CINECA SCAI

SuperComputing Applications and Innovation

Quantum Annealing with continuous variables: Low-Rank Matrix Factorization

Daniele Ottaviani
CINECA

Quantum Computing and High Performance Computing
CINECA Casalecchio di Reno, Bologna, 18-12-2018

QUBO Problems with real variables

We define a QUBO problem with real variables as a Quadratic Unconstrained Optimization problem with unknown variables expressed as:

$$x = c \cdot \sum_{e=0}^{N-1} 2^e q_e, \quad c = 10^{-a}, \text{ for some } a \in \mathbb{N}$$

For example, the QUBO problem associated with the simple equation $x - b = 0$ is:

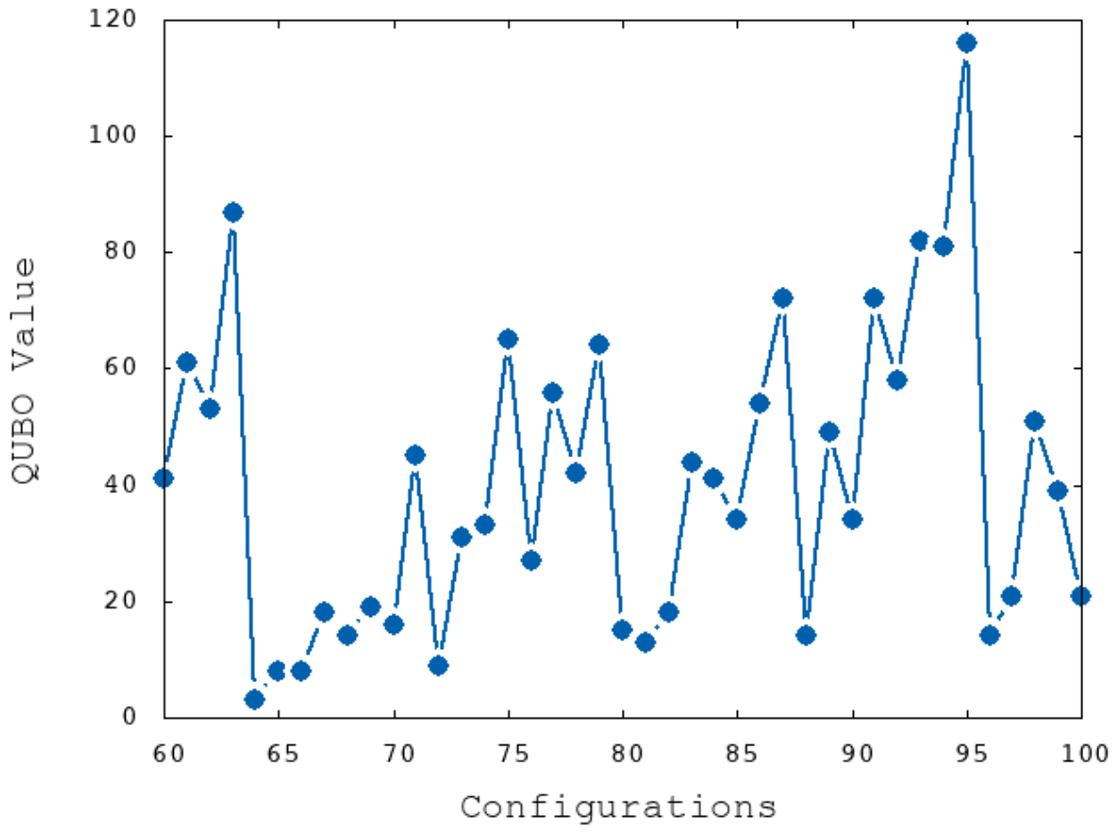
$$\min_{\mathbf{q}=(q_0,\ \ldots,\ q_{N-1})} \left(\sum_{e=0}^{N-1} \left(c^2 2^{2e} - bc2^{e+1} \right) q_e + \sum_{e<f} \left(c^2 2^{e+f+1} \right) q_e q_f \right)$$

Considering $x - b = 0$ as $\min_{x \in \mathbb{R}} (x - b)^2$

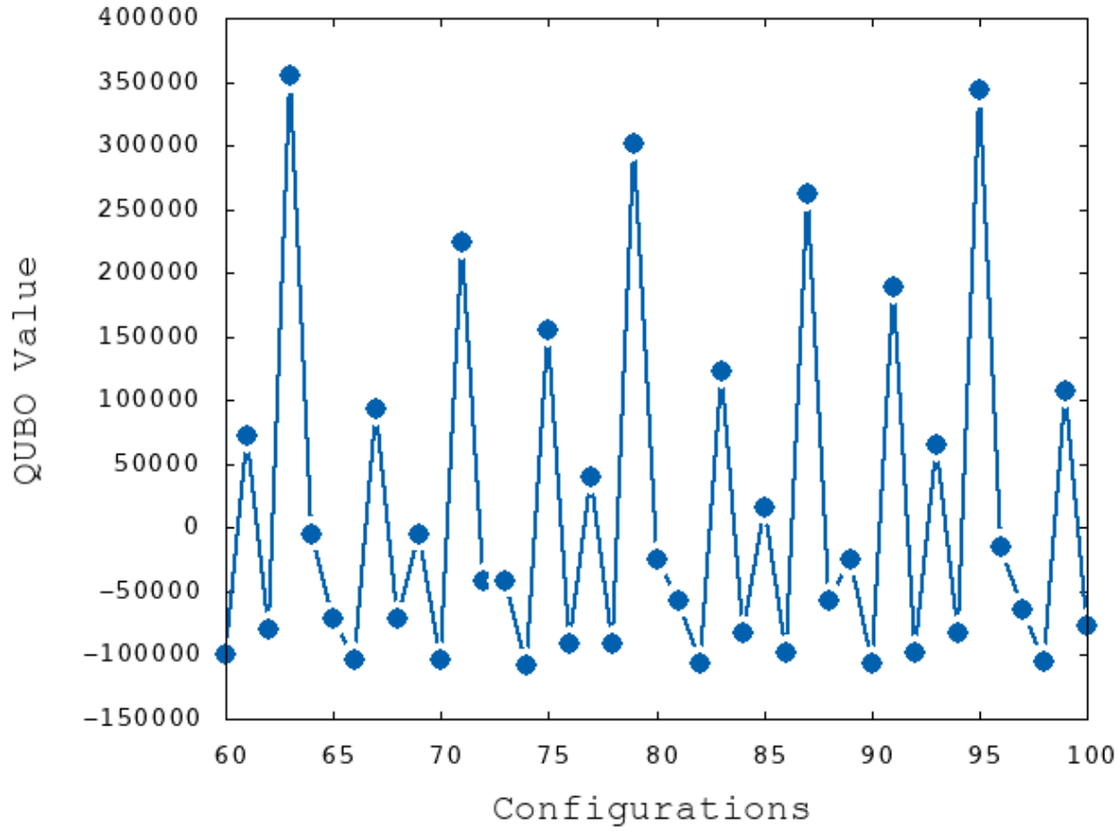
Graphical representation

**QUBO problems of this kind are particularly difficult to solve.
Especially with annealing techniques.**

This is due to the exponential dependence of the coefficients from the binary variable indices, which create numerous local minima very similar to the global minimum.



"Normal" QUBO landscape



"Real-variables" QUBO landscape

Solving a linear system

We have chosen to solve a linear system $A\mathbf{x} = b$, where

$$\mathbf{x} = (x_1, x_2, x_3) \text{ and } x_i \in [0,1].$$

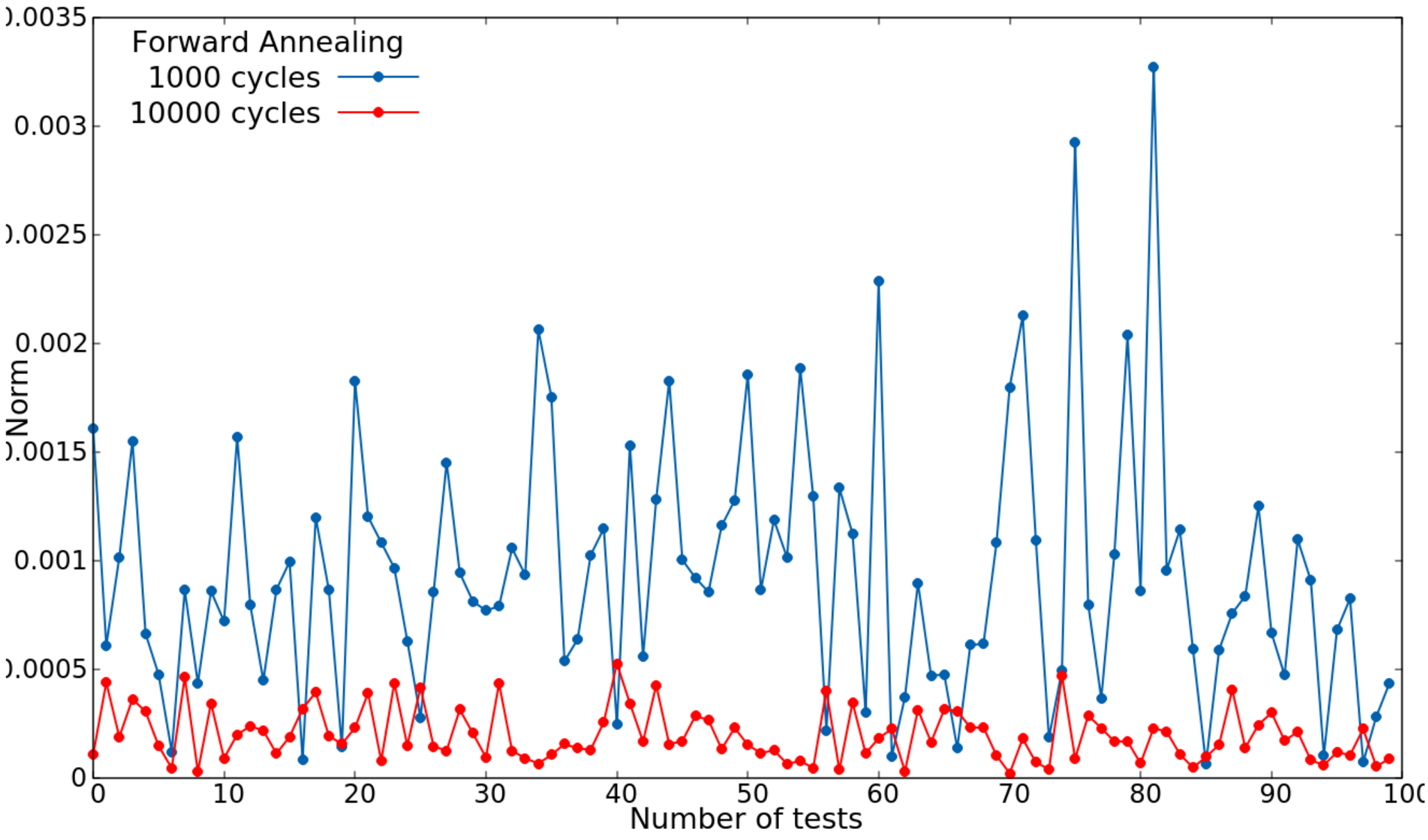
We represent $x_i = c \cdot \sum_{e=0}^{9} 2^e q_e, \quad c = 10^{-3}$ ($N = 10$, $a = 3$).

We will find $\mathbf{x}$ solving $\min_{\mathbf{x} \in [0,1]^3} ||A\mathbf{x} - b||_2^2$

$$\begin{bmatrix} 1.301 & 0.125 & 0.187 \\ 0.440 & 0.342 & 0.082 \\ 0.672 & 0.709 & 0.802 \\ 0.218 & 0.427 & 0.520 \\ 0.024 & 0.036 & 0.038 \end{bmatrix} \cdot \begin{bmatrix} 0.178 \\ 0.333 \\ 0.489 \end{bmatrix} = \begin{bmatrix} 0.365 \\ 0.232 \\ 0.748 \\ 0.435 \\ 0.035 \end{bmatrix}$$

Attempt number 1: Forward Annealing

100 attempts with 1,000 and 10,000 annealing cycles



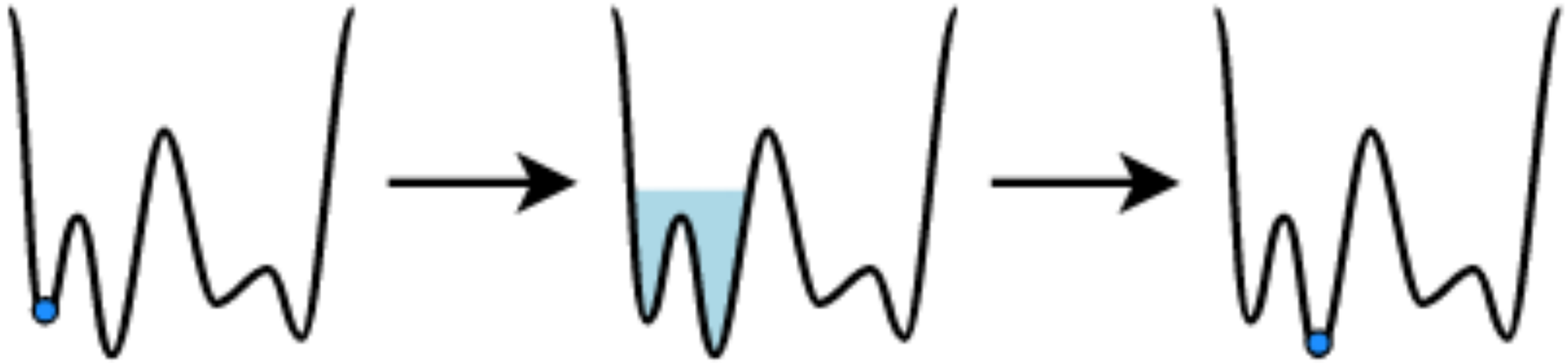
$$\begin{bmatrix} 0.179 \\ 0.338 \\ 0.483 \end{bmatrix}$$

$$\left(\begin{bmatrix} 0.178 \\ 0.333 \\ 0.489 \end{bmatrix} \right)$$

Local refinement of solutions:

Reverse Annealing

Introduced with the last D-Wave model, DWAVE2000Q



Starting point chosen by the user

Backward Annealing

Forward Annealing

During the Backward Annealing phase, the transverse field slowly increases up to a value chosen by the user (*Reversal Distance)*

The last Forward Annealing phase is a LOCAL quantum annealing search: how much local depends on the reversal distance value.

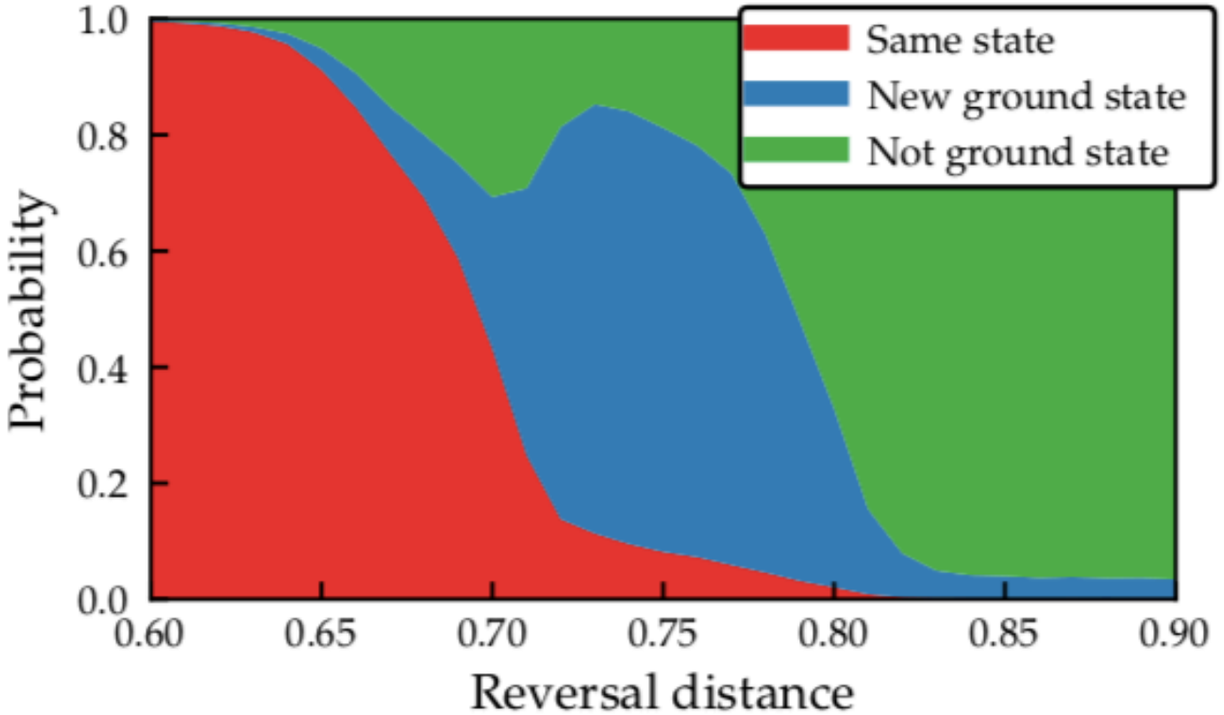
Image taken from *Reverse Quantum Annealing for Local Refinement of Solutions,* D-Wave White Papers, 2017

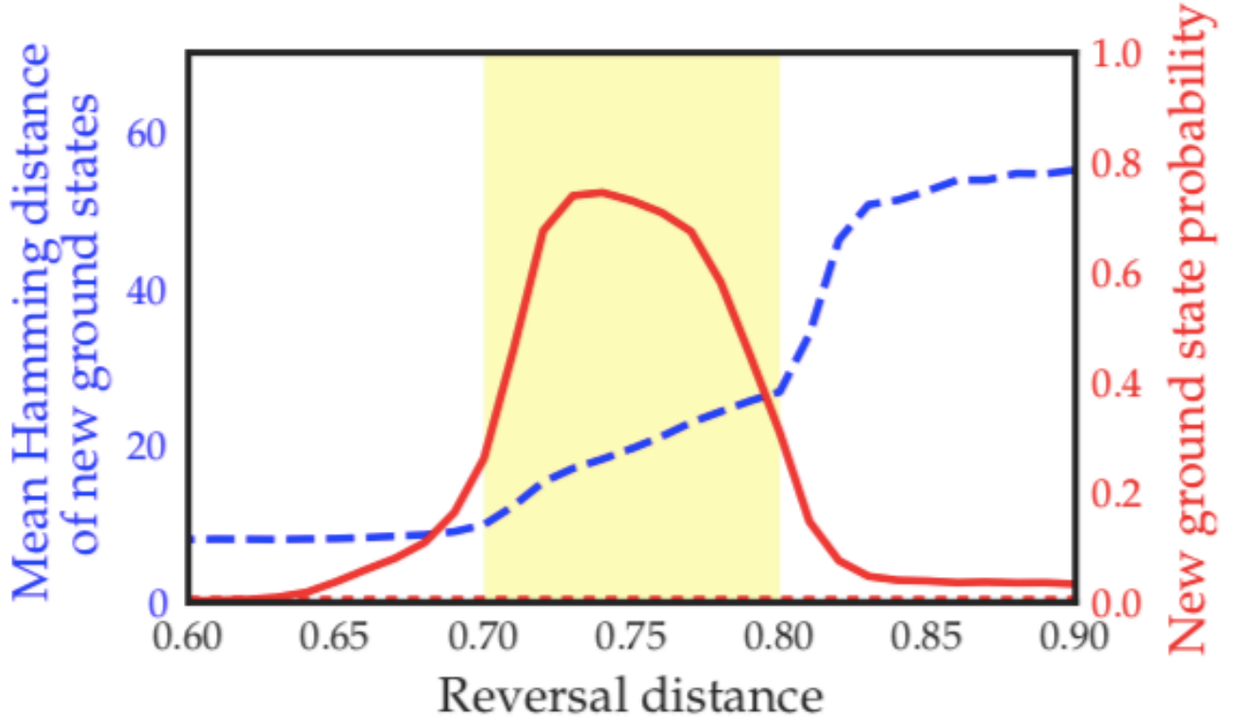
Tuning the reversal distance

D:wave
The Quantum Computing Company™

Reverse Quantum Annealing for Local Refinement of Solutions

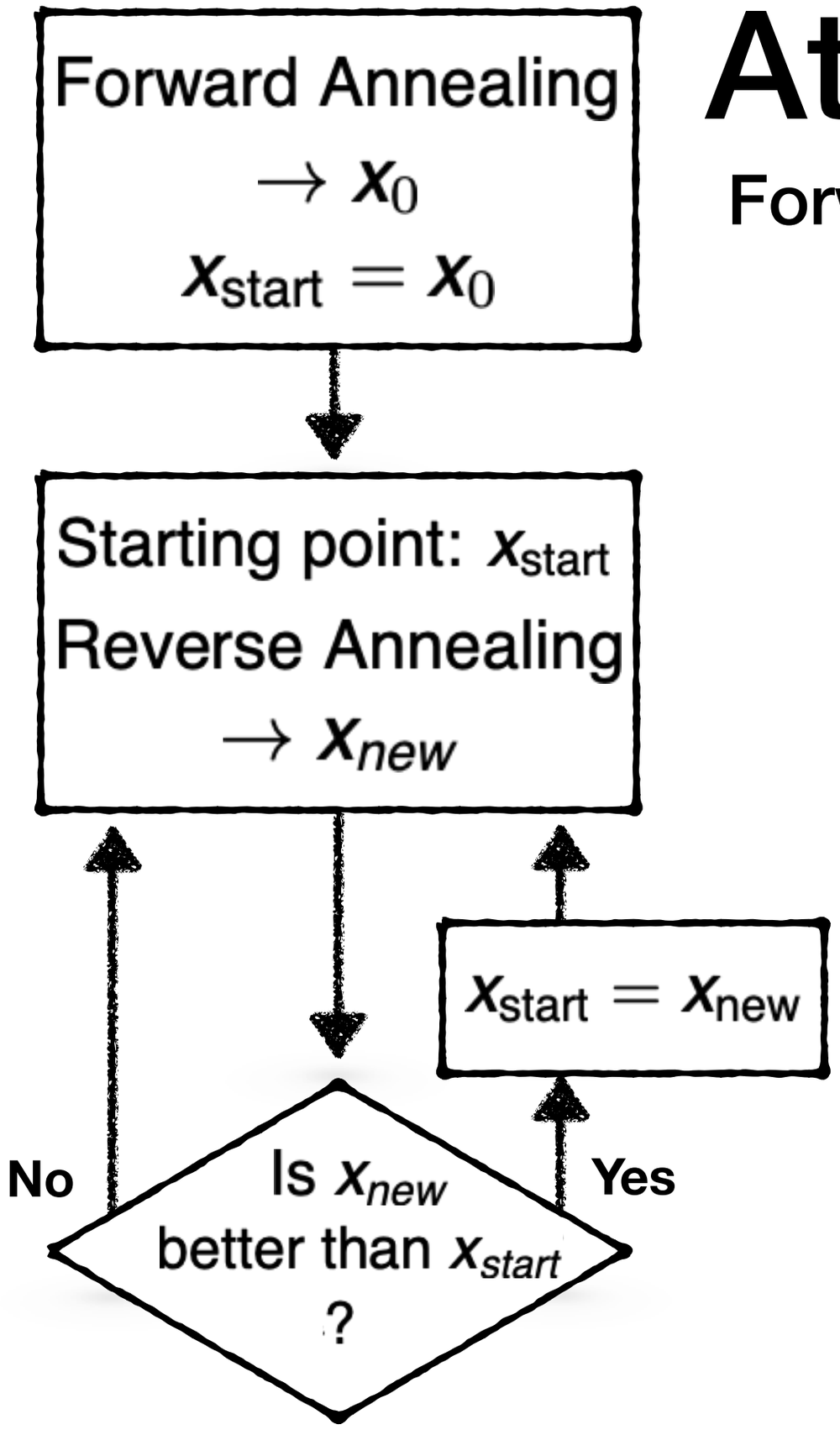
WHITEPAPER





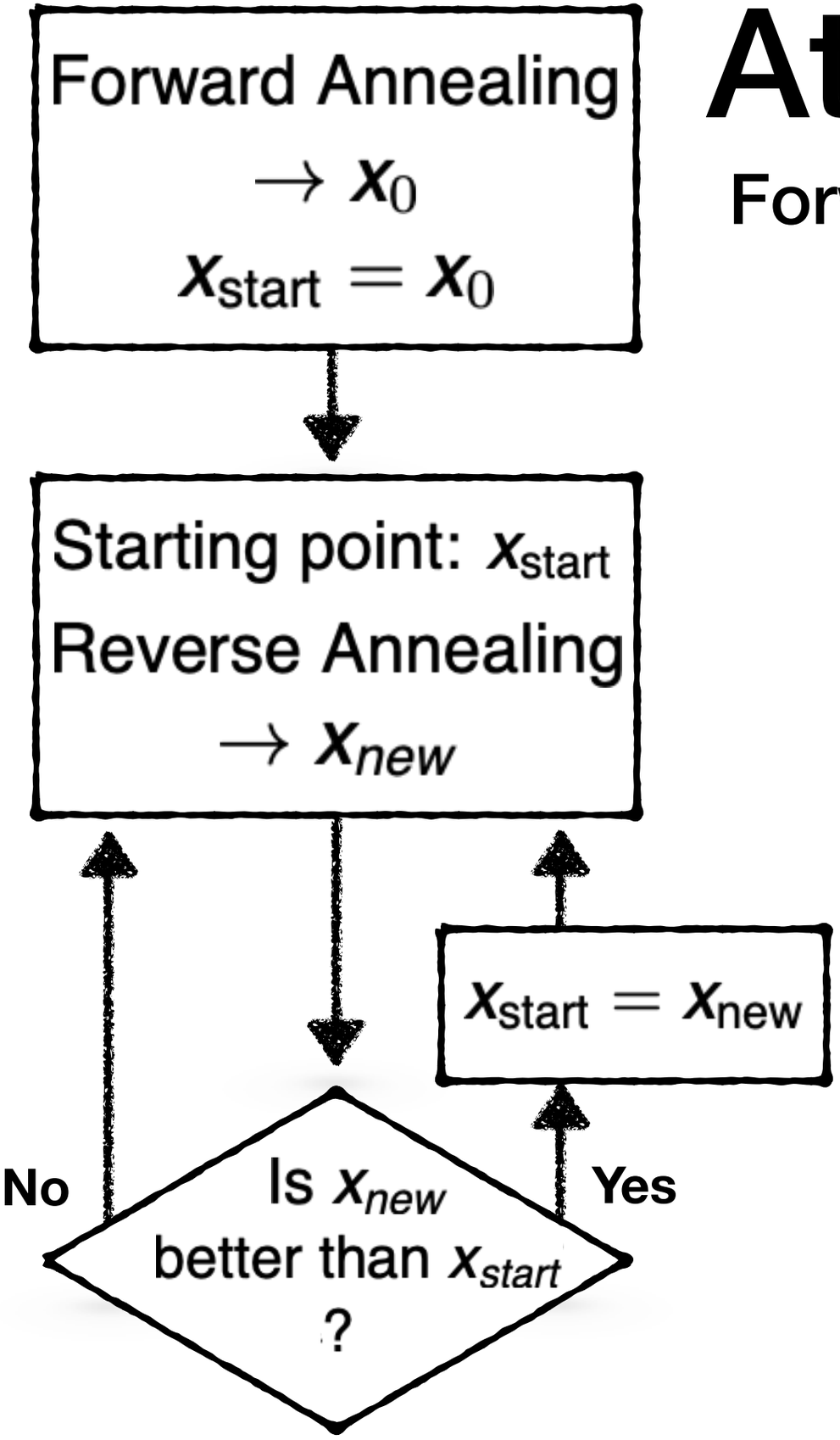
Attempt number 2:

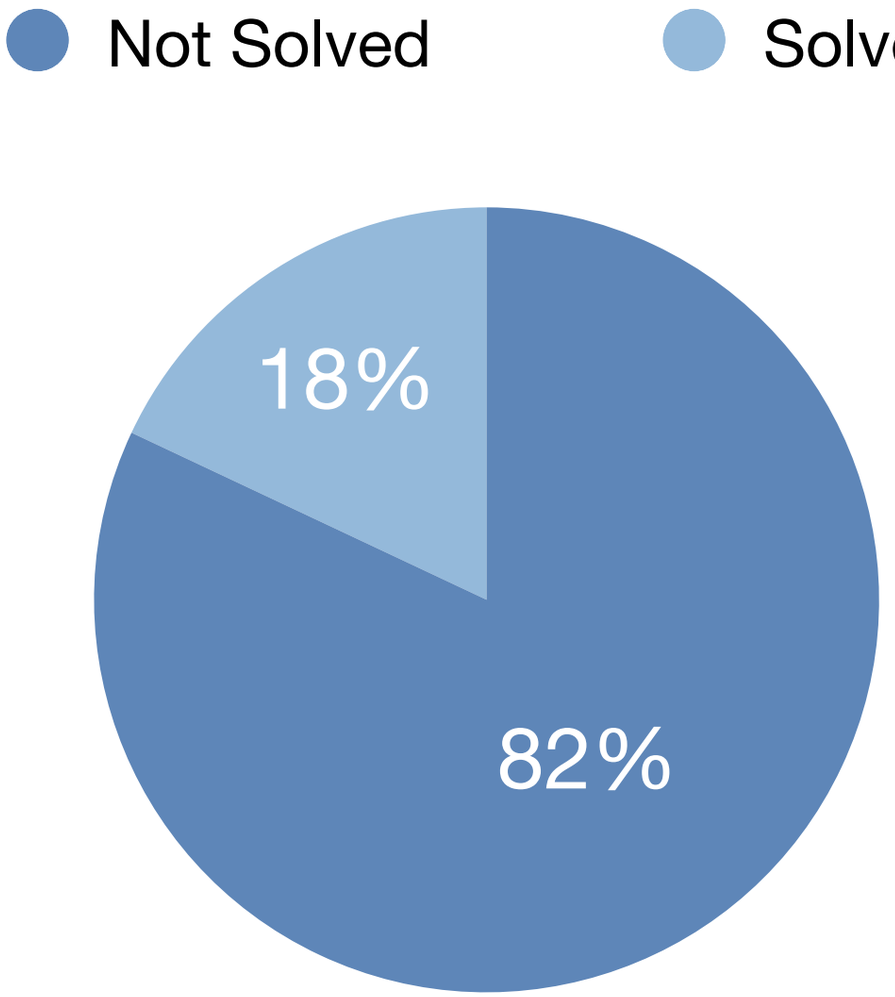
Forward Annealing + Reverse Annealing



Attempt number 2:

Forward Annealing + Reverse Annealing

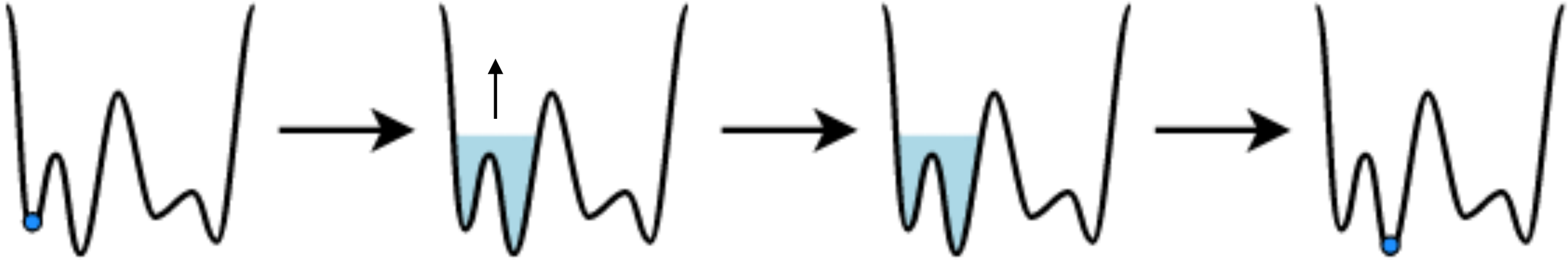




Pausing the annealing process

Being able to pause the annealing process is another of the new features introduced with the latest D-WAVE quantum annealer.

We can use the pause during a Reverse Annealing search in this way:



Starting point chosen by the user

Backward Annealing

Pause

Forward Annealing

Why pause? Because pausing the annealing process means better exploration of the selected zone, increasing the chances of obtaining a new global minimum.

But pay attention: pause can't be too long. For two main reasons:
1) it increase the computational time of each annealing cycle.
2) if it is too long, it may also risk to increase the search radius more than desired.

Correlation between pause and search radius

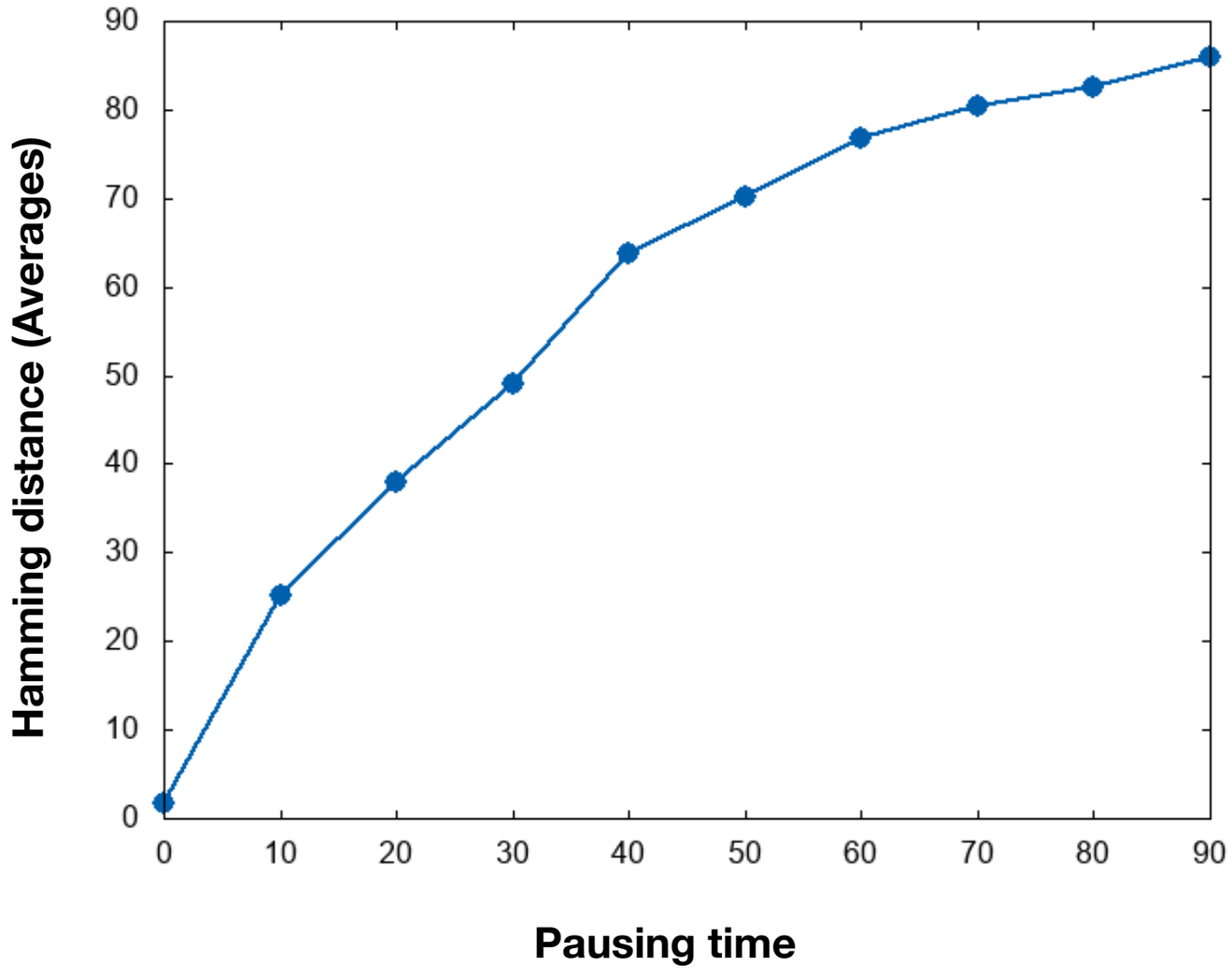
We can realize a posteriori the search radius of a reverse annealing search by analyzing the average distance between the solutions found by each cycle.

To do this, we choose the Hamming distance, a function written to calculate the distance between vectors of binary numbers.

We have observed that there is a correlation between the pause time and the average distance between the solutions obtained with each annealing cycle

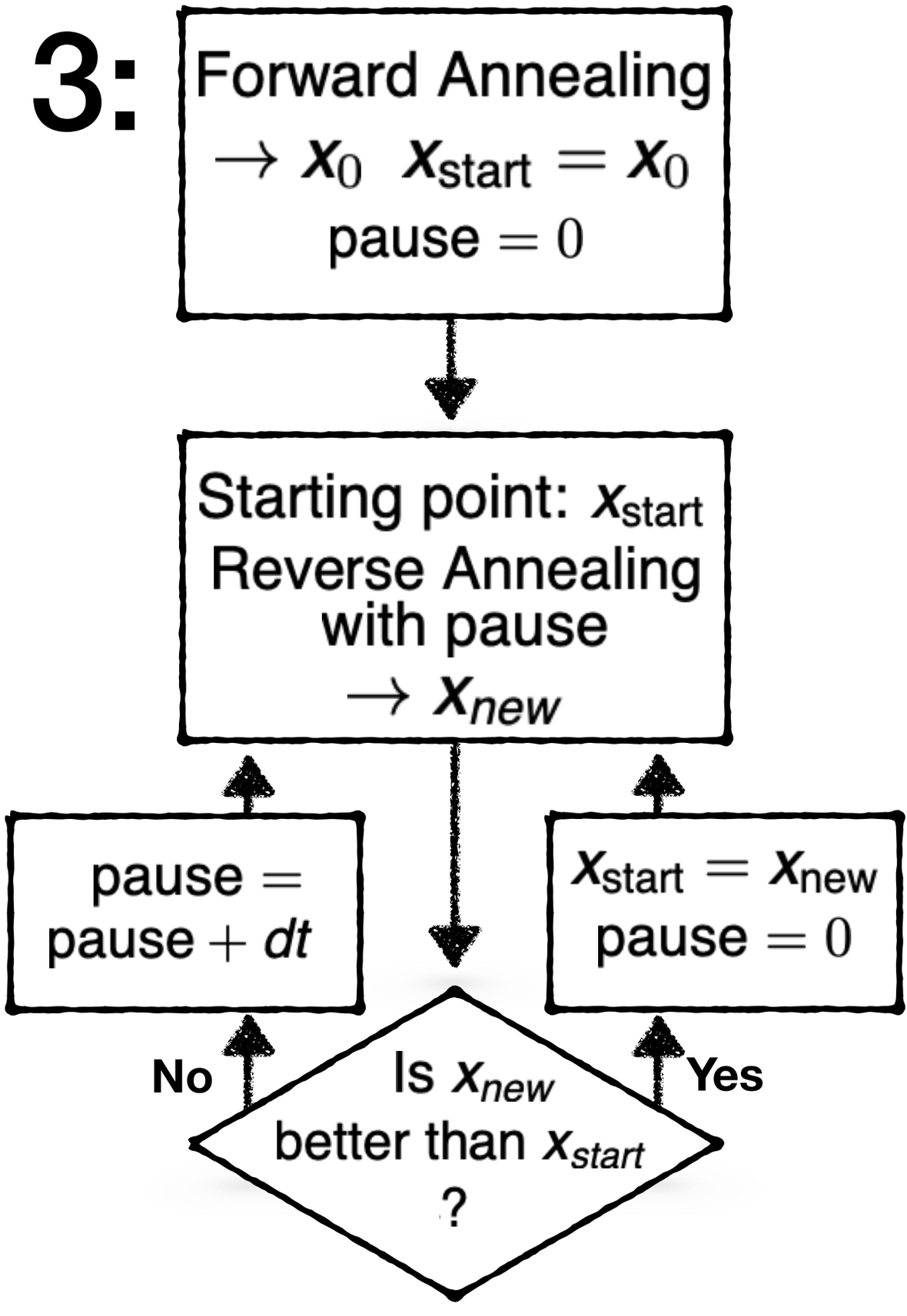
As with the reversal distance, here too we have to be careful about the right break time:

too little is not enough,
too much can lead to wrong results



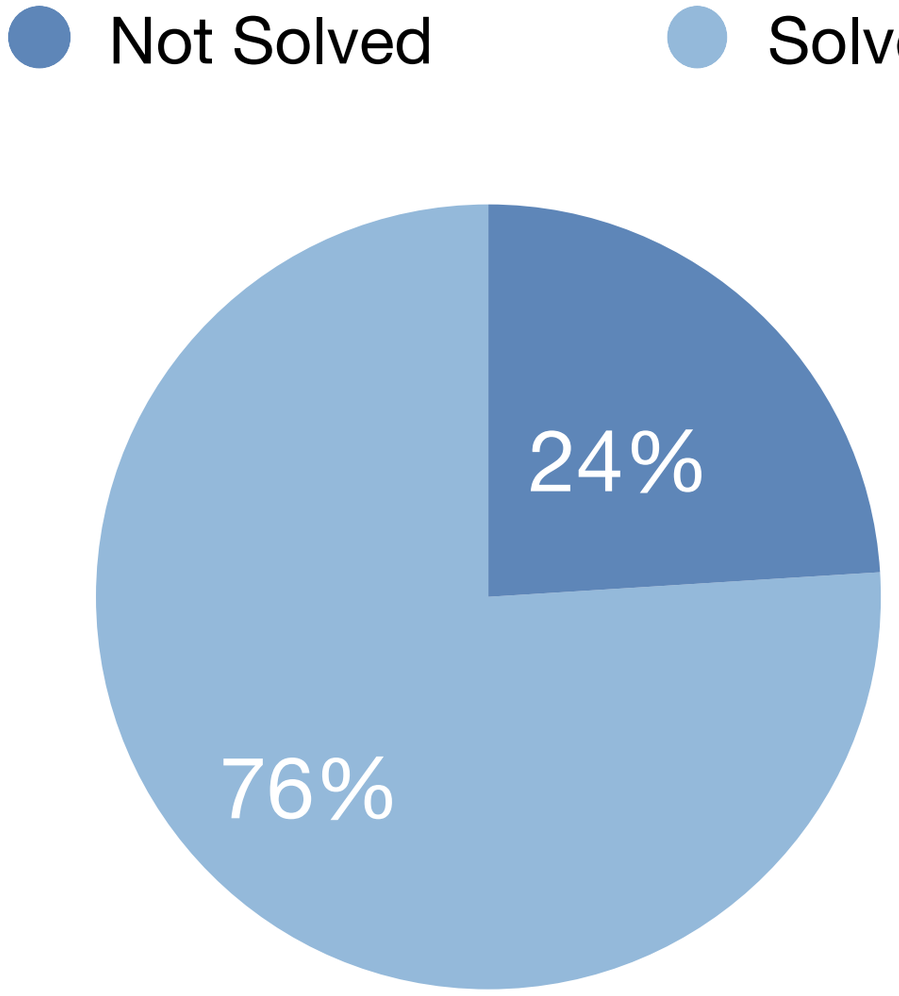
Attempt number 3:

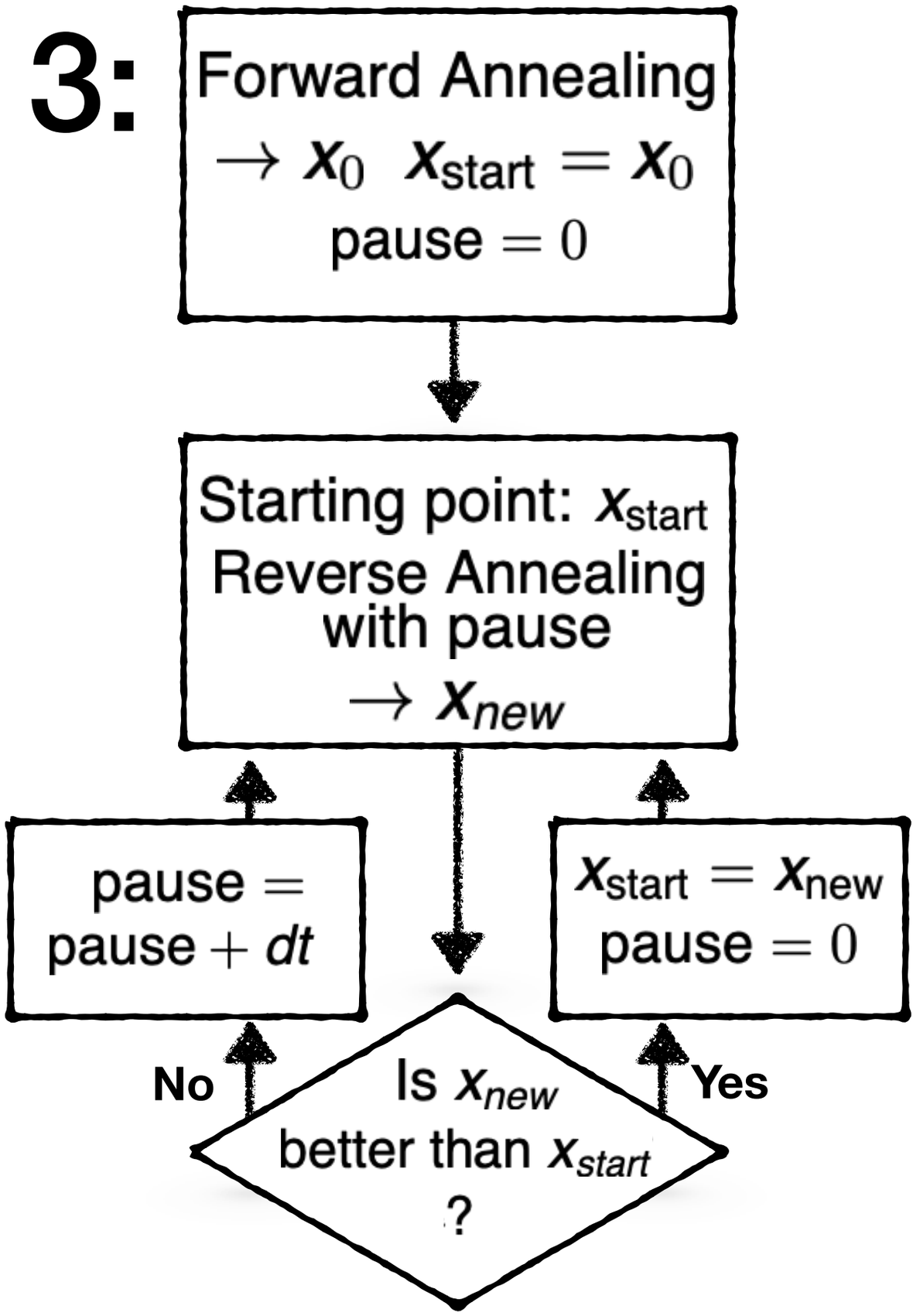
Forward Annealing + Reverse Annealing with pause



Attempt number 3:

Forward Annealing + Reverse Annealing with pause



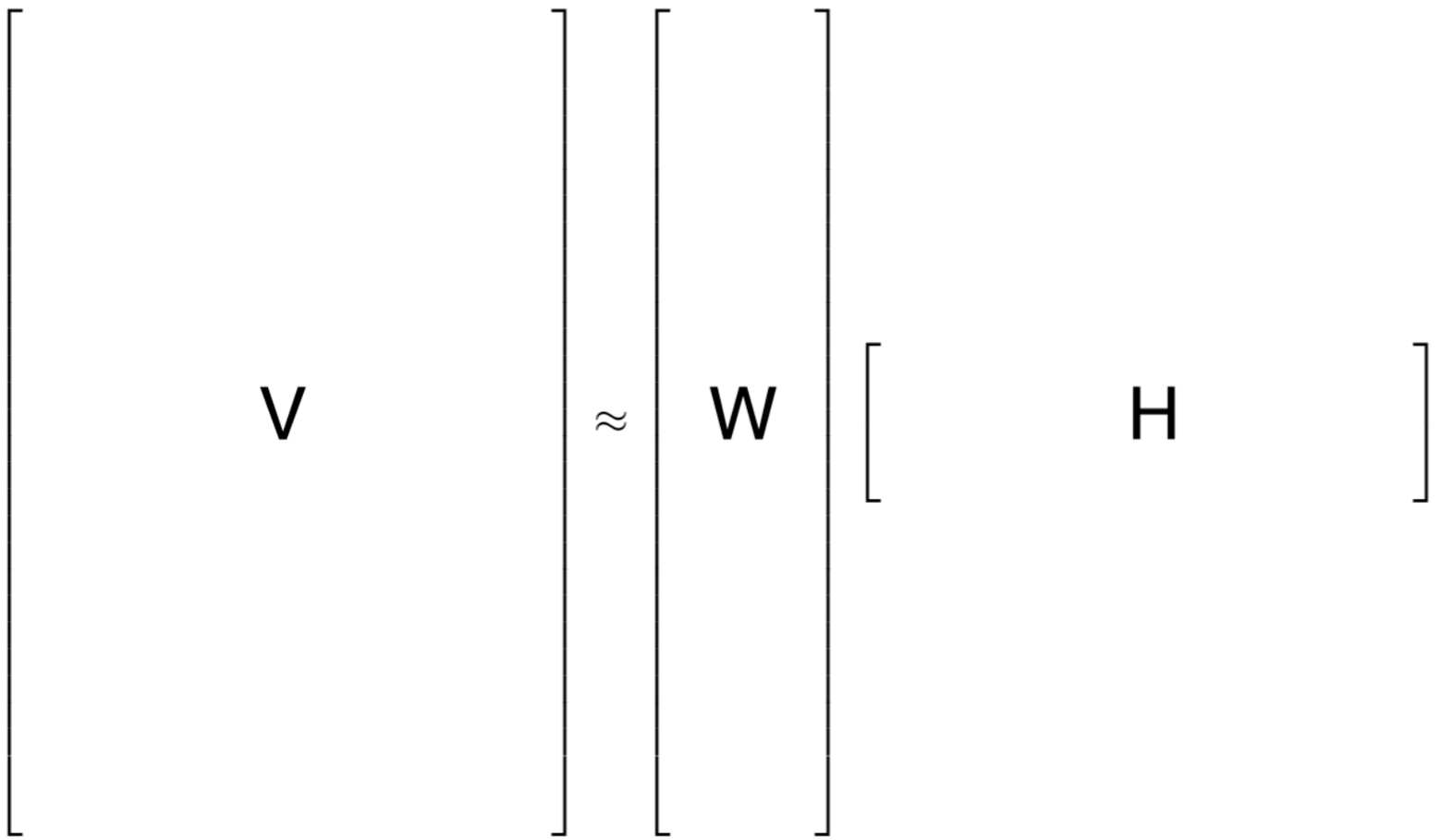


Low-rank Nonnegative Matrix Factorization

Given $V \in \mathbb{R}^{n \times m}$, find $W \in \mathbb{R}^{n \times k}$ and $H \in \mathbb{R}^{k \times m}$ such as

$$V \simeq WH, \qquad W_{ij} \geq 0 \quad H_{ij} \geq 0$$

Usually, k is a very small parameter

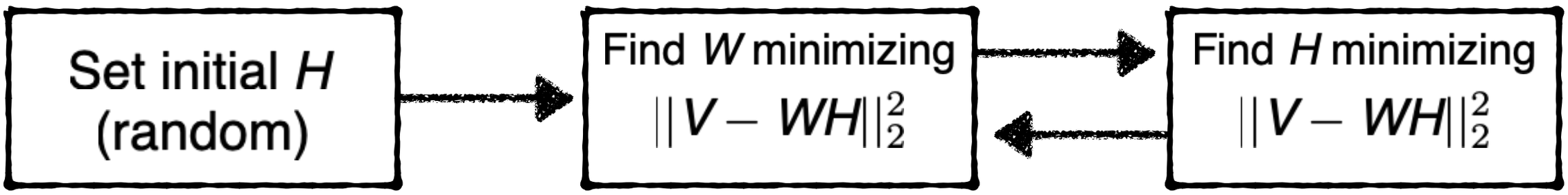


Our case

We want to perform a k=2 NMF.

$$V \simeq WH, \qquad W_{ij} \in [0,1] \quad H_{ij} \geq 0 \qquad V = \begin{bmatrix} 0.421 & 0.503 \\ 0.386 & 0.505 \end{bmatrix}$$

To calculate the factorization, we have chosen an ALS (Alternating Least Squares) approach:



D-WAVE SOLVER

Reverse annealing + pause.

CLASSIC SOLVER

Python library *lsqnonneg.py*

Problem decomposition of the D-WAVE part:

$$\min_{W \in [0,1]^{n \times k}} ||V - WH||_2^2 \quad \Rightarrow \quad \min_{W_i \in [0,1]^k} ||V_i - H^T W_i||_2^2 \qquad \forall i \in \{1, \ldots, n\}$$

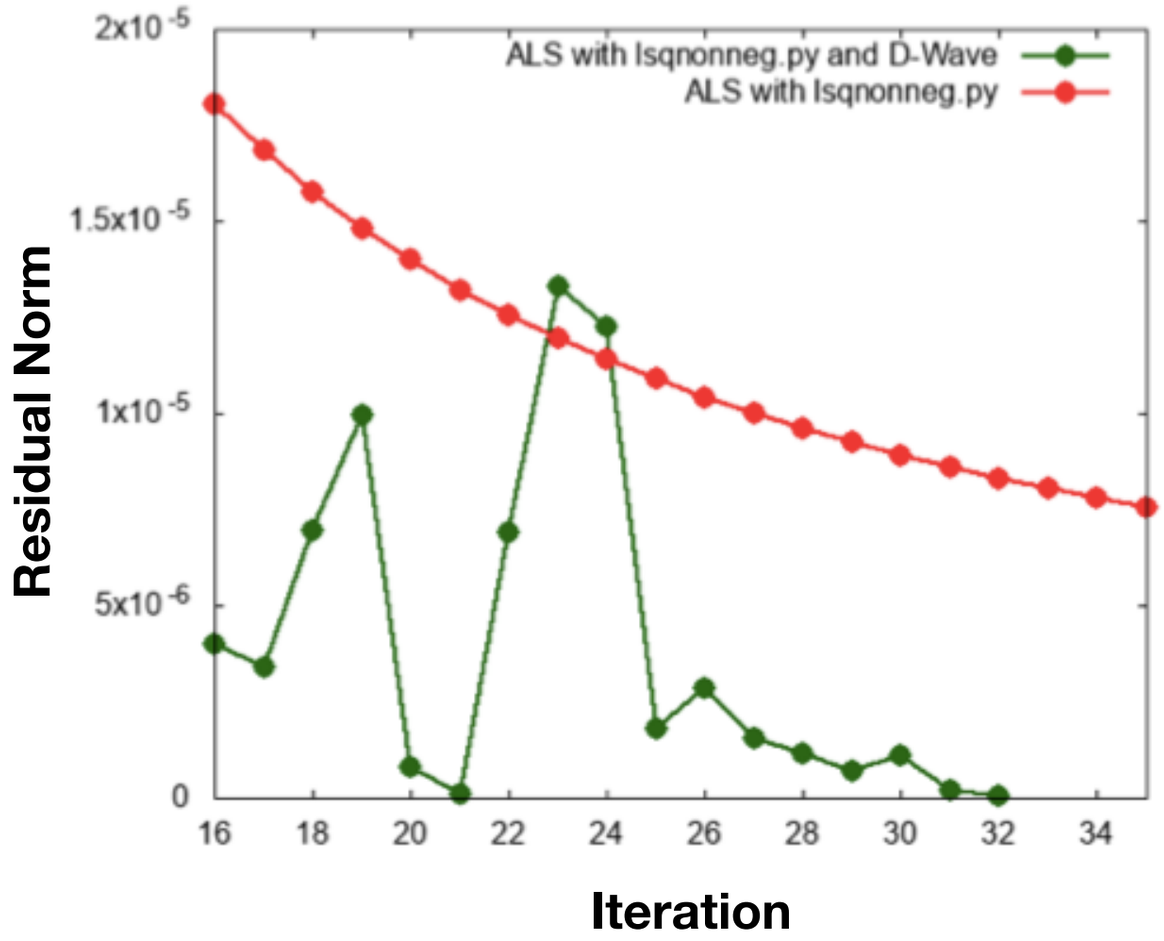
Results

We have tested our mixed DWAVE-classic algorithm versus the same algorithm entirely written with the python library lsqnonneg.py

We generated 35 random initial matrices "H".
For each of them, we started a double factorization, with both the algorithms.
We measure the goodness of a factorization with the value of the norm

$$||V - WH||_2^2$$

	D-Wave/ python	python
Convergence rate	100%	100%
Number of Iterations (Average)	40	10000
Best result: Residual Norm (Iterations)	5.98e-08 (32)	2.25e-07 (10000)



QPU Timing

$$T_{annealing} \times N_{cycles} \times 10^{-6} \text{ s}$$

QPU Timing for solving a linear system

$$T_{rev} = (T_{annealing} \times N_{cycles} + N_{calls} \times (T_{pause} + T_{annealing}) \times N_{cycles}) \times 10^{-6} \text{ s}$$

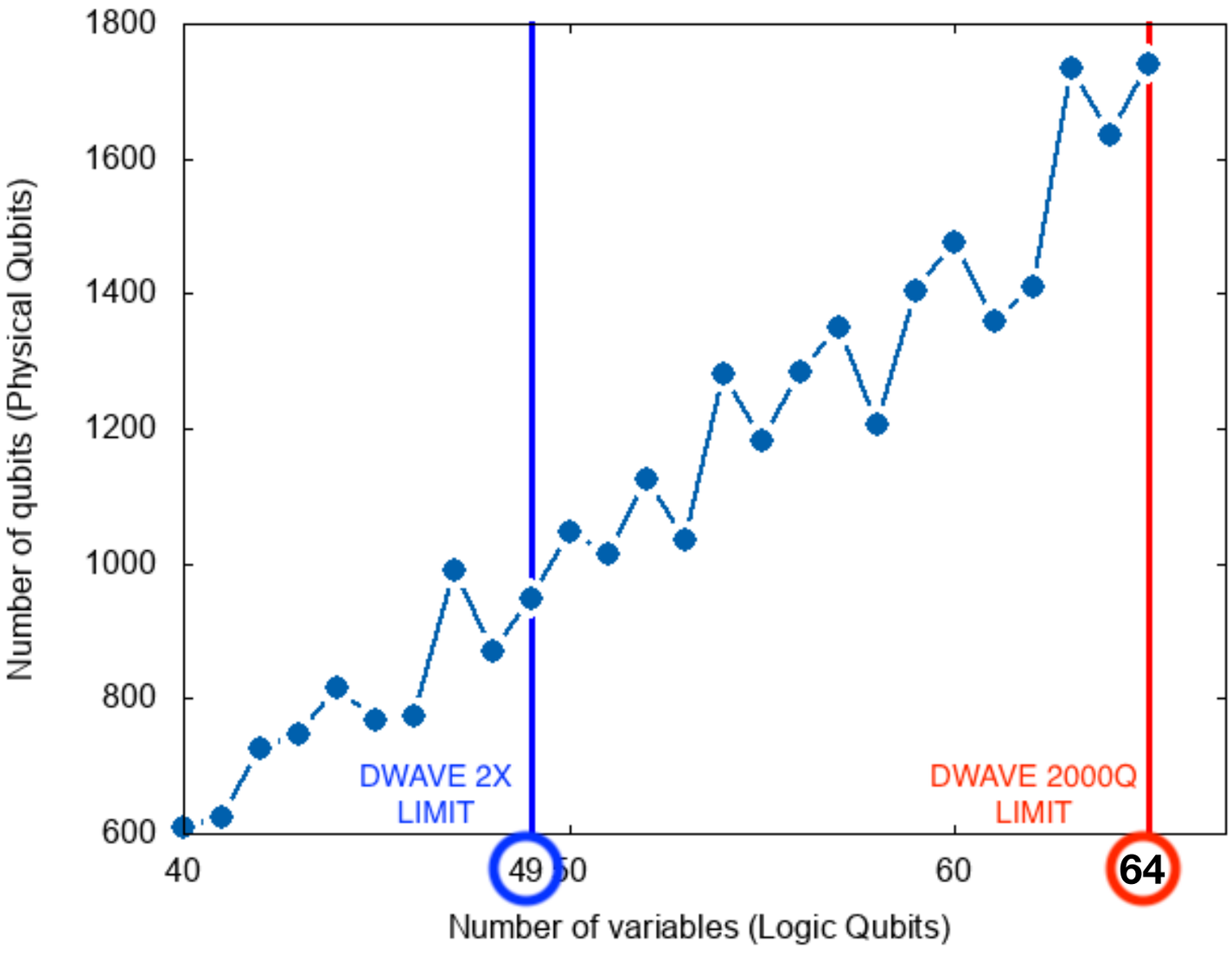
$$(1 \times 10000 + 10 \times (20 + 1) \times 10000) \times 10^{-6} = 21 \text{ s}$$

QPU Timing for solving Low-rank NMF

$$T_{fact} = N_{iterations} \times N_{rows} \times T_{rev}$$

$$40 \times 10 \times 21 = 8400 \text{ s}$$

Qubits utilization



DWAVE 2000Q

64 binary variables with a full connected graph

$$x_i = c \cdot \sum_{e=0}^{9} 2^e q_e,$$

$$c = 10^{-3}$$

We need 10 qubits for a single variable

Linear system: maximum 6 variables

Matrix factorization: only limitation on rank k. Maximum k=6.
No limitations for number of rows and columns of the matrix to factorize.

Thank you!