Probing the quantum-classical boundary with compression software Supplementary material

Hou Shun Poh,[1] Marcin Markiewicz,[2,3] Paweł Kurzyński,[1,4]
Alessandro Cerè,[1] Dagomir Kaszlikowski,[1,5] and Christian Kurtsiefer[1,5]

[1]*Center for Quantum Technologies, National University of Singapore, 3 Science Drive 2, Singapore 117543*
[2]*Faculty of Physics, University of Warsaw, ul. Pasteura 5, PL-02-093 Warszawa, Poland*
[3]*Institute of Theoretical Physics and Astrophysics, University of Gdansk, ul. Wita Stwosza 57, PL-80-952, Gdansk, Poland*
[4]*Faculty of Physics, Adam Mickiewicz University, Umultowska 85, 61-614 Poznań, Poland*
[5]*Department of Physics, National University of Singapore, 2 Science Drive 3, Singapore 117542*

SYMMETRIZATION OF DETECTOR EFFICIENCIES

The experimental setup (Fig. 3 in main article) uses four APDs: D_{HA}, D_{VA} (Alice), and D_{HB}, D_{VB} (Bob) to register photon pair events in the two spatial modes. By denoting events at D_H and D_V as 1 and 0, the four possible output patterns are 00, 01, 10, and 11, where the least and most significant bit corresponds to the Alice and Bob mode, respectively. Due to differences in the the losses in the transmitted and reflected port of the PBS, efficiencies in coupling light into the APDs, and the quantum efficiencies of APDs, the detection efficiencies for the four output combinations are different. The resulting effective pair efficiencies are then given by the product of the contributing detection efficiencies η_{VB}, η_{HB}, η_{VA}, and η_{HA}.

This asymmetry will skew the statistics of the measurement results. We symmetrize the effective pair efficiencies for each (θ_A, θ_B) measuring also the following settings for the half wave plates: $(\theta_A + 45°, \theta_B)$, $(\theta_A, \theta_B + 45°)$, and $(\theta_A + 45°, \theta_B + 45°)$. This procedure swaps the output ports of the PBS at which each polarization state is detected. The resulting outcomes are then interleaved, providing an uniform detection probability for the four possible outcomes. The effective pair detection efficiency for all four combinations is then $(\eta_{VB}\eta_{VA} + \eta_{VB}\eta_{HA} + \eta_{HB}\eta_{VA} + \eta_{HB}\eta_{HA})/4$.

CHOICE OF COMPRESSOR

In order to evaluate the NCDs of the binary strings, we need to choose a compression algorithm that performs close to the Shannon limit. This is necessary to ensure that it does not introduce any unintended artifacts that lead to an overestimation of the violation. Preferably we want to work in the regime where the obtained NCDs always underestimate the violation. For this purpose, we characterized four compression algorithms implemented by freely available compression programs: *lzma* [1], *bzip2* [2], *gzip* [3], and *lzw* [4]. To eliminate the overhead associated with the compression of ASCII text files, we save data in a binary format.

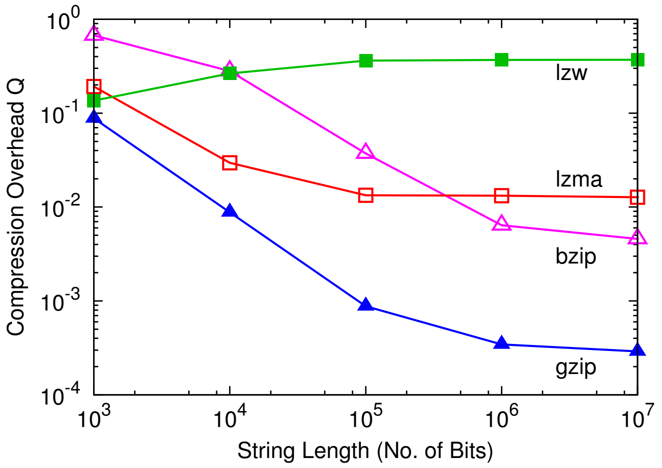

FIG. 1. Comparison of the compression overhead Q obtained using four different compression algorithms on pseudorandom strings of varying lengths. The expected value for an ideal compressor is 0. From this characterization we can exclude *lzw* as a useful compressor for our application.

For this characterization and a simulation of the experiment, we need to generate a "random" string of bits (1, 0) or pairs of bits (00, 01, 10, and 11) of various length with various probability distributions. We generate these strings using the *MATLAB* [5] function *randsample()* that uses the pseudo random number generator *mt19937ar* with a long period of $2^{19937} - 1$. It is based on the Mersenne Twister [6], with ziggurat [7] as the algorithm that generates the required probability distribution. The complexity of this (deterministic) source of pseudorandom numbers should be high enough to *not* be captured as algorithmic.

The first part of this characterization involves establishing the minimum string length required for the compression algorithms to perform consistently. We start by generating binary strings, x, with equal probability of 1's and 0's, i.e. random strings, of varying length. For each x, we evaluate the compression overhead Q as

$$Q = \frac{C(x) - H(x)}{l(x)}\,. \tag{1}$$

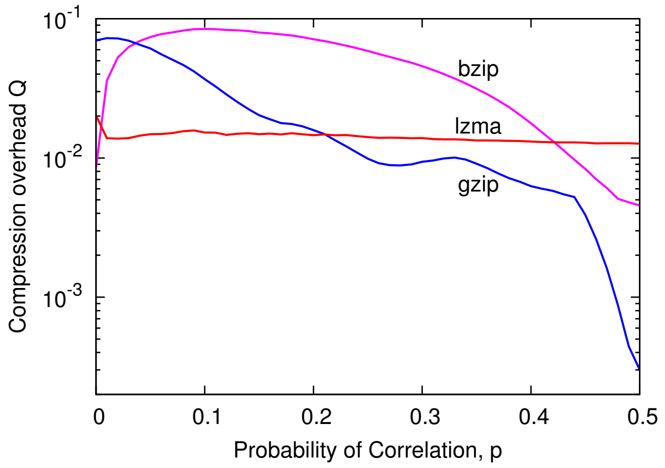

FIG. 2. Compression overhead Q for the string xy as a function of the probability of pairwise correlation p between the bits of the generating strings x and y for three different compressors: *bzip*, *gzip*, and *lzma*.

For a good compressor, we expect Q to be close to 0. From Fig. 1, it can be seen that for all the compressors, Q starts to converge after about 10^5 bits, setting the minimum string length required for the compressors to work consistently. The *lzw* compressor fails this test, converging to a Q of 0.37 for long string, while *bzip2*, *gzip*, and *lzma* give a Q below 10^{-1}.

In the second part of this characterization, test the compressors with strings with a known amount of correlation. We generate a random string x of length 10^7 using the same technique already described. We then generate a second string y of equal length and with probability p of being correlated to x. For $p = 0$ the two strings are equal, i.e. perfectly correlated. For $p = 0.5$ they are uncorrelated.

The two strings x and y are then combined to form the string xy: to avoid artifacts due to the limited data block size of the compression algorithms, the elements of x and y are interleaved. We then compress xy and evaluate the compression overhead Q as a function of p. The results for different compressors are shown in Fig. 2. Although there are ranges of p where *bzip* and *gzip* perform better than *lzma*, the latter shows a more uniform performance over the entire interval of p. It is reasonable to assume that the use of *lzma* should reduce the possibility of artifacts in the estimation of the NCD also for the data obtained from the experiment.

[1] I. Pavlov, http://www.7-zip.org/sdk.html.
[2] J. Seward, http://www.bzip.org/.
[3] J.-l. Gailly and M. Adler, http://www.gzip.org/.
[4] T. Welch, Computer **17**, 8 (1984).
[5] MATLAB R2010a, The MathWorks, Inc., Natick, Massachusetts, United States.
[6] M. Matsumoto and T. Nishimura, ACM Transactions on Modeling and Computer Simulation (TOMACS) **8**, 3 (1998).
[7] G. Marsaglia and W. W. Tsang, Journal of Statistical Software **5**, 1 (2000).