \mid |\psi_i\rangle = |\psi_{A,k}\rangle \otimes |\psi_{B,l}\rangle, i = 1, \cdots, d\}$, where $|\psi_{A,k}\rangle$ is a state vector of subsystem A and $|\psi_{B,l}\rangle$ one of subsystem B:

$$\rho = \sum_i^{d_A}\sum_j^{d_B}\sum_k^{d_A}\sum_l^{d_B} \alpha_{i,j}(|\psi_{A,i}\rangle\langle\psi_{A,k}\rangle \otimes (|\psi_{B,j}\rangle\langle\psi_{B,l}|), \tag{2.20}$$

Then the partial trace $\mathrm{tr}_B(\rho)$ is

$$\rho_B\rho = \sum_i^{d_A}\sum_j^{d_B}\sum_k^{d_A}\sum_l^{d_B} \alpha_{i,j}\left(|\psi_{A,i}\rangle\langle\psi_{A,k}|\,\mathrm{tr}(|\psi_{B,j}\rangle\langle\psi_{B,l}|)\right) \tag{2.21}$$

$$= \sum_i^{d_A}\sum_k^{d_A} \alpha_{i,j}\left(|\psi_{A,i}\rangle\langle\psi_{A,k}|\sum_j^{d_B}\sum_l^{d_B}\langle\psi_{B,j}|\psi_{B,l}\rangle\right). \tag{2.22}$$

2.4 Entanglement

As already mentioned in Section 2.2, there are quantum mechanical systems for which the state of the system as a whole is not completely determined by the states of the individual subsystems. Take for example a system that consists of two subsystems. Let us consider the following state of the combined system

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.23}$$

There is no way to write $|\psi^+\rangle$ as a tensor product of the states of subsystems A and B, i.e.

$$|\psi^+\rangle \neq |\psi_A\rangle \otimes |\psi_B\rangle. \tag{2.24}$$

A state that has this property is called a non-separable or entangled state.

Now imagine that the two subsystems are spatially separated, i.e. one qubit is at Alice's location and the other qubit is at Bob's location but the state of the system that includes both qubits is still described by Equation (2.23). A consequence of the system being in the state $|\psi^+\rangle$ is that if the qubits are measured locally, i.e. if Alice measures the system with the set of operators $\{|0\rangle\langle 0|\otimes\mathbb{1},\ |1\rangle\langle 1|\otimes\mathbb{1}\}$ and Bob uses the operators $\{\mathbb{1}\otimes|0\rangle\langle 0|,\ \mathbb{1}\otimes|1\rangle\langle 1|\}$, they will get the same measurement result with certainty, given that they choose the same measurement basis. Although Alice and Bob measure the same value, the outcome of their measurement will be random. To see this, we use the knowledge of 2.3 and calculate the probabilities p_0 and p_1 using Formula (2.14) as well as the corresponding post-measurement states $|\phi_0\rangle$ and $|\phi_1\rangle$ using Formula (2.15) with e.g. Alice's set of measurement operators $\{|0\rangle\langle 0|\otimes\mathbb{1},\ |1\rangle\langle 1|\otimes\mathbb{1}\}$.

If Alice and Bob choose two different sets of measurement operators to measure the state $|\psi^+\rangle$ described by Equation (2.23), they will get the same result only with a certain probability.

Let Alice and Bob be far away from each other and let them perform the measurements on the state (2.23) within a small enough time window, so that their qubits cannot communicate through any known physical interaction. By the theory of relativity interaction cannot happen faster than the time it takes for a photon to travel in vacuum from Alice to Bob. Assume furthermore that Alice and Bob repeat the experiment with different measurement bases rotated to each other with different angles. One can show that the results of the measurement performed in this way violate the so called Bell inequality [1] which gives a bound for the value of a special correlation function under the assumption that the measurement results of Alice and Bob were predetermined by so called "hidden variables" which can only influence each other through the way of physical interaction (i.e. communication). So either the theory of quantum mechanics has to be considered wrong or the assumptions that were made to derive the Bell inequality do not conform with reality. The first experiment that was able to verify the violation of Bell's inequality while excluding the possibility of communication between the subsystems of Alice and Bob during the time of the measurement was performed in 1982 by Aspect et. al. [20] with polarization-entangled photons. Since then, many experiments with different physical realizations of entanglement have shown that Bell's inequality are violated in nature. Thus, nature behaves in a non-local and/or non-predetermined manner.

The above described setting, where there are qubits at two different locations, is called bipartite system. Not every state of the bipartite system can be used to violate the Bell inequalities. For example any separable state, i.e. a state that can be written as a tensor product of the states of the individual subsystems cannot be used to violate the Bell inequality. Those states that maximally violate the Bell inequality are called Bell or maximally entangled states and have the form

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|0+\rangle + |1-\rangle), \tag{2.25}$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|0+\rangle - |1-\rangle), \tag{2.26}$$

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0-\rangle + |0+\rangle), \tag{2.27}$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|0-\rangle - |1+\rangle). \tag{2.28}$$

We denote the basis states of Alice's subsystem as $|0\rangle$ and $|1\rangle$ and those of Bob $|+\rangle$ and $|-\rangle$ to make clear which qubits are at which location.

Despite of the violation of the Bell inequalities being a behaviour that cannot be explained by classical physics, entangled states cannot be used to communicate. This is indicated by the description of the subsystems by the partial trace of the combined system. The partial trace describes the absence of knowledge about the other subsystem. Let the combined system be in the state $|\psi^+\rangle$. Thus, the density operator is

$$\rho = |\psi^+\rangle\langle\psi^+| \tag{2.29}$$

$$= \frac{1}{2}\left(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|\right) \tag{2.30}$$

$$= \frac{1}{2}\left(|0\rangle\langle 0| \otimes |0\rangle\langle 0| + |0\rangle\langle 1| \otimes |0\rangle\langle 1| + |1\rangle\langle 0| \otimes |1\rangle\langle 0| + |1\rangle\langle 1| \otimes |1\rangle\langle 1|\right). \tag{2.31}$$

If we now trace out the subsystem of e.g. Bob to get the state of the subsystem of

Alice, we get

$$\begin{aligned}\rho_A &= \mathrm{tr}_B(\rho) && (2.32)\\ &= \frac{1}{2}\left(|0\rangle\langle 0|\underbrace{\langle 0|0\rangle}_{1}+|0\rangle\langle 1|\underbrace{\langle 0|1\rangle}_{0}+|1\rangle\langle 0|\underbrace{\langle 1|0\rangle}_{0}+|1\rangle\langle 1|\underbrace{\langle 1|1\rangle}_{1}\right) && (2.33)\\ &= \frac{1}{2}\left(|0\rangle\langle 0|+|1\rangle\langle 1|\right) && (2.34)\\ &= \frac{1}{2}\mathbb{1}. && (2.35)\end{aligned}$$

So the state ρ_A is a mixed state where the probability that the system is in state $|0\rangle$ is exactly $\frac{1}{2}$. No measurement on this state can reveal any information about Bob's subsystem.

Chapter 3

Entanglement Measures

3.1 Local Operations and Classical Communication

Since most applications of entanglement involve local operations and classical communication (LOCC) it may make sense to assume that those operations are freely available when the question of whether one state is more entangled than the other is addressed. For example one could ask whether the (pure) state $|\psi_1\rangle$ can be turned into $|\psi_2\rangle$ with only LOCC. If it is possible, the state $|\psi_1\rangle$ is considered to have the same or more entanglement than $|\psi_2\rangle$ since it is at least as useful as $|\psi_2\rangle$ for any application.

In the present work, classical communication is usually not regarded as free and it would thus be more appropriate to exclude the ability to communicate in the definition of an entanglement measure. Nevertheless, we use the standard notion of entanglement in the present work, so that the results of this work can be compared directly to previously known results in the field of entanglement distillation research.

3.2 Entropy of Entanglement

For bipartite pure states $|\psi\rangle$ one often uses the so called entropy of entanglement denoted as $E(|\psi\rangle)$ as a measure. It is defined as the Von Neumann entropy $S(\rho_{\psi,A})$ (see e.g. [18, p. 510]) of the density operator that describes one of the subsystems, e.g. Alice's:

$$\rho_{\psi,A} := \mathrm{tr}_B |\psi\rangle\langle\psi|, \tag{3.1}$$

so that

$$E(|\psi\rangle) := S(\rho_{\psi,A}) = -\rho_{\psi,A} \log_2 \rho_{\psi,A}. \tag{3.2}$$

To compute E, one first computes the vector of eigenvalues λ of the density matrix $\rho_{\psi,A}$ and then uses

$$E(|\psi\rangle) = S(\rho_{\psi,A}) = H(\lambda) := -\sum_i^d \lambda_i \log_2 \lambda_i, \tag{3.3}$$

where d is the dimension of the Hilbert space describing Alice's subsystem. The entropy of entanglement cannot be increased by LOCC as can be seen from the discussion in Section 4.3.

3.3 Fidelity and the Fully Entangled Fraction

The term fidelity is used for the probability for a state described by the density operator ρ to yield a pure state $|\psi_{\text{ref}}\rangle$ when the corresponding system is measured with a set of operators that project on orthogonal subspaces containing the operator $|\psi_{\text{ref}}\rangle\langle\psi_{\text{ref}}|$. According to Formula (2.17), this probability is

$$\varphi_{|\psi_{\text{ref}}\rangle}(\rho) := \operatorname{tr}(|\psi_{\text{ref}}\rangle\langle\psi_{\text{ref}}|\rho) = \langle\psi_{\text{ref}}|\rho|\psi_{\text{ref}}\rangle. \tag{3.4}$$

In the present work, the symbol $\varphi_{|\psi_{\text{ref}}\rangle}(\rho)$ is used to denote the fidelity of state ρ with respect to the reference state $|\psi_{\text{ref}}\rangle$. If the subscript of φ is omitted, the fidelity with respect to the Bell state $|\psi^+\rangle$ is meant.

The fully entangled fraction $F(\rho)$ [5] is defined to be the maximal fidelity with respect to a maximally entangled state that one can get if local unitary operations are allowed to be performed before the (hypothetical) measurement. This entanglement measure is known to be well suited for the distinction of separable states from entangled states [21] and is very useful in the context of optimization of entanglement distillation protocols as will become clear in Chapter 5. Moreover, the fully entangled fraction is related to protocols that use a Bell state as a resource so that the fully entangled fractions determines the best achievable error rate of the quantum algorithm. Mathematically, the definition of the fully entangled fraction is

Definition 3.3.1. *The following equation defines the fully entangled fraction $F(\cdot)$ of a state described by the density operator ρ of a bipartite two-qubit system represented by the four-dimensional Hilbert space $\mathscr{H}_A \otimes \mathscr{H}_B$, where $\mathscr{H}_A$ and $\mathscr{H}_B$ are two-dimensional Hilbert spaces.*

$$F(\rho) = \max_{|\phi\rangle:m.e.} \langle\phi|\rho|\phi\rangle, \tag{3.5}$$

where $|\phi\rangle$ denotes any maximally entangled state.

For bipartite systems, Definition 3.3.1 can also be written in terms of unitary operations as follows

$$F(\rho) = \max_{U_A,U_B} \langle\phi^+|(U_A \otimes U_B)\rho(U_A^\dagger \otimes U_B^\dagger)|\phi^+\rangle, \tag{3.6}$$

where U_A and U_B denote arbitrary unitary operations on the Hilbert spaces $\mathscr{H}_A$ and $\mathscr{H}_B$ respectively and $|\phi^+\rangle = \frac{1}{\sqrt{2}}(|0+\rangle + |1-\rangle)$. This equivalence holds because in the bipartite case, the maximally entangled states are the Bell states (Section 2.4), which form a complete basis set and can be interconverted to each other by local unitary operations without communication [5].

Also notice that if the system is in a pure state, i.e. $\rho = |\psi\rangle\langle\psi|$, then the fully entangled fraction is simply

$$F(\rho) = \max_{U_A,U_B} |\langle\phi^+|(U_A \otimes U_B)|\psi\rangle|^2. \tag{3.7}$$

In the following, we will derive some properties of the fully entangled fraction that will be used in Chapter 4 in the context of the Schmidt projection method.

Lemma 3.3.1. *Let $\rho = \sum_i c_i\rho_i$ be a linear combination of density matrices ρ_i with $c_i \in [0,1] \subseteq \mathbb{R}$, then:*

$$F(\rho) \leq \sum_i c_i F(\rho_i), \tag{3.8}$$

where $F(\rho)$ is the fully entangled fraction (Definition 3.3.1).

Proof.

$$F(\rho) = \max_{|\phi\rangle\text{: m.e.}} \langle\phi|(\sum_i c_i\rho_i)|\phi\rangle \tag{3.9}$$

$$= \max_{|\phi\rangle\text{: m.e.}} \sum_i c_i\langle\phi|\rho_i|\phi\rangle \tag{3.10}$$

$$\leq \sum_i c_i \max_{|\phi\rangle\text{: m.e.}} \langle\phi|\rho_i|\phi\rangle \tag{3.11}$$

The last equality holds because the coefficients c_i are non-negative and since an increase in the value of any sum over real numbers implies an increase in at least one term over which the sum is taken. □

Note that Theorem 1 in [22] gives a condition under which Equation 3.8 holds with equality, given that the sum on the right hand side is a convex combination of pure states.

Lemma 3.3.2. *Let $\mathscr{H}_A$ and $\mathscr{H}_B$ be two-dimensional Hilbert spaces and let $|\psi\rangle$ be a separable pure state of the combined system, i.e. $\rho := |\psi\rangle\langle\psi|$ with $|\psi\rangle := |\psi_A\rangle \otimes |\psi_B\rangle$ for some states $|\psi_A\rangle \in \mathscr{H}_A$ and $|\psi_B\rangle \in \mathscr{H}_B$. Then, the fully entangled fraction of the state ψ of the combined system is always less than or equal to* 0.5.

Proof. The pure states can be written in general as $|\psi_A\rangle := \alpha_A|0\rangle + \beta_A|1\rangle$ and $|\psi_B\rangle := \alpha_B|+\rangle + \beta_B|-\rangle$ for some complex coefficients $\alpha_A, \alpha_B, \beta_A, \beta_B \in \mathbb{C}$ with $|\alpha_A|^2 + |\beta_A|^2 = 1$ and $|\alpha_B|^2 + |\beta_B|^2 = 1$.

If any unitary operation of the form $U_A \otimes U_B$ is applied on a tensor product state of the form $|\psi_A\rangle \otimes |\psi_B\rangle$, the state of the combined system can still be written as a tensor product:

$$(U_A \otimes U_B)(|\psi_A\rangle \otimes |\psi_B\rangle) = (U_A|\psi_A\rangle \otimes U_B|\psi_B\rangle) = (|\tilde{\psi}_A\rangle \otimes |\tilde{\psi}_B\rangle). \tag{3.12}$$

with $|\tilde{\psi}_A\rangle := U_A|\psi_A\rangle = \tilde{\alpha}_A|0\rangle + \tilde{\beta}_A|1\rangle$ and $|\tilde{\psi}_B\rangle := U_B|\psi_B\rangle = \tilde{\alpha}_B|+\rangle + \tilde{\beta}_B|-\rangle$.

With the above definitions, the state of the combined system after application of any local unitary operations is

$$|\tilde{\psi}_A\rangle \otimes |\tilde{\psi}_B\rangle = \tilde{\alpha}_A\tilde{\alpha}_B|0+\rangle + \tilde{\alpha}_A\tilde{\beta}_B|0-\rangle + \tilde{\beta}_A\tilde{\alpha}_B|1+\rangle + \tilde{\beta}_A\tilde{\beta}_B|1-\rangle \tag{3.13}$$

.

Therefore we have

$$|\langle\phi^+|(|\tilde{\psi}_A\rangle \otimes |\tilde{\psi}_B\rangle)|^2 = |\left(\frac{1}{\sqrt{2}}(\langle 0+| + \langle 1-|)\right)(|\tilde{\psi}_A\rangle \otimes |\tilde{\psi}_B\rangle)|^2 \tag{3.14}$$

$$= \frac{1}{2}|\tilde{\alpha}_A\tilde{\alpha}_B + \tilde{\beta}_A\tilde{\beta}_B|^2 \tag{3.15}$$

$$\leq \frac{1}{2}(|\tilde{\alpha}_A\tilde{\alpha}_B|^2 + |\tilde{\beta}_A\tilde{\beta}_B|^2) \qquad \text{subadditivity} \tag{3.16}$$

$$= \frac{1}{2}\left(|\tilde{\alpha}_A\tilde{\alpha}_B|^2 + (1-|\tilde{\alpha}_A|^2)(1-|\tilde{\alpha}_B|^2)\right) \tag{3.17}$$

$$= \frac{1}{2}\left(2|\tilde{\alpha}_A\tilde{\alpha}_B|^2 - |\tilde{\alpha}_A|^2 - |\tilde{\alpha}_B|^2 + 1\right) \tag{3.18}$$

$$= \frac{1}{2}\underbrace{\left(\underbrace{|\tilde{\alpha}_A|^2}_{\geq 0}\underbrace{(|\tilde{\alpha}_B|^2-1)}_{\leq 0} + \underbrace{|\tilde{\alpha}_B|^2}_{\geq 0}\underbrace{(|\tilde{\alpha}_A|^2-1)}_{\leq 0} + 1\right)}_{\leq 1} \tag{3.19}$$

$$\leq \frac{1}{2}. \tag{3.20}$$

Since in the above derivation we allow any unitary operation of the form $U_A \otimes U_B$, the fully entangled fraction has to fulfil

$$F\left[(|\psi_A\rangle \otimes |\psi_B\rangle)(\langle\psi_A| \otimes \langle\psi_B|)\right] \leq \frac{1}{2}. \tag{3.21}$$

□

Lemma 3.3.3. *Let $\mathscr{H}_A$ and $\mathscr{H}_B$ be two-dimensional Hilbert spaces and let ρ be a separable mixed state of the combined system, i.e. $\rho := \rho_A \otimes \rho_B$ for some density operators ρ_A on $\mathscr{H}_A$ and ρ_B on $\mathscr{H}_B$. Then, the fully entangled fraction of ρ is always less than or equal to* 0.5.

Proof. By definition, the density matrices ρ_A and ρ_B can be written as

$$\rho_A := \sum_i^n p_{A,i}|\psi_{A,i}\rangle\langle\psi_{A,i}| \quad \text{and} \tag{3.22}$$

$$\rho_B := \sum_i^n p_{B,i}|\psi_{B,i}\rangle\langle\psi_{B,i}|, \tag{3.23}$$

where each sum goes over an ensemble of n pure states $|\psi_{A,i}\rangle$ and $|\psi_{B,i}\rangle$ of the systems A and B respectively. For these ensembles, $p_{A,i}$ and $p_{B,i}$ denote the probabilities with which the systems A and B are in state $|\psi_{A,i}\rangle$ and $|\psi_{B,i}\rangle$ respectively. With these definitions, we have

$$\begin{aligned}
F(\rho) &:= F(\rho_A \otimes \rho_B) && (3.24)\\
&= F\left(\left(\sum_i^n p_{A,i}|\psi_{A,i}\rangle\langle\psi_{A,i}|\right) \otimes \left(\sum_i^n p_{B,i}|\psi_{B,i}\rangle\langle\psi_{B,i}|\right)\right) && (3.25)\\
&= F\left(\sum_i^n\sum_j^n p_{A,i}p_{B,j}(|\psi_{A,i}\rangle \otimes |\psi_{B,j}\rangle)(\langle\psi_{A,i}| \otimes \langle\psi_{B,j}|)\right) && (3.26)\\
&\leq \sum_i^n\sum_j^n p_{A,i}p_{B,j}F\left((|\psi_{A,i}\rangle \otimes |\psi_{B,j}\rangle)(\langle\psi_{A,i}| \otimes \langle\psi_{B,j}|)\right) && \text{by Lemma3.3.1}\\
& && (3.27)\\
&\leq \sum_i^n\sum_j^n p_{A,i}p_{B,j}\frac{1}{2} = \frac{1}{2}\sum_i^n p_{A,i}\sum_j^n p_{B,j} = \frac{1}{2} && \text{by Lemma3.3.2.}\\
& && (3.28)
\end{aligned}$$

□

Chapter 4

Entanglement Distillation Protocols

Given a certain number n of copies of a particular non-separable state, the goal of Entanglement Distillation is to obtain a number m of (close to) perfectly entangled states as high as possible using only local operations and, if needed, classical communication. For states that are not perfectly entangled, the best achievable fraction m/n has to be smaller than 1. This is true by definition: any state that can be converted to a maximally entangled state by LOCC is by itself maximally entangled (See Section 3.1). Furthermore, the optimal fraction m/n is non-increasing under LOCC since if it could be increased it would not be optimal. Thus, the optimal fraction m/n serves as an entanglement measure, which is usually called the distillable entanglement E_D.

In the context of optimization (see Chapter 5) and in the case when finite n is considered, it may also be useful to ask the modified question of how much of the initial entanglement can be concentrated in a certain number m of qubits (where $m < n$) instead of requiring the m output states to be perfectly entangled. This is the scenario we will consider most in this work.

Table C.1 summarizes the properties of some previously known entanglement distillation protocols.

4.1 Single Pair Purification

Protocols exist, that act on a single pair of qubits distributed between Alice and Bob. Examples for such protocols are the Procrustean method, entanglement swapping and local filtering (see Table C.1). These protocols rely on the fact that with a certain measurement procedure one can project the state of the two-qubit system onto a maximally entangled state. This projection belongs to a particular measurement outcome that only occurs with a certain probability. Therefore Alice and Bob have to "post-select" the maximally entangled states, which means that they discard their qubits if the measurement outcomes differ from the one that corresponds to the maximally entangled state. Such a protocol can be executed by local operations and classical communication (LOCC). Since every two states that can be interconverted by LOCC to each other with certainty are considered to be equally entangled, it follows that whenever the probability p_{succ} with which a maximally entangled state is obtained by such a single pair purification protocol is smaller than one, the input state has to be less than maximally entangled. This suggests the usefulness of the best achievable p_{succ} as an entanglement measure. This (non-additive) entanglement measure is called the entanglement

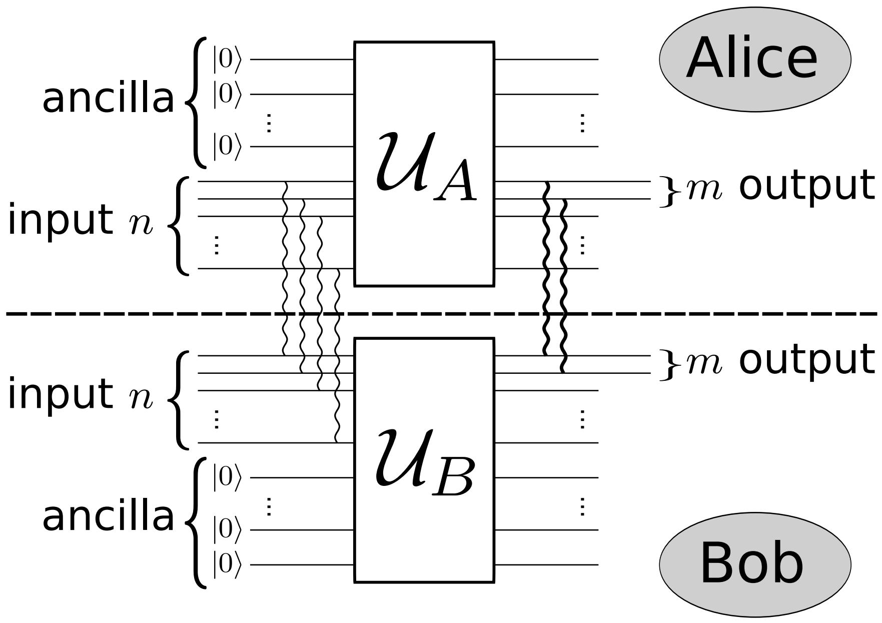

Figure 4.1: General scheme of an entanglement protocol without communication. Entangled Pairs of qubits are connected by a wavy line. The initial entanglement of the n input qubit pairs shared by Alice and Bob is concentrated into m pairs of stronger entanglement by the unitary operations $\mathscr{U}_A$ and $\mathscr{U}_B$. The ancilla qubits can be used to perform irreversible operations such as measurements on the n input qubits as part of these unitary operations.

of single pair purification [23] usually denoted as E_s.

4.2 Protocols without communication

Figure 4.1 shows the general scheme of an entanglement protocol without communication. The entanglement protocol is completely determined by the two unitary operations $\mathscr{U}_A$ and $\mathscr{U}_B$. In order to be able to perform irreversible operations such as measurements on the n qubits, a number of ancilla qubits can be added.

Among the protocols shown in Table C.1, there are two protocols that work without communication: the Schmidt projection method by Bennet et al [12] for pure states and the No-communication Random Permutation Protocol by Ambainis and Yang [8] for mixed states. The reader may wonder why such protocols can exist at all. In the case of the Schmidt projection method this is possible because Alice and Bob can gain some information about their state by an incomplete measurement on the n input qubits and use this information to concentrate the entanglement into a number $m < n$ qubits. In the Random Permutation Protocol, Alice and Bob make use of shared source of random numbers (public coin) and some initially shared perfectly entangled states. The improvement of the fidelity with the Random Permutation Protocol is very limited.

4.3 General Limitations and Optimality

The protocols shown in Table C.1 give lower bounds on the distillable entanglement of the states they act on. There also exist several upper bounds on distillable entanglement. First of all since entanglement protocols are a special case of general entanglement transformations, all limits known to general entanglement transformations apply also for entanglement protocols. As shown in [24], a pure state $|\psi\rangle$ can be transformed into another pure state $|\phi\rangle$ if and only if

$$\lambda_\psi \prec \lambda_\phi, \tag{4.1}$$

where λ_ψ and λ_ϕ are the vectors of eigenvalues of $\rho_{\psi,A} = \mathrm{tr}_B|\psi\rangle\langle\psi|$ and $\rho_{\phi,A} = \mathrm{tr}_B|\phi\rangle\langle\phi|$. The relation $\prec$ means that λ_ψ is majorized by λ_ϕ, i.e. $\sum_i(\lambda_\psi^\downarrow)_i \leq \sum_i(\lambda_\phi^\downarrow)_i$, where $\lambda_\psi^\downarrow$ $(\lambda_\phi^\downarrow)$ have the same elements as λ_ψ (λ_ϕ) but are sorted decreasingly. From the above necessary and sufficient condition for entanglement conversion under LOCC, it is easy to derive that the entropy of entanglement cannot be increased under LOCC. The majorization condition is also discussed in [18, pp. 573–581], where they also pose an exercise to prove a similar condition for entanglement transformation of pure states without the ability to communicate, namely

Theorem 4.3.1. *(Exercise 12.22 from [18, p. 579]) A pure state $|\psi\rangle$ can be transformed into another pure state $|\phi\rangle$ without communication if and only if*

$$\lambda_\psi \cong \lambda_\phi \otimes \mathbf{x}, \tag{4.2}$$

where $\mathbf{x}$ is a vector of non-negative elements summing up to 1, λ_ψ and λ_ϕ are the vectors of eigenvalues of $\rho_{\psi,A} = \mathrm{tr}_B|\psi\rangle\langle\psi|$ and $\rho_{\phi,A} = \mathrm{tr}_B|\phi\rangle\langle\phi|$. The relation $\cong$ means that the vectors λ_ψ and $\lambda_\phi \otimes \mathbf{x}$ have the same elements but possibly with different order.

The proof of Theorem 4.3.1 is postponed to Appendix A. While 4.3.1 is relatively easy to proof for pure states since we can use the Schmidt decomposition to find the relationship between λ_ψ and λ_ϕ, it becomes more complicated when one or both of the input and output states of the transformation are mixed states. The Scenario involving mixed states is investigated by the numerical optimization described in Chapter 5.

In [13] it is shown that for an arbitrary *pure* state $|\psi\rangle$ with entropy of entanglement E_{init}, the optimal probability of distilling more than nE_{init} maximally entangled states goes to zero as n goes to infinity. On the other hand, the Schmidt projection method (see Table C.1 and Section 4.4) produces a number of $m = nE_{\mathrm{init}}$ maximally entangled states as n approaches infinity. Thus the Schmidt projection method is *asymptotically* optimal. Furthermore, in [13] it is also shown that $m < nE_{\mathrm{init}}$ is a sufficient condition to distill m maximally entangled states out of $n \to \infty$ copies of *any* pure state.

An overview of this and other previously known limits to the existence of entanglement protocols is given in Table C.2. All of these limits apply to entanglement protocols without communication. Those limits that are also valid for protocols with communication are indicated by an asterisk (*) in the column denoted as “comm” in Table C.2.

4.4 A Note On The Schmidt Projection Method For Two Copies

The Schmidt projection method by Bennet et al [12] is an entanglement distillation method for pure states that works without communication. The procedure is as follows: first, an incomplete measurement is performed to project the system into a subspace where the state is maximally entangled if the measurement outcome k is larger than 0 and smaller than n. Then the maximally entangled state is transferred into a standard form such as m copies of a particular Bell state i.e. $(|\phi^+\rangle)^{\otimes m}$.

As pointed out in [12], this procedure only becomes efficient for a large number of copies n. Nevertheless, there is a non-zero probability of obtaining a maximally entangled state even for $n = 2$ copies. In the context of the optimization technique described in Chapter 5, the question arose whether it is possible to concentrate enough entanglement into the subsystem of the first pair of qubits, so that the fully entangled fraction of the mixed state that results from application of the protocol is higher than for one of two copies of a given input state. I.e.:

$$F(\mathrm{tr}_{A_2}\,\mathrm{tr}_{B_2}[\mathscr{E}(|\psi_{\mathrm{input}}\rangle \otimes |\psi_{\mathrm{input}}\rangle)]) \overset{?}{>} F(|\psi_{\mathrm{input}}\rangle), \tag{4.3}$$

where $\mathscr{E}$ denotes the Schmidt projection method that takes a pure state as input.

For 2 copies, the answer to this question is negative:

Theorem 4.4.1. *The Schmidt projection method, if applied on two copies of a pure initial state, does not lead to a state with higher fully entangled fraction than the initial state if no post-selection is allowed.*

Note that this result does not mean the Schmidt projection method is not useful at all for entanglement distillation with two copies of an imperfectly entangled state. There is a non-zero probability of obtaining a measurement result from that one knows that the post-measurement state is a perfectly entangled pure state. It just means that it is necessary to select those outputs of the protocol that have been successful and throw away those that were not successful (post-selection).

Theorem 4.4.1, which is proven in Appendix B, leads to the question whether it is possible at all to increase the fully entangled fraction, if only two copies of a pure state are given as input to any entanglement distillation protocol without communication and without post-selection. This question turns out to be hard to answer analytically, but it is addressed by numerical optimization in Chapter 5.

Interestingly, it is possible to increase the fully entangled fraction of the first pair of qubits if $n = 3$ copies of $|\psi\rangle$ are given. It can be done by the following procedure:

Protocol 4.4.1. *(Schmidt projection method for 3 copies of a pure state)*

1. *Apply the Schmidt projection method on the first two copies.*
2. *If the method was not successful in obtaining a maximally entangled state, flip the third copy of $|\psi\rangle$ with the first pair of qubits (conditional swap). If necessary, apply the local unitary transformations on the first qubit in Alice's and Bob's location such that the fidelity of the first pair of qubits w.r.t. a maximally entangled state equals the fully entangled fraction of the initial state.*

Claim 4.4.1. *Given 3 copies of a pure input state $|\psi_{input}\rangle = c|0+\rangle + s|1-\rangle$, $c,s \in \mathbb{R}$* [1] *that has a fully entangled fraction $F(|\psi_{input}\rangle) = \frac{1}{2}(c+s)^2$, the above described protocol produces a mixed state with a fidelity of $\frac{1}{2}(1-2c^2s^2)(c+s)^2 + 2c^2s^2 > F(|\psi_{input}\rangle)$ if the initial state $|\psi_{input}\rangle$ was entangled, i.e. $c > 0, s > 0$.*

Proof. Since the Schmidt projection method applied on two copies of $|\psi_{\mathrm{input}}\rangle$ produces a maximally entangled state with probability $2c^2s^2$ (see [12]), and in the other case the protocol swaps the output of the Schmidt projection method with the untouched third copy of the initial state, the resulting mixed state is

$$\rho_{\mathrm{out}} = 2c^2s^2|\phi^+\rangle\langle\phi^+| + (1-2c^2s^2)|\psi_{\mathrm{input}}\rangle\langle\psi_{\mathrm{input}}|. \tag{4.4}$$

[1]By the Schmidt decomposition every pure state of two qubits can be brought into this form if the computational basis is chosen appropriately (see e.g. [18, p. 109]).

The fidelity with respect to the Bell state $|\phi^+\rangle$ of the resulting state is:

$$\begin{aligned}\langle\phi^+|\rho_{\text{out}}|\phi^+\rangle &= 2c^2s^2 + (1-2c^2s^2)F(|\psi_{\text{input}}\rangle\langle\psi_{\text{input}}|) && (4.5)\\ &= 2c^2s^2 + (1-2c^2s^2)(c+s)^2 && (4.6)\end{aligned}$$

This quantity is larger than $F(|\psi_{\text{input}}\rangle)$ if $c, s > 0$ and $c \neq s$. $\square$

Note that Protocol 4.4.1 can also be used to distill entanglement from the following mixed state:

$$\rho_{\text{input}} = \sum_{i=1}^{k} p_i \left(|\psi_i\rangle\langle\psi_i|\right)^{\otimes 3}, \tag{4.7}$$

where $|\psi_i\rangle = c_i|0+\rangle + s_i|1-\rangle$ with $c_i^2 + s_i^2 = 1$ and $0 < c_i < s_i$ for $i = 1, \cdots, k$. And $p_i > 0$, $i = 1, \cdots, k$ with $\sum_{i=1}^{k} p_i = 1$. This can be easily seen since by Lemma 3.3.1 the fully entangled fraction is less than the weighted sum of the fully entangled fractions of $|\psi_i\rangle$, $i = 1, \cdots k$ with weights p_i. Furthermore, the Schmidt projection protocol $\mathscr{E}$ acts linearly on ρ_{input} such that

$$\mathscr{E}(\rho_{\text{input}}) = \sum_{i=1}^{k} p_i \mathscr{E}\left(\left(|\psi_i\rangle\langle\psi_i|\right)^{\otimes 3}\right). \tag{4.8}$$

Let the output of the protocol be only the first pair of qubits, i.e. tracing out the second and third pair,

$$\rho_{\text{output}} = \text{tr}_{A_{2,3},B_{2,3}}[\mathscr{E}(\rho_{\text{input}})] \tag{4.9}$$

Thus, the fidelity of the first pair of qubits of the output state ρ_{output} will be

$$\langle\phi^+|\rho_{\text{output}}|\phi^+\rangle = \sum_{i=1}^{k} p_i \varphi_i, \tag{4.10}$$

where φ_i is the fidelity one obtains by applying Protocol 4.4.1 on 3 copies of the pure input state $|\psi_i\rangle$ that is by Claim 4.4.1 larger than the fully entangled fraction of $|\psi_i\rangle$. It follows that the fully entangled fraction of ρ_{output} must be larger than the fully entangled fraction of ρ_{input}.

Chapter 5

Numerical Optimization

The task of finding deterministic protocols for entanglement distillation of an arbitrary bipartite state without communication (see Section 4.2) is addressed by the following optimization problem:

Problem 5.0.1. *(Optimization Problem)*

> *Given the input state $\rho_{input} \in \mathscr{H}_{2^{2n}}$, find $\mathscr{U}_A \in \mathbb{C}^{n\times n}$ and $\mathscr{U}_B \in \mathbb{C}^{n\times n}$ which minimize*
>
> $$f(\mathscr{U}_A, \mathscr{U}_B) = 1 - \langle \phi^+ | \operatorname{tr}_{S_A} \operatorname{tr}_{S_B} [(\mathscr{U}_A \otimes \mathscr{U}_B)\rho_{input}(\mathscr{U}_A^\dagger \otimes \mathscr{U}_B^\dagger)] | \phi^+ \rangle, \quad (5.1)$$
>
> *where* tr_{S_A} (tr_{S_B}) *denotes the partial trace over the subsystem of Alice (Bob) without her (his) first Qubit,*
>
> *subject to the constraint of $\mathscr{U}_A, \mathscr{U}_B \in \mathbb{C}^{2^n\times 2^n}$ being unitary matrices, i.e.*
>
> $$\mathscr{U}_A^\dagger \mathscr{U}_A = \mathscr{U}_A \mathscr{U}_A^\dagger = \mathscr{U}_B^\dagger \mathscr{U}_B = \mathscr{U}_B \mathscr{U}_B^\dagger = \mathbb{1}. \quad (5.2)$$

The Optimization Problem 5.0.1 can be converted into an unconstrained problem by an appropriate parameterization of the unitary matrices $\mathscr{U}_A$ and $\mathscr{U}_B$ (see Section 5.1). A difficulty with this approach is, that parameterizations of unitary matrices are always non-linear and cannot be written as polynomials with a finite number of coefficients. Therefore, this approach requires an efficient algorithm to perform unconstrained non-linear optimization (non-linear programming). Several algorithms exist to perform local non-linear optimization, e.g. Newton's method (see e.g. [25]). In contrast, no algorithm is known, that deterministically performs global optimization of a general non-linear continuous objective function of several variables. In order to circumvent this difficulty, several heuristics can be employed such as starting the Local optimization with a number of randomly distributed initial search points. A class of heuristics that has been quite successful in many applications is the so called evolutionary algorithms. Due to the stochastic nature of evolutionary algorithms, they are well suited for objective functions with a large number of parameters. This work investigates the use of CMA-ES (Covariance Matrix Adaptation - Evolution Strategy) [9], which is a modern evolutionary algorithm for continuous objective functions (Section 5.2). As test cases serve the problems of finding protocols to distill entanglement from 3 or 2 copies of a pure input state in Section 5.3 and 5.4 respectively and from 3 copies of a mixed input state in Section 5.5.

As an alternative approach one could try to directly solve the Optimization Problem 5.0.1. The real valued objective function $f(\mathscr{U}_A, \mathscr{U}_B)$ is a positive polynomial of fourth order with complex coefficients and complex-valued variables, which are the elements of the unitary matrices in its argument. The constraints are equalities on positive polynomials of second order (also with complex variables and coefficients). Those polynomials can be converted into polynomials of eighth and forth order respectively in the real valued elements of the real and imaginary parts of the unitary matrices by substituting the complex numbers by $z = a + ib$, for $a, b \in \mathbb{R}$ and analytically performing the matrix-vector multiplications. Lasserre showed in [26] that polynomial optimization problems can be approximated by a sequence of convex linear matrix inequality (LMI) problems. Furthermore, Waki et. al. [27] proposed how to handle equality constraints and provided a practical implementation of their algorithm [28]. The treatment of Optimization Problem 5.0.1 by means of polynomial optimization is not further investigated in the present work but proposed as subject for future research.

5.1 Parameterization of Unitary Matrices

The goal of this section is to find a parameterization

$$\mathscr{P} : \mathbb{R}^k \to U(N); \mathbf{x} \in \mathbb{R}^k \mapsto \mathbf{U} = \mathscr{P}(\mathbf{x}) \in U(N), \tag{5.3}$$

where $U(N)$ denotes the group of N-dimensional unitary matrices i.e.

$$U(N) := \{\mathbf{U} \in \mathbb{C}^{N\times N} \mid \mathbf{U}^\dagger \mathbf{U} = \mathbf{U}\mathbf{U}^\dagger = \mathbb{1}\}. \tag{5.4}$$

It is known from group theory [29], that there exists the exponential map

$$\exp : u(N) \to U(N); \mathbf{X} \in u(N) \mapsto \mathbf{U} := e^{\mathbf{X}} \in U(N), \tag{5.5}$$

$u(N)$ denotes the corresponding N-dimensional algebra $u(N) = \{\mathbf{X} \in \mathbb{C}^{N\times N} \mid \mathbf{X}^\dagger = -\mathbf{X}\}$ and $e^{\mathbf{X}}$ is the matrix exponential. This exponential map can also be written as

$$\mathbf{U} = e^{i\mathbf{H}}, \tag{5.6}$$

where $\mathbf{H} = \mathbf{H}^\dagger$ is a hermitean matrix.

Therefore, a simple parameterization of unitary operations is obtained by parameterizing hermitean matrices $\mathbf{H}$. This can be done by mapping $\frac{1}{2}N(N+1)$ real parameters to the elements of a symmetric matrix $\mathbf{S}$ and $\frac{1}{2}N(N-1)$ parameters to the elements of a skew-symmetric matrix $\mathbf{K}$ such that $\mathbf{H} = \mathbf{S} + i\mathbf{K}$. In total this parameterization has $k = N^2$ parameters.

The above described parameterization has a major drawback: the matrix exponential is expensive to compute. Fortunately, there is a divide-and-conquer algorithm found by C. Jarlskog [30, 29] that computes the exponential map (5.5) using $\mathscr{O}(N^2)$ elementary operations.

Since the computing time an optimization algorithm needs to converge to the optimum strongly depends on the number of parameters k and the number of dimensions N grows exponentially with the number of Qubits, it is sometimes useful to restrict the optimization to a subset of all possible solutions. In the case of unitary matrices one can restrict the search to the subset of orthogonal matrices that only have $N(N-1)$ degrees of freedom in addition to one parameter (the determinant) that can be either plus or minus one. A recursive parameterization of orthogonal matrices of determinant +1 or -1 that is efficient to compute has been found by R. Raffenetti and K. Ruedenberg [31]. The parameterization of orthogonal matrices has the advantage, that the derivatives of the matrix elements with respect to the parameters can be computed and it can be efficiently inverted [32]. The disadvantage is, that the sign of the determinant is an additional binary parameter.

5.2 Black-Box Optimization With CMA-ES

In the following, the CMA-ES optimization algorithm [9] is introduced, that has been used to obtain the results presented in Sections 5.3, 5.4 and 5.5. A detailed discussion of CMA-ES is given in [33]. The algorithm was designed to solve a general multivariate single-objective optimization problem with the goal that the user of the algorithm a-priori needs very little knowledge about the objective function. This kind of treatment is called "black-box" optimization (BBO).

The origin of CMA-ES is the so called evolution strategy that was developed in the early 1970's by Rechenberg and Schwefel in their Ph.D. theses [34, 35]. As for the evolution strategy, CMA-ES maintains a set of strategy parameters that describe how to sample a number λ of guesses of the solution to the optimization problem, i.e. the vector that minimizes the objective function. Following the idea and terminology of biological evolution theory, these guesses are called "children", "offspring" or simply "population". For each of these guesses, the objective function is evaluated to get a so called fitness value. Using the ranking based on the objective function value, the μ best candidate solutions ("parents") are selected. To conclude one iteration (also called "generation"), the strategy parameters are updated by machine learning based on the properties of the selected parent solutions as well as information about the previous iterations. This last step of the iteration is called self-adaptation. The entire procedure is repeated many times so that a larger and larger part of the space of possible solution vectors is explored.

In CMA-ES, the λ offspring $\mathbf{x}_k^{(g)}$ of generation g are sampled from a multivariate normal distribution: [33]

$$\mathbf{x}_k^{(g)} \sim \mathcal{N}\left(\mathbf{m}^{(g)}, (\sigma^{(g)})^2\mathbf{C}^{(g)}\right) \qquad \text{for } k = 1, \cdots, \lambda. \tag{5.7}$$

The strategy parameters are the mean vector $\mathbf{m}^{(g)}$, the covariance matrix $\mathbf{C}^{(g)}$ and a scalar $\sigma^{(g)}$ that scales the covariance matrix and is thus called "step length". The mean vector is updated by a weighted average over the μ selected parents. The default setting for the number of parents is $\mu = \frac{1}{2}\lambda$. It is reasoned in [33] that this is a good choice if the weights are chosen to be linearly decreasing from the parent with the best fitness value to the one among the μ selected parents with the worst fitness.

In order to store information about previous iterations that can be used to update the covariance matrix $\mathbf{C}$ and step length σ, there are two additional strategy parameters, the so called cumulated evolution paths $\mathbf{p}_c^{(g)}$ and $\mathbf{p}_\sigma^{(g)}$. These evolution paths are weighted sums over the difference between the mean vectors $\mathbf{m}^{(g)}$ and $\mathbf{m}^{(g+1)}$ of two consecutive iterations (referred to as "steps"). The weights are chosen to be exponentially fading, so that the most recent step gains the highest weight. The evolution paths can be calculated by update rules given in [33]. In the update of $\mathbf{p}_c^{(g+1)}$, the effect of the global step size $\sigma^{(g)}$ on the evolution path is undone, so that this vector only reflects the direction from the mean vector $\mathbf{m}$ of the previous iteration to the one of the present iteration. The principal components of the covariance matrix are then adapted to $\mathbf{p}_c^{(g+1)}$. This update is usually combined with the so called rank-μ update [36] that takes into account the differences between the strategy mean vector $\mathbf{m}^{(g)}$ and each of the μ best offspring of generation g separately. Using the rank-μ update, more information about the every single individual is incorporated into the update of the covariance matrix so that larger population sizes will lead to faster learning of the covariance matrix.

On the other hand, $\mathbf{p}_\sigma^{(g+1)}$ is calculated by transforming the steps with $\mathbf{C}^{-1/2}$ in order to negate the effect of the covariance matrix. The magnitude of $\mathbf{p}_\sigma$ is compared to an analytically calculated expectation value under the assumption that the μ best offspring are distributed around the strategy mean vector $\mathbf{m}^{(g)}$ by the multivariate normal distribution $\mathscr{N}(\mathbf{m}^{(g)}, \mathbb{1})$, where $\mathbb{1}$ denotes the identity matrix. If the actual evolution path $\mathbf{p}_\sigma^{(g+1)}$ is longer (shorter) than the expected length, this means that the step length $\sigma^{(g)}$ was too short (long) and thus will be increased (decreased) for the next iteration.

As explained in [9], the adaptation of the covariance matrix improves the probability for the algorithm to find the global optimum compared to traditional evolution strategies. Namely, the global search property largely depends on the step size, which is adjusted by the self-adaptation part of CMA-ES. A large number of offspring λ should also make the search more global since it increases the density of points that are evaluated in the search space.

5.3 Results for 3 copies of a pure initial state with 2 ancilla qubits per party

We want to find entanglement distillation protocols with input given by three copies of the pure state

$$|\psi_{\text{input}}\rangle = \sqrt{0.1}|0+\rangle + \sqrt{0.9}|1-\rangle \tag{5.8}$$

together with two ancilla qubits on Alice's and Bob's location. Thus, the input state that determines Optimization Problem 5.0.1 is

$$\rho_{\text{input}} = (|\psi_{\text{input}}\rangle\langle\psi_{\text{input}}|)^{\otimes 3} \otimes |00\rangle\langle 00| \otimes |++\rangle\langle ++| \tag{5.9}$$

One copy of the input state $|\psi_{input}\rangle$ has a fidelity of 0.8. For this problem, we know that we can achieve a fidelity of 0.836 by Protocol 4.4.1, which corresponds to an objective function value of 0.164 provided that Alice and Bob each have two ancilla qubits to do the Schmidt projection measurement. So this problem can serve as a test to evaluate the suitability of the optimization algorithm to find new entanglement distillation protocols.

5.3.1 Optimization over a Single Orthogonal Matrix

In order to reduce the dimensionality of the optimization problem, the number of parameters have been reduced by only optimizing over orthogonal matrices (see Section 5.1) and by setting $\mathscr{U}_B \overset{!}{=} \mathscr{U}_A$ ("symmetric distillation"). Note that Protocol 4.4.1 leads to an orthogonal matrix and is a case of symmetric distillation.

Under these restrictions, the operation of one party on the three local copies and on two ancilla qubits, is represented by an orthogonal matrix of dimension $d = 2^{3+2} = 32$. The corresponding parameterization has $\frac{1}{2}d(d-1)+1 = 497$ parameters. This number includes one additional parameter, which has been chosen to be the first element of the parameter vector. The sign of this parameter determines the sign of the determinant, whereas the magnitude of the parameter has no effect on the objective function value.

The optimization has been started multiple times, each time with a different initial search point and a different seed of the random number generator.

Interestingly, there were several runs of the optimization that converged to an objective function value of 0.164. This objective function value corresponds to an end state with a fidelity of 0.836 that equals the fully entangled fraction of the output state that is produced by Protocol 4.4.1. Since the fidelity of the output state is higher than the initial fully entangled fraction of 0.8, we conclude that it is possible to find entanglement distillation protocols by the presented optimization method. The optimization runs that resulted in a protocol that achieves a higher fidelity than the initial fully entangled fraction will be called successful optimization runs.

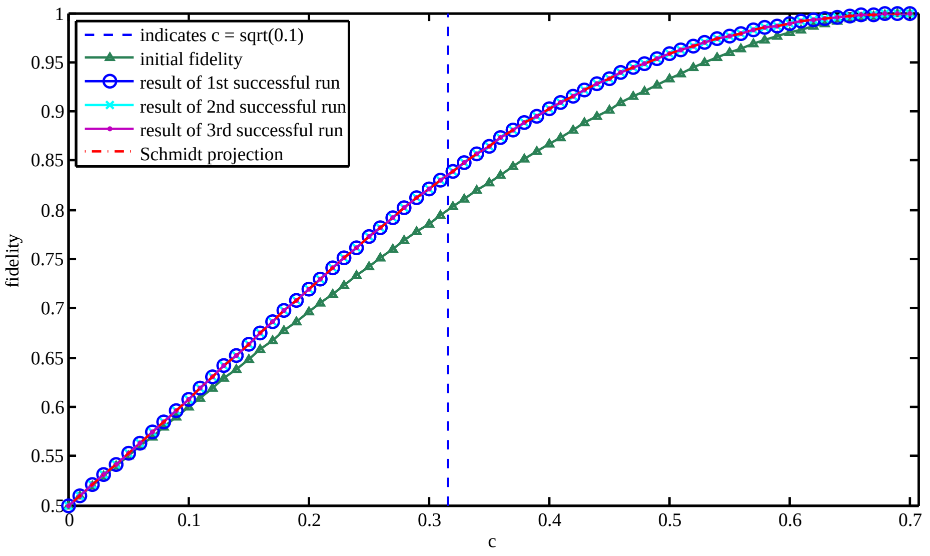

Figure 5.1: The fidelity of the state one obtains by applying different protocols on three copies of the input state $|\psi\rangle = c|0+\rangle + \sqrt{1-c^2}|1-\rangle$ for different values of c. For comparison, the fidelity of the input state is indicated by the green line with triangles. It can be seen, that all protocols lead to a state of the same fidelity. Three of those protocols have been obtained by numerical optimization using the CMA-ES optimization algorithm. They are referred to as first, second and third successful run. By the term 'Schmidt projection' in the legend, Protocol 4.4.1 is meant. The vertical dashed line indicates $c = \sqrt{0.1}$ that corresponds to the input state for which these protocols have been optimized in the three successful optimization runs.

The initial search point of the first and second successful run has been chosen by sampling from a Gaussian distribution around the zero vector with a standard deviation of 0.0814. By the parameterization of orthogonal matrices, the zero-vector corresponds to the identity matrix. For the third successful run, the initial search point has been chosen to be exatly the zero vector. Each of these runs have been started with an initial step width of $\sigma = 0.01$ that turned out to lead to faster convergence compared with larger step widths. The optimal initial step width is hard to find due to the high computational cost of each optimization run.

Figure 5.1 shows the fidelity that one gets if the protocols found by the three optimization runs are applied on initial states of the form $c|0+\rangle + \sqrt{1-c^2}|1-\rangle$. for different values of c. Keep in mind that the optimization has been performed for an initial sate with $c = 0.1$ (5.8). While the unitary matrices that make up these protocols differ, the final states obtained by these protocols have the same fidelity for every possible input state that equals to the fidelity achieved by Protocol 4.4.1. It can be seen from Figure 5.2, that the density matrices of the end states of the protocols found by the first and third successful run are essentially (up to small deviations) the same as for the end state of Protocol 4.4.1.

While these runs converged to the same objective function value, the determined orthogonal matrices differ between each run. This is illustrated by Figure 5.3.

Furthermore, all of these optimization runs show a slightly different convergence behaviour. It can be seen from Figure 5.4, that the convergence of all optimization runs has stalled near to the initial fitness of 0.2 before the optimization converged to the final fitness value of 0.164 and reached the stopping criteria. The first and the second run needed about 10000 function evaluations until the optimization came close to the final objective function value, whereas in the third run, the fitness value has stalled over a longer period of about 0.9×10^5 function evaluations.

Only those optimization runs were successful, that have been started with an initial

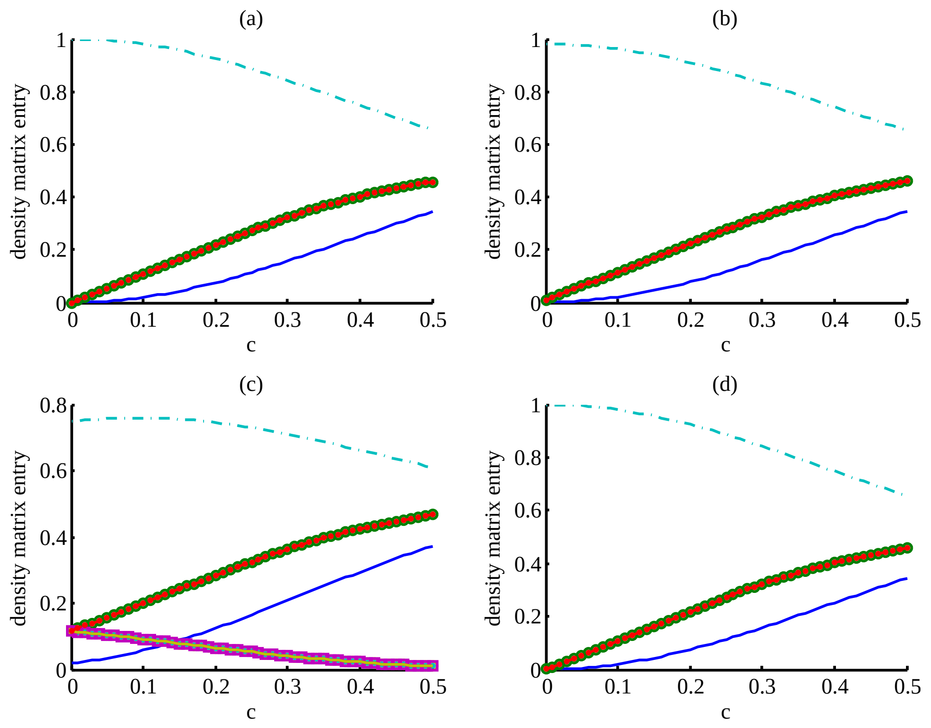

Figure 5.2: Plots of the elements of the density matrices of the output state obtained by applying four different protocols on three copies of the input state $|\psi\rangle = c|0+\rangle + \sqrt{1-c^2}|1-\rangle$ for different values of c. Only those elements of the 4×4 real density matrix that considerably differ from zero are shown. Panel (a): Protocol 4.4.1, Panel (b), (c) and (d): first second and third successful run of the CMA-ES optimization algorithm. Solid blue line: $\rho_{1,1}$, green circles with red dots: $\rho_{4,1}$ and $\rho_{1,4}$, dash-dotted cyan line: $\rho_{4,4}$, magenta squares with brown line in Panel (c): $\rho_{2,2}$, $\rho_{3,2}$, $\rho_{2,3}$ and $\rho_{3,3}$. This figure reveals, that only the protocol of the second successful optimization run leads to a different density matrix than the others.

parameter vector that is close to the zero vector. For those optimization runs for which the elements of the parameter vector have been chosen to be uniformly distributed in the range $[-\pi, +\pi]$, the optimization converged to an objective function value of 0.203. This value corresponds to a fidelity of 0.797, which is above the initial fidelity of 0.8. In some cases the optimization simply converged to a fitness value 0.2 that corresponds exactly to the input fidelity and also could be a local minimum. Table C.3 gives an overview over all optimization runs that have been performed.

A possible explanation for it to be beneficial to choose the initial parameters close to zero may be that due to the angular nature of the parameterization of orthogonal matrices, parameters close to zero or a multiple of π will lead to sparse matrices. As can be seen from Figure 5.3, the optimal orthogonal matrices are semi-sparse in the sense that in each column there is a particular element that has a much larger magnitude (from 2 to 10^5 times larger) than the other non-zero elements of the column.

5.3.2 Unrestricted Optimization over 2 Unitary Matrices

In this section, we present the results of the optimization over two different unitary matrices $\mathscr{U}_A$ and $\mathscr{U}_B$, i.e. without restriction to orthogonal matrices. To do the optimization, the parameterization of unitary matrices presented in Section 5.1 was used. Since each unitary matrix has N^2 degrees of freedom, where $N = 2^n$ is the dimension of the Hilbert space of $n = 5$ qubits, we have 2048 parameters in total.

Table C.4 gives an overview of all optimization runs that have been performed for

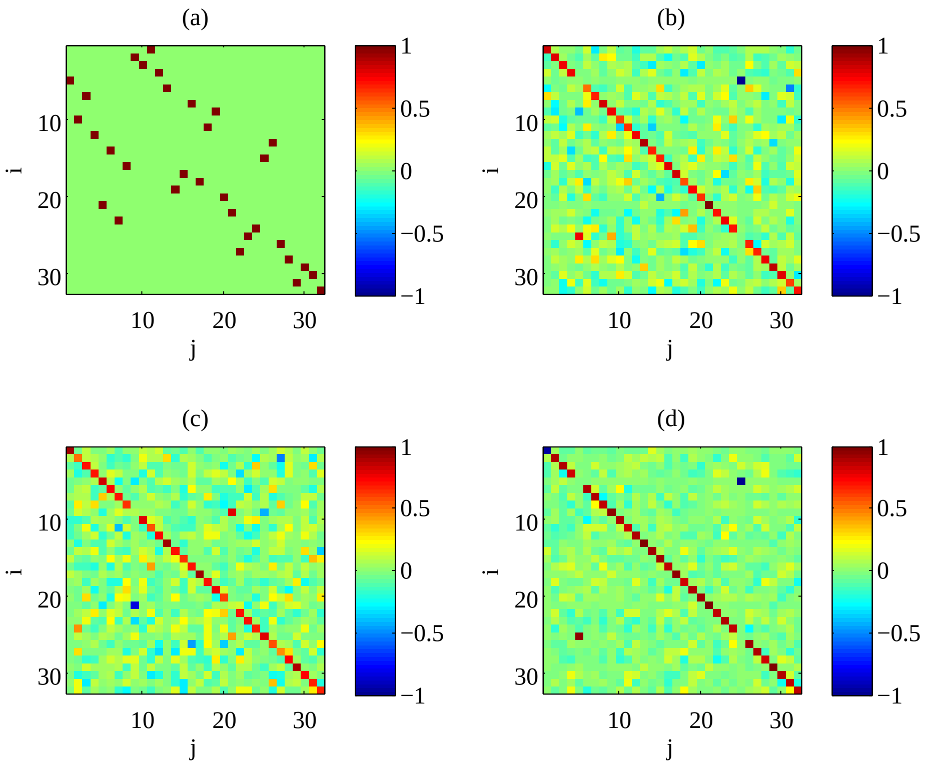

Figure 5.3: Visualization of four different orthogonal 32×32 matrices that all lead to a state with the same increased fully entangled fraction if simultaneously applied by Alice and Bob on their part of 3 copies of an initially entangled pure state together with two local ancilla qubits per party. The color of each pixel indicates the value of the corresponding matrix entry $(\mathbf{U})_{i,j}$. (a) is determined by hand according to Protocol 4.4.1. (b), (c) and (d) are the results of the first, second and third successful run of the optimization with CMA-ES for different initial search points and seeds of the random number generator.

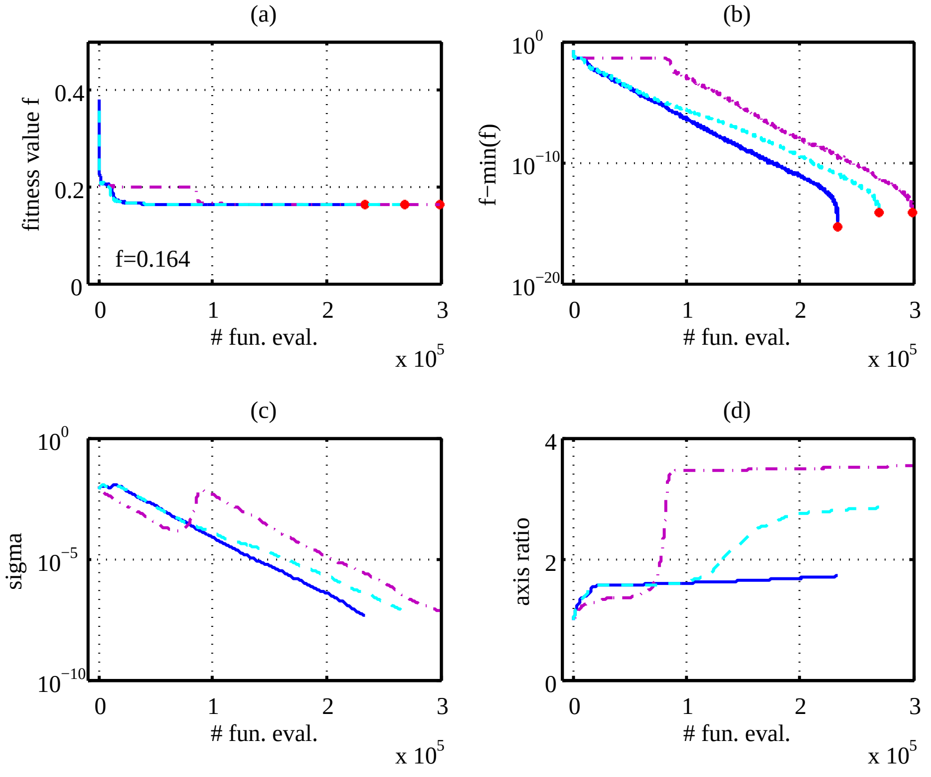

Figure 5.4: Convergence behaviour of three runs of the CMA-ES algorithm when used to find an entanglement distillation protocol without communication for 3 copies of a pure state. The optimization was limited to orthogonal matrices and the operations performed by Alice and Bob have been constrained to be the same. The dimensionality of the search space is 497. The fully entangled fraction of the given input state is 0.8, which corresponds to a fitness value of $f = 0.2$. The best fidelity achieved was 0.836, which corresponds to a fitness value of $f = 0.164$. The blue solid line corresponds to the first successful run, the cyan dashed line to the second and the magenta dashed-dotted line to the third run. The abscissae correspond to the number of objective function evaluations that have been done by the black-box optimization algorithm to reach the corresponding state. Panel (a): the best objective function value found after each iteration. Panel (b): the difference to the final objective function value. Panel (c): the parameter σ of the CMA-ES algorithm, by which the standard deviations of the sampling distribution are scaled (step size). Panel (d): the ratio between the longest and shortest principle axis length of the covariance matrix of the sampling distribution.

the unrestricted optimization. As for the symmetric orthogonal case (Section 5.3.1), it depends on the starting point whether the optimization converges to a protocol that is successful in producing a state below the initial fidelity. But unlike the orthogonal case, the initial parameters do not have to be close to zero. It can be seen from Figure 5.5, that the behaviour of the self-adaptation of σ and the covariance matrix is very similar for two runs with different starting point, even if one run is successful and the other fails. The best protocol that has been found within the limited number of function evaluations for the general unitary case, produces a state with fidelity 0.83272 from the input state (run 8.2 in Table C.4). This value is close to 0.836, which is the fidelity achieved by Protocol 4.4.1. Figure 5.6 shows that the unitary matrices of the protocol found by the optimization are sparse and from Figure 5.7 it can be seen that the found protocol does not perform as good as Protocol 4.4.1 especially for input states that are close to maximally entangled (i.e. $c \approx \frac{1}{\sqrt{2}}$ in Equation (5.8)) this contrasts with the case where the optimization was restricted to symmetric distillation by orthogonal matrices (c.f. Figure 5.1). Because the number of function evaluations needed to reach the best found objective function value ("needed # f.e." in Table C.4) is close to the total number of function evaluations that have been performed, we expect that the solution can be improved by running the optimization algorithm for a longer time.

5.4 Results for 2 copies of a pure initial state

For the case of only two copies of a pure input state we have shown in Section 4.4, that the Schmidt projection cannot be used to increase the fully entangled fraction without post-selection. Thus, it is an interesting question whether it is possible to find another entanglement distillation protocol that can perform this task. However, none of the tried optimization runs performed on two copies of the state $0.1|0+\rangle + 0.9|1-\rangle$ resulted in a protocol that leads to a state with higher fidelity than the fully entangled 0.2 of the initial state (see Table C.5). This is a sign that possibly there is no non-interactive protocol that increases the fully entangled fraction of two copies of a pure input state.

5.5 Results for 3 copies of a Werner state

It has been tried to use the same optimization method as used for 3 copies of a pure state (Section 5.3) to distill entanglement from 3 copies of a mixed state. The state chosen for this purpose was the Werner state:

$$\rho_{\text{input}} = \left(1 - \frac{3}{4}p\right)|\phi^+\rangle\langle\phi^+| + \frac{1}{4}p\left(|\phi^-\rangle\langle\phi^-| + |\psi^+\rangle\langle\psi^+| + |\psi^-\rangle\langle\psi^-|\right). \quad (5.10)$$

For the optimization p in (5.10) has been fixed to a value of $p = 0.1$. This state has an initial fidelity w.r.t. $|\phi^+\rangle$ of 0.925, which corresponds to a fitness value of 0.075. We know from [8] that the maximal fidelity we can hope to produce from any number of copies of the Werner state ρ_{input} (5.10) is upper bounded by $1 - \frac{p}{2} = 1 - 0.05 = 0.95$ (see Table C.2). It can be seen from Table C.6, that none of the performed optimization runs was successful. The problem is, that the computational complexity of the objective function evaluation is higher than for the case of a pure input state: In the former case one only has to perform matrix vector multiplication, which has complexity $\mathcal{O}(N^2)$ where N is the dimension of the Hilbert space. On the other hand, in order to apply the unitary operations on the mixed input state ρ_{input} one has to calculate $\mathbf{U}\rho_{\text{input}}\mathbf{U}^\dagger$, which involves two matrix multiplications. For that, the number of elementary operations is

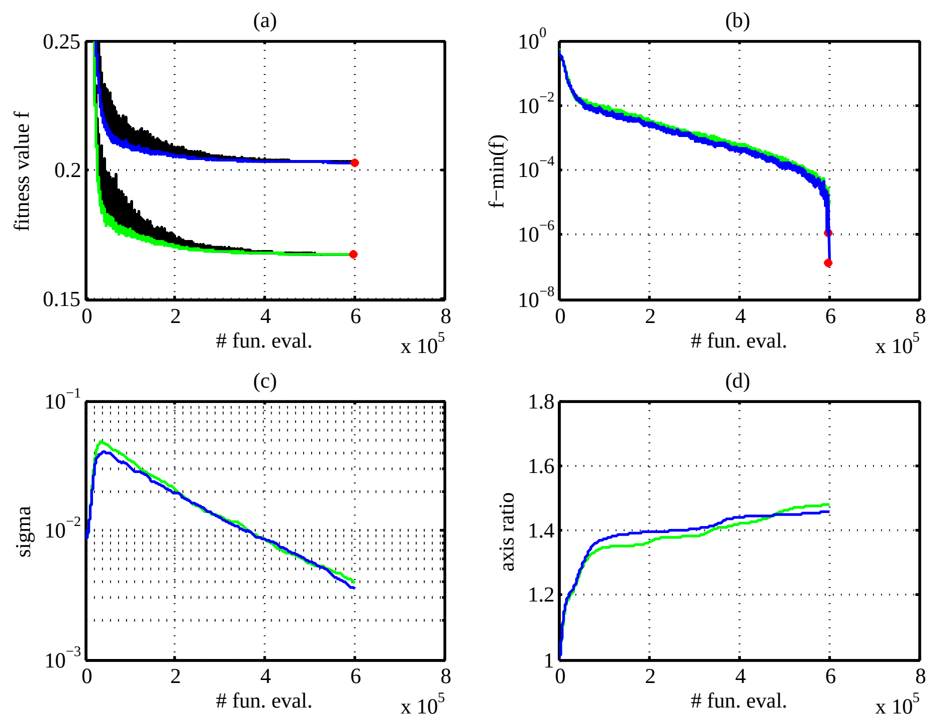

Figure 5.5: Convergence behaviour of three runs of the CMA-ES algorithm when used to find an entanglement distillation protocol without communication for 3 copies of a pure state. The search space is 2048-dimensional, which corresponds to the general case of optimization over 2 unitary matrices. The fully entangled fraction of the given input state is 0.8, which corresponds to a fitness value of $f = 0.2$. Each of the runs shown in this figure has a different starting point chosen uniformly between $-\pi$ and $+\pi$. The green line corresponds to the first successful run (id 8.2 in Table C.4), the blue line to an unsuccessful run (id 15.1 in Table C.4). The abscissae correspond to the number of objective function evaluations that have been performed by the black-box optimization algorithm to reach the corresponding state. Panel (a): coloured lines show the best objective function value found after each iteration (black lines show the fitness of the worst offspring in the corresponding iteration). Panel (b): the difference to the final objective function value. Panel (c): the parameter σ of the CMA-ES algorithm, by which the standard deviations of the sampling distribution are scaled (step size). Panel (d): the ratio between the longest and shortest principle axis length of the covariance matrix of the sampling distribution.

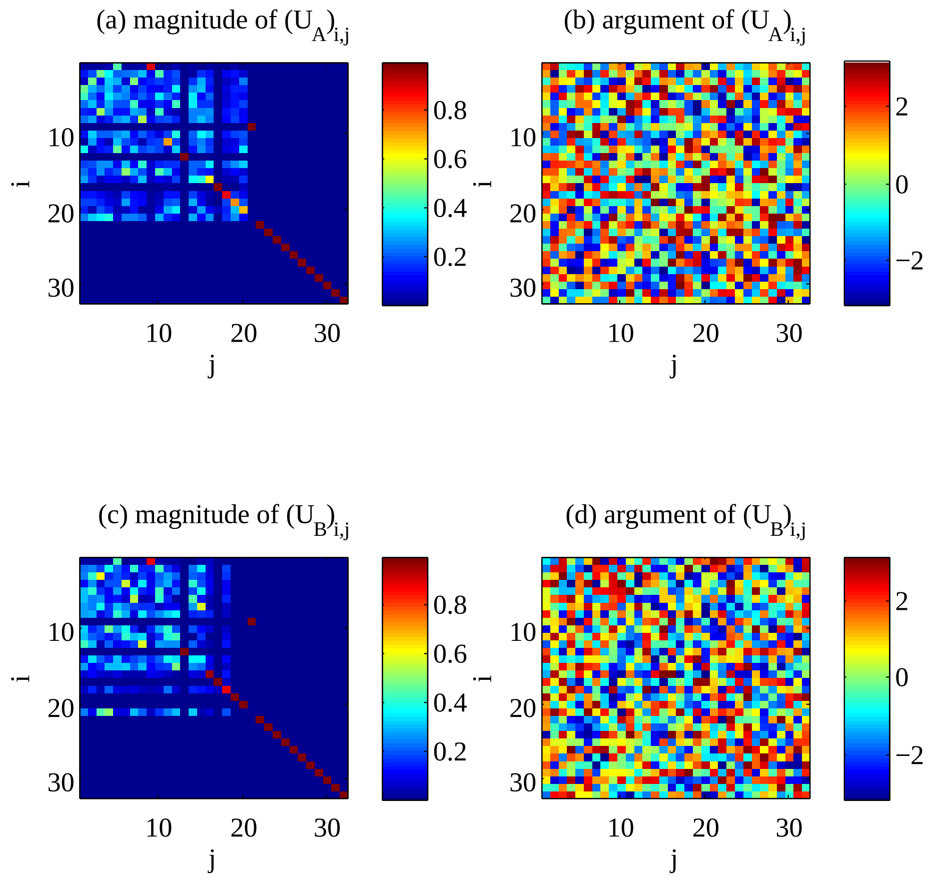

Figure 5.6: Visualization of the 32×32 unitary matrices $\mathbf{U_A}$ and $\mathbf{U_B}$ found by optimization run 8.2 shown in Table C.4. If $\mathbf{U_A}$ ($\mathbf{U_B}$) is applied by Alice (Bob) on her (his) part of 3 copies of the pure state (5.8) together with two local ancilla qubits per party, a fidelity of 0.83272 results. Panel (a): the magnitude of the complex entries of U_A. Panel (b): the argument $\arg\left((U_A)_{i,j}\right) = \arctan\left(\mathrm{Im}\left((U_A)_{i,j}\right)/\mathrm{Re}\left((U_A)_{i,j}\right)\right)$. Panel (c): the magnitude of the complex entries of U_B. Panel (d): the argument $\arg\left((U_B)_{i,j}\right) = \arctan\left(\mathrm{Im}\left((U_B)_{i,j}\right)/\mathrm{Re}\left((U_B)_{i,j}\right)\right)$.

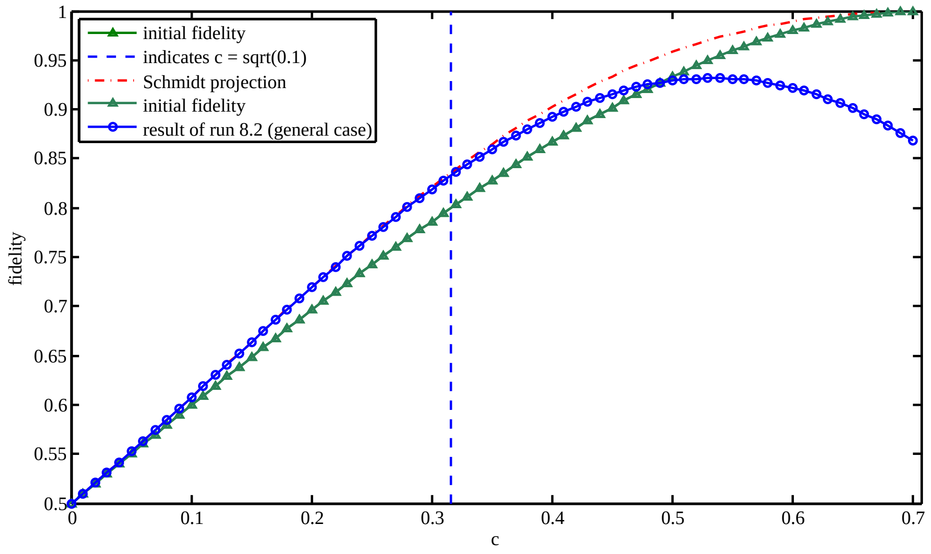

Figure 5.7: The fidelity of the state one obtains by applying different protocols on three copies of the input state $|\psi\rangle = c|0+\rangle + \sqrt{1-c^2}|1-\rangle$ for different values of c compared to the initial fidelity indicated by the green line with triangles. The performance of the best protocol obtained by numerical optimization over general unitary matrices with limited number of function evaluations (optimization run 8.2 in Table C.4) is indicated by the blue solid line with circles. By the term 'Schmidt projection' in the legend, Protocol 4.4.1 is meant. The vertical dashed line indicates $c = \sqrt{0.1}$ corresponding to the input state for which optimization run 8.2 has been performed.

on the order of $\mathscr{O}(N^3)$ with the naive approach [1].

[1]There exist algorithms for matrix-matrix multiplication with lower complexity such as Strassen's algorithm [37] for which the number of scalar multiplications is on the order of $\mathscr{O}(N^{\log_2 7})$.

Chapter 6

Conclusions

After summarizing the properties of previously known entanglement distillation protocols and limits to the distillability with and without communication, we have investigated the use of the Schmidt projection method without communication for the cases where only 2 or 3 copies of an entangled state are given. It is possible to increase the fully entangled fraction with the Schmidt projection without post-selection if and only if more than two copies of an entangled pure state are given. We derived a non-interactive entanglement distillation protocol for mixed states from the Schmidt projection method for three copies of a pure input state. Furthermore, we demonstrated the possibility to find an equivalent protocol to the Schmidt projection method for 3 copies by numerical optimization. For 2 copies of a pure input state the numerical optimization was unsuccessful, indicating that no protocol for 2 copies of an entangled pure state that increases the fully entangled fraction exists. Finally, we investigated the performance of the optimization algorithm if the input state is a mixed state.

Acknowledgements

Many thanks go to my advisor Dejan Dukaric for the great assistance he gave me during the time I was working on the present project, for the patience he had in answering my numerous questions and giving the present work the right direction. My special gratitude belongs to Prof. Stefan Wolf for the opportunity to do the present thesis in his group and for the well understandable and motivating lectures he held at ETH Zürich. His lectures have sparked my interest in quantum information and computation as well as general information science.

Appendix A

Proof of Theorem 4.3.1

We are going to proof Theorem 4.3.1:

$$|\psi\rangle \text{ can be non-interactively transformed to } |\phi\rangle \Longleftrightarrow \lambda_\psi \cong \lambda_\phi \otimes \mathbf{x}, \tag{A.1}$$

where the elements of vector $\mathbf{x}$ are non-negative and sum up to 1.

Proof. "$\Longleftarrow$": Assume that the right hand side of Equivalence (A.1) is true for some pure states $|\psi\rangle$ and $|\phi\rangle$. Let $|\psi_E\rangle$ be some pure state with $\rho_{\psi_E,B} = \mathrm{tr}_B|\psi_E\rangle\langle\psi_E|$ having eigenvalues given by the vector $\mathbf{x}$ and let $|\psi_R\rangle := |\phi\rangle \otimes |\psi_E\rangle$. Furthermore, let the density operator $\rho_{\psi_R,A} = \mathrm{tr}_B|\psi_R\rangle\langle\psi_R|$ of Alice's subsystem have eigenvalues given by the vector $\lambda_\rho \otimes \mathbf{x}$.

The Schmidt decomposition (see e.g. [18, p. 109]) of any bipartite state $|\varphi\rangle$ is

$$|\varphi\rangle = \sum_{i=1}^{d} \sqrt{c_i}|\varphi_{A,i}\rangle|\varphi_{B,i}\rangle, \tag{A.2}$$

for orthogonal sets of basis vectors $\{|\varphi_{A,i}\rangle\ |i = 1,\cdots d\}$ and $\{|\varphi_{B,i}\rangle\ |i = 1,\cdots d\}$, where d is the dimension of the Hilbert space describing Alice's and Bob's subsystem (which for simplicity and without loss of generality are assumed to be of the same dimensionality) and $\mathbf{c} \in \mathbb{R}^d$ is the vector of squared Schmidt coefficients with $\sum_i^d c_i = 1$. The corresponding density matrix is

$$|\varphi\rangle\langle\varphi| = \sum_{i=1}^{d}\sum_{j=1}^{d} \sqrt{c_i c_j}|\varphi_{A,i}\rangle\langle\varphi_{A,i}| \otimes |\varphi_{B,i}\rangle\langle\varphi_{B,i}|. \tag{A.3}$$

Thus, the partial trace over Bob's subsystem is

$$\rho_{\varphi,A} = \mathrm{tr}_B|\varphi\rangle\langle\varphi| = \sum_{i=1}^{d} c_i|\varphi_{A,i}\rangle\langle\varphi_{A,i}|. \tag{A.4}$$

So the squared Schmidt coefficients are exactly the eigenvalues of $\rho_{\varphi,A}$. By the assumption that for the state $|\psi_R\rangle$ the density matrix of Alice's subsystem has the same eigenvalues as for the state $|\psi\rangle$, it follows that the Schmidt coefficients of $|\psi_R\rangle$ and $|\psi\rangle$ are the same. So to convert $|\psi\rangle$ into $|\psi_R\rangle$ Alice and Bob just have to apply local unitary operations $\mathbf{U}_A$ and $\mathbf{U}_B$ that transform the orthogonal sets of basis vectors $\{|\varphi_{A,i}\rangle\ |i = 1,\cdots d\}$ and $\{|\varphi_{B,i}\rangle\ |i = 1,\cdots d\}$ from the Schmidt decomposition of $|\psi\rangle$ into the ones of the Schmidt decomposition of $|\psi_R\rangle$. Finally, $|\phi\rangle$ can be obtained by the partial trace over the subsystem of $|\psi_E\rangle$ in $|\psi_R\rangle = |\phi\rangle \otimes |\psi_E\rangle$.

"$\Longrightarrow$": Let us assume that the pure state $|\psi\rangle$ has been converted into the pure state $|\phi\rangle$ without communication. It follows that

$$|\psi_R\rangle = |\phi\rangle \otimes |\psi_E\rangle = (\mathbf{U}_A \otimes \mathbf{U}_B)|\psi\rangle. \tag{A.5}$$

From the derivation above, we know that the vectors λ_ψ and λ_{ψ_R} are equal to the vector of squared Schmidt coefficients of $|\psi\rangle$ and $|\psi_R\rangle$ respectively. It is easy to see that the local unitary transformation $\mathbf{U}_A \otimes \mathbf{U}_B$ does not change the values of the Schmidt coefficients of $|\psi\rangle$. As above, we define $\mathbf{x}$ to be the vector of eigenvalues of $\rho_{\psi_E,A} = \mathrm{tr}_B|\psi_E\rangle\psi_E|$. Since

$$\mathrm{tr}_B|\psi_R\rangle\langle\psi_R| = \mathrm{tr}_B\left[|\phi\rangle\langle\phi| \otimes |\psi_E\rangle\langle\psi_E|\right] = \rho_{\phi,A} \otimes \rho_{\psi_E,A}, \tag{A.6}$$

it follows that $\lambda_\psi \cong \lambda_\phi \otimes \mathbf{x}$

$\square$

Appendix B

Proof of Theorem 4.4.1

In the following, the mathematical statement and proof of Theorem 4.4.1 is given.

Definition B.0.1. *Let the Schmidt-Projection measurement be defined by the measurement operators $P_0 := P_{0,A} \otimes P_{0,B}$, $P_1 := P_{1,A} \otimes P_{1,B}$, $P_2 := P_{2,A} \otimes P_{2,B}$ where*

$$P_{0,A} := |00\rangle\langle 00|, \qquad P_{0,B} := |{+}{+}\rangle\langle {+}{+}|, \tag{B.1}$$

$$P_{1,A} := |01\rangle\langle 01| + |10\rangle\langle 10|, \qquad P_{1,B} := |{+}{-}\rangle\langle {+}{-}| + |{-}{+}\rangle\langle {-}{+}|, \tag{B.2}$$

$$P_{2,A} := |11\rangle\langle 11|, \qquad P_{2,B} := |{-}{-}\rangle\langle {-}{-}|. \tag{B.3}$$

The subscripts of the projection operators denote the measurement outcomes.

Theorem B.0.1. *Mathematical satement of Theorem 4.4.1:*

Let $|\psi_{\text{input}}\rangle := c|0+\rangle + s|1-\rangle \in \mathscr{H}_A \otimes \mathscr{H}_B$ with $c, s \in \mathbb{R}$, $c^2 + s^2 = 1$, where $|0\rangle$ and $|1\rangle$ are orthogonal states in the two-dimensional Hilbert space $\mathscr{H}_A$ while $|+\rangle$ and $|-\rangle$ are orthogonal states in the two-dimensional Hilbert space $\mathscr{H}_B$. Based on that, let $|\psi_{\text{copy}}\rangle := (|\psi_{\text{input}}\rangle)^{\otimes 2} \in \mathscr{H}_{A_1} \otimes \mathscr{H}_{B_1} \otimes \mathscr{H}_{A_2} \otimes \mathscr{H}_{B_2}$, where $\mathscr{H}_{A_1} \otimes \mathscr{H}_{B_1}$ and $\mathscr{H}_{A_2} \otimes \mathscr{H}_{B_2}$ denote the Hilbert spaces of the first and second copy respectively. Furthermore let ρ_{pm} be the state of the system after a Schmidt-Projection measurement (Definition B.0.1). Then

$$F(|\psi_{\text{input}}\rangle\langle\psi_{\text{input}}|) > \max_{\mathscr{U}_A, \mathscr{U}_B} F(\text{tr}_{A_2}[\text{tr}_{B_2}[(\mathscr{U}_A \otimes \mathscr{U}_B)\rho_{\text{pm}}(\mathscr{U}_A^\dagger \otimes \mathscr{U}_B^\dagger)]]), \tag{B.4}$$

where $F(\cdot)$ denotes the fully entangled fraction (Definition 3.3.1), $\mathscr{U}_A$ denotes an arbitrary unitary operator on the subspace $\mathscr{H}_{A_1} \otimes \mathscr{H}_{A_2}$ whereas $\mathscr{U}_B$ denotes an arbitrary unitary operator on $\mathscr{H}_{B_1} \otimes \mathscr{H}_{B_2}$ and $\text{tr}_{A_2}[\cdot]$ and $\text{tr}_{B_2}[\cdot]$ denote the partial trace over the subspaces $\mathscr{H}_{A_2}$ and $\mathscr{H}_{B_2}$.

Note that the “standardization process” as described in [12] is not probabilistic in this special case of only two given copies of the initial state, since if the measurement outcome of the Schmidt-Projection measurement (Definition B.0.1) is 1, then the corresponding state can be transformed into a maximally entangled state with local unitary operations that are applied before tracing out one of the entangled pairs of qubits.

Proof. Since

$$|\psi_{\text{copy}}\rangle := (|\psi_{\text{input}}\rangle)^{\otimes 2} = c^2|00{+}{+}\rangle + s^2|11{-}{-}\rangle + cs(|01{+}{-}\rangle + |10{-}{+}\rangle) \tag{B.5}$$

$$= c^2|00{+}{+}\rangle + s^2|11{-}{-}\rangle + \sqrt{2}cs|\psi_1\rangle \tag{B.6}$$

with $|\psi_1\rangle := \frac{1}{\sqrt{2}}(|01+-\rangle + |10-+\rangle)$ the state ρ_{pm} after application of the Schmidt-Projection measurement (Definition B.0.1) is

$$\begin{aligned}
\rho_{\text{pm}} &= c^4|00++\rangle\langle 00++| + s^4|11--\rangle\langle 11--| + 2c^2s^2|\psi_1\rangle\langle\psi_1| && \text{(B.7)}\\
&= c^4|00\rangle\langle 00| \otimes |++\rangle\langle ++| && \text{(B.8)}\\
&\quad + s^4|11\rangle\langle 11| \otimes |--\rangle\langle --| && \text{(B.9)}\\
&\quad + 2c^2s^2|\psi_1\rangle\langle\psi_1|. && \text{(B.10)}
\end{aligned}$$

Therefore, the state after application of the unitary operator $\mathscr{U}_A \otimes \mathscr{U}_B$ is

$$\begin{aligned}
\tilde{\rho}_{\text{pm}} :&= (\mathscr{U}_A \otimes \mathscr{U}_B)\rho_{\text{pm}}(\mathscr{U}_A^\dagger \otimes \mathscr{U}_B^\dagger) && \text{(B.11)}\\
&= c^4\mathscr{U}_A|00\rangle\langle 00|\mathscr{U}_A^\dagger \otimes \mathscr{U}_B|++\rangle\langle ++|\mathscr{U}_B^\dagger && \text{(B.12)}\\
&\quad + s^4\mathscr{U}_A|11\rangle\langle 11|\mathscr{U}_A^\dagger \otimes \mathscr{U}_B|--\rangle\langle --|\mathscr{U}_B^\dagger && \text{(B.13)}\\
&\quad + 2c^2s^2(\mathscr{U}_A \otimes \mathscr{U}_B)|\psi_1\rangle\langle\psi_1|(\mathscr{U}_A^\dagger \otimes \mathscr{U}_B^\dagger). && \text{(B.14)}\\
&= c^4|\varphi_0^A\rangle\langle\varphi_0^A| \otimes |\varphi_0^B\rangle\langle\varphi_0^B| && \text{(B.15)}\\
&\quad + s^4|\varphi_2^A\rangle\langle\varphi_2^A| \otimes |\varphi_2^B\rangle\langle\varphi_2^B| && \text{(B.16)}\\
&\quad + 2c^2s^2|\tilde{\psi}_1\rangle\langle\tilde{\psi}_1|. && \text{(B.17)}
\end{aligned}$$

with $|\varphi_0^A\rangle := \mathscr{U}_A|00\rangle$, $|\varphi_0^B\rangle := \mathscr{U}_B|++\rangle$, $|\varphi_2^A\rangle := \mathscr{U}_A|11\rangle$, $|\varphi_2^B\rangle := \mathscr{U}_B|--\rangle$ and $|\tilde{\psi}_1\rangle := (\mathscr{U}_A \otimes \mathscr{U}_B)|\psi_1\rangle$.

Now the partial trace (linear operator) over the second copy of the distributed pair of qubits is

$$\begin{aligned}
\rho_{\text{output}} &:= \operatorname{tr}_{A_2}[\operatorname{tr}_{B_2}[\tilde{\rho}_{\text{pm}}]] && \text{(B.18)}\\
&= c^4(\operatorname{tr}_{A_2}[|\varphi_0^A\rangle\langle\varphi_0^A|]) \otimes (\operatorname{tr}_{B_2}[|\varphi_0^B\rangle\langle\varphi_0^B|]) && \text{(B.19)}\\
&\quad + s^4(\operatorname{tr}_{A_2}[|\varphi_2^A\rangle\langle\varphi_2^A|]) \otimes (\operatorname{tr}_{B_2}[|\varphi_2^B\rangle\langle\varphi_2^B|]) && \text{(B.20)}\\
&\quad + 2c^2s^2\operatorname{tr}_{A_2}[\operatorname{tr}_{B_2}[|\tilde{\psi}_1\rangle\langle\tilde{\psi}_1|]] && \text{(B.21)}\\
&= c^4\rho_{0,A} \otimes \rho_{0,B} + s^4\rho_{2,A} \otimes \rho_{2,B} + 2c^2s^2\rho_1, && \text{(B.22)}
\end{aligned}$$

with $\rho_{0,A} := \operatorname{tr}_{A_2}[|\varphi_0^A\rangle\langle\varphi_0^A|]$, $\rho_{0,B} := \operatorname{tr}_{B_2}[|\varphi_0^B\rangle\langle\varphi_0^B|]$, $\rho_{2,A} := \operatorname{tr}_{A_2}[|\varphi_2^A\rangle\langle\varphi_2^A|]$, $\rho_{2,B} := \operatorname{tr}_{B_2}[|\varphi_2^B\rangle\langle\varphi_2^B|]$ and $\rho_1 := \operatorname{tr}_{A_2}[\operatorname{tr}_{B_2}[|\tilde{\psi}_1\rangle\langle\tilde{\psi}_1|]]$.

Because of Lemma 3.3.1, Lemma 3.3.3 and because $F(\cdot) \leq 1$ by definition, the following inequalities hold

$$\begin{aligned}
F(\rho_{\text{output}}) &\leq c^4F(\rho_{0,A} \otimes \rho_{0,B}) && && \text{(B.23)}\\
&\quad + s^4F(\rho_{2,A} \otimes \rho_{2,B}) && && \text{(B.24)}\\
&\quad + 2c^2s^2F(\rho_1) && \text{by Lemma 3.3.1} && \text{(B.25)}\\
&\leq (c^4+s^4)\frac{1}{2} + 2c^2s^2F(\rho_1) && \text{by Lemma 3.3.3} && \text{(B.26)}\\
&\leq (c^4+s^4)\frac{1}{2} + 2c^2s^2 && && \text{(B.27)}\\
&= \frac{1}{2}(c^4+s^4+2c^2s^2) + c^2s^2 && && \text{(B.28)}\\
&= \frac{1}{2} + c^2s^2 && && \text{(B.29)}
\end{aligned}$$

The inequality has to hold even if we maximize over the unitary operators $\mathscr{U}_A$ and $\mathscr{U}_B$ since the right-hand-side of this inequality is constant, whereas $F(\rho_{\text{output}})$ allows an arbitrary local unitary operation $\mathscr{U}_A \otimes \mathscr{U}_B$ to be performed before taking the partial trace over one of the copies.

But the initial entanglement was

$$\begin{aligned} F(|\psi_{\text{input}}\rangle\langle\psi_{\text{input}}|) &= \left|\langle\phi^+|(c|0+\rangle + s|1-\rangle)\right|^2 && \text{(B.30)} \\ &= \left|\frac{1}{\sqrt{2}}(c+s)\right|^2 && \text{(B.31)} \\ &= \frac{1}{2}(c^2+s^2+2*cs) && \text{(B.32)} \\ &= \frac{1}{2}+cs. && \text{(B.33)} \end{aligned}$$

Thus, since $c, s \in [0,1]$

$$F(\rho_{\text{output}}) \leq F(|\psi_{\text{input}}\rangle\langle\psi_{\text{input}}|) \qquad \text{(B.34)}$$

with equality if and only if $c=0$ or $s=0$ (separable input state). □

Appendix C

Tables

C.1 Entanglement Distillation Protocols

Table C.1: Some previously known entanglement distillation protocols. Refer to the cited articles in the column "reference" for the description and proof of correctness of these protocols. The protocols take n states of the form given in the column "input state" and produce m output states of fidelity φ_{out} w.r.t. a specific maximally entangled state if they succeed (with probability p_{succ}). The protocols listed above the thin line are protocols that are applied on pure states, whereas those below that line are protocols for mixed states. Whether the protocol includes classical communication from Alice to Bob in one or both directions is shown in the column denoted as "comm.". In the column named "prev. knowledge", the variables that have to be known in advance, i.e. before the execution of the protocol, are listed. An approximate-equality symbol "$\approx$" in the column "m" indicates that the corresponding value is realized in the limit as $n \to \infty$. The four Bell states are denoted as $|\phi_{ij}\rangle$, where $i, j \in \{0,1\}$. Note that the order of the enumeration of the Bell states causes no loss of generality, since Alice and Bob can permute the Bell states by local unitary operations[5].

n	m	input state	name of the protocol	comm.	prev. knowledge	success prob. p_{succ}	output fidelity φ_{out}	reference
1	1	$\|\psi\rangle = c\|0+\rangle + s\|1-\rangle$	Procrustean (Filtering)	1-way	c,s	$2\min(c^2, s^2)$	1	[12, 38, 39]
1	1	$\|\psi\rangle = \psi_{AA} \otimes \psi_{AB}$, where $\psi_{AA} = c\|00\rangle + s\|11\rangle$ is the sate of a pair of qubits completely in Alice's possession, whereas $\psi_{AB} = c\|0+\rangle + s\|1-\rangle$ belongs to a pair that is distributed to Alice and Bob.	Entanglement Swapping	1-way	none	$2c^2s^2$	1 with probability p_{succ} or else $\varphi_{00} = \frac{1}{2}(1-2c^2+2c^4)^{-1}$, $\varphi_{01} = 1 - \varphi_{00}$	[23]
$\gg 1$	$\approx nE_{\text{init}}$	$\|\psi\rangle^{\otimes n} = (c\|0+\rangle + s\|1-\rangle)^{\otimes n}$ with entropy of entanglement $E_{\text{init}} > 0$	Schmidt Projection	no	none	1 as $n \to \infty$	1	[12, 39, 40]
1	1	$\varphi\|\phi^+\rangle\langle\phi^+\| + (1-\varphi)\|0+\rangle\langle 0+\|$	Filtering	2-way	none	$\varphi\varepsilon + (1-\varphi)\varepsilon^2$	$\varphi\varepsilon / p_{\text{succ}}$	[41, 42]
2	1	$\rho_{\text{pair}} = \rho_{W,1} \otimes \rho_{W,2}$, where $\rho_{W,1}$ and $\rho_{W,2}$ are *Werner states* i.e. $\rho_{W,k} = \frac{1}{3}(4\varphi_k - 1)\|\phi^+\rangle\langle\phi^+\| + \frac{1}{3}(1-\varphi_k)\mathbb{1}_{AB}$ with $\varphi_1, \varphi_2 > \frac{1}{2}$	BBPSSW (for recursive and nested schemes)	2-way	none	$\frac{1}{9}(5 - 2\varphi_2 + \varphi_1(8\varphi_2 - 2))$	$\frac{1}{9}(1 - \varphi_2 + \varphi_1(10\varphi_2 - 1))/p_{\text{succ}}$	[4, 5, 43, 42]
2	1	$\rho_{\text{pair}} = \rho_1 \otimes \rho_2$, where ρ_1 and ρ_2 are *Bell diagonal* i.e. $\rho_k = \sum_{ij,k} \varphi_{ij,k} \|\phi_{ij,k}\rangle\langle\phi_{ij,k}\|$ with $\varphi_{00,1}, \varphi_{00,2} > \frac{1}{2}$	DEJMPS (for recursive and nested schemes)	2-way	none	$(\varphi_{00,1} + \varphi_{11,1})(\varphi_{00,2} + \varphi_{11,2}) + (\varphi_{01,1} + \varphi_{10,1})(\varphi_{01,2} + \varphi_{10,2})$	$\varphi_{00,\text{out}} = (\varphi_{00,1}\varphi_{00,2} + \varphi_{11,1}\varphi_{11,2})/p_{\text{succ}}$ $\varphi_{01,\text{out}} = (\varphi_{01,1}\varphi_{01,2} + \varphi_{10,1}\varphi_{10,2})/p_{\text{succ}}$ $\varphi_{10,\text{out}} = (\varphi_{00,1}\varphi_{11,2} + \varphi_{11,1}\varphi_{00,2})/p_{\text{succ}}$ $\varphi_{11,\text{out}} = (\varphi_{01,1}\varphi_{10,2} + \varphi_{10,1}\varphi_{01,2})/p_{\text{succ}}$	[6, 44, 43, 42]
≥ 2	$< n$	any $\rho = \rho_{\text{input}}$, where $\rho_{\text{input}} \in \mathscr{H}_{2^{2n}}$ with $\varphi(\rho_{\text{input}}) = \langle\phi^+\|\rho\|\phi^+\rangle \geq 1 - \varepsilon$	Random Hashing	1-way	none	$\geq 1 - \varepsilon$	$1 - 2^{m-n}/(1-\varepsilon)$	[7, 8, 45]
≥ 2	$< n$	any $\rho = \rho_{\text{input}} \otimes (\|\phi^+\rangle\langle\phi^+\|)^{\otimes k}$, where $\rho_{\text{input}} \in \mathscr{H}_{2^{2n}}$ with $\varphi(\rho_{\text{input}}) = \langle\phi^+\|\rho\|\phi^+\rangle \geq 1 - \varepsilon$ and n and m chosen such that nk/m and m/k are integers.	Random Permutation	2-way	none	1	$1 - \frac{2^m - 2^k}{2^m} \frac{2^n}{2^n - 1} \varepsilon$	[7, 8, 45]
≥ 2	$< n$	same input as above but with a shared source of random numbers (public coin)	No-communication Random Permutation	no	none	1	$1 - \frac{3}{4} \frac{2^n - \frac{2}{3}}{2^n - 1} \varepsilon$	[8, 45]
$\gg 1$	$\approx n(1 - S(\rho))$	$\rho^{\otimes n}$, where ρ is *Bell diagonal* i.e. $\rho = \sum_{i,j} \varphi_{ij} \|\phi_{ij}\rangle\langle\phi_{ij}\|$ with von Neumann entropy $S(\rho) < 1$ combined with a number $n(S(\rho) + \delta)$, $\delta \geq 0$ of maximally entangled states	Breeding	1-way	φ_{ij}	1 as $n \to \infty$	1	[4, 5, 42]
$\gg 1$	$\approx n(1 - S(\rho))$	$\rho^{\otimes n}$, where ρ is *Bell diagonal* i.e. $\rho = \sum_{i,j} \varphi_{ij} \|\phi_{ij}\rangle\langle\phi_{ij}\|$ with von Neumann entropy $S(\rho) < 1$	Hashing	1-way	φ_{ij}	1 as $n \to \infty$	1	[5, 46, 42]

C.2 Limits to Distillability

Table C.2: Some previously known limits to the existence of entanglement distillation protocols for particular input states. Refer to the cited articles for the proofs of these limits (given in the column "refence"). *m* denotes the number of output states which have higher fidelity φ_{out} than each of the *n* given input states. A "**0**" for "max. *m*" means that no distillation protocol fulfilling this particular requirements exists. The limits given in the columns "max. *m*" and "max. φ_{out}" only apply if the requirements specified in the other columns are met. An asterisk (*) indicates that the corresponding quantity does not matter for the limit, i.e. the limitation applies for any value of this quantity. The limits listed above the thin line are limits that apply to protocols that work on pure states, whereas those below that line are limitations on the existence of protocols for mixed states. Whether the protocol is allowed to include classical communication from Alice to Bob in one or both directions is shown in the column denoted as "comm.". In the column named "prev. knowledge", the variables that are given in advance, i.e. before the execution of the protocol, are listed. The column "required p_{succ}" specifies what the desired success probability is. Note that a value of 1 means that the protocol has to be deterministic in the sense that it always has to succeed. A value of one in column "required φ_{out}" means that the limitation only applies to protocols that generate maximally entangled output pairs.

n	m	input state	name of the corresponding noise model (if any)	comm.	prev. knowledge	required p_{succ}	required φ_{out}	max. m	max. φ_{out}	reference
*	*	any pure state with entropy of entanglement $E(\|\psi\rangle) \geq k$	Entanglement Model	*	only k	1	*	**0**		[45]
$\gg 1$	*	any pure state $\|\psi\rangle$ with entropy of entanglement E_{init}		*	*	> 0	1	nE_{init} for $n \to \infty$		[13]
*	*	any $\rho = \rho_{\text{input}} \otimes (\|\phi^+\rangle\langle\phi^+\|)^{\otimes k}$, where $\rho_{\text{input}} \in \mathscr{H}_{2^{2n}}$ with $\varphi_{\text{init}} = \varphi(\rho_{\text{input}}) \geq 1-\varepsilon$ and $k < m$	Fidelity Model	*	only ε and k	1	*		$1 - \frac{2^m - 2^k}{2^m} \frac{2^n}{2^n - 1} \varepsilon$	[7, 45]
*	*	same input as above but with a shared source of random numbers (public coin)	Fidelity Model	no	only ε and k	1	*		$1 - \frac{3}{4} \frac{2^{2n}}{2^{2n} - 1} \varepsilon$	[8, 45]
*	*	$\rho = \frac{1}{2^n \binom{n}{r}} \sum_{\mathbf{v}:\deg(\mathbf{v})=r} \|\phi_{\mathbf{v}}\rangle\langle\phi_{\mathbf{v}}\|$, where $\mathbf{v} \in \{0,1,*\}^n$, $\deg(\mathbf{v}) := \|\{i : v_i \neq *\}\|$ and $\|\phi_{\mathbf{v}}\rangle := \bigotimes_{j=0}^{n-1} \|\phi_j\rangle$ with $\|\phi_j\rangle := \begin{cases} \|0+\rangle & \text{if } v_j = 0 \\ \|1-\rangle & \text{if} v_j = 1 \\ \|\phi^+\rangle & \text{if } v_j = * \end{cases}$	Bounded Measurement Model	no	*	1	*	**0**		[8, 45]
*	*	$\rho = \frac{1}{3^n \binom{n}{r}} \sum_{\mathbf{v}:\deg(\mathbf{v})=r} P_{\mathbf{v}} \|(\phi^+\rangle\langle\phi^+\|)^{\otimes n} P_{\mathbf{v}}^\dagger$, $\mathbf{v} := (x_0, x_1, \ldots, x_{n-1}, z_0, z_1, \ldots, z_{n-1})$, $\deg(\mathbf{v}) := \|\{i : x_i \neq 0 \vee z_i \neq 0\}\|$, $P_{\mathbf{v}} := X^{x_0} Z^{z_0} \otimes \ldots \otimes X^{x_{n-1}} Z^{z_{n-1}}$	Bounded Corruption Model	no	*	1	*		$1 - \frac{r}{2n}$	[45]
*	*	$\rho = (\rho_W)^{\otimes n}$, where ρ_p is a Werner state: $\rho_W = (1 - \frac{3p}{4})\|\phi^+\rangle\langle\phi^+\| + \frac{p}{4}(\|\phi^-\rangle\langle\phi^-\| + \|\psi^+\rangle\langle\psi^+\| + \|\psi\rangle\langle\psi^-\|)$	Depolarization Model	no	*	1	*		$1 - \frac{p}{2}$	[8, 45]
*	*	$\rho(p) = (\Omega_\alpha)^{\otimes n}$, where $\Omega_\alpha = \frac{1+\alpha}{2}\|\phi^+\rangle\langle\phi^+\| + \frac{1-\alpha}{2}\|\phi^-\rangle\langle\phi^-\|$	Weakly Entangled Mixed States	no	*	1	*	**0**		[14]
2	*	$\rho(p) = p\rho_0 + (1-p)\rho_1$, $p \in [0,1]$, where $\rho(p) \in S$ and S is a *twirlable* family of bipartite mixed states, e.g the familiy of all Werner states (see reference).		*	*	1	*	**0**		[15]

C.3 Optimization Runs

In the following, tables summarizing the properties of the performed numerical optimizations are presented.

Table C.3: Listing of the performed optimization runs for the case of symmetric distillation with an orthogonal matrix for 3 copies of a pure input state together with 2 ancilla qubits per party (Section 5.3.1). The dimensionality of the search space was 497. The "run id" is an arbitrary identifier of the run. In the column "initial parameters", either the distribution with which the initial parameters have been sampled is given or else the parameters were exactly the same as in a previous optimization run ("same as ..."). "resumed from ..." means that the corresponding run has been continued from of a former run. The term "param." in the column denoted as "det.", means that the determinant was set by the sign of an additional parameter. For an explanation of the algorithm specific parameters σ and λ, refer to Section 5.2. The value given in the column "total # f. e." is the total number of function evaluations, whereas the value given in "needed # f. e." is the number of function evaluations needed to reach the final fitness value given in the column "final fitness" up to an accuracy of 5 decimals. The given fitness values can be converted directly to fidelities by subtracting the fitness from 1. The fidelity of the input state was 0.8 for all runs. Thus, only those runs with fitness < 0.2 were successful in finding an entanglement distillation protocol. The time the CPU used to perform the optimization is given in the column "runtime CPU". The optimizations have been performed on a cluster which automatically distributes the jobs to specific nodes. Those nodes have different CPUs. The speed of the CPU is indicated by the so called "CPU factor". The corresponding CPU models are: 6.5: AMD Opteron 250, 7.5: AMD Opteron 8380, 8.0: AMD Opteron 2220

run id	initial parameters	det.	init. σ	λ	total # f. e.	needed # f. e.	initial fitness	final fitness	runtime CPU	CPU factor
1	$\mathscr{N}(\mathbf{0},(0.0814)^2)$	param.	0.01	22	233289	79729	0.45097	0.16400	n/a	n/a
2	$\mathscr{N}(\mathbf{0},(0.0814)^2)$	param.	0.01	22	269699	87539	0.48775	0.16400	n/a	n/a
z	all zero	param.	0.01	22	299223	146191	0.20000	0.16400	n/a	n/a
3	$\mathscr{U}([-\pi,+\pi])$	param.	$2\pi/\sqrt{12}$	22	89167	28205	0.45196	0.32802	n/a	n/a
4	$\mathscr{U}([-\pi,+\pi])$	param.	0.01	22	2442287	249767	0.48753	0.20300	n/a	n/a
5	same as 4	param.	0.01	22	42571	38831	0.48753	0.20333	n/a	n/a
6	same as 4	param.	0.01	22	28601	28183	0.48753	0.20434	39min	6.5
6.1	resumed from 6	param.	-	22	300015	258171	-	0.20300	4h 06min	7.5
7	same as 4	param.	0.01	22	300015	287013	0.48753	0.20300	4h 27min	7.5
8	same as 4	param.	0.01	22	300015	286947	0.48753	0.20300	5h 19min	7.5
10	$\mathscr{U}([-\pi,+\pi])$	+1	0.01	22	300015	239053	0.44144	0.20300	6h 52min	6.5
11	same as 10	+1	0.01	22	300015	260503	0.44144	0.20300	6h 49min	6.5
12	same as 10	-1	0.01	22	300015	223081	0.44144	0.20300	7h 13min	6.5

table continued on next page

run id	initial parameters	det.	init. σ	λ	total # f. e.	needed # f. e.	initial fitness	final fitness	runtime CPU	CPU factor
13	same as 10	-1	0.01	22	300015	231947	0.44144	0.20300	6h 48min	6.5
15	$\mathscr{U}([-\pi,+\pi])$	+1	0.01	22	300015	220221	0.41733	0.20300	7h 02min	6.5
16	$\mathscr{U}([-\pi,+\pi])$	+1	0.01	22	300015	235027	0.46305	0.20300	6h 49min	6.5
17	$\mathscr{U}([-\pi,+\pi])$	-1	0.01	22	300015	254783	0.43582	0.20000	7h 20min	6.5
18	$\mathscr{U}([-\pi,+\pi])$	param.	0.01	22	300015	251021	0.44511	0.20300	5h 31min	8.0
18.1	resumed from 18	param.	-	22	600007	251021	-	0.20300	5h 45min	8.0
19	$\mathscr{U}([-\pi,+\pi])$	param.	0.01	22	6403	6007	0.50830	0.20735	9min	6.5
19.1	resumed from 19	param.	-	22	1000011	257709	-	0.20000	18h 19min	8.0
20	$\mathscr{U}([-\pi,+\pi])$	param.	0.01	22	300015	256807	0.46339	0.20300	6h 59min	6.5

Table C.4: Listing of the performed optimization runs for the case of general entanglement distillation for 3 copies of a pure input state together with 2 ancilla qubits per party (Section 5.3.2). The dimensionality of the search space was 2048. The “run id” is an arbitrary identifier of the run. In column “initial parameters”, either the distribution with which the initial parameters have been sampled is given or else the parameters were exactly the same as in a previous optimization run (“same as ...”) or the corresponding run is a continuation of a former run (“resumed from ...”). For an explanation of the algorithm specific parameters σ and λ, refer to Section 5.2. The value given in the column “total # f. e.” is the total number of function evaluations, whereas the value given in “needed # f. e.” is the number of function evaluations needed to reach the final fitness value given in the column “final fitness” up to an accuracy of 5 decimals. The given fitness values can be converted directly to fidelities by subtracting the fitness from 1. The fidelity of the input state was 0.8 for all runs. Thus, only those runs with fitness < 0.2 were successful in finding an entanglement distillation protocol within the limited number of function evaluations. The time the CPU used to perform the optimization is given in the column “runtime CPU”. The optimizations have been performed on a cluster which automatically distributes the jobs to specific nodes. Those nodes have different CPUs. The speed of the CPU is indicated by the so called “CPU factor”. The corresponding CPU models are: 6.5: AMD Opteron 250, 7.5: AMD Opteron 8380, 8.0: AMD Opteron 2220

run id	initial parameters	init. σ	λ	total # f. e.	needed # f. e.	initial fitness	final fitness	runtime CPU	CPU factor
1	$\mathscr{U}([-\pi,\pi])$	0.01	26	44773	44461	0.73277	0.17946	11h 33min	6.5
2	same as 1	0.01	26	100023	99867	0.73277	0.17532	17h 56min	6.5
3	same as 1	0.01	26	95083	92587	0.73277	0.17578	n/a	n/a
8	same as 1	0.01	26	100023	100023	0.73277	0.17511	8h 08min	8.0
8.1	resumed from 8	-	26	300015	291490	-	0.16842	20h 53min	6.5
8.2	resumed from 8.1	-	26	600003	593867	-	0.16728	23h 57min	8.0
9	same as 1	0.01	26	100023	99815	0.73277	0.17534	10h 14min	6.5
9.1	resumed from 9	-	26	300015	297181	-	0.16825	20h 39min	6.5
15	$\mathscr{U}([-\pi,\pi])$	0.01	26	200019	196919	0.71485	0.20524	16h 53min	8.0
15.1	resumed from 15	-	26	600003	591033	-	0.20296	32h 20min	8.0
16	$\mathscr{U}([-\pi,\pi])$	0.01	26	200019	199161	0.71485	0.20520	16h 40min	8.0
16.1	resumed from 16	-	26	600003	590487	-	0.20036	32h 45min	8.0
17	$\mathscr{U}([-\pi,\pi])$	0.01	26	200019	195833	0.67071	0.17284	16h 05min	8.0

Table C.5: Listing of the performed optimization runs for the case of entanglement distillation for 2 copies of a pure input state together with 2 ancilla qubits per party (Section 5.4). The "run id" is an arbitrary identifier of the run. In the column "initial parameters", either the distribution with which the initial parameters have been sampled is given or else the parameters were exactly the same as in a previous optimization run ("same as …") or the corresponding run has been continued from a former run ("resumed from …"). The dimensionality of the search space depends on the restrictions that have been applied. If the value in the column "sym." is true, then the unitary matrices $\mathbf{U}_A$ and $\mathbf{U}_B$ were restricted to be the same. If the column "orth." contains a true, then $\mathbf{U}_A$ and $\mathbf{U}_B$ were restricted to be orthogonal matrices. In the case of orthogonal matrices, the sign of the determinant is either set by a parameter which is indicated by the term "param" in the column denoted as "det.". Otherwise, the column "det." contains the value to which the determinant was set. For an explanation of the algorithm specific parameters σ and λ, refer to Section 5.2. The value given in the column "total # f. e." is the total number of function evaluations, whereas the value given in "needed # f. e." is the number of function evaluations needed to reach the final fitness value given in the column "final fitness" up to an accuracy of 5 decimals. The given fitness values can be converted directly to fidelities by subtracting the fitness from 1. The fidelity of the input state was 0.8 for all runs. Thus, only those runs with fitness < 0.2 were successful in finding an entanglement distillation protocol within the limited number of function evaluations. The time the CPU used to perform the optimization is given in "runtime CPU". The optimizations have been performed on a cluster which automatically distributes the jobs to specific nodes. Those nodes have different CPUs. The speed of the CPU is indicated by the so called "CPU factor". The corresponding CPU models are: 6.5: AMD Opteron 250, 7.5: AMD Opteron 8380, 8.0: AMD Opteron 2220

run id	initial param.	sym.	orth.	det.	init. σ	λ	total # f. e.	needed # f. e.	initial fitness	final fitness	runtime CPU	CPU factor
sr1	$\mathscr{U}([-\pi,\pi])$	true	true	1	0.1	18	226063	51913	0.50584	0.20000	0h 25min	8.0
sr2	$\mathscr{U}([-\pi,\pi])$	true	true	-1	0.1	18	164665	23779	0.50472	0.20000	0h 19min	8.0
sr3	all zero	true	true	-1	0.1	18	41113	0	0.20000	0.20000	0h 05min	7.5
sr4	all zero	true	true	1	0.1	18	41419	0	0.20000	0.20000	0h 05min	8.0
r1	all zero	false	true	param.	0.1	20	125601	0	0.20000	0.20000	0h 19min	7.5
r2	all zero	false	true	param.	0.1	20	232441	0	0.20000	0.20000	0h 38min	7.5
r3	$\mathscr{U}([-\pi,\pi])$	false	true	1	0.1	20	748661	13421	0.55897	0.20000	1h 02min	8.0
r4	$\mathscr{U}([-\pi,\pi])$	false	true	1	0.1	20	583841	4978	0.51957	0.23000	1h 35min	8.0
g1	$\mathscr{U}([-\pi,\pi])$	false	false	-	0.1	22	300015	283845	0.77006	0.20001	1h 40min	8.0
g1.2	resumed from g1	false	false	-	-	22	1000011	15750	-	0.20000	4h 18min	8.0
g2	$\mathscr{U}([-\pi,\pi])$	false	false	-	0.1	22	300015	197935	0.77387	0.20000	1h 36min	7.5
g2.2	resumed from g2	false	false	-	-	22	567909	197935	-	0.20000	1h 51min	8.0

Table C.6: Listing of the performed optimization runs for the case of entanglement distillation for 3 copies of a Werner state (5.10) together with 2 ancilla qubits per party (Section 5.5). The "run id" is an arbitrary identifier of the run. In the column "initial parameters", either the distribution with which the initial parameters have been sampled is given or else the parameters were exactly the same as in a previous optimization run ("same as ...") or the corresponding run is a continuation of a former run ("resumed from ..."). The dimensionality of the search space depends on the restrictions that have been applied. If the value in the column "sym." is true, then the unitary matrices $\mathbf{U}_A$ and $\mathbf{U}_B$ were restricted to be the same. If the column "orth." contains a true, then $\mathbf{U}_A$ and $\mathbf{U}_B$ were restricted to be orthogonal matrices. In the case of orthogonal matrices, the sign of the determinant is either set by a parameter which is indicated by the term "param" in the column denoted as "det.". Otherwise, the column "det." contains the value to which the determinant was set. For an explanation of the algorithm specific parameters σ and λ, refer to Section 5.2. The value given in the column "total # f. e." is the total number of function evaluations, whereas the value given in "needed # f. e." is the number of function evaluations needed to reach the final fitness value given in the column "final fitness" up to an accuracy of 5 decimals. The given fitness values can be converted directly to fidelities by subtracting the fitness from 1. The fidelity of the input state was 0.925 for all runs. Thus, only those runs with fitness smaller than 0.075 were successful in finding an entanglement distillation protocol within the limited number of function evaluations. The time the CPU used to perform the optimization is given in "runtime CPU". The optimizations have been performed on a cluster which automatically distributes the jobs to specific nodes. Those nodes have different CPUs. The speed of the CPU is indicated by the so called "CPU factor". The corresponding CPU models are: 6.5: AMD Opteron 250, 7.5: AMD Opteron 8380, 8.0: AMD Opteron 2220

run id	initial param.	sym.	orth.	det.	init. σ	λ	total # f. e.	needed # f. e.	initial fitness	final fitness	runtime CPU	CPU factor
sr1	$\mathscr{U}([-\pi,\pi])$	true	true	param.	0.01	22	91983	91631	0.50373	0.12013	34h 20min	6.5
sr2	same as 1	true	true	param.	0.01	22	91983	91125	0.50373	0.12009	34h 05min	6.5
sr3	all zero	true	true	+1	0.01	22	100013	1	0.07500	0.07500	18h 34min	8.0
sr4	all zero	true	true	+1	0.01	22	100013	1	0.07500	0.07500	18h 20min	8.0
sr5	all zero	true	true	-1	0.01	22	100013	1	0.11888	0.11888	18h 34min	8.0
sr6	all zero	true	true	-1	0.01	22	100013	1	0.11888	0.11888	18h 28min	8.0
s1	$\mathscr{U}([-\pi,\pi])$	true	false	-	0.01	24	12577	12565	0.71130	0.37129	28h 33min	8.0
s1.1	resumed from s1	true	false	-	-	24	47665	46199	-	0.17446	34h 48min	8.0
s2	$\mathscr{U}([-\pi,\pi])$	true	false	-	0.01	24	51721	51516	0.67241	0.13511	34h 38min	8.0
g1	$\mathscr{U}([-\pi,\pi])$	false	false	-	0.01	26	8009	7984	0.74997	0.53858	24h 20min	6.5
g2	same as g1	false	false	-	0.01	26	8009	7984	0.74997	0.54449	18h 44min	8.0

Appendix D

Source Code

Here we provide the MATLAB source code to the numerical optimization which has been used to obtain the results stated in Chapter 5.

The central routine which evaluates the objective function, is in the file "FITNESS.M" (Section D.9). The CMA-ES optimization algorithm is implemented in "CMAES.M" a file which has been published by Nico Hansen [47] (not redistributed in this report). In order to initialize all the parameters of the algorithm and the function which computes the objective function value one can use the function implemented in the file "TEST_CONTOPT.M" (Section D.41).

D.1 ANGLE_CROSSOVER.M

```
% angle_crossover.m: 10.01.2011, Yves Salathe
function x = angle_crossover(parents, options, nvars, FitnessFcn,
    unused,thisPopulation)
% crossover function which can be used in a genetic algortihm
% The genome, which is used as an input for the fitness function is
% encoded as follows: every pair of bits encodes an angle
% 00: 0 degrees
% 01: 90 degrees
% 10: 180 degrees
% 11: 270 degrees
% The crossover works by selecting a random pair of bits in the genome
% of two parent, split the genome at that position and recombine the
% genomes to produce two offsprings
r = randi(nvars/2,length(parents)/2)*2-2;
x = zeros(length(parents)/2,nvars);
for i = 1:size(x,1)
    x(i,1:r(i)) = thisPopulation(parents(2*i-1),1:r(i));
    x(i,(r(i)+1):end) = thisPopulation(parents(2*i),(r(i)+1):end);
end
```

D.2 CHOOSE_K_FROM_N.M

```
% choose_k_from_n.m: 15.03.2011, Yves Salathe
function M = choose_k_from_n(n,k,ind1,ind2,j,M)
% choose k elements from a list of n elements in every possible way and
% return a matrix containing the choices
```

```
if nargin<6
    M = zeros(nchoosek(n,k),k);
    ind1 = 1;
    ind2 = 1;
    j = 1;
end
    for i = ind1:(n-k+ind2)
     %   if i<n
            l = nchoosek(n-i,k-ind2);
            M(j:(j+l-1),ind2) = repmat(i,l,1);


        if ind2<k
            M = choose_k_from_n(n,k,i+1,ind2+1,j,M);
        end
        j=j+l;
    end
```

D.3 COMPUTATIONAL_BASIS.M

```
% computational_basis.m: 10.05.2010, Yves Salathe
function psi = computational_basis(dim, state_nr)
% this function produces the qubit state |state_nr> in the
    computational basis
% of dimension given by dim.
if (state_nr >= dim); error('input argument "state_nr" has to be
    smaller than "dim"'); end;
psi = sparse(dim,1);
psi(state_nr+1) = 1;
return
```

D.4 CONDITIONAL_SWAP.M

```
% conditional_swap.m: 07.01.2011, Yves Salathe
function U = conditional_swap(ancilla_side,n_copies,n_ancillas)
% A unitary operation that swaps the states of two qubits based on the
% sate of the ancilla qubits
% ancilla_side can be either 'left' or 'right' depending on whether
% the ancilla qubits are multplied by tensor product from the left
% or from the right onto the state space

dim_copies = 2^n_copies;
dim_ancillas = 2^n_ancillas;
dim = dim_copies*dim_ancillas;
swap_states = [0,2];
P_Qubits = sparse([1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]); % permutes a
    pair of Qubits

if (strcmp(ancilla_side,'left'))
    ancillas_left = true;
elseif (strcmp(ancilla_side,'right'))
    ancillas_left = false;
else
    error('ancilla_side has to be either "left" or "right"');
end;
```

```
U = sparse(dim,dim);
for i = 0:dim_ancillas-1
    ancillas_state = computational_basis(dim_ancillas,i);
    ancillas_state = ancillas_state*ancillas_state';
    U_loc = speye(dim_copies);
    if ismember(i,swap_states)
        for j = 1:n_copies-1
            P = kron(speye(2^(n_copies-j-1)),kron(P_Qubits,speye(2^(j
                -1))));
            U_loc = P*U_loc;
        end
    end
    if ancillas_left
       U_loc = kron(ancillas_state,U_loc);
    else
       U_loc = kron(U_loc,ancillas_state);
    end
    U = U + U_loc;
end
```

D.5 CROSSOVER.M

```
% crossover.m: 17.05.2010, Yves Salathe
function P = crossover(n,k,l)
% permutation that acts on a n-Qubit system which moves the Qubit at
% position k to position l

dim = 2^n;
direction = +1; % +1 = forward movement; -1 = backward
if l < k; direction = -1; end;

P_Qubits = sparse([1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]); % permutes a
    pair of Qubits
P = speye(dim);
for i = k:direction:(l-direction)
    P_loc = P_Qubits;
    if direction == 1
      if i > 1; P_loc = kron(speye(2^(i-1)),P_loc); end;
      if i < n-1; P_loc = kron(P_loc,speye(2^(n-i-1))); end;
    else
      if i > 2; P_loc = kron(speye(2^(i-2)),P_loc); end;
      if i < n; P_loc = kron(P_loc,speye(2^(n-i))); end;
    end
    P = P_loc*P;
end

return;
```

D.6 DECODE_DISCRETE_ANGLES.M

```
% decode_discrete_angles.m: 10.01.2011, Yves Salathe
function t = decode_discrete_angles(x)
% decoder function which can be used in a genetic algortihm
% The genome, which is used as an input for the fitness function is
% encoded as follows: every pair of bits encodes an angle
% 00: 0 degrees
% 01: 90 degrees
% 10: 180 degrees
% 11: 270 degrees
angles = [0; pi/2; pi; 3*pi/2];
if mod(length(x)-1,2) ~= 0; error('x has to have length 2*n+1, where n
    is an integer'); end;
n = (length(x)-1)/2;
% first bit tells sign of determinant
detsig = 2*x(1)-1;
% decode x
B = reshape(x(2:end),n,2);
s = 2*B(:,1)+B(:,2)+1;
t = [detsig;angles(s)];
```

D.7 ENTROPY_OF_ENTANGLEMENT.M

```
% entropy_of_entanglement.m: 11.03.2011, Yves Salathe
function E = entropy_of_entanglement(x,dimA,dimB)
% function that computes the entropy of entanglement E for a state x
% x can be either a vector (pure state) or a density matrix

% partial trace over Bob's subsystem
if(length(x) ~= dimA*dimB); error('x has to be the state vector or
    density matrix of a bipartite system where the subsystem of Alice (
    Bob) has dimension dimA (dimB)'); end;
rhoA = TrX(x,2,[dimA,dimB]);
% compute the eigenvalues of rhoA
lambda = eig(rhoA);
% compute the Von Neumann entropy of rhoA
tol = 1e-14;
if any(lambda+tol<0); error('rhoA should not have negative eigenvalues'
    ); end;
lambda = lambda(lambda>tol);
E = -sum(lambda.*log2(lambda));
```

D.8 EXPECTED_E.M

```
% expected_E.m: 17.05.2010, Yves Salathe
function E = expected_E(x,n)
% A function which calculates the expected entropy of entanglement for
    the
% Schmidt Projection Method if used with n input qubits on a state
    given
% by: sqrt(x)*|0> + sqrt(1-x^2)*|1>. If n and x are vectors, then the
% expected entanglement is calculated for every combination of the
% elements of n and x and the results are returned in a matrix.
x = x(:);
E = zeros(length(x),length(n));
for i = 1:length(n);
    for k = 1:n(i)-1;
        E(:,i) = E(:,i) + x.^(n(i)-k).*(1-x).^k.*nchoosek(n(i),k).*log2
            (nchoosek(n(i),k));
    end;
end;
```

D.9 FITNESS.M

```
% fitness.m: 08.01.2011, Yves Salathe
function [fit,rho_new] = fitness(x,args)
% measure the fitness of a distillation protocol specified by the input
% vector x. args is a structure which controls the interpretation of
    the
% input vector x (i.e. mapping to unitary matrices) as well as the
% calculation of the fitness value (e.g. adding a penalty).

% generate unatry operators that act on the subsystems of Alice and Bob
[Ua,Ub] = args.operator_fn(x,args);

% form the overall unitary operator
% permutation of the second and third qubit is needed to perform
% the operations individually on the distinct subsystems of Alice (Ua)
% and Bob (Ub)
U = kron(Ua,Ub);

U = args.P'*U*args.P;

if not(isempty(args.psi_init)) % pure initial state
    % apply the unitary operator to the state
    new_global_state = U*args.psi_init_global;
else % mixed initial state
    new_global_state = U*args.rho_init_global*U';
end

% trace out all ancilla bits and all copies except one
rho_new = TrX_fast(new_global_state,args.trace_out,...
        args.trace_keep,args.trace_dimkeep,args.trace_dimtrace,...
        args.trace_rdim,args.trace_n);

% measure the new fidelity
F = abs(args.psi_ref'*rho_new*args.psi_ref);

% the corresponding fitness is the distance to one (which is be
    minimized)
fit = 1-F;

% penalize some solutions
if not(isempty(args.penalty_fn))
    fit = fit + args.penalty_fn(x,args,Ua,Ub,U,new_global_state,rho_new
        ,F);
end
```

D.10 GATES.M

```
% gates.m: 17.05.2010, Yves Salathe
function G = gates(n_qubits)
% construct a set of quantum gates

dim = 2^n_qubits;

theta = acos(3/5);

H = sparse([1 1; 1 -1]/sqrt(2));
R = sparse([1 0; 0 exp(2*pi*1i*theta)]);
CNOT = sparse([1 0 0 0; 0 1 0 0; 0 0 0 1; 0 0 1 0]);
```

```
single_qubit_op = {H,R};
two_qubit_op = {CNOT};

G = {};

% single qubit operations
for i = 1:length(single_qubit_op)
  for j = 1:n_qubits
    U = single_qubit_op{i};
    if j > 1; U = kron(speye(2^(j-1)),U); end;
    if j < n_qubits; U = kron(U,speye(2^(n_qubits-j))); end;
    G = [ G, {U} ];
  end
end

% two qubit operations
for i = 1:length(two_qubit_op)
  for j = 1:n_qubits
    U_loc = two_qubit_op{i};
    if j < n_qubits-1; U = kron(U,speye(2^(n_qubits-j-1))); end;
    for k = 1:(j-1)
      U = U_loc;
      if j > 2; U = kron(speye(2^(j-2)),U); end;
      if j < n_qubits; U = kron(U,speye(2^(n_qubits-j))); end;
      P = crossover(n_qubits,k,j);
      G = [ G, {P'*U*P} ];
    end
    for k = (j+1):n_qubits
      U = U_loc;
      if j > 1; U = kron(speye(2^(j-1)),U); end;
      if j < n_qubits-1; U = kron(U,speye(2^(n_qubits-1-j))); end;
      P = crossover(n_qubits,k,j+1);
      G = [ G, {P'*U*P} ];
    end
  end
end

return;
```

D.11 HERMITIAN_INVERSE.M

```
%hermitean_inverse.m: 18.05.2010, Yves Salathe
function [re im] = hermitian_inverse(H)
% inverse of the mapping to a hermitian nxn matrix out of input
% vectors re and im
% the dimension n of the matrix is determined by the length m of re
% such that m = n*(n+1)/2
% im has to be of length k = n*(n-1)/2
n = size(H,1);
if n ~= size(H,2); error('H has to be a square matrix'); end;
m = .5*n*(n+1);
% check length of im
k = .5*n*(n-1);
re = zeros(m,1);
im = zeros(k,1);
re_ind = 1;
im_ind = 1;
for i=1:1:n
   re_ind2 = re_ind+n-i;
   im_ind2 = im_ind+n-i-1;
   % real part
   re(re_ind:re_ind2) = real(H(i,i:n));
   % add imaginary part
   im(im_ind:im_ind2) = imag(H(i,(i+1):n));
   re_ind = re_ind2+1;
   im_ind = im_ind2+1;
end
```

D.12 HERMITIAN.M

```
% hermitean.m: 17.05.2010, Yves Salathe
function H = hermitian(re, im)
% form a hermitian nxn matrix out of input vectors re and im
% the dimension n of the matrix is determined by the length m of re
% such that m = n*(n+1)/2
% im has to be of length k = n*(n-1)/2
m = length(re);
n = .5*(sqrt(1+8*m)-1);
% check whether n is an integer
if (n - floor(n) ~= 0), error('the length m of the input vector re has
    to be chosen in such a way that m = n*(n+1)/2 has an integer
    solution n.'), end;
% check length of im
k = .5*n*(n-1);
if (length(im)~=k), error(['the input vector im has to be of length ',
    num2str(k)]), end;
H = zeros(n);
re_ind = 1;
im_ind = 1;
% transform im into a column vector
im = im(:);
for i=1:1:n
   re_ind2 = re_ind+n-i;
   im_ind2 = im_ind+n-i-1;
   % real part
   H(i,i:n) = re(re_ind:re_ind2);
   H((i+1):n,i) = re(re_ind+1:re_ind2);
   % add imaginary part
   H(i,(i+1):n) = H(i,(i+1):n) + 1i*im(im_ind:im_ind2)';
   H((i+1):n,i) = H((i+1):n,i) - 1i*im(im_ind:im_ind2);
   re_ind = re_ind2+1;
   im_ind = im_ind2+1;
end
```

D.13 IDENTITY.M

```
% identity.m: 07.01.2011, Yves Salathe
function [ Ua,Ub ] = identity( args )
% simple function which returns two identity matrices with the
% properties given by args. This function is useful to initialize the
% optimization
    n_copies = args.n_copies;
    n_ancillas_A = args.n_ancillas_A;
    n_ancillas_B = args.n_ancillas_B;
    Ua = speye(2^(n_copies+n_ancillas_A));
    Ub = speye(2^(n_copies+n_ancillas_B));
```

```
end
```

D.14 INIT_FITNESS_ARGS.M

```
% fitness_args.m: 08.01.2011, Yves Salathe
function fitness_args = init_fitness_args(n_copies,n_ancillas_A,
    n_ancillas_B,initstate,operator_fn,operator_args,penalty_fn,
    penalty_args)

% computational basis
psi_0 = [1 0]';
psi_1 = [0 1]';
psi_00 = kron(psi_0,psi_0);
psi_11 = kron(psi_1,psi_1);

% arguments for fitness function
fitness_args = struct('n_copies',{}...
    ,'n_ancillas_A',{}...
    ,'n_qubits_A',{}...
    ,'n_qubits_B',{}...
    ,'dim_A',{}...
    ,'dim_B',{}...
    ,'dim',{}...
    ,'psi_ref',{}...
    ,'psi_init',{}...
    ,'psi_init_global',{}...
    ,'rho_init',{}...
    ,'rho_init_V',{}...
    ,'rho_init_D',{}...
    ,'rho_init_S',{}...
    ,'rho_init_global',{}...
    ,'F_init',{}...
    ,'P',{}...
    ,'dim_subsystems',{}...
    ,'trace_out',{}...
    ,'trace_keep',{}...
    ,'trace_dimkeep',{}...
    ,'trace_dimtrace',{}...
    ,'trace_rdim',{}...
    ,'trace_n',{}...
    ,'operator_fn',{}...
    ,'operator_args',{});

fitness_args(1).n_copies = n_copies;
fitness_args(1).n_ancillas_A = n_ancillas_A;
fitness_args(1).n_ancillas_B = n_ancillas_B;
fitness_args(1).n_qubits_A = n_copies+n_ancillas_A;
fitness_args(1).n_qubits_B = n_copies+n_ancillas_B;

% calculate the dimensions of the subsystems of Alice and Bob
fitness_args(1).dim_A = 2^fitness_args.n_qubits_A;
fitness_args(1).dim_B = 2^fitness_args.n_qubits_B;

% dimension of the combined system
fitness_args(1).dim = fitness_args.dim_A*fitness_args.dim_B;


% fully entangled state (reference)
fitness_args(1).psi_ref = (psi_00+psi_11)/sqrt(2);

% copy the initial state
```

```
initstate_global = initstate;
for i = 2:n_copies
    initstate_global = kron(initstate_global,initstate);
end

if isvector(initstate_global) % initial state is pure
    % add ancillas (initialized in the state psi_0)
    for i = 1:n_ancillas_A % add ancillas of Alice
        initstate_global = kron(initstate_global,psi_0);
    end
    for i = 1:n_ancillas_B % add ancillas of Bob
        initstate_global = kron(initstate_global,psi_0);
    end
    % initial pure state
    fitness_args(1).psi_init = initstate;
    fitness_args(1).psi_init_global = initstate_global;
    % initial density matrix
    fitness_args(1).rho_init = initstate*initstate';
    fitness_args(1).rho_init_global = initstate_global*initstate_global
        ';
else % initial state is mixed
    % add ancillas (initialized in the state psi_0)
    for i = 1:n_ancillas_A % add ancillas of Alice
        initstate_global = kron(initstate_global,psi_0*psi_0');
    end
    for i = 1:n_ancillas_B % add ancillas of Bob
        initstate_global = kron(initstate_global,psi_0*psi_0');
    end
    % initial density matrix
    fitness_args(1).rho_init = initstate;
    fitness_args(1).rho_init_global = initstate_global;
end

% eigenvalue decomposition of the initial density matrix (e.g. for
    penalty functions)
if issparse(fitness_args.rho_init)
    [V,D] = eigs(fitness_args.rho_init,4);
else
    [V,D] = eig(fitness_args.rho_init);
end
fitness_args(1).rho_init_V = V;
fitness_args(1).rho_init_D = D;

% Von Neumann entropy of the initial density matrix (e.g. for penalty
    functions)
e = diag(D);
fitness_args(1).rho_init_S = abs(-dot(e,log2(e+(e==0))));

% fidelity of the initial state
fitness_args(1).F_init = abs(fitness_args.psi_ref'*
    fitness_args.rho_init*fitness_args.psi_ref);

% permutation that simplfies operations on the distinct subsystems of
    Alice
% and Bob
fitness_args(1).P = permutation(n_copies,n_ancillas_A,n_ancillas_B);

% arguments for the tracing
keep_subsystem = 1;
dim_subsystems = repmat(4,1,n_copies);
if(n_ancillas_A > 0), dim_subsystems = [dim_subsystems 2^n_ancillas_A];
     end;
if(n_ancillas_B > 0), dim_subsystems = [dim_subsystems 2^n_ancillas_B];
     end;
n_subsystems = length(dim_subsystems);
```

```
trace_out = [1:(keep_subsystem-1) (keep_subsystem+1):n_subsystems];

[trace_out,dim_subsystems,keep,dimkeep,dimtrace,rdim,n] ...
        = TrX_fast_args(trace_out,dim_subsystems,fitness_args.dim);
fitness_args(1).dim_subsystems = dim_subsystems;
fitness_args(1).trace_out = trace_out;
fitness_args(1).trace_keep = keep;
fitness_args(1).trace_dimkeep = dimkeep;
fitness_args(1).trace_dimtrace = dimtrace;
fitness_args(1).trace_rdim = rdim;
fitness_args(1).trace_n = n;

% pointer to function that creates the unitary operation
fitness_args(1).operator_fn = operator_fn;

% arguments for the function that creates the unitary operation
fitness_args(1).operator_args = operator_args;

% pointer to function that penalizes some solutions ({} means no
    peanlty)
fitness_args(1).penalty_fn = penalty_fn;

% arguments for the function that penalizes some solutions
fitness_args(1).penalty_args = penalty_args;
```

D.15 INIT_OPERATOR_ARGS.M

```
% init_operator_args.m: 08.01.2011, Yves Salathe
function operator_args = init_operator_args(n_copies,n_ancillas_A,
    n_ancillas_B,unitary,symmetry,onlyreal,determinant)
if(nargin < 7)
    determinant = [];
end
operator_args = struct('n_real_entries_A',{}...
    ,'n_real_entries_B',{}...
    ,'n_imag_entries_A',{}...
    ,'n_imag_entries_B',{}...
    ,'n_entries_total',{}...
    ,'ind_max_real_A',{}...
    ,'ind_min_imag_A',{}...
    ,'ind_max_imag_A',{}...
    ,'ind_min_real_B',{}...
    ,'ind_max_real_B',{}...
    ,'ind_min_imag_B',{}...
    ,'unitary',{}...
    ,'symmetry',{}...
    ,'onlyreal',{}...
    ,'determinant',{}...
    );
% calculate the dimensions of the subsystems of Alice and Bob
dim_A = (2^n_copies)*(2^n_ancillas_A);
dim_B = (2^n_copies)*(2^n_ancillas_B);

dim = dim_A*dim_B;

if unitary
    if not(onlyreal)
        operator_args(1).n_real_entries_A = .5*dim_A*(dim_A+1);
        operator_args(1).n_real_entries_B = .5*dim_B*(dim_B+1);
    else
        operator_args(1).n_real_entries_A = .5*dim_A*(dim_A-1);
        operator_args(1).n_real_entries_B = .5*dim_B*(dim_B-1);
        if isempty(determinant)
           operator_args(1).n_real_entries_A = operator_args(1)
               .n_real_entries_A + 1;
           operator_args(1).n_real_entries_B = operator_args(1)
               .n_real_entries_B + 1;
        end
    end
    operator_args(1).n_imag_entries_A = .5*dim_A*(dim_A-1);
    operator_args(1).n_imag_entries_B = .5*dim_B*(dim_B-1);
else
    operator_args(1).n_real_entries_A = dim_A*dim_A;
    operator_args(1).n_real_entries_B = dim_B*dim_B;
    operator_args(1).n_imag_entries_A = dim_A*dim_A;
    operator_args(1).n_imag_entries_B = dim_B*dim_B;
end
operator_args(1).n_entries_total = operator_args.n_real_entries_A +
    operator_args.n_imag_entries_A + operator_args.n_real_entries_B +
    operator_args.n_imag_entries_B;
operator_args(1).ind_max_real_A = operator_args.n_real_entries_A;
if (not(onlyreal))
    operator_args(1).ind_min_imag_A = operator_args.ind_max_real_A+1;
    operator_args(1).ind_max_imag_A = operator_args.ind_max_real_A+
        operator_args.n_imag_entries_A;
    operator_args(1).ind_min_real_B = operator_args.ind_max_imag_A+1;
    operator_args(1).ind_max_real_B = operator_args.ind_max_imag_A+
        operator_args.n_real_entries_B;
else
    operator_args(1).ind_min_real_B = operator_args.ind_max_real_A+1;
    operator_args(1).ind_max_real_B = operator_args.ind_max_real_A+
        operator_args.n_real_entries_B;
end
if (not(onlyreal))
    operator_args(1).ind_min_imag_B = operator_args.ind_max_real_B+1;
end
operator_args(1).unitary = unitary;
operator_args(1).symmetry = symmetry;
operator_args(1).onlyreal = onlyreal;
operator_args(1).determinant = determinant;
return;
```

D.16 MATRIX_DIFFERENCES.M

```
% matrix_differences.m: 10.01.2011, Yves Salathe
function [d,J] = matrix_differences(x,T,detT,ind)
% computes the difference between a matrix T and the
% matrix parameterized by the vector x. This is useful
% as simple fitness function in a numerical optimization
% to find a certain parameterization of a matrix
Tx = orthogonal([detT;x]);
d = Tx(ind)-T(ind);

if nargout > 1 % Jacobian required
    J = zeros(length(d),length(x));
    for j = 1:length(x)
        dT = orthogonal_deriv([detT;x],j);
        J(:,j) = dT(ind);
    end
end
```

D.17 MATRIX_DISTANCE.M

```
% matrix_distance.m: 10.01.2011, Yves Salathe
function [d,g] = matrix_distance(x,T,detT)
% computes a distance measure between a matrix T and the
% matrix parameterized by the vector x. This is useful
% as simple fitness function in a numerical optimization
% to find a certain parameterization of a matrix
x = x(:);
Tx = orthogonal([detT;x]);
diff = Tx-T;
d = sum(sum(diff.^2));

if nargout > 1 % gradient required
    g = zeros(length(x),1);
    for i = 1:length(x)
        dT = orthogonal_deriv([detT;x],i);
        g(i) = 2*sum(sum(diff.*dT));
    end
end
```

D.18 MATRIX_DISTANCE_TO_FUN.M

```
% matrix_distance.m: 10.01.2011, Yves Salathe
function [d,g] = matrix_distance_to_fun(x,T,fun,varargin)
% computes a distance measure between a matrix T and the
% matrix parameterized by the vector x using the mapping computed by
    the
% function fun. This is useful
% as simple fitness function in a numerical optimization
% to find a certain parameterization of a matrix
x = x(:);
Tx = fun(x);
diff = Tx-T;
d = sum(sum(diff.^2));

if nargout > 1 && not(isempty(varargin)) % gradient required
    g = zeros(length(x),1);
    for i = 1:length(x)
        dT = varargin{1}(x,i);
        g(i) = 2*sum(sum(diff.*dT));
    end
end
```

D.19 MEASUREMENT_UNITARY.M

```
% measurement_unitary.m: 07.01.2011, Yves Salathe
function U = measurement_unitary(subspaces,ancilla_side,
    dim_untouched_system)
% function that generates a unitary matrix that represents a
    measurement
% of the projections onto the subspaces specified by the matrices
% in the cell array 'subspaces'
```

```
if (strcmp(ancilla_side,'left'))
    ancillas_left = true;
elseif (strcmp(ancilla_side,'right'))
    ancillas_left = false;
else
    error('ancilla_side has to be either "left" or "right"');
end;

dim_measured_system = size(subspaces{1},1);
n_ancillas = ceil(log2(length(subspaces)));
dim_ancillas = 2^n_ancillas;
dim = dim_measured_system*dim_ancillas*dim_untouched_system;
U = sparse(dim,dim);
for i = 0:length(subspaces)-1
    ancillas_input = computational_basis(dim_ancillas,i);
    for j = 0:length(subspaces)-1
        U_loc = kron(subspaces{j+1},speye(dim_untouched_system));
        ancillas_output = computational_basis(dim_ancillas,mod(i+j,
            length(subspaces)));
        ancillas_operator = ancillas_output*ancillas_input';
        if ancillas_left
           U_loc = kron(ancillas_operator,U_loc);
        else
           U_loc = kron(U_loc,ancillas_operator);
        end
        U = U + U_loc;
    end
end

for i = length(subspaces):(dim_ancillas-1)
    ancillas_state = computational_basis(dim_ancillas,i);
    ancillas_operator = ancillas_state*ancillas_state';
    U_loc = speye(dim_measured_system*dim_untouched_system);
    if ancillas_left
        U_loc = kron(ancillas_operator,U_loc);
    else
        U_loc = kron(U_loc,ancillas_operator);
    end
    U = U + U_loc;
end

return
```

D.20 OPERATOR_ANY.M

```
% operator_any.m: 08.01.2011, Yves Salathe
function [Ua,Ub] = operator_any(x,args)
% Build a matrix out of the input vector x
% directly interpret the entries of the input vector as the elements of
% the matrix. args specifies whether the input vector contains only the
% real part of the matrix and whether the unitary operations are
% determined separately for Alice and Bob or if they are the same for
% both.

Ua = reshape(x(1:args.operator_args.ind_max_real_A),args.dim_A,
    args.dim_A);
if (not(args.operator_args.onlyreal))
    Ua = Ua + 1i*reshape(x(args.operator_args.ind_min_imag_A:
        args.operator_args.ind_max_imag_A),args.dim_A,args.dim_A);
end
```

```
if(not(args.operator_args.symmetry))
    Ub = reshape(x(args.operator_args.ind_min_real_B:
        args.operator_args.ind_max_real_B),args.dim_B,args.dim_B);
    if not(args.operator_args.onlyreal)
        Ub = Ub + 1i*reshape(x(args.operator_args.ind_min_real_B:
            args.operator_args.ind_max_real_B),x(
            args.operator_args.ind_min_imag_B:end),args.dim_B,
            args.dim_B);
    end
else
    Ub = Ua;
end
```

D.21 OPERATOR_NORMALIZED.M

```
% operator_normalized.m:
function [Ua,Ub] = operator_normalized(x,args)
% Build a matrix out of the input vector x:
% normalize each column so that it sums up to 1

Ua = reshape(x,args.dim_A-1,args.dim_A);
s = sum(Ua.^2);
Ua = [Ua; zeros(1,args.dim_A)];
Ua(end,s<1) = sqrt(1-s(s<1));
Ua(:,s>1) = Ua(:,s>1)./repmat(s(s>1),args.dim_A,1);
end
Ub = Ua;
```

D.22 ORTHOGONAL_DERIV.M

```
% orthogonal_deriv.m: 10.01.2011, Yves Salathe
function dT = orthogonal_deriv(x,i,determinant)
% form a orthogonal nxn matrix out of input vectors x
% the dimension dim of the matrix is determined by the length m of x
% such that m = n(n-1)/2
% This function computes the derivative of this parameterization w.r.t.
% the i-th parameter
%
% this function uses the recursive parameterization described in the
    paper
% Raffenetti, R. C. and Ruedenberg, K. (1969), Parametrization of an
    orthogonal matrix in terms of generalized eulerian angles.
    International Journal of Quantum Chemistry, 4: 625âĂŞ634. doi: 10
    .1002/qua.560040725
%
% see also
%Hoffman, D., Raffenetti, R. & Ruedenberg, K. 1972 Generalization of
    Euler angles to N-dimensional orthogonal matrices. J. Mathem. Phys.
     13, 528âĂŞ532. (doi:10.1063/1.1666011)

if isempty(determinant)
    detsig = (x(1)>=0)-(x(1)<0);
    x = x(2:end);
else
    detsig = determinant;
end
```

```
x = x(:);
m = length(x);
if (i < 1 || i > m); error('i has to be an index between 1 and length(x
    )-1'); end;
dim = .5*(1+sqrt(1+8*m));
if (dim - floor(dim) ~= 0), error('the length m (+1 if no determinant
    is given) of the input vector re has to be chosen in such a way
    that m = n*(n+1)/2 has an integer solution n.'), end;
pihalf = pi/2;
indp = triu(repmat((1:dim)',1,dim),1);
indq = triu(repmat(1:dim,dim,1),1);
indp = indp(indp>0);
indq = indq(indq>0);
p = indp(i);
q = indq(i);
ind = (q-1)*(q-2)/2;
if ind > 0
    dT = orthogonal([detsig;x(1:ind)]);
else
    dT = detsig;
end
ind = ind + 1;
gamma = [x(ind:(ind+q-2));pihalf];
a = cos(gamma);
b = sin(gamma);
t = [dT, zeros(size(dT,1),1); zeros(1,size(dT,2)), 1];
s = zeros(size(t));
sigma = zeros(size(t));
s(1,:) = [zeros(1,size(dT,2)) -1];
for k = 1:(q-1)
    s(k+1,:) = b(k)*t(k,:)+a(k)*s(k,:);
end
% in the paper it reads t(:,k) but I think that's wrong.
sigma(p+1,1:(end-1)) = a(p)*t(p,1:(end-1))-b(p)*s(p,1:(end-1));
sigma(p+1,end) = -b(p)*s(p,end);
for k = (p+1):(q-1)
    sigma(k+1,:) = a(k)*sigma(k,:);
end
dT=[zeros(p-1,q); -b(p)*t(p,:)-a(p)*s(p,:); repmat(-b((p+1):q),1,q).*
    sigma((p+1):q,:)];
ind = ind + q - 1;
for n = (q+1):dim
    gamma = [x(ind:(ind+n-2));pihalf];
    t = [dT, zeros(size(dT,1),1); zeros(1,size(dT,2)), 1];
    s = zeros(size(t));
    for k = 1:(n-1)
        s(k+1,1:(n-1)) = sin(gamma(k))*t(k,1:(n-1))+cos(gamma(k))*s(k
            ,1:(n-1));
    end
    dT = repmat(cos(gamma),1,n).*t - repmat(sin(gamma),1,n).*s;
    ind = ind + n - 1;
end
```

D.23 ORTHOGONAL_DISCRETE.M

```
% orthogonal_discrete.m: 10.01.2011, Yves Salathe
% form an orthogonal matrix by a discrete set of Euler angles
function T = orthogonal_discrete(x)
angles = [0; pi/2; pi; 3*pi/2];
if mod(length(x)-1,2) ~= 0; error('x has to have length 2*n+1, where n
    is an integer'); end;
```

```
n = (length(x)-1)/2;
% first bit tells sign of determinant
detsig = 2*x(1)-1;
% decode x
B = reshape(x(2:end),n,2);
s = 2*B(:,1)+B(:,2)+1;
T = orthogonal([detsig;angles(s)]);
```

D.24 ORTHOGONAL_INVERSE.M

```
% orthogonal_inverse.m: 10.01.2011, Yves Salathe
function x = orthogonal_inverse(T,method,varargin)
% from an orthogonal nxn matrix determine the parameter vector x
% the dimension dim of the matrix is determined by the length m of x
% such that m = n(n-1)/2
methods = {'lsqnonlin';'fsolve';'fminunc';'fminsearch';'cmaes';'ga_cont
    ';'ga_discrete';'simulannealbnd'};
method_num = 0;
for i = 1:length(methods)
    if strcmp(method,methods{i}); method_num = i; end;
end
dim = size(T,1);
m = .5*dim*(dim-1);
detT = det(T);
fmin = Inf;
ind = triu(ones(dim),1)>0;
bestx = 2*pi*rand(m,1);
bestf = Inf;
iter = 1;
maxiter = 10;
while (fmin>1e-5 && iter <= maxiter)
    if not(isempty(varargin))
        x_init = varargin{1};
    else
        x_init = 2*pi*rand(m,1);
    end
    if     method_num == 1
        f = @(x) matrix_differences(x,T,detT,ind);
        options = optimset('Jacobian','on','Diagnostics','on','TolFun'
            ,1e-6,'TypicalX',pi*ones(m,1),'Display','iter');
        [x,fmin] = lsqnonlin(f,x_init,[],[],options);
    elseif method_num == 2
        f = @(x) matrix_differences(x,T,detT,ind);
        options = optimset('Jacobian','on','Diagnostics','on','TolFun'
            ,1e-6,'TypicalX',pi*ones(m,1),'Display','iter');
        [x,fmin] = fsolve(f,x_init,options);
        fmin = sum(sum(fmin.^2));
    elseif method_num == 3
        f = @(x) matrix_distance(x,T,detT);
        options = optimset('GradObj','on','Diagnostic','on','TolFun',1e
            -6,'TypicalX',pi*ones(m,1),'Display','iter');
        [x,fmin] = fminunc(f,x_init,options);
    elseif method_num == 4
        f = @(x) matrix_distance(x,T,detT);
        options = optimset('TolFun',1e-6);
        [x,fmin] = fminsearch(f,x_init,options);
    elseif method_num == 5
        OPTS = cmaes;
        %OPTS.DiagonalOnly='(1+100*N/sqrt(popsize))';   % C is diagonal
            for given iterations, 1==always
        OPTS.TolFun = 1e-6;
        OPTS.SaveVariables = 'off';
        OPTS.DispModulo='5  % [0:Inf], disp messages after every i-th
            iteration';
        [x,fmin] = cmaes('matrix_distance',x_init,pi*ones(m,1),OPTS,T,
            detT);
    elseif method_num == 6
        f = @(x) matrix_distance(x,T,detT);
        options = gaoptimset('Display','final');
        [x,fmin] = ga(f,m,options);
    elseif method_num == 7
        f = @(x) matrix_distance_to_fun(decode_discrete_angles([(detT
            +1)/2,x]),T,@orthogonal);
        options = gaoptimset('PopulationType','bitstring','MutationFcn'
            ,@mutationuniform,'CrossoverFcn',@angle_crossover);
        [x,fmin] = ga(f,2*m,options);
        x = decode_discrete_angles([(detT+1)/2,x]);
        x = x(2:end);
    elseif method_num == 8
        f = @(x) matrix_distance(x,T,detT);
        options = saoptimset('PlotFcns',{@saplotbestx,...
                @saplotbestf,@saplotx,@saplotf});
        [x,fmin] = simulannealbnd(f,x_init,zeros(m,1),2*pi*ones(m,1),
            options);
    else
        error(['method ', method, ' unknown']);
    end
    if fmin < bestf
        bestx = x;
        bestf = fmin;
    end
    disp(['iter = ', num2str(iter), ', fmin = ', num2str(fmin), ',
        bestf = ', num2str(bestf)]);
    save(['variables_iter_' num2str(iter) '.mat'],'-mat','x','fmin','
        bestx','bestf','method_num');
    save(['variables_x_iter_' num2str(iter) '.txt'],'-ASCII','x');
    % method_num = mod(method_num,length(methods)) + 1; % next time try
         another method
    iter = iter + 1;
end
if (fmin>1e-5); error('no good approximation of T has been found'); end
    ;
x = bestx(:);
x = [detT;x];
```

D.25 ORTHOGONAL.M

```
% orthogonal.m: 10.01.2011, Yves Salathe
function T = orthogonal(x,determinant)
% form an orthogonal nxn matrix out of input vectors x
% the dimension dim of the matrix is determined by the length m+1 of x
% such that m = n(n-1)/2
% the first component of the vector just indicates the sign of the
% determinant of the unitary matrix
%
% this function uses the recursive parameterization described in the
    paper
% Raffenetti, R. C. and Ruedenberg, K. (1969), Parametrization of an
    orthogonal matrix in terms of generalized eulerian angles.
    International Journal of Quantum Chemistry, 4: 625âĂŞ634. doi: 10
    .1002/qua.560040725
%
```

```
% see also
%Hoffman, D., Raffenetti, R. & Ruedenberg, K. 1972 Generalization of
    Euler angles to N−dimensional orthogonal matrices. J. Mathem. Phys.
     13, 528âĂŞ532. (doi:10.1063/1.1666011)

if isempty(determinant)
    T = (x(1)>=0)−(x(1)<0);
    x = x(2:end);
else
    T = determinant;
end
x = x(:);
m = length(x);
dim = .5*(1+sqrt(1+8*m));
if (dim − floor(dim) ~= 0), error('the length m (+1 if no determinant
    is given) of the input vector re has to be chosen in such a way
    that m = n*(n−1)/2 has an integer solution n.'), end;
pihalf = pi/2;
ind = 1;
for n = 2:dim
    gamma = [x(ind:(ind+n−2));pihalf];
    t = [T, zeros(size(T,1),1); zeros(1,size(T,2)), 1];
    s = zeros(size(t));
    s(1,:) = [zeros(1,size(T,2)) −1];
    for k = 1:(n−1)
        s(k+1,:) = sin(gamma(k))*t(k,:)+cos(gamma(k))*s(k,:);
    end
    T = repmat(cos(gamma),1,n).*t − repmat(sin(gamma),1,n).*s;
    ind = ind + n − 1;
end
```

D.26 PENALTY_IDENT.M

```
% penalty_ident.m: 17.05.2011, Yves Salathe
function p = penalty_ident(x,args,Ua,Ub,U,new_global_state,rho_new,F)
% penalize those solutions that do leave the initial state identical to
% what it was before application of the protocol
% do not penalize any solution that has a better fidelity than the
    inital
% fidelity
p = 0;
if F<args.F_init
    [V,D] = eig(rho_new);
    e = diag(D);
    e = e+(e==0);
    Srel = abs(−trace(args.rho_init*V*diag(log2(e))*V'))−
        args.rho_init_S;
    p = max(args.penalty_args.p_max−args.penalty_args.factor*Srel,0);
end
return;
```

D.27 PERMUTATION.M

```
% permutation.m: 08.01.2011, Yves Salathe
function P = permutation(n_copies,n_ancillas_A,n_ancillas_B)
% permute qubits of Alice and Bob in such a way, that the first n
    Qubits
```

```
% comprise the subsystem of Alice and the last n Qubits the one of Bob.
% This is useful in order to apply separate unitary operations on the
% two subsystems (i.e. kron(Ua,Ub))

% calculate the dimensions of the subsystems of Alice and Bob
dim_ancillas_A = 2^n_ancillas_A;
dim_ancillas_B = 2^n_ancillas_B;
dim_A = (2^n_copies)*dim_ancillas_A;
dim_B = (2^n_copies)*dim_ancillas_B;

% dimension of the entire system
dim = dim_A*dim_B;

% the following operator permutes two neighbouring qubits
P_Qubits = sparse([1 0 0 0; 0 0 1 0; 0 1 0 0; 0 0 0 1]); % permutes a
    pair of Qubits

% prepare permutation operator that acts on the whole system
P = speye(dim);


% permutation of the qubits, so that the unitary operations Ua and Ub
    act
% on the distinct subsystems of Alice and Bob
% (don't touch the ancillary qubits as they are initially separated)
for i = 1:(n_copies-1)
    n_permutations = n_copies-i;
    P_loc = speye(2^i);
    for j = 1:n_permutations
        P_loc = kron(P_loc,P_Qubits);
    end
    P_loc = kron(P_loc,speye(2^(i+n_ancillas_A+n_ancillas_B)));
    P = P_loc*P;
end


% now permute the ancillas of alice to the middle
for i = 1:n_ancillas_A
    for j = 1:n_copies
        P_loc = speye(2^(2*n_copies-j+i-1));
        P_loc = kron(P_loc,P_Qubits);
        P_loc = kron(P_loc,speye(2^(n_ancillas_A+n_ancillas_B-i+j-1)));
        P = P_loc*P;
    end
end
```

D.28 PLOTCMAESDAT_FITNESS.M

```
function plotcmaesdat(figNb, step, filenameprefix,filenameextension,
    objectvarname,logf)
% PLOTCMAESDAT;
% PLOTCMAES(FIGURENUMBER_iBEGIN_iEND, FILENAMEPREFIX, FILENAMEEXTENSION
    , OBJECTVARNAME);
%   plots output from CMA-ES, e.g. cmaes.m, Java class
    CMAEvolutionStrategy...
%   mod(figNb,100)==1 plots versus iterations.
%
% PLOTCMAES([101 300]) plots versus iteration, from iteration 300.
% PLOTCMAES([100 150 800]) plots versus function evaluations, between
    iteration 150 and 800.
%
```

```
% Upper left subplot: blue/red: function value of the best solution in
    the
%   recent population, cyan: same function value minus best
%   ever seen function value, green: sigma, red: ratio between
%   longest and shortest principle axis length which is equivalent
%   to sqrt(cond(C)).
% Upper right plot: time evolution of the distribution mean (default)
    or
%   the recent best solution vector.
% Lower left: principle axes lengths of the distribution ellipsoid,
%   equivalent with the sqrt(eig(C)) square root eigenvalues of C.
% Lower right: magenta: minimal and maximal "true" standard deviation
%   (with sigma included) in the coordinates, other colors: sqrt(diag(C
    ))
%   of all diagonal elements of C, if C is diagonal they equal to the
%   lower left.
%
% Files [FILENAMEPREFIX name FILENAMEEXTENSION] are used, where
%   name = axlen, OBJECTVARNAME (xmean|xrecentbest), fit, or stddev.
%
manual_mode = 1;

  if nargin < 1 || isempty(figNb)
    figNb = 325;
  end
  if nargin < 2 || isempty(step)
    step = 1;
  end
  if nargin < 3 || isempty(filenameprefix)
    filenameprefix = 'outcmaes';
  end
  if nargin < 4 || isempty(filenameextension)
    filenameextension = '.dat';
  end
  if nargin < 5 || isempty(objectvarname)
    objectvarname = 'xmean';
    objectvarname = 'xrecentbest';
  end
  if nargin < 6 || isempty(logf)
    logf = false;
  end

  % load data
  % d.x = load([filenameprefix objectvarname filenameextension]);
  % d.x = load([filenameprefix 'xmean' filenameextension]);
  % d.x = load([filenameprefix 'xrecentbest' filenameextension]);
  d.f = load([filenameprefix 'fit' filenameextension]);
  d.std = load([filenameprefix 'stddev' filenameextension]);
%  d.D = load([filenameprefix 'axlen' filenameextension]);

  % interpret entries in figNb for cutting out some data
  if length(figNb) > 1
    iend = inf;
    istart = figNb(2);
    if length(figNb) > 2
      iend = figNb(3);
    end
    figNb = figNb(1);
%     d.x = d.x(d.x(:,1) >= istart & d.x(:,1) <= iend, :);
    d.f = d.f(d.f(:,1) >= istart & d.f(:,1) <= iend, :);
    d.std = d.std(d.std(:,1) >= istart & d.std(:,1) <= iend, :);
%     d.D = d.D(d.D(:,1) >= istart & d.D(:,1) <= iend, :);
  end

  % set up figure window
  if manual_mode
    figure(figNb);  % just create and raise the figure window
  else  % try to prevent raise
    if evalin('caller', 'countiter') <= 2 && evalin('caller', 'irun')
        == 1
      figure(324);
    elseif gcf ~= 324
      if ismember(324, findobj('Type', 'figure'))
        set(0, 'CurrentFigure', 324);  % prevents raise of figure
            window
      else
        figure(324);
      end
    end
  end

  % decide for x-axis
  iabscissa = 2; % 1== versus iterations, 2==versus fevals
  if mod(figNb,100) == 1
    iabscissa = 1; % a short hack
  end
%  if iabscissa == 1
%    xlab ='iterations';
%  elseif iabscissa == 2
%    xlab = 'function evaluations';
%  end

%  if size(d.x, 2) < 1000
%    minxend = 1.03*d.x(end, iabscissa);
%  else
%    minxend = 0;
%  end

  % plot fitness etc
  foffset = 0;
  dfit = d.f(:,6)-min(d.f(:,6));
  [ignore idxbest] = min(dfit);
  dfit(dfit<1e-98) = NaN;
%  subplot(2,2,1); hold off;
  dd = abs(d.f(:,7:8)) + foffset;
  dd(d.f(:,7:8)==0) = NaN;
  subplot(2,2,1)
  if logf
      semilogy(d.f(1:step:end,iabscissa), dd(1:step:end,:), '-k'); hold
           on;
  else
      plot(d.f(1:step:end,iabscissa), dd(1:step:end,:), '-k'); hold on;
  end
  % additional fitness data, for example constraints values
  if size(d.f,2) > 8
    dd = abs(d.f(:,9:end)) + 10*foffset;   % a hack
    % dd(d.f(:,9:end)==0) = NaN;
    subplot(2,2,1)
    if logf
        semilogy(d.f(1:step:end,iabscissa), dd(1:step:end), '-m'); hold
             on;
    else
        plot(d.f(1:step:end,iabscissa), dd(1:step:end), '-m'); hold on;
    end
    if size(d.f,2) > 12
        if logf
            semilogy(d.f(1:step:end,iabscissa),abs(d.f(1:step:end,[7 8
                11 13]))+foffset,'-k'); hold on;
        else
```

```
          plot(d.f(1:step:end,iabscissa),abs(d.f(1:step:end,[7 8 11
              13]))+foffset,'-k'); hold on;
      end
   end
end

if logf
    idx = find(d.f(1:step:end,6)>1e-98);  % positive values
    if ~isempty(idx)  % otherwise non-log plot gets hold
      semilogy(d.f(idx,iabscissa), d.f(idx,6)+foffset, 'b'); hold on;
    end
    idx = find(d.f(1:step:end,6) < -1e-98);  % negative values
    if ~isempty(idx)
      semilogy(d.f(idx, iabscissa), abs(d.f(idx,6))+foffset,'r');
          hold on;
    end
    semilogy(d.f(1:step:end,iabscissa),abs(d.f(1:step:end,6))+foffset
        ,'-b'); hold on;
else
    plot(d.f(1:step:end,iabscissa), d.f(1:step:end,6)+foffset, '.-b')
        ; hold on;
end


subplot(2,2,2)
semilogy(d.f(1:step:end,iabscissa),dfit(1:step:end),'-c'); hold on;
subplot(2,2,4)
plot(d.f(1:step:end,iabscissa),(d.f(1:step:end,4)),'-r'); hold on; %
    AR
subplot(2,2,3)
semilogy(d.std(1:step:end,iabscissa), [max(d.std(1:step:end,6:end),
    [], 2) ...
      min(d.std(1:step:end,6:end), [], 2)], '-m'); % max,min std
maxval = max(d.std(end,6:end));
minval = min(d.std(end,6:end));
text(d.std(end,iabscissa), maxval, sprintf('%.0e', maxval));
text(d.std(end,iabscissa), minval, sprintf('%.0e', minval));

subplot(2,2,3)
semilogy(d.std(1:step:end,iabscissa),(d.std(1:step:end,3)),'-g'); %
    sigma
% plot best f
subplot(2,2,2)
semilogy(d.f(idxbest,iabscissa),min(dfit),'*c'); hold on;
subplot(2,2,1)
if logf
  semilogy(d.f(idxbest,iabscissa),abs(d.f(idxbest,6))+foffset,'*r');
      hold on;
else
  plot(d.f(idxbest,iabscissa),abs(d.f(idxbest,6))+foffset,'*r'); hold
       on;
end

subplot(2,2,1)
ax = axis;
% ax(2) = max(minxend, ax(2));
axis(ax);

if logf
  xannote = 10^(log10(ax(1)) + 0.05*(log10(ax(2))-log10(ax(1))));
  yannote = 10^(log10(ax(3)) + 0.05*(log10(ax(4))-log10(ax(3))));
else
  xannote = ax(1) + 0.05*(ax(2)-ax(1));
  yannote = ax(3) + 0.05*(ax(4)-ax(3));
end
```

```
  text(xannote, yannote, ...
       [ 'f=' num2str(d.f(end,6), '%.15g') ]);

text_xlabel = '# fun. eval.';
subplot(2,2,1)
title('(a)')
xlabel(text_xlabel);
ylabel('fitness value f');
grid on;
subplot(2,2,2)
title('(b)')
xlabel(text_xlabel);
ylabel('f-min(f)');
grid on;
subplot(2,2,3)
title('(c)')
xlabel(text_xlabel);
ylabel('sigma');
grid on;
subplot(2,2,4)
title('(d)')
xlabel(text_xlabel);
ylabel('axis ratio');
grid on;

%  title('blue:abs(f), cyan:f-min(f), green:sigma, red:axis ratio');
%  ax([3,4]) = [1e-9, 1e5];
%  axis(ax);

%  % plot x-values
%  subplot(2,2,2); hold off;
%  plot(d.x(:,iabscissa), d.x(:,6:end),'-'); hold on;
%  ax = axis;
%  ax(2) = max(minxend, ax(2));
%  axis(ax);
%
%  % add some annotation lines
%  [ignore idx] = sort(d.x(end,6:end));
%  % choose no more than 25 indices
%  idxs = round(linspace(1, size(d.x,2)-5, min(size(d.x,2)-5, 25)));
%  yy = repmat(NaN, 2, size(d.x,2)-5);
%  yy(1,:) = d.x(end, 6:end);
%  yy(2,idx(idxs)) = linspace(ax(3), ax(4), length(idxs));
%  plot([d.x(end,iabscissa) ax(2)], yy, '-');
%  plot(repmat(d.x(end,iabscissa),2), [ax(3) ax(4)], 'k-');
%  for i = idx(idxs)
%    text(ax(2), yy(2,i), ...
%         ['x(' num2str(i) ')=' num2str(yy(1,i), '%.3g')]);
%  end
%  title(['Object Variables (' num2str(size(d.x, 2)-5) '-D)']);grid on;
%
%  subplot(2,2,3); hold off; semilogy(d.D(:,iabscissa), d.D(:,6:end),
    '-');
%  ax = axis;
%  ax(2) = max(minxend, ax(2));
%  axis(ax);
%  title('Principle Axes Lengths');grid on;
%  xlabel(xlab);
%
%  subplot(2,2,4); hold off;
%  % semilogy(d.std(:,iabscissa), d.std(:,6:end), 'k-'); hold on;
%  % remove sigma from stds
%  d.std(:,6:end) = d.std(:,6:end) ./ (d.std(:,3) * ones(1,size(d.std
    ,2)-5));
%  semilogy(d.std(:,iabscissa), d.std(:,6:end), '-'); hold on;
%  if 11 < 3  % max and min std
%    semilogy(d.std(:,iabscissa), [d.std(:,3).*max(d.std(:,6:end), [],
    2) ...
%                        d.std(:,3).*min(d.std(:,6:end), [], 2)], '-m',
     'linewidth', 2);
%    maxval = max(d.std(end,6:end));
%    minval = min(d.std(end,6:end));
%    text(d.std(end,iabscissa), d.std(end,3)*maxval, sprintf('max=%.0e
    ', maxval));
%    text(d.std(end,iabscissa), d.std(end,3)*minval, sprintf('min=%.0e
    ', minval));
%  end
%  ax = axis;
%  ax(2) = max(minxend, ax(2));
%  axis(ax);
%  % add some annotation lines
%  [ignore idx] = sort(d.std(end,6:end));
%  % choose no more than 25 indices
%  idxs = round(linspace(1, size(d.x,2)-5, min(size(d.x,2)-5, 25)));
%  yy = repmat(NaN, 2, size(d.std,2)-5);
%  yy(1,:) = d.std(end, 6:end);
%  yy(2,idx(idxs)) = logspace(log10(ax(3)), log10(ax(4)), length(idxs))
    ;
%  semilogy([d.std(end,iabscissa) ax(2)], yy, '-');
%  semilogy(repmat(d.std(end,iabscissa),2), [ax(3) ax(4)], 'k-');
%  for i = idx(idxs)
%    text(ax(2), yy(2,i), [' ' num2str(i)]);
%  end
%  title('Standard Deviations in Coordinates divided by sigma');grid on
    ;
%  xlabel(xlab);

  if figNb ~= 324
    % zoom on;  % does not work in Octave
  end
  drawnow;
```

D.29 PLOT_FIDELITY.M

```
% plot_fidelity.m: 28.02.2011, Yves Salathe
function [F,rho_new,rho,rhonz,ind_row,ind_col,lables,E_input,E_output,
    E_max] = plot_fidelity(n_copies,n_ancillas_A,n_ancillas_B,Ua,Ub,c)
% this function plots the fidelity of a protocol specified by the
% unitary matrices Ua and Ub if applied on the pure state
% c|00> + sqrt(1-c^2)|11>
psi_0 = [1 0]';
psi_1 = [0 1]';
psi_00 = kron(psi_0,psi_0);
psi_11 = kron(psi_1,psi_1);
psi_ref = (psi_00+psi_11)/sqrt(2);
F = zeros(length(c),1);
E_input = zeros(length(c),1);
E_max = zeros(length(c),1);
E_output = zeros(length(c),1);
rho_new = cell(length(c),1);
rho = zeros(length(c),16);

% arguments for the tracing
keep_subsystem = 1;
dim_subsystems = repmat(4,1,n_copies);
```

```
if(n_ancillas_A > 0), dim_subsystems = [dim_subsystems 2^n_ancillas_A];
     end;
if(n_ancillas_B > 0), dim_subsystems = [dim_subsystems 2^n_ancillas_B];
     end;
dimA = 2^(n_copies+n_ancillas_A);
dimB = 2^(n_copies+n_ancillas_B);
n_subsystems = length(dim_subsystems);
trace_out = [1:(keep_subsystem-1) (keep_subsystem+1):n_subsystems];
P = permutation(n_copies,n_ancillas_A,n_ancillas_B);
for j = length(c):-1:1
    initstate = c(j)*psi_00+sqrt(1-c(j)^2)*psi_11;
    % also measure the entropy of entanglement of initstate
    E_input(j) = entropy_of_entanglement(initstate,2,2);
    % copy the initial state
    initstate_global = initstate;
    for i = 2:n_copies
        initstate_global = kron(initstate_global,initstate);
    end
    % add ancillas (initialized in the state psi_0)
    for i = 1:n_ancillas_A % add ancillas of Alice
        initstate_global = kron(initstate_global,psi_0);
    end
    for i = 1:n_ancillas_B % add ancillas of Bob
        initstate_global = kron(initstate_global,psi_0);
    end
    % also measure the entropy of entanglement of initstate_global
    %E_max(j) = entropy_of_entanglement(P*initstate_global,dimA,dimB);


    % form the overall unitary operator
    % permutation of the second and third qubit is needed to perform
    % the operations individually on the distinct subsystems of Alice (
        Ua)
    % and Bob (Ub)
    U = kron(Ua,Ub);
    U = P'*U*P;
    % apply the unitary operator to the state
    new_global_state = U*initstate_global;
    % compute the eigenvalues of rhoA
    rhoA = TrX(P*new_global_state,2,[dimA,dimB]);
    lambda = eig(rhoA);

    % trace out all ancilla bits and all copies except one
    rho_new{j} = TrX(new_global_state,trace_out,dim_subsystems);
    % measure the new fidelity
    F(j) = abs(psi_ref'*rho_new{j}*psi_ref);
    % also measure the entropy of entanglement
    E_output(j) = entropy_of_entanglement(rho_new{j},2,2);
    % compute the eigenvalues of rhoA
    %rhoA2 = TrX(rho_new{j},2,[2,2]);
    %lambda2 = eig(rhoA2);
    E_max(j) = upper_E(lambda);
    rho(j,:) = rho_new{j}(:);
end
figure()
subplot(3,1,1)
plot(c,F)
subplot(3,1,2)
plot(c,E_input,'-^',c,E_max,'-v',c,E_output,'-x')
subplot(3,1,3)
rhonz = [];
ind_row = [];
ind_col = [];
labels = {};
for i = 1:16
```

```
    if(nnz(abs(rho(:,i)>1e−12))>0)
        rhonz = [rhonz rho(:,i)];
        [row col] = ind2sub([4 4],i);
        ind_row = [ind_row; row];
        ind_col = [ind_col; col];
        labels = [labels,{['\rho_{' num2str(row) ',' num2str(col) '}'
            ]}];
    end
end
plot(repmat(c(:),1,size(rhonz,2)),rhonz)
legend(labels);
```

D.30 QUBIT_STATES.M

```
% qubit_states.m: 10.05.2010, Yves Salathe
function states = qubit_states(n_qubits)
% this function generates cell array containing the state vectors of
    all
% possible states of an n_qubit system

dim_loc = 2^n_qubits;
qubit = {[1 0]',[0 1]'};
states = cell(dim_loc,1);
for i = 1:dim_loc
    states{i} = qubit{mod(i−1,2)+1};
    d = floor((i−1)/2);
    for j = 1:n_qubits−1
        states{i} = kron(qubit{mod(d,2)+1},states{i});
        d = floor(d/2);
    end
    if (d ~= 0); error(['failed to generate state ',num2str(i)]); end;
end
```

D.31 READ_ANCILLA_STATES.M

```
% read_ancilla_states.m: 17.05.2010, Yves Salathe
function states = read_ancilla_states(psi,n_ancillas_A,n_ancillas_B)
% function that measures the state of the ancilla qubits in the
% computational basis

dim = size(psi,1);
tol = 1e−15;
dim_ancillas_A = 2^n_ancillas_A;
dim_ancillas_B = 2^n_ancillas_B;
dim_ancillas = dim_ancillas_A*dim_ancillas_B;
dim_copies = dim/dim_ancillas;
if (dim_copies ~= floor(dim_copies)); error('the dimension of psi has
    to be dividable by the dimensions of the ancillas'); end;
dim_subsystems = [dim_ancillas_A dim_copies dim_ancillas_B];
trace_over = [1 3];
states = cell(dim_ancillas_A,dim_ancillas_B);
for i = 0:dim_ancillas_A−1
    ancilla_state_A = computational_basis(dim_ancillas_A,i);
    ancilla_state_A = ancilla_state_A*ancilla_state_A';
    for j = 0:dim_ancillas_B−1
        ancilla_state_B = computational_basis(dim_ancillas_B,j);
        ancilla_state_B = ancilla_state_B*ancilla_state_B';
        % project onto ancilla state
        P = kron(ancilla_state_A,kron(speye(dim_copies),ancilla_state_B
            ));
        psi_new = P*psi;
        % trace out the ancillas after the projection
        state = TrX(psi_new,trace_over,dim_subsystems);
        if(norm(state(:))>tol)
            states{i+1,j+1} = state;
        else
            states{i+1,j+1} = 0;
        end
    end
end
```

D.32 SCHMIDT_PROJECTION_MEASUREMENT_UNITARY.M

```
% schmidt_projection_measurement_unitary.m: 08.01.2011, Yves Salathe
function U = schmidt_projection_measurement_unitary(ancilla_side,
    n_copies, n_untouched_copies)
% generates a unitary operation that represents the Schmidt projection
% measurement on two qubits using two additional ancilla qubits to make
% the measurement reversible.
% ancilla_side can be either 'left' or 'right' depending on whether
% the ancilla qubits are multplied by tensor product from the left
% or from the right onto the state space

dim_copies = 2^(n_copies-n_untouched_copies);
dim_untouched_copies = 2^n_untouched_copies;

psi_0 = computational_basis(dim_copies,0);
psi_1 = computational_basis(dim_copies,1);
psi_2 = computational_basis(dim_copies,2);
psi_3 = computational_basis(dim_copies,3);

subspaces = {psi_0*psi_0', ...
    (psi_1*psi_1' + psi_2*psi_2'), ...
    psi_3*psi_3'};

U = measurement_unitary(subspaces,ancilla_side,dim_untouched_copies);

return
```

D.33 SCHMIDT_PROJECTION_METHOD_UNITARY.M

```
% schmidt_projection_method_unitary.m: 08.01.2011, Yves Salathe
function [Ua,Ub] = schmidt_projection_method_unitary(args)
% generates two unitary matrices that represent the Schmidt projection
% method on two pairs of qubits using some ancilla qubits.
    n_copies = args.n_copies;
    n_ancillas_A = args.n_ancillas_A;
    n_ancillas_B = args.n_ancillas_B;
    n_untouched_copies_A = args.operator_args.n_untouched_copies_A;
    n_untouched_copies_B = args.operator_args.n_untouched_copies_B;
    % Schmidt Projection
    Ua = schmidt_projection_measurement_unitary('right',n_copies,
        n_untouched_copies_A);
    Ub = schmidt_projection_measurement_unitary('right',n_copies,
        n_untouched_copies_B);
    % Standardization
    Ua = kron(standardization_unitary(),speye(2^(n_untouched_copies_A+
        n_ancillas_A)))*Ua;
    Ub = kron(standardization_unitary(),speye(2^(n_untouched_copies_B+
        n_ancillas_B)))*Ub;
    % Conditional Swap
    Ua = conditional_swap('right',n_copies,n_ancillas_A)*Ua;
    Ub = conditional_swap('right',n_copies,n_ancillas_B)*Ub;
return
```

D.34 SKEW_SYMMETRIC_INVERSE.M

```
% skew_symmetric_inverse.m: 08.01.2011
function re = skew_symmetric_inverse(H)
% from a skew-symmetric nxn matrix, determine the input vector re of
    the
% parameterization that is done in skew_symmetric.m
% the dimension n of the matrix is determined by the length m of re
% such that m = n*(n+1)/2
n = size(H,1);
if n ~= size(H,2); error('H has to be a square matrix'); end;
m = .5*n*(n-1);
re = zeros(m,1);
re_ind = 1;
for i=1:1:n
   re_ind2 = re_ind+n-i-1;
   % real part
   re(re_ind:re_ind2) = real(H(i,(i+1):n));
   re_ind = re_ind2+1;
end
```

D.35 SKEW_SYMMETRIC.M

```
% skew_symmetric.m: 08.01.2011, Yves Salathe
function H = skew_symmetric(re)
% form a skew-symmetric nxn matrix out of input vector re of
% the dimension n of the matrix is determined by the length m of re
% such that m = n*(n+1)/2
m = length(re);
n = .5*(sqrt(1+8*m)+1);
% check whether n is an integer
if (n - floor(n) ~= 0), error('the length m of the input vector re has
    to be chosen in such a way that m = n*(n-1)/2 has an integer
    solution n.'), end;
H = zeros(n);
re_ind = 1;
for i=1:1:n
   re_ind2 = re_ind+n-i-1;
   % real part
   H(i,(i+1):n) = re(re_ind:re_ind2);
   H((i+1):n,i) = -re(re_ind:re_ind2);
   re_ind = re_ind2+1;
end
```

D.36 STANDARDIZATION_UNITARY.M

```
% standardization_unitary.m: 17.05.2010, Yves Salathe
function U = standardization_unitary()
% the standardization process used in the Schmidt projection method for
% 2 qubits

U = computational_basis(4,1)*computational_basis(4,0)'...
    + computational_basis(4,0)*computational_basis(4,1)'...
    + computational_basis(4,2)*computational_basis(4,2)'...
    + computational_basis(4,3)*computational_basis(4,3)';
```

D.37 SYMMETRIC.M

```
% symmetric.m: 17.05.2010, Yves Salathe
function H = symmetric(x)
% form symmetric matrix out of input vector x

m = length(x);
n = (sqrt(1+8*m)-1)/2;
H = zeros(n);
ind = 1;
for i=1:1:n
   ind2 = ind+n-i;
   H(i,i:n) = x(ind:ind2);
   H((i+1):n,i) = x(ind+1:ind2);
   ind = ind2+1;
end
```

D.38 TEST_CONTOPT_3COPIES.M

```
%test_contopt.m: 18.05.2010, Yves Salathe
% script to test the continuous optimization
% tries to find the optimal protocol for a pure state
psi_0 = [1 0]';
psi_1 = [0 1]';
psi_00 = kron(psi_0,psi_0);
psi_11 = kron(psi_1,psi_1);
initstate = sqrt(.1)*psi_00 + sqrt(.9)*psi_11;
operator_args = init_operator_args(3,2,2);
fitness_args = init_fitness_args(3,2,2,initstate,@unitary_continuous,
    operator_args,{},{});
OPTS = cmaes;
cmaes('fitness',randn(2048,1),ones(2048,1),OPTS,fitness_args)
```

D.39 TEST_CONTOPT_3COPIES_SYMMETRY.M

```
% test_cmaes_3copies_symmetry_onlyreal.m: 08.01.2011, Yves Salathe
% script to test the continuous optimization
% tries to find the optimal protocol for 3 copies of a pure state
% starts with the schmidt projection with conditional swap as the
% initial guess
% always does the same operation for Alice and Bob
% only takes into account orthogonal matrices instead of general
    unitary
% operations

n_copies = 3;
n_ancillas_A = 2;
n_ancillas_B = 2;
c = sqrt(.1);
s = sqrt(.9);
operator_args = struct('n_untouched_copies_A',1,'n_untouched_copies_B'
    ,1);
initstate = full(c*computational_basis(4,0)+s*computational_basis(4,3))
    ;
phi_plus = full(sqrt(.5)*(computational_basis(4,0)+computational_basis
    (4,3)));
% init_F = abs(dot(initstate,phi_plus))^2
% operator_fn = @(x,args) schmidt_projection_method_unitary(args);
%operator_fn = @(x,args) identity(args);
% fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B,
    initstate,operator_fn,operator_args,{},{});
% [fit,rho_new] = fitness({},fitness_args)

% expected fidelity
%p = .5*(1+2*c^2*s^2)
% actual fidelity
%F = 1-fit

% optimization
tol = 1e-10;
%[Ua,Ub] = schmidt_projection_method_unitary(fitness_args);
%if norm(Ua-Ub) > tol; error('operations Ua and Ub have to be identical
    '); end;
%if norm(Ua*Ua'-eye(size(Ua))) > tol; error('operations Ua and Ub have
    to be unitary'); end;
%if norm(sum(sum(abs(imag(Ua))))) > tol; error('operations Ua and Ub
    have to be real valued'); end;
% find parameters for Ua
%x_init = orthogonal_inverse2(Ua);
dim = 2^(n_copies+n_ancillas_A);
x_init = 2*pi*rand(dim^2)-pi;
size(x_init)
operator_args = init_operator_args(n_copies,n_ancillas_A,n_ancillas_B,
    true,true,false);
fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B,
    initstate,@unitary_continuous2,operator_args,{},{});
init_F = fitness(x_init,fitness_args)
OPTS = cmaes;
%OPTS.DiagonalOnly='(1+100*N/sqrt(popsize))';  % C is diagonal for
    given iterations, 1==always
OPTS.SaveVariables = 'off';
OPTS.DispModulo='5  % [0:Inf], disp messages after every i-th iteration
    ';
cmaes('fitness',x_init,ones(length(x_init),1)/100,OPTS,fitness_args)
save('variables_test_sp_3_onlyreal.mat');
```

D.40 TEST_CONTOPT_3COPIES_SYMMETRY_ONLYREAL.M

```
% test_contopt_3copies_symmetry_onlyreal.m: 08.01.2011, Yves Salathe
% script to test the continuous optimization
% tries to find the optimal protocol for 3 copies of a pure state
% starts with the schmidt projection with conditional swap as the
% initial guess
% always does the same operation for Alice and Bob
% only takes into account orthogonal matrices instead of general
    unitary
% operations

n_copies = 3;
n_ancillas_A = 2;
n_ancillas_B = 2;
c = sqrt(.1);
s = sqrt(.9);
operator_args = struct('n_untouched_copies_A',1,'n_untouched_copies_B'
    ,1);
initstate = full(c*computational_basis(4,0)+s*computational_basis(4,3))
    ;
phi_plus = full(sqrt(.5)*(computational_basis(4,0)+computational_basis
    (4,3)));
% init_F = abs(dot(initstate,phi_plus))^2
% operator_fn = @(x,args) schmidt_projection_method_unitary(args);
%operator_fn = @(x,args) identity(args);
% fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B,
    initstate,operator_fn,operator_args,{},{});
% [fit,rho_new] = fitness({},fitness_args)

% expected fidelity
%p = .5*(1+nchoosek(2,1)*c^2*s^2)
% actual fidelity
%F = 1-fit

% optimization
tol = 1e-10;
%[Ua,Ub] = schmidt_projection_method_unitary(fitness_args);
%if norm(Ua-Ub) > tol; error('operations Ua and Ub have to be identical
    '); end;
%if norm(Ua*Ua'-eye(size(Ua))) > tol; error('operations Ua and Ub have
    to be unitary'); end;
%if norm(sum(sum(abs(imag(Ua))))) > tol; error('operations Ua and Ub
    have to be real valued'); end;
% find parameters for Ua
%x_init = orthogonal_inverse2(Ua);
dim = 2^(n_copies+n_ancillas_A);
x_init = 2*pi*rand(dim*(dim-1)/2+1)-pi;
size(x_init)
operator_args = init_operator_args(n_copies,n_ancillas_A,n_ancillas_B,
    true,true,true);
fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B,
    initstate,@unitary_continuous,operator_args,{},{});
init_F = fitness(x_init,fitness_args)
OPTS = cmaes;
%OPTS.DiagonalOnly='(1+100*N/sqrt(popsize))';  % C is diagonal for
    given iterations, 1==always
OPTS.SaveVariables = 'off';
OPTS.DispModulo='5  % [0:Inf], disp messages after every i-th iteration
    ';
 [xmin, ...      % minimum search point of last iteration
      fmin, ...      % function value of xmin
      counteval, ... % number of function evaluations done
      stopflag, ...  % stop criterion reached
      out, ...     % struct with various histories and solutions
      bestever ... % struct containing overall best solution (for
          convenience)
          ] = cmaes('fitness',x_init,ones(length(x_init),1)/100,OPTS,
              fitness_args)
save('variables_test_sp_3_onlyreal.mat');
```

D.41 TEST_CONTOPT.M

```
% test_contopt.m: 08.01.2011, Yves Salathe
function test_contopt(n_copies,n_ancillas_A,n_ancillas_B,symmetry,
    onlyreal,init_operation,init_state,method,maxFevals,parallel,
    determinant,sigma_init,seed,resume)
% function to test the continuous optimization
% tries to find the optimal protocol for 3 copies of a pure state
% starts with the schmidt projection with conditional swap as the
% initial guess
% always does the same operation for Alice and Bob
% only takes into account orthogonal matrices instead of general
    unitary
% operations
%
% An example of how to call this function:
%
% test_contopt(3,2,2,false,false,'rand','pure','cmaes','600000')
%
% The above command starts an optimization run on 3 copies of a pure
% state with 2 ancilla qubits per party without restricting U_A and U_B
     to
% be the same and without restricting the search to orthogonal
    matrices. The
% optimization starts with a random parameter vector where every
    element
% is uniformly distributed in [-pi,pi]. The optimization algorithm will
% be CMA-ES and the maximal number of function evaluations will be
% restricted to 600000.

% default options
if nargin < 1 || isempty(n_copies)
    n_copies = 3
end
if nargin < 2 || isempty(n_ancillas_A)
    n_ancillas_A = 2
end
if nargin < 3 || isempty(n_ancillas_B)
    n_ancillas_B = 2
end
if nargin < 4 || isempty(onlyreal)
    onlyreal = false
end
if nargin < 5 || isempty(onlyreal)
    onlyreal = false
end
if nargin < 6 || isempty(init_operation)
    init_operation = 'rand'
end
if nargin < 7 || isempty(init_state)
    init_state = 'pure'
end
if nargin < 8 || isempty(method)
    method = 'cmaes'
end
if nargin < 9 || isempty(maxFevals)
    maxFevals = 'Inf'
```

```
end
if nargin < 10 || isempty(parallel)
    parallel = false
end
if nargin < 11
    determinant = [];
end
if nargin < 12 || isempty(sigma_init)
    sigma_init = 1.0e-2*ones(length(x_init),1);
end
if nargin < 13 || isempty(seed)
    seed = sum(100*clock())
end
if nargin < 14 || isempty(resume)
    if exist('variables.mat')
        resume = true
    else
        resume = false
    end
end

% set the seed of the random number generator
stream = RandStream('mt19937ar');
reset(stream,seed);
RandStream.setDefaultStream(stream);

phip = sqrt(.5)*(computational_basis(4,0) + computational_basis(4,3));
phim = sqrt(.5)*(computational_basis(4,0) - computational_basis(4,3));
psip = sqrt(.5)*(computational_basis(4,1) + computational_basis(4,2));
psim = sqrt(.5)*(computational_basis(4,1) - computational_basis(4,2));

if strcmp(init_state,'pure')
    phip = full(phip);
    phim = full(phim);
    psip = full(psip);
    psim = full(psim);
    c = sqrt(.1);
    s = sqrt(.9);
    initstate = full(c*computational_basis(4,0)+s*computational_basis
        (4,3));
    init_F = abs(phip'*initstate)^2
else strcmp(init_state,'werner')
    p = .1
    initstate = (1-3*p/4)*phip*phip' + p/4*(phim*phim'+psip*psip'+psim*
        psim');
    init_F = abs(phip'*initstate*phip)
end


% optimization
tol = 1e-10;
dimA = 2^(n_copies+n_ancillas_A);
if onlyreal == false
    mA = dimA^2;
else
    if isempty(determinant)
        mA = .5*dimA*(dimA-1)+1;
    else
        mA = .5*dimA*(dimA-1);
    end
end
mB = 0;
if not(symmetry)
    dimB = 2^(n_copies+n_ancillas_B);
    if onlyreal == false
```

```
        mB = dimB^2;
    else
        if isempty(determinant)
            mB = .5*dimB*(dimB-1)+1;
        else
            mB = .5*dimB*(dimB-1);
        end
    end
end
m = mA+mB;
if strcmp(init_operation,'rand')
    if onlyreal && isempty(determinant)
        if symmetry
            x_init = [0;2*pi*rand(m-1,1)-pi];

        else
            x_init = [0;2*pi*rand(mA-1,1)-pi];
            x_init = [x_init;0;2*pi*rand(mB-1,1)-pi];
        end
    else
        x_init = 2*pi*rand(m,1)-pi;
    end
elseif strcmp(init_operation,'zeros')
    x_init = zeros(m,1);
elseif strcmp(init_operation,'sp') && onlyreal == true
    operator_fn = @(x,args) schmidt_projection_method_unitary(args);
    operator_args = struct('n_untouched_copies_A',1,'
        n_untouched_copies_B',1);
    %operator_fn = @(x,args) identity(args);
    fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B
        ,initstate,operator_fn,operator_args,{},{});
    [Ua,Ub] = schmidt_projection_method_unitary(fitness_args);
    if norm(Ua-Ub) > tol; error('operations Ua and Ub have to be
        identical'); end;
    if norm(Ua*Ua'-eye(size(Ua))) > tol; error('operations Ua and Ub
        have to be unitary'); end;
    if norm(sum(sum(abs(imag(Ua))))) > tol; error('operations Ua and Ub
         have to be real valued'); end;
    % find parameters for Ua
    x_init = orthogonal_inverse2(Ua);
    if not(symmetry)
        x_init = [xinit;xinit];
    end
end
size(x_init)
operator_args = init_operator_args(n_copies,n_ancillas_A,n_ancillas_B,
    true,symmetry,onlyreal,determinant);
operator_fn = @unitary_continuous;
fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B,
    initstate,operator_fn,operator_args,{},{});
init_fit = fitness(x_init,fitness_args)

resume_nr = 1;
time_start = [];
wall_time_start = [];
time_end = [];
wall_time_end = [];
if resume
    local_maxFevals = maxFevals;
    load('variables.mat');
    maxFevals = local_maxFevals;
    clear local_maxFevals
    resume = true;
    resume_nr = 2;
end
```

```
time_start = [time_start; cputime()]
wall_time_start = [wall_time_start; clock()]
save('variables.mat');
if strcmp(method,'cmaes')
    OPTS = cmaes;
    %OPTS.DiagonalOnly='(1+100*N/sqrt(popsize))';  % C is diagonal for
        given iterations, 1==always
    OPTS.SaveVariables = 'on';
    if resume
        OPTS.Resume = 'true';
    else
        OPTS.Seed = 'noreset';
    end
    if parallel
        OPTS.EvalParallel = 'yes'
        fitfun = 'fitness_parallel'
    else
        fitfun = 'fitness';
    end
    OPTS.StopFunEvals = maxFevals
    OPTS.DispModulo='5  % [0:Inf], disp messages after every i-th
        iteration';
    [xmin, ...      % minimum search point of last iteration
      fmin, ...      % function value of xmin
      counteval, ... % number of function evaluations done
      stopflag, ...  % stop criterion reached
      out, ...      % struct with various histories and solutions
      bestever ... % struct containing overall best solution (for
          convenience)
    ] = cmaes(fitfun,x_init,sigma_init,OPTS,fitness_args)
end
time_end = [time_end; cputime()]
wall_time_end = [wall_time_end; clock()]
save('variables.mat');
```

D.42 TEST_SCHMIDT_PROJECTION_GENERAL.M

```
%test_schmidt_projection_general.m: 07.01.2011, Yves Salathe
function test_schmidt_projection_general(n_copies,n_ancillas_A,
    n_ancillas_B,n_untouched_copies_A,n_untouched_copies_B)
% function to test the effect that the Schmidt projection method has on
% several copies of a certain pure state

operator_args = struct('n_untouched_copies_A',n_untouched_copies_A,'
    n_untouched_copies_B',n_untouched_copies_B);
initstate = full(sqrt(.1)*computational_basis(4,0)+sqrt(.9)*
    computational_basis(4,3));
phi_plus = full(sqrt(.5)*(computational_basis(4,0)+computational_basis
    (4,3)));
init_F = abs(dot(initstate,phi_plus))^2
operator_fn = @(x,args) schmidt_projection_method_unitary(args);
%operator_fn = @(x,args) identity(args);
fitness_args = init_fitness_args(n_copies,n_ancillas_A,n_ancillas_B,
    initstate,operator_fn,operator_args,{},{});
[fit,rho_new] = fitness({},fitness_args)
```

D.43 TRX_FAST_ARGS.M

```
% TrX_fast_args.m: 3.3.2011, Yves Salathe
function [sys,dim,keep,dimkeep,dimtrace,rdim,n] = TrX_fast_args(sys,dim
    ,dim_total)
% The function TrX_density_fast computes the partial trace over an
    density
% matrix or state vector of dimension dim_total in a fast manner
% using arguments
% that have been precomputed by TrX_density_fast_args
%
% this program is derived from:
%
% TRX   Partial trace
% requires: nothing
% author: Toby Cubitt
% license: GPL2
%
%    RHO = TrX(PSI,SYS,DIM) traces out the subsystems spcified in
%    vector SYS of state PSI (a state vector or densitry matrix) whose
%    subsystem dimensions are specified by the vector DIM.


%% Copyright (C) 2004-2009 Toby Cubitt
%%
%% This program is free software; you can redistribute it and/or
%% modify it under the terms of the GNU General Public License
%% as published by the Free Software Foundation; either version 2
%% of the License, or (at your option) any later version.
%%
%% This program is distributed in the hope that it will be useful,
%% but WITHOUT ANY WARRANTY; without even the implied warranty of
%% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%% GNU General Public License for more details.
%%
%% You should have received a copy of the GNU General Public License
%% along with this program; if not, write to the Free Software
%% Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
%% MA 02110-1301, USA.


% check arguments
if any(sys > length(dim)) || any(sys < 0)
  error('Invalid subsystem in SYS')
end
if (length(dim) == 1 && mod(dim_total/dim,1) ~= 0)...
   || dim_total ~= prod(dim)
  error('Size of state PSI inconsistent with DIM');
end


if exist('setdiff')
  sys = setdiff(sys,find(dim == 1));
else
  % octave
  sys = complement(find(dim == 1),sys);
end

%dim = dim(find(dim ~= 1));
% 3.3.2011, Yves Salathe: logical indexing is faster than find
dim(dim == 1) = [];


% calculate systems, dimensions, etc.
n = length(dim);
rdim = dim(end:-1:1);
keep = [1:n];
```

```
keep(sys) = [];
dimtrace = prod(dim(sys));
dimkeep = dim_total/dimtrace;
```

D.44 TRX_FAST.M

```
% TrX_fast.m: 3.3.2011, Yves Salathe
function x = TrX_fast(p,sys,keep,dimkeep,dimtrace,rdim,n)
% this function computes the partial trace over a density matrix or
% state vector in a
% fast manner using arguments that have been precomputed by
% TrX_density_fast_args
%
% this program is derived from:
%
% TRX   Partial trace
% requires: nothing
% author: Toby Cubitt
% license: GPL2
%
%    RHO = TrX(PSI,SYS,DIM) traces out the subsystems spcified in
%    vector SYS of state PSI (a state vector or densitry matrix) whose
%    subsystem dimensions are specified by the vector DIM.


%% Copyright (C) 2004-2009 Toby Cubitt
%%
%% This program is free software; you can redistribute it and/or
%% modify it under the terms of the GNU General Public License
%% as published by the Free Software Foundation; either version 2
%% of the License, or (at your option) any later version.
%%
%% This program is distributed in the hope that it will be useful,
%% but WITHOUT ANY WARRANTY; without even the implied warranty of
%% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%% GNU General Public License for more details.
%%
%% You should have received a copy of the GNU General Public License
%% along with this program; if not, write to the Free Software
%% Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
%% MA 02110-1301, USA.

if any(size(p) == 1)
  % state vector
  if size(p,1) == 1
    p = p';
  end
  % reshape state vector to "reverse" ket on traced subsystems into a
      bra,
  % then take outer product
  perm = n+1-[keep(end:-1:1),sys];
  x = reshape(permute(reshape(p,rdim),perm),[dimkeep,dimtrace]);
  x = x*x';


else
  % density matrix

  % reshape density matrix into tensor with one row and one column
      index
  % for each subsystem, permute traced subsystem indices to the end,
  % reshape again so that first two indices are row and column
  % multi−indices for kept subsystems and third index is a flattened
     index
  % for traced subsystems, then sum third index over "diagonal" entries
  perm = n+1−[keep(end:−1:1),keep(end:−1:1)−n,sys,sys−n];
  x = reshape(permute(reshape(p,[rdim,rdim]),perm),...
            [dimkeep,dimkeep,dimtrace^2]);
  x = sum(x(:,:,[1:dimtrace+1:dimtrace^2]),3);

end
```

D.45 TrX.m

```
% trx.m: not self−written
function x = TrX(p,sys,dim)

% TRX   Partial trace
% requires: nothing
% author: Toby Cubitt
% license: GPL2
%
%    RHO = TrX(PSI,SYS,DIM) traces out the subsystems spcified in
%    vector SYS of state PSI (a state vector or densitry matrix) whose
%    subsystem dimensions are specified by the vector DIM.


%% Copyright (C) 2004−2009 Toby Cubitt
%%
%% This program is free software; you can redistribute it and/or
%% modify it under the terms of the GNU General Public License
%% as published by the Free Software Foundation; either version 2
%% of the License, or (at your option) any later version.
%%
%% This program is distributed in the hope that it will be useful,
%% but WITHOUT ANY WARRANTY; without even the implied warranty of
%% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%% GNU General Public License for more details.
%%
%% You should have received a copy of the GNU General Public License
%% along with this program; if not, write to the Free Software
%% Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
%% MA 02110−1301, USA.


% check arguments
if any(sys > length(dim)) || any(sys < 0)
  error('Invalid subsystem in SYS')
end
if (length(dim) == 1 && mod(length(p)/dim,1) ~= 0)...
  || length(p) ~= prod(dim)
  error('Size of state PSI inconsistent with DIM');
end


% remove singleton dimensions
% 3.3.2011, Yves Salathe: check with 'exist' commented out to improve
% performance
%if exist('setdiff')
  % matlab
  sys = setdiff(sys,find(dim == 1));
%else
```

```
  % octave
%  sys = complement(find(dim == 1),sys);
%end

%dim = dim(find(dim ~= 1));
% 3.3.2011, Yves Salathe: logical indexing is faster than find
dim(dim == 1) = [];


% calculate systems, dimensions, etc.
n = length(dim);
rdim = dim(end:-1:1);
keep = [1:n];
keep(sys) = [];
dimtrace = prod(dim(sys));
dimkeep = length(p)/dimtrace;


if any(size(p) == 1)
  % state vector
  if size(p,1) == 1
    p = p';
  end
  % reshape state vector to "reverse" ket on traced subsystems into a
      bra,
  % then take outer product
  perm = n+1-[keep(end:-1:1),sys];
  x = reshape(permute(reshape(p,rdim),perm),[dimkeep,dimtrace]);
  x = x*x';


else
  % density matrix

  % reshape density matrix into tensor with one row and one column
      index
  % for each subsystem, permute traced subsystem indices to the end,
  % reshape again so that first two indices are row and column
  % multi-indices for kept subsystems and third index is a flattened
      index
  % for traced subsystems, then sum third index over "diagonal" entries
  perm = n+1-[keep(end:-1:1),keep(end:-1:1)-n,sys,sys-n];
  x = reshape(permute(reshape(p,[rdim,rdim]),perm),...
              [dimkeep,dimkeep,dimtrace^2]);
  x = sum(x(:,:,[1:dimtrace+1:dimtrace^2]),3);

end
```

D.46 UNITARITY_PENALTY.M

```
% unitarity_penality.m: 08.01.2011, Yves Salathe
function p = unitarity_penalty(x,args,Ua,Ub,U,new_global_state,rho_new,
    F)
% penalizes those solutions that do not lead to a unitary matrix
p = max(max(abs(Ua*Ua'-eye(size(Ua)))));
if p > .1
    p = Inf;
end
if(not(args.operator_args.symmetry))
    s = max(max(abs(Ub*Ub'-eye(size(Ub)))));
    if s > .1
```

```
        p = Inf;
    else
        p = p + s
    end
end
```

D.47 UNITARY_CONTINUOUS.M

```
% unitary_continous.m: 08.01.2011, Yves Salathe
function [Ua,Ub] = unitary_continuous(x,args)
% Build unitary operators out of the input vector x:
% The first dim_A*(dim_A+1)/2 entries of x specify the real part of
% the hermitian matrix Ha which will yield the unitary operation that
    acts
% on Alice's subsystem. The succeeding dim_A*(dim_A-1)/2 entries are
% interpreted as the imaginary part of Ha.
% In the same manner, the remaining entries specify the real and
    imaginary
% parts of Hb, which belongs to Bob's subsystem.
%
% x has to have length 2*n*(n+1), where n is the length of each of the
    two
% subspaces of Alice and Bob
x = x(:);
if (not(args.operator_args.onlyreal))
    Ua = unitary([x(1:args.operator_args.ind_max_real_A);x(
        args.operator_args.ind_min_imag_A:
        args.operator_args.ind_max_imag_A)]);
else
    Ua = orthogonal(x(1:args.operator_args.ind_max_real_A),
        args.operator_args.determinant);
end

if(not(args.operator_args.symmetry))
    if not(args.operator_args.onlyreal)
        Ub = unitary([x(args.operator_args.ind_min_real_B:
            args.operator_args.ind_max_real_B);x(
            args.operator_args.ind_min_imag_B:end)]);
    else
        Ub = orthogonal(x(args.operator_args.ind_min_real_B:
            args.operator_args.ind_max_real_B),
            args.operator_args.determinant);
    end
else
    Ub = Ua;
end
```

D.48 UNITARY_DISCRETE.M

```
% unitary_discrete.m: 17.05.2011, Yves Salathe
function U = unitary_discrete(x,gates_A,gates_B,dim_A,dim_B)
% build unitary operations from a set of gates. Which gates to take is
% specified by the vector x

% the length of x has to be dividable by two
n = length(x)/2;
```

```
if(floor(n)~=n), error('the length of the input vector x has to be
    dividable by two'); end;

Ua = eye(dim_A);
Ub = eye(dim_B);

for i = 1:n
  Ua = Ua*gates_A{x(i)};
  Ub = Ub*gates_B{x(n+i)};
end
U = kron(Ua,Ub);

return;
```

D.49 UNITARY.M

```
% uniatary.m: 26.02.2011, Yves Salathe
function T = unitary(x)
% map the n^2 parameters in the input vector x to a nxn unitary matrix
%
% this function uses the parameterization from the paper
% C. Jarlskog,
% Recursive parametrization and invariant phases of unitary matrices
% J. Math. Phys. 47, 013507 (2006); doi:10.1063/1.2159069 (13 pages)
%
% see also:
% Kazuyuki Fujii, Kunio Funahashi, Takayuki Kobayashi,
% Jarlskog's Parametrization of Unitary Matrices and Qudit Theory
% eprint: arXiv:quant-ph/0508006v3
% (Submitted on 1 Aug 2005 (v1), last revised 23 Nov 2005 (this version
    , v3))

x = x(:);
m = length(x);
dim = sqrt(m);
if (dim - floor(dim) ~= 0); error('the length m of the input vector has
     to be chosen in such a way, that dim = sqrt(m) is an integer');
    end;
theta = x(1:dim);
%T = spdiags(exp(1i*theta),[0],dim,dim);
T = diag(exp(1i*theta));
ind_re = dim + 1;
ind_im = .5*dim*(dim+1) + 1;
for j = 2:dim
    e1 = speye(j-1);
    e2 = speye(dim-j);
    zer = sparse(j-1,dim-j);
    zer2 = sparse(1,dim-j);
    z = x(ind_re:(ind_re+j-2)) + 1i* x(ind_im:(ind_im+j-2));
    n = norm(z);
    z = z./n;
    c = cos(n);
    s = sin(n);
    M = (1-c)*(z*z');
    sz = s*z;
    T = T*[ e1 - M,   sz,     zer; ...
              -sz',     c,      zer2; ...
              zer',     zer2',       e2];
    ind_re = ind_re + j - 1;
    ind_im = ind_im + j - 1;
end
```

D.50 UPPER_E.M

```
% upper_E.m: 11.03.2011, Yves Salathe
function E = upper_E(lambda_psi,a)
% this function calculates the upper bound for the entropy of
% entanglement that a pure state can have after application of any
% protocol without communication on a pure state with
% lambda_psi being the eigenvalues of rho_A =tr_B(|psi><psi|),
% where tr_B denotes the partial trace over Bob's subsystem.
% see exercise 12.22 in the book
% M. A. Nielsen and I. L. Chuang. Quantum Computation and Quantum
    Information.
% 1st ed. Cambridge University Press, Oct. 2000. ISBN: 0521635039.
E = 0;
tol = 1e-10;
lambda_psi = lambda_psi(lambda_psi>tol);
m = length(lambda_psi);
if mod(m,2)==0
    M = choose_k_from_n(m,m/2);
    Mf = flipud(M);
    for i = 1:size(M,1)
    %for lambda_phi = 0:.01:.
        lpsi1 = sort(lambda_psi(M(i,:)));
        l = sum(lpsi1);
        lpsi2 = sort(lambda_psi(Mf(i,:)));
%         f = @(l) sum(((1-l)/l*lpsi1-lpsi2).^2)+(s-l).^2;
%         gs = GlobalSearch;
%         problem = createOptimProblem('fmincon','x0',.3,...
%             'objective',f,'lb',0,'ub',.5);
%         [l,fval] = run(gs,problem);
        if (all(abs((1-l)*lpsi1-l*lpsi2)<tol))
            E = max(E,-l*log2(l)-(1-l)*log2(1-l));
        end
    end
end
```

Bibliography

[1] J. S. Bell. “On the Einstein-Podolsky-Rosen paradox”. In: *Physics* 1.3 (1964), pp. 195–200.

[2] C. H. Bennett et al. “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels”. In: *Phys. Rev. Lett.* 70.13 (Mar. 1993), pp. 1895–1899. DOI: 10.1103/PhysRevLett.70.1895.

[3] C. H. Bennett and S. J. Wiesner. “Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states”. In: *Phys. Rev. Lett.* 69.20 (Nov. 1992), pp. 2881–2884. DOI: 10.1103/PhysRevLett.69.2881.

[4] C. H. Bennett et al. “Purification of Noisy Entanglement and Faithful Teleportation via Noisy Channels”. In: *Phys. Rev. Lett.* 76.5 (Jan. 1996), pp. 722–725. DOI: 10.1103/PhysRevLett.76.722.

[5] C. H. Bennett et al. “Mixed-state entanglement and quantum error correction”. In: *Phys. Rev. A* 54.5 (Nov. 1996), pp. 3824–3851. DOI: 10.1103/PhysRevA.54.3824.

[6] D. Deutsch et al. “Quantum Privacy Amplification and the Security of Quantum Cryptography over Noisy Channels”. In: *Phys. Rev. Lett.* 77.13 (Sept. 1996), pp. 2818–2821. DOI: 10.1103/PhysRevLett.77.2818.

[7] A. Ambainis, A. Smith, and K. Yang. “Extracting quantum entanglement (general entanglement purification protocols)”. In: *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on.* 2002, pp. 82 –91. DOI: 10.1109/CCC.2002.1004345.

[8] A. Ambainis and K. Yang. “Towards the classical communication complexity of entanglement distillation protocols with incomplete information”. In: *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on.* June 2004, pp. 305 –319. DOI: 10.1109/CCC.2004.1313853.

[9] N. Hansen and A. Ostermeier. “Completely Derandomized Self-Adaptation in Evolution Strategies”. In: *Evol. Comput.* 9.2 (2001), pp. 159–195. ISSN: 1063-6560. DOI: 10.1162/106365601750190398.

[10] A. Einstein, B. Podolsky, and N. Rosen. “Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?” In: *Physical Review Online Archive (Prola)* 47.10 (May 1935), pp. 777–780. DOI: 10.1103/PhysRev.47.777.

[11] C. H. Bennet and G. Brassard. “Quantum Cryptography: Public Key Distribution and Coin Tossing”. In: *International Conference on Computers, Systems & Signal Processing Bangalore, India.* 1984.

[12] C. H. Bennett et al. “Concentrating partial entanglement by local operations”. In: *Phys. Rev. A* 53.4 (Apr. 1996), pp. 2046–2052. DOI: 10.1103/PhysRevA.53.2046.

[13] H.-K. Lo and S. Popescu. "Concentrating entanglement by local actions: Beyond mean values". In: *Phys. Rev. A* 63.2 (Jan. 2001), p. 022301. DOI: 10.1103/PhysRevA.63.022301.

[14] D. D. Dukaric and S. Wolf. "A Limit on Non-Locality Distillation". In: *arXiv eprints* (Aug. 2008). arXiv:quant-ph/0808.3317.

[15] A. J. Short. "No Deterministic Purification for Two Copies of a Noisy Entangled State". In: *Phys. Rev. Lett.* 102.18 (May 2009), p. 180502. DOI: 10.1103/PhysRevLett.102.180502.

[16] A. Messiah. *Quantum Mechanics, Two Volumes Bound as One*. Dover Publications Inc., Mineola, New York, 1999.

[17] C. Cohen-Tannoudji, B. Diu, and F. Laloe. *Quantum Mechanics (2 vol. set)*. Wiley-Interscience, Oct. 2006. ISBN: 0471569526.

[18] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. 1st ed. Cambridge University Press, Oct. 2000. ISBN: 0521635039.

[19] P. J. Mohr, B. N. Taylor, and D. B. Newell. "CODATA recommended values of the fundamental physical constants: 2006". In: *Reviews of Modern Physics* 80.2, 633 (2008), p. 633. DOI: 10.1103/RevModPhys.80.633.

[20] A. Aspect, J. Dalibard, and G. Roger. "Experimental Test of Bell's Inequalities Using Time- Varying Analyzers". In: *Phys. Rev. Lett.* 49.25 (Dec. 1982), pp. 1804–1807. DOI: 10.1103/PhysRevLett.49.1804.

[21] J. Grondalski, D. M. Etlinger, and D. F. V. James. "The fully entangled fraction as an inclusive measure of entanglement applications". In: *Physics Letters A* 300.6 (2002), pp. 573 –580. ISSN: 0375-9601. DOI: 10.1016/S0375-9601(02)00884-8.

[22] M.-J. Zhao and Z.-X. W. Zong-Guo Li Shao-Ming Fei. "A Note on Fully Entangled Fraction". In: *arXiv eprints* (June 2010). arXiv:1006.3502v2.

[23] S. Bose, V. Vedral, and P. L. Knight. "Purification via entanglement swapping and conserved entanglement". In: *Phys. Rev. A* 60.1 (July 1999), pp. 194–197. DOI: 10.1103/PhysRevA.60.194.

[24] M. A. Nielsen. "Conditions for a Class of Entanglement Transformations". In: *Phys. Rev. Lett.* 83.2 (July 1999), pp. 436–439. DOI: 10.1103/PhysRevLett.83.436.

[25] A. Mordecai. *Nonlinear Programming: Analysis and Methods*. Dover Publishing, Oct. 2003. ISBN: 0-486-43227-0.

[26] J. B. Lasserre. "Global Optimization with Polynomials and the Problem of Moments". In: *SIAM Journal on Optimization* 11.3 (2001), pp. 796–817. DOI: 10.1137/S1052623400366802.

[27] H. Waki et al. "Sums of Squares and Semidefinite Program Relaxations for Polynomial Optimization Problems with Structured Sparsity". In: *SIAM Journal on Optimization* 17.1 (2006), pp. 218–242. DOI: 10.1137/050623802.

[28] H. Waki et al. "Algorithm 883: SparsePOP—A Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems". In: *ACM Trans. Math. Softw.* 35 (2 July 2008), 15:1–15:13. ISSN: 0098-3500. DOI: 10.1145/1377612.1377619.

[29] K. Fujii, K. Funahashi, and T. Kobayashi. "Jarlskog's Parametrization of Unitary Matrices and Qudit Theory". In: *ArXiv Quantum Physics e-prints* (July 2005). arXiv:quant-ph/0508006.

[30] C. Jarlskog. “Recursive parametrization and invariant phases of unitary matrices”. In: *Journal of Mathematical Physics* 47.1, 013507 (2006), p. 013507. DOI: 10.1063/1.2159069.

[31] R. C. Raffenetti and K. Ruedenberg. “Parametrization of an orthogonal matrix in terms of generalized eulerian angles”. In: *International Journal of Quantum Chemistry* 4.S3B (1969), pp. 625–634. ISSN: 1097-461X. DOI: 10.1002/qua.560040725.

[32] D. K. Hoffman, R. C. Raffenetti, and K. Ruedenberg. “Generalization of Euler Angles to N-Dimensional Orthogonal Matrices”. In: *Journal of Mathematical Physics* 13.4 (1972), pp. 528–533. DOI: 10.1063/1.1666011.

[33] N. Hansen. “The CMA Evolution Strategy: a comparing review”. In: *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*. Ed. by J. Lozano et al. Springer, 2006, pp. 75–102.

[34] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Fromman-Holzboog, 1973.

[35] H.-P. Schwefel. “Evolutionsstrategie und numerische Optimierung”. Dr.-Ing. Dissertation. Technische Universität Berlin,Fachbereich Verfahrenstechnik, 1975.

[36] N. Hansen, S. Muller, and P. Koumoutsakos. “Reducing the time complexity of the derandomized Evolution Strategy with covariance matrix adaptation (CMA-ES)”. In: *Evolutionary Computation* 11.1 (2003), pp. 1–18.

[37] V. Strassen. “Gaussian elimination is not optimal”. In: *Numerische Mathematik* 13 (4 1969). 10.1007/BF02165411, pp. 354–356. ISSN: 0029-599X.

[38] P. G. Kwiat et al. “Experimental entanglement distillation and ‘hidden’ nonlocality”. In: *Nature* 409.6823 (2001), pp. 1014–1017. DOI: 10.1038/35059017.

[39] S. Popescu and D. Rohrlich. “The joy of entanglement”. In: *Introduction to Quantum Computation*. Ed. by H.-K. Lo, S. Popescu, and T. P. Spiller. Singapore: World-Scientific, 1998, pp. 29–48.

[40] J. Preskill. *Lecture Notes for Physics 229: Quantum Information and Computation*.

[41] N. Gisin. “Hidden quantum nonlocality revealed by local filters”. In: *Physics Letters A* 210.3 (1996), pp. 151 –156. ISSN: 0375-9601. DOI: 10.1016/S0375-9601(96)80001-6.

[42] W. Dür and H.-J. Briegel. “Purification and Distillation”. In: *Lectures on Quantum Information*. Ed. by D. Bruß and G. Leuchs. Wiley-VCH, 2007, pp. 177–208. ISBN: 3527405275.

[43] W. Dür and H.-J. Briegel. “Entanglement Purification for Quantum Computation”. In: *Phys. Rev. Lett.* 90.6 (Feb. 2003), p. 067901. DOI: 10.1103/PhysRevLett.90.067901.

[44] C. Macchiavello. “On the analytical convergence of the QPA procedure”. In: *Physics Letters A* 246.5 (1998), pp. 385 –388. ISSN: 0375-9601. DOI: 10.1016/S0375-9601(98)00550-7.

[45] K. Yang. “On the communication complexity of correlation and entanglement distillation”. AAI3126935. PhD thesis. Pittsburgh, PA, USA, 2004.

[46] H. Aschauer. “Quantum communication in noisy environments”. PhD thesis. Ludwig Maximilians Universität München, 2004.

[47] N. Hansen. *CMA Evolution Strategy Source Code*. `http://www.lri.fr/~hansen/cmaes_inmatlab.html`. Accessed March 15, 2011.