Computer Vision on the D-Wave 2000Q

Pankaj Gupta
Adobe Systems Incorporated

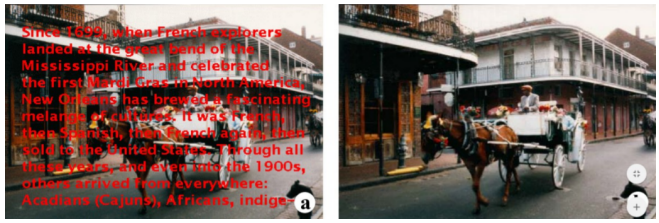
Nike Dattani*
Harvard-Smithsonian Center for Astrophysics
Department of Chemistry and Chemical Biology, McMaster University and
Fukui Institute for Fundamental Chemistry, Kyoto University, Kyoto, Japan
(Dated: 10th November 2019)

If you get robbed and the only way to get your wallet or purse back is for police to determine the identity of the thief based on a blurry surveillance video of a moving suspect, you might wish that a computer can "calculate" a sharp and clear version of the required image which shows clearly who it is that has your money. This is more complicated than the simple face-recognition algorithms used by Facebook for automatic tagging, or by Google for image search, because it involves first predicting what the unblurry version of an image should look like, and may involve video footage rather than simply stationary photographs.

The state-of-the-art method for solving this problem in industry is by solving a QUBO problem, and some examples from the classical-computer QUBO literature are below:

QUBO based in-painting by from Ref. [1].

QUBO based in-painting by from Ref. [1].



QUBO based de-blurring from Ref. [2].

QUBO based de-noising from Ref. [3].

Therefore this is a real-world problem that is "perfect" for the D-Wave machines, because the industry standard for solving the problem is to run a QUBO optimization on a classical computer. Image de-noising is performed by detectives for recovering partial finger-prints, by museums for reconstructing antique paintings, and by doctors for medical imaging, but in these cases the number of pixels (and hence the number of binary variables in the QUBO problem) is small. For terabyte-sized telescope images of galaxies, inter-stellar media, and possible life-containing exo-planets, runtimes are colossal and we welcome any speed-up even if only by a constant factor of 10 or even 2.

The algorithm works by minimizing two functions of binary variables simultaneously:

$$f_C(X_C) = \sum_{i=1}^{K} \alpha_i \log(1 + \frac{1}{2}(J_i \cdot X_C)^2) \tag{1}$$

$$f_v(X_v) = \frac{(N_v - X_v)^2}{2\sigma^2} \tag{2}$$

* n.dattani@cfa.harvard.edu

The first function was calibrated by using billions of good and bad images such that the minimization of this function removes as much noise as possible. Minimizing the second function helps to maintain fidelity to the original image (its minimum occurs when the new pixel N_v at location v is the same as the original pixel X_v, but it is minimized simultaneously with Eq. 1 so our new pixels will not be exactly the same). C denotes a clique of size 2×2, yielding 4-local interactions in the Hamiltonian. More accurate results can be obtained by using, for example, a 3×3 mask for the image processing, but this has never been applied in the literature since it would require quadratizing a 9-local Hamiltonian, which would greatly increase the number of auxiliary qubits required.

The number of binary variables (logical qubits) Q_L is equal to the number of pixels in the image. At each iteration, we would either accept the pixel from I or from P depending on whether the variable associated with that pixel is 0 or 1. The clique terms in Eq. 1 are quartic. We used auxiliary qubits to quadratize this quartic function, and what we are left with is a quadratic unconstrained boolean optimization (QUBO) problem involving Q_2 qubits coupled at most quadratically. Once we have quadratized the problem into QUBO form, we can chimerize it so that it can run on D-Wave annealers. Efficient chimerization is an active area of research, with new methods being proposed frequently, however we estimate the number of physical qubits Q_P needed by $\mathcal{O}(Q_2^2)$, which we know is an upper bound in the case for chimerizing a fully connected graph. Therefore, with $Q_A = 2048$ qubits available, we can have at least $Q_2 = 45$ qubits in the QUBO problem, so we will explore images of size 3×3 pixels ($Q_L = 9$ logical qubits) and 4×4 pixels ($Q_L = 16$ logical qubits). We do not investigate smaller images because the size of the image processing mask is 2×2. Below we show that the computer vision algorithm still works even for small images:

I. CHIMERIZATION OF A QUBO PROBLEM FOR DENOISING A 3×3 IMAGE

$$I = \begin{pmatrix} 201 & 180 & 195 \\ 183 & 209 & 210 \\ 173 & 171 & 162 \end{pmatrix} P = \begin{pmatrix} 242 & 230 & 255 \\ 234 & 252 & 239 \\ 238 & 204 & 241 \end{pmatrix}$$

Initial number of binary variables (logical qubits): $Q_L = 9$.

Number of qubits after quadratization: 29

Graph connectivity for 29 spin Ising problem:

$M =$

11768	14864	0	3284	14344	0	0	0	0	-11820	0	0	0	0	-7288	-13068	-12018	0	0	0	0	0	0	0	0	0	0	0	0
14864	44700	10500	1170	60370	27952	0	0	0	-11820	-29144	0	0	-2504	0	-13068	-12018	0	-32400	-20472	-12064	0	0	0	0	0	0	0	0
0	10500	33716	0	3694	24318	0	0	0	0	-29144	0	0	0	0	0	0	-1762	0	-20472	-12064	0	0	0	0	0	0	0	0
3284	1170	0	111320	12740	0	-620	16326	0	-11820	0	-17192	0	-2504	-7288	0	-12018	0	0	0	0	0	-41416	-7756	-2760	0	0	0	0
14344	60370	3694	12740	78878	1478	10878	11602	4486	-11820	-29144	-17192	-3770	-2504	-7288	-13068	0	-1762	-32400	0	-12064	-9702	0	-7756	-2760	0	-816	-4060	-9072
0	27952	24318	0	1478	29988	0	11230	4238	0	-29144	0	-3770	0	0	0	0	-1762	-32400	-20472	0	0	0	0	0	-4268	0	-4060	-9072
0	0	0	-620	10878	0	45574	1250	0	0	0	-17192	0	0	0	0	0	0	0	0	0	-9702	-41416	0	-2760	0	0	0	0
0	0	0	16326	11602	11230	1250	143330	7026	0	0	-17192	-3770	0	0	0	0	0	0	0	0	-9702	-41416	-7756	0	-4268	-816	0	-9072
0	0	0	0	4486	4238	0	7026	5396	0	0	0	-3770	0	0	0	0	0	0	0	0	0	0	0	0	-4268	-816	-4060	0
-11820	-11820	0	-11820	-11820	0	0	0	0	35460	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-29144	-29144	0	-29144	-29144	0	0	0	0	87432	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-17192	-17192	0	-17192	-17192	0	0	0	25788	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-3770	-3770	0	-3770	-3770	0	0	0	11310	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-2504	0	-2504	-2504	0	0	0	0	0	0	0	0	2504	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-7288	0	0	-7288	-7288	0	0	0	0	0	0	0	0	0	7288	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-13068	-13068	0	0	-13068	0	0	0	0	0	0	0	0	0	0	13068	0	0	0	0	0	0	0	0	0	0	0	0	0
-12018	-12018	0	-12018	0	0	0	0	0	0	0	0	0	0	0	0	12018	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1762	0	-1762	-1762	0	0	0	0	0	0	0	0	0	0	0	1762	0	0	0	0	0	0	0	0	0	0	0
0	-32400	0	0	-32400	-32400	0	0	0	0	0	0	0	0	0	0	0	0	32400	0	0	0	0	0	0	0	0	0	0
0	-20400	-20472	0	0	-20472	0	0	0	0	0	0	0	0	0	0	0	0	0	20472	0	0	0	0	0	0	0	0	0
0	-12064	-12064	0	-12064	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12064	0	0	0	0	0	0	0	0
0	0	0	0	-9702	0	-9702	-9704	0	0	0	0	0	0	0	0	0	0	0	0	0	9702	0	0	0	0	0	0	0
0	0	0	-41416	0	0	-41416	-41416	0	0	0	0	0	0	0	0	0	0	0	0	0	0	41416	0	0	0	0	0	0
0	0	0	-7756	-7756	0	0	-7756	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15512	0	0	0	0	0
0	0	0	-2760	-2760	0	-2760	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5520	0	0	0	0
0	0	0	0	0	-4268	0	-4268	-4268	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4268	0	0	0
0	0	0	0	-816	0	0	-816	-816	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1632	0	0
0	0	0	0	-4060	-4060	0	0	-4060	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4060	0
0	0	0	0	-9072	-9072	0	-9072	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9072

The function to optimize, corresponding to the above matrix is:

$$11768x_1 + 14864x_1x_2 + 3284x_1x_4 + 14344x_1x_5 - 11820x_1x_{10} - 7288x_1x_{15} - 13068x_1x_{16} - 12018x_1x_{17} + 44700x_2 + 10500x_2x_3 + 1170x_2x_4 + 60370x_2x_5 + 27952x_2x_6 - 11820x_2x_{10} - 29144x_2x_{11} - 2504x_2x_{14} - 13068x_2x_{16} - 12018x_2x_{17} - 32400x_2x_{19} - 20472x_2x_{20} - 12064x_2x_{21} + 33716x_3 + 3694x_3x_5 + 24318x_3x_6 - 29144x_3x_{11} - 1762x_3x_{18} - 20472x_3x_{20} - 12064x_3x_{21} + 111320x_4 + 12740x_4x_5 - 620x_4x_7 + 16326x_4x_8 - 11820x_4x_{10} - 17192x_4x_{12} - 2504x_4x_{14} - 7288x_4x_{15} - 12018x_4x_{17} - 41416x_4x_{23} - 7756x_4x_{24} - 2760x_4x_{25} + 78878x_5 + 1478x_5x_6 + 10878x_5x_7 + 11602x_5x_8 + 4486x_5x_9 - 11820x_5x_{10} - 29144x_5x_{11} - 17192x_5x_{12} - 3770x_5x_{13} - 2504x_5x_{14} - 7288x_5x_{15} - 13068x_5x_{16} - 1762x_5x_{18} - 32400x_5x_{19} - 12064x_5x_{21} - 9704x_5x_{22} - 7756x_5x_{24} - 2760x_5x_{25} - 816x_5x_{27} - 4060x_5x_{28} - 9072x_5x_{29} + 29988x_6 + 11230x_6x_8 + 4238x_6x_9 - 29144x_6x_{11} - 3770x_6x_{13} - 1762x_6x_{18} - 32400x_6x_{19} - 20472x_6x_{20} - 4268x_6x_{26} - 4060x_6x_{28} - 9072x_6x_{29} + 45574x_7 + 1250x_7x_8 - 17192x_7x_{12} - 9702x_7x_{22} - 41416x_7x_{23} - 2760x_7x_{25} + 143330x_8 + 7026x_8x_9 - 17192x_8x_{12} - 3770x_8x_{13} - 9702x_8x_{22} - 41416x_8x_{23} - 7756x_8x_{24} - 4268x_8x_{26} - 816x_8x_{27} - 9072x_8x_{29} + 5396x_9 - 3770x_9x_{13} - 4268x_9x_{26} - 816x_9x_{27} - 4060x_9x_{28} + 35460x_{10} + 87432x_{11} + 25788x_{12} + 11310x_{13} + 2504x_{14} + 7288x_{15} + 13068x_{16} + 12018x_{17} + 1762x_{18} + 32400x_{19} + 20472x_{20} + 12064x_{21} + 9702x_{22} + 41416x_{23} + 15512x_{24} + 5520x_{25} + 4268x_{26} + 1632x_{27} + 4060x_{28} + 9072x_{29}$$

Figure 1.

The embeddings obtained for the Chimera architecture are:

logical qubit 0 :	[339, 467, 595, 723, 851, 864, 973, 979, 981, 989, 992, 997, 1005, 1013]	chain length : 14
logical qubit 1 :	[590, 594, 598, 606, 614, 617, 622, 630, 638, 745]	chain length : 10
logical qubit 2 :	[251, 252, 379, 382, 507, 635, 636, 763, 891, 1019, 1021, 1147, 1148]	chain length : 13
logical qubit 3 :	[331, 332, 340, 344, 348, 354, 356, 364, 372, 380, 482, 610, 738, 866]	chain length : 14
logical qubit 4 :	[242, 370, 498, 626, 628, 754, 882, 1010, 1138]	chain length : 9
logical qubit 5 :	[346, 351, 359, 367, 375, 383, 474, 602, 605, 730, 858, 986]	chain length : 12
logical qubit 6:	[243, 371, 499, 627, 755, 759, 767, 883, 1011, 1139]	chain length : 10
logical qubit 7 :	[578, 633, 706, 711, 761, 834, 838, 846, 854, 862, 870, 878, 886, 889, 894]	chain length : 15
logical qubit 8 :	[234, 237, 245, 253, 362, 490, 618, 620, 743, 746, 751, 874, 1002, 1130]	chain length : 14
logical qubit 9 :	[241, 369, 497, 621, 625, 629, 637, 753, 881, 1009, 1137]	chain length : 11
logical qubit 10:	[459, 587, 715, 843, 844, 852, 860, 971, 1099, 1100, 1108, 1116, 1124, 1132, 1140]	*chain length : 15*
logical qubit 11 :	[232, 360, 488, 616, 744, 872, 876, 884, 892, 1000, 1128]	*chain length : 11*
logical qubit 12 :	[200, 204, 212, 217, 220, 228, 236, 244, 328, 456, 584, 589, 597, 712, 840]	*chain length : 15*
logical qubit 13 :	[448, 450, 452, 460, 466, 468, 476, 484, 492, 500, 508, 576, 704, 710]	*chain length : 14*
logical qubit 14 :	[219, 347, 350, 358, 366, 374, 475, 588, 596, 603, 604, 612, 731, 859]	*chain length : 14*
logical qubit 15 :	[336, 464, 592, 720, 848, 976, 982, 990, 998, 1006, 1014, 1022]	*chain length : 12*
logical qubit 16 :	[718, 726, 734, 742, 750, 752, 758, 766]	*chain length : 8*
logical qubit 17 :	[504, 632, 717, 725, 733, 741, 749, 757, 760, 765]	*chain length : 10*
logical qubit 18 :	[361, 462, 470, 478, 486, 489, 491, 494, 502, 510, 619, 747, 875, 1003]	*chain length : 14*
logical qubit 19 :	[352, 461, 469, 477, 480, 485, 493, 501, 505, 509, 608, 736]	*chain length : 12*
logical qubit 20 :	[378, 506, 634, 762, 845, 853, 861, 869, 873, 877, 885, 890, 893, 1001]	*chain length : 14*
logical qubit 21 :	[708, 716, 724, 732, 740, 748, 756, 764]	*chain length : 8*
logical qubit 22 :	[345, 473, 601, 729, 735, 857, 985, 988, 996, 1004, 1012, 1020]	*chain length : 12*
logical qubit 23 :	[472, 600, 728, 842, 847, 855, 856, 863, 871, 879, 887, 895]	*chain length : 12*
logical qubit 24 :	[337, 465, 593, 721, 849, 977, 983, 991, 999, 1007, 1015, 1023]	*chain length : 12*
logical qubit 25 :	[355, 463, 471, 479, 483, 487, 495, 503, 511, 611, 739, 867]	*chain length : 12*
logical qubit 26 :	[330, 333, 341, 349, 357, 365, 368, 373, 381, 458, 496, 586, 624, 714, 719]	*chain length : 15*
logical qubit 27 :	[457, 585, 591, 599, 607, 615, 623, 631, 639, 713, 841]	*chain length : 11*
logical qubit 28 :	[225, 353, 481, 609, 613, 737, 865, 868, 993, 1121, 1125, 1133, 1141, 1149]	*chain length : 14*

(3)

The embedded problem is:

{(866, 870): 16326.0, (344, 350): -1822.0, (362, 367): 4238.0, (354, 358): -1822.0, (587, 590): -29144.0, (628, 636): 3694.0, (627, 628): 5439.0, (620, 628): 2243.0, (857, 862): -41416.0, (1009, 1013): -11820.0, (361, 367): -16200.0, (1011, 1012): -41416.0, (992, 998): -4356.0, (723, 726): -12018.0, (738, 742): -12018.0, (891, 893): -4021.3333333333335, (602, 606): 27952.0, (882, 884): -17192.0, (626, 631): -4060.0, (634, 636): -4021.3333333333335, (883, 884): -17192.0, (865, 870): -9072.0, (355, 359): -2134.0, (1010, 1014): -13068.0, (371, 372): -620.0, (490, 495): -2134.0, (370, 373): -408.0, (597, 605): -3770.0, (858, 862): 11230.0, (474, 479): -2134.0, (634, 638): -6032.0, (498, 500): -2504.0, (595, 598): 14854.0, (489, 617): -16200.0, (625, 630): -5910.0, (379, 383): 24318.0, (242, 245): 2243.0, (761, 764): -4851.0, (866, 871): -3878.0, (872, 878): -8596.0, (711, 719): -816.0, (234, 236): -3770.0, (754, 759): 5439.0, (761, 767): 625.0, (602, 607): -4060.0, (882, 885): -12064.0, (369, 372): -11820.0, (339, 340): 3284.0, (242, 244): -3770.0, (754, 756): -9702.0, (474, 478): -16200.0, (867, 870): -4268.0, (466, 594): -2504.0, (1011, 1015): -2760.0, (856, 862): -2585.3333333333335, (632, 636): -881.0, (858, 860): -29144.0, (378, 382): -4021.3333333333335, (979, 982): -4356.0, (1138, 1141): -9072.0, (840, 846): -3770.0, (592, 598): -13068.0, (595, 596): -7288.0, (608, 614): -20472.0, (635, 638): 10500.0, (624, 628): -408.0, (482, 484): -2504.0, (370, 375): 1478.0, (626, 630): 60370.0, (362, 365): -816.0, (755, 756): -9702.0, (739, 743): -2134.0, (626, 629): -5910.0, (345, 348): -41416.0, (507, 509): -20472.0, (617, 621): -5910.0, (360, 364): -17192.0, (352, 359): -10236.0, (370, 374): -7288.0, (745, 873): -6032.0, (610, 612): -1822.0, (763, 765): -881.0, (874, 878): 7026.0, (619, 622): -16200.0, (618, 623): -4060.0, (337, 340): -2760.0, (1138, 1140): -29144.0, (882, 886): 11602.0, (976, 981): -4356.0, (883, 886): 625.0, (498, 502): -32400.0, (889, 892): -8596.0, (610, 614): 1170.0, (474, 477): -10236.0, (842, 846): -2585.3333333333335, (882, 887): -7756.0, (353, 359): -4536.0, (347, 348): -1822.0, (745, 750): -12018.0, (889, 895): -2585.3333333333335, (754, 757): -1762.0, (625, 628): -5910.0, (344, 472): -3878.0, (605, 613): -4536.0, (370, 372): 12740.0, (1010, 1015): -2760.0, (706, 708): -4851.0, (1140, 1148): -29144.0, (1010, 1013): 14344.0, (730, 733): -1762.0}

II. CHIMERIZATION OF A QUBO PROBLEM FOR DENOISING A 4×4 IMAGE

$$I = \begin{pmatrix} 224 & 255 & 253 & 255 \\ 124 & 220 & 243 & 250 \\ 255 & 255 & 255 & 255 \\ 106 & 95 & 115 & 127 \end{pmatrix} P = \begin{pmatrix} 189 & 255 & 254 & 255 \\ 97 & 224 & 233 & 255 \\ 255 & 241 & 251 & 250 \\ 118 & 109 & 127 & 171 \end{pmatrix}$$

Initial number of binary variables (logical qubits): $Q_L = 16$.
Number of qubits after quadratization: $Q_Q = 44$
Graph connectivity for 44-spin ising problem:

$$M = \begin{bmatrix}
7560 & 66 & 0 & 0 & -3595 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -24 & -1 & -79 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
66 & -271 & 52 & 0 & 41 & 374 & -748 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -152 & 0 & 0 & 0 & -4 & 0 & -1 & -79 & 0 & -273 & -498 & -95 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 52 & 3469 & 328 & 0 & 189 & 1288 & 154 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -152 & -400 & 0 & 0 & 0 & 0 & 0 & 0 & -283 & 0 & -498 & -95 & 0 & -356 & -214 & -190 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 328 & 867 & 0 & 0 & -551 & 161 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -400 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -43 & 0 & -214 & -190 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3595 & 41 & 0 & 0 & 12795 & 379 & 0 & 0 & -4522 & -955 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -206 & 0 & -4 & -24 & 0 & -79 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -98 & -191 & -184 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
15 & 374 & 189 & 0 & 379 & 765 & -1430 & 0 & 244 & 38 & 0 & 0 & 0 & 0 & 0 & 0 & -152 & 0 & -206 & 0 & -4 & -24 & -1 & 0 & -283 & -273 & 0 & -95 & 0 & 0 & 0 & 0 & -155 & 0 & -191 & -184 & -587 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -748 & 1288 & -551 & 0 & -1430 & 57165 & 151 & 0 & -1597 & 0 & 4668 & 0 & 0 & 0 & 0 & -152 & -400 & 0 & 0 & 0 & 0 & 0 & 0 & -283 & -273 & -498 & 0 & -43 & -356 & 0 & -190 & 0 & 0 & 0 & 0 & -587 & -903 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 154 & 161 & 0 & 0 & 151 & 4168 & 0 & 0 & 0 & 1069 & 0 & 0 & 0 & 0 & 0 & -400 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -43 & -356 & -214 & 0 & 0 & 0 & 0 & 0 & 0 & -903 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -4522 & 244 & 0 & 0 & 21202 & -113 & 0 & 0 & 3614 & -195 & 0 & 0 & 0 & 0 & -206 & -2534 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -155 & -98 & 0 & -184 & 0 & 0 & 0 & -4108 & -2446 & -2034 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -955 & 38 & -1597 & 0 & -113 & 32094 & 0 & 0 & 2817 & -2819 & 2504 & 0 & 0 & 0 & -206 & -2534 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -155 & -98 & -191 & 0 & -587 & 0 & -2781 & 0 & -2446 & -2034 & -4096 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4668 & 1069 & 0 & 0 & 0 & 7223 & 0 & 0 & 558 & -1024 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -903 & 0 & 0 & 0 & 0 & 0 & -274 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3614 & 2817 & 0 & 0 & 456 & 3898 & 0 & 0 & 0 & 0 & 0 & -2534 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2781 & -4108 & 0 & -2034 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -195 & -2819 & 0 & 0 & 3898 & 15395 & 1762 & 0 & 0 & 0 & 0 & -2534 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2781 & -4108 & -2446 & 0 & -4096 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2504 & 0 & 558 & 0 & 1762 & -618 & 1185 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4096 & -274 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1024 & 0 & 0 & 1185 & 1875 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -274 \\
0 & -152 & -152 & 0 & 0 & -152 & -152 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 456 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -400 & -400 & 0 & 0 & -400 & -400 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 600 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -206 & -206 & 0 & 0 & -206 & -206 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 618 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2534 & -2534 & 0 & 0 & -2534 & -2534 & 0 & 0 & 0 & 0 & 0 & 7602 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -4 & 0 & 0 & -4 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-24 & 0 & 0 & 0 & -24 & -24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 48 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-79 & -79 & 0 & 0 & -79 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 158 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -283 & 0 & 0 & -283 & -283 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 283 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -273 & 0 & 0 & 0 & -273 & -273 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 546 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -498 & -498 & 0 & 0 & 0 & -498 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 498 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -95 & -95 & 0 & 0 & -95 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 95 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -43 & 0 & 0 & -43 & -43 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 43 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -356 & 0 & 0 & 0 & -356 & -356 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 356 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -214 & -214 & 0 & 0 & 0 & -214 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 428 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -190 & -190 & 0 & 0 & -190 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 380 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -155 & 0 & 0 & -155 & -155 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 155 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -98 & 0 & 0 & 0 & -98 & -98 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 196 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -191 & -191 & 0 & 0 & 0 & -191 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 191 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -184 & -184 & 0 & 0 & -184 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 184 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -587 & -587 & 0 & 0 & -587 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 587 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -903 & -903 & 0 & 0 & 0 & -903 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1806 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2781 & 0 & 0 & -2781 & -2781 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2781 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4108 & 0 & 0 & 0 & -4108 & -4108 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4108 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2446 & -2446 & 0 & 0 & 0 & -2446 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2446 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2034 & -2034 & 0 & 0 & -2034 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2034 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4096 & 0 & 0 & 0 & -4096 & -4096 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8192 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -274 & 0 & 0 & -274 & -274 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 274
\end{bmatrix}$$

The function to optimize, corresponding to the above matrix is:

$$7560x_1+66x_1x_2-3595x_1x_5+15x_1x_6-24x_1x_{22}-x_1x_{23}-79x_1x_{24}-271x_2+52x_2x_3+41x_2x_5+374x_2x_6-748x_2x_7-152x_2x_{17}-4x_2x_{21}-x_2x_{23}-79x_2x_{24}-273x_2x_{26}-498x_2x_{27}-95x_2x_{28}+3469x_3+328x_3x_4+189x_3x_6+1288x_3x_7+154x_3x_8-152x_3x_{17}-400x_3x_{18}-283x_3x_{25}-498x_3x_{27}-95x_3x_{28}-356x_3x_{30}-214x_3x_{31}-190x_3x_{32}+867x_4-551x_4x_7+161x_4x_8-400x_4x_{18}-43x_4x_{29}-214x_4x_{31}-190x_4x_{32}+12795x_5+379x_5x_6-4522x_5x_9-955x_5x_{10}-206x_5x_{19}-4x_5x_{21}-24x_5x_{22}-79x_5x_{24}-98x_5x_{34}-191x_5x_{35}-184x_5x_{36}+765x_6-1430x_6x_7+244x_6x_9+38x_6x_{10}-152x_6x_{17}-206x_6x_{19}-4x_6x_{21}-24x_6x_{22}-x_6x_{23}-283x_6x_{25}-273x_6x_{26}-95x_6x_{28}-155x_6x_{33}-191x_6x_{35}-184x_6x_{36}-587x_6x_{37}+57165x_7+151x_7x_8-1597x_7x_{10}+4668x_7x_{12}-152x_7x_{17}-400x_7x_{18}-283x_7x_{25}-273x_7x_{26}-498x_7x_{27}-43x_7x_{29}-356x_7x_{30}-190x_7x_{32}-587x_7x_{37}-903x_7x_{38}+4168x_8+1069x_8x_{12}-400x_8x_{18}-43x_8x_{29}-256x_8x_{30}-214x_8x_{31}-903x_8x_{38}+21202x_9-113x_9x_{10}+3614x_9x_{13}-195x_9x_{14}-206x_9x_{19}-2534x_9x_{20}-155x_9x_{33}-98x_9x_{34}-184x_9x_{36}-4108x_9x_{40}-2446x_9x_{41}-2034x_9x_{42}+32094x_{10}+2817x_{10}x_{13}-2819x_{10}x_{14}+2504x_{10}x_{15}-206x_{10}x_{19}-2534x_{10}x_{20}-155x_{10}x_{33}-98x_{10}x_{34}-191x_{10}x_{35}-587x_{10}x_{37}-2781x_{10}x_{39}-2446x_{10}x_{41}-2034x_{10}x_{42}-4096x_{10}x_{43}+7223x_{12}+558x_{12}x_{15}-1024x_{12}x_{16}-903x_{12}x_{38}-274x_{12}x_{44}+456x_{13}+3898x_{13}x_{14}-2534x_{13}x_{20}-2781x_{13}x_{39}-4108x_{13}x_{40}-2034x_{13}x_{42}+15395x_{14}+1762x_{14}x_{15}-2534x_{14}x_{20}-2781x_{14}x_{39}-4108x_{14}x_{40}-2446x_{14}x_{41}-4096x_{14}x_{43}-618x_{15}+1185x_{15}x_{16}-4096x_{15}x_{43}-274x_{15}x_{44}+1875x_{16}-274x_{16}x_{44}+456x_{17}+600x_{18}+618x_{19}+7602x_{20}+4x_{21}+48x_{22}+x_{23}+158x_{24}+283x_{25}+546x_{26}+498x_{27}+95x_{28}+43x_{29}+356x_{30}+428x_{31}+380x_{32}+155x_{33}+196x_{34}+191x_{35}+184x_{36}+587x_{37}+1806x_{38}+2781x_{39}+4108x_{40}+2446x_{41}+2034x_{42}+8192x_{43}+274x_{44}$$

The embeddings are:

logical qubit 0 : [720, 848, 953, 956, 964, 968, 972, 976, 980, 984, 988, 996, 1004, 1012, 1020, 1081, 1112, 1209, 1337, 1465, 1593, 1721, 1849] *chain length : 23*
logical qubit 1 : [728, 818, 820, 828, 836, 844, 852, 856, 860, 868, 876, 884, 892, 946, 1074, 1202, 1330, 1458, 1586, 1714, 1842] *chain length : 21*
logical qubit 2 : [1496, 1583, 1591, 1599, 1607, 1615, 1623, 1624, 1631, 1639, 1647, 1655, 1663, 1752] *chain length : 14*
logical qubit 3 : [738, 866, 994, 1122, 1219, 1222, 1227, 1230, 1238, 1246, 1250, 1254, 1262, 1270, 1278, 1355, 1478, 1483, 1486, 1611, 1739] *chain length : 21*
logical qubit 4 : [1362, 1455, 1463, 1471, 1479, 1487, 1490, 1495, 1503, 1511, 1519, 1527, 1535, 1618, 1746] *chain length : 15*
logical qubit 5 : [1345, 1473, 1601, 1710, 1718, 1726, 1729, 1734, 1742, 1750, 1758, 1766, 1774, 1782, 1790] *chain length : 15*
logical qubit 6 : [747, 875, 1003, 1127, 1131, 1135, 1143, 1146, 1151, 1259, 1387, 1515, 1643, 1771] *chain length : 14*
logical qubit 7 : [731, 859, 987, 1111, 1115, 1118, 1119, 1126, 1134, 1142, 1150, 1243, 1371, 1499, 1627, 1755, 1883] *chain length : 17*
logical qubit 8 : [752, 880, 1008, 1136, 1264, 1392, 1520, 1648, 1776, 1904, 1908, 1916] *chain length : 12*
logical qubit 9 : [705, 819, 821, 829, 833, 837, 845, 853, 861, 869, 877, 885, 893, 947, 1075, 1203, 1331, 1459, 1587, 1715, 1843] *chain length : 21*
logical qubit 10: [744, 749, 872, 1000, 1128, 1256, 1384, 1512, 1640, 1644, 1768, 1896, 2024, 2030, 2032, 2038, 2046] *chain length : 17*
logical qubit 11 : [1700, 1708, 1716, 1724, 1732, 1740, 1748, 1756, 1764, 1772, 1780, 1788] *chain length : 12*
logical qubit 12 : [682, 684, 692, 700, 708, 712, 716, 724, 732, 740, 748, 756, 764, 810, 938, 1066, 1194, 1322, 1450, 1578, 1706, 1834] *chain length : 22*
logical qubit 13 : [753, 757, 765, 881, 1009, 1137, 1265, 1393, 1521, 1649, 1777, 1905] *chain length : 12*
logical qubit 14 : [737, 865, 993, 1121, 1249, 1312, 1316, 1324, 1332, 1340, 1348, 1356, 1364, 1368, 1369, 1372, 1377, 1379, 1380, 1388, 1396, 1404, 1440, 1568, 1696] *chain length : 25*
logical qubit 15 : [723, 817, 822, 830, 838, 846, 850, 851, 854, 862, 870, 878, 886, 894, 945, 1073, 1201, 1329, 1457, 1585, 1713, 1841] *chain length : 22*
logical qubit 16 : [730, 735, 743, 751, 759, 767, 858, 986, 1114, 1242, 1370, 1498, 1502, 1626, 1754, 1882] *chain length : 16*
logical qubit 17 : [714, 842, 970, 1098, 1226, 1325, 1333, 1341, 1349, 1354, 1357, 1365, 1373, 1381, 1389, 1397, 1405, 1482, 1610, 1738] *chain length : 20*
logical qubit 18 : [715, 843, 955, 959, 967, 971, 975, 983, 991, 999, 1007, 1015, 1023, 1083, 1099, 1211, 1339, 1467, 1595, 1723, 1851] *chain length : 21*
logical qubit 19 : [683, 686, 694, 702, 710, 718, 726, 734, 742, 750, 758, 766, 811, 939, 1067, 1195, 1323, 1451, 1579, 1707, 1835] *chain length : 21*
logical qubit 20 : [1452, 1460, 1468, 1476, 1484, 1489, 1492, 1500, 1505, 1508, 1516, 1524, 1532, 1633, 1761] *chain length : 15*
logical qubit 21 : [1497, 1582, 1590, 1598, 1606, 1614, 1622, 1625, 1630, 1638, 1646, 1654, 1662, 1753] *chain length : 14*
logical qubit 22 : [755, 883, 1011, 1139, 1267, 1395, 1523, 1651, 1779, 1907, 2035, 2036, 2044] *chain length : 13*
logical qubit 23 : [721, 849, 954, 958, 966, 974, 977, 982, 990, 998, 1006, 1014, 1022, 1082, 1210, 1338, 1466, 1594, 1722, 1850] *chain length : 20*
logical qubit 24 : [754, 882, 1010, 1138, 1266, 1394, 1522, 1650, 1778, 1906, 2034, 2037, 2045] *chain length : 13*
logical qubit 25 : [707, 808, 815, 823, 831, 835, 839, 847, 855, 863, 871, 879, 887, 895, 936, 1064, 1192, 1320, 1448, 1576, 1704, 1832] *chain length : 22*
logical qubit 26 : [745, 873, 1001, 1129, 1257, 1385, 1510, 1513, 1518, 1526, 1534, 1641, 1769] *chain length : 13*
logical qubit 27 : [760, 888, 1016, 1144, 1272, 1400, 1528, 1656, 1784, 1912, 2040] *chain length : 11*
logical qubit 28 : [763, 891, 1019, 1147, 1275, 1403, 1531, 1652, 1659, 1660, 1787, 1915] *chain length : 12*
logical qubit 29 : [1612, 1617, 1620, 1628, 1745, 1824, 1831, 1833, 1839, 1847, 1855, 1863, 1871, 1873, 1879, 1887, 1895, 1897, 1899, 1903, 1911, 1919] *chain length : 22*
logical qubit 30 : [713, 841, 969, 1097, 1225, 1327, 1335, 1343, 1351, 1353, 1359, 1367, 1375, 1383, 1391, 1399, 1407, 1481, 1609, 1737] *chain length : 20*
logical qubit 31 : [746, 874, 1002, 1125, 1130, 1133, 1141, 1149, 1258, 1386, 1514, 1642, 1770, 1898] *chain length : 14*
logical qubit 32: [729, 857, 985, 1080, 1084, 1092, 1100, 1106, 1108, 1113, 1116, 1124, 1132, 1140, 1148, 1208, 1241, 1336, 1464, 1592, 1720, 1848] *chain length : 22*
logical qubit 33 : [736, 864, 992, 1120, 1248, 1252, 1260, 1268, 1276, 1376, 1504, 1632, 1636, 1760] *chain length : 14*
logical qubit 34 : [1363, 1449, 1453, 1461, 1469, 1477, 1485, 1491, 1493, 1501, 1509, 1517, 1525, 1533, 1577, 1705] *chain length : 16*
logical qubit 35 : [1703, 1711, 1719, 1727, 1735, 1736, 1743, 1751, 1759, 1767, 1775, 1783, 1791] *chain length : 13*
logical qubit 36 : [739, 867, 944, 949, 957, 965, 973, 978, 981, 989, 995, 997, 1005, 1013, 1021, 1072, 1123, 1200, 1251, 1328, 1456, 1584, 1712, 1840] *chain length : 24*
logical qubit 37 : [1507, 1635, 1709, 1717, 1725, 1733, 1741, 1749, 1757, 1763, 1765, 1773, 1781, 1789] *chain length : 14*
logical qubit 38 : [704, 832, 960, 1088, 1199, 1207, 1215, 1216, 1223, 1231, 1239, 1247, 1255, 1263, 1271, 1279, 1344, 1472, 1600, 1728, 1856] *chain length : 21*
logical qubit 39 : [1360, 1488, 1581, 1589, 1597, 1605, 1613, 1616, 1619, 1621, 1629, 1637, 1645, 1653, 1744, 1747, 1875, 1878, 1886, 1894, 1902, 1910, 1918] *chain length : 23*
logical qubit 40 : [1326, 1334, 1342, 1350, 1352, 1358, 1366, 1374, 1378, 1382, 1390, 1398, 1406, 1480, 1506, 1608, 1634, 1762] *chain length : 18*
logical qubit 41 : [1197, 1205, 1213, 1218, 1221, 1229, 1237, 1240, 1245, 1253, 1261, 1269, 1277, 1346, 1474, 1602, 1730, 1858] *chain length : 18*
logical qubit 42 : [1274, 1347, 1402, 1475, 1530, 1603, 1658, 1731, 1786, 1837, 1845, 1853, 1857, 1859, 1861, 1869, 1877, 1885, 1893, 1901, 1909, 1914, 1917] *chain length : 23*
logical qubit 43 : [761, 889, 1017, 1145, 1273, 1401, 1529, 1657, 1661, 1785, 1913, 2041] *chain length : 12*

(4)

The embedded problem is:

{(1752, 1758): 94.5, (1122, 1126): 80.5, (1392, 1398): -2446.0, (1008, 1015): -206.0, (1130, 1135): -63.3333, (1618, 1622): -12.0, (1387, 1389): -400.0, (1123, 1127): -293.5, (1393, 1396): 1762.0, (1843, 1845): -4096.0, (1755, 1757): -903.0, (832, 837): -1390.5, (1753, 1758): -12.0, (1129, 1135): -249.0, (875, 876): -748.0, (976, 982): -13.16667, (1771, 1773): -903.0, (1723, 1726): -206.0, (752, 756): 3614.0, (682, 686): -844.6667, (1355, 1357): -133.3333, (1586, 1591): 52.0, (1131, 1133): -63.3333, (1496, 1502): -76.0, (704, 708): -1390.5, (1402, 1404): -2048.0, (1473, 1476): -2.0, (883, 884): -1.0, (865, 870): 592.5, (1776, 1783): -184.0, (1225, 1230): -71.3333, (1627, 1631): 154.0, (1771, 1772): 4668.0, (1138, 1143): -283.0, (818, 823): -91.0, (1642, 1647): -190.0, (889, 894): -274.0, (1763, 1764): -903.0, (1355, 1359): -71.3333, (1626, 1631): -76.0, (1904, 1910): -2054.0, (1721, 1726): 15.0, (1905, 1909): -4096.0, (1473, 1479): 189.5, (705, 710): -2534.0, (1203, 1205): -2034.0, (835, 836): -91.0, (968, 974): -13.16667, (857, 861): -155.0, (1736, 1742): -92.0, (1785, 1788): -274.0, (1610, 1615): -400.0, (864, 869): -98.0, (1715, 1718): 38.0, (1147, 1151): -43.0, (728, 735): -76.0, (1746, 1751): -184.0, (752, 758): -2534.0, (683, 684): -844.6667, (1449, 1455): -63.66674, (1465, 1471): -3595.0, (1489, 1495): -1.333333, (1259, 1262): -275.5, (1520, 1527): -4522.0, (1371, 1373): -400.0, (1755, 1756): 1069.0, (1459, 1461): -191.0, (1011, 1012): -1.0, (1720, 1726): -155.0, (1714, 1718): 374.0, (1705, 1710): -95.5, (1704, 1710): -273.0, (1194, 1197): -2034.0, (1601, 1607): 94.5, (880, 885): -113.0, (1329, 1332): 592.5, (954, 956): -13.16667, (1761, 1766): -2.0, (1264, 1269): -2034.0, (753, 756): 3898.0, (1611, 1615): 328.0, (1905, 1908): -97.5, (1656, 1663): -95.0, (1003, 1005): -293.5, (1347, 1348): -2048.0, (1146, 1149): -63.3333, (1641, 1647): -498.0, (1754, 1758): -152.0, (865, 869): 1252.0, (1482, 1486): -133.3333, (1265, 1271): -2781.0, (747, 751): -152.0, (1771, 1899): -356.0, (888, 892): -95.0, (858, 860): -76.0, (1146, 1150): 50.3333, (1601, 1606): -12.0, (1779, 1782): -1.0, (1490, 1493): -63.66674, (867, 869): -293.5, (712, 718): -844.6667, (977, 980): -13.16667, (1371, 1375): -214.0, (1649, 1653): -2054.0, (1466, 1471): -79.0, (881, 885): -2819.0, (1275, 1278): -43.0, (1136, 1140): -155.0, (1490, 1492): -1.333333, (1784, 1790): -95.0, (1778, 1782): -283.0, (752, 757): -97.5, (1147, 1150): -43.0, (1264, 1268): -98.0, (1505, 1511): -1.333333, (1331, 1332): 1252.0, (1458, 1463): 41.0, (1593, 1598): -24.0, (1481, 1486): -71.3333, (1746, 1750): 189.5, (1776, 1782): 244.0, (1258, 1262): -95.0, (1491, 1495): -63.66674, (1393, 1398): -2446.0, (843, 845): -206.0, (1131, 1134): 50.3333, (1497, 1503): -12.0, (849, 852): -79.0, (1194, 1199): -1390.5, (753, 758): -2534.0, (1515, 1518): -249.0, (1467, 1471): -206.0, (1401, 1404): -274.0, (1119, 1127): 50.3333, (1459, 1463): -955.0, (1712, 1718): -587.0, (1473, 1477): -95.5, (1504, 1511): -98.0, (1226, 1230): -133.3333, (875, 879): -273.0, (856, 863): -91.0, (1617, 1623): -178.0, (1627, 1628): -178.0, (875, 877): -1597.0, (1122, 1127): -275.5, (1905, 1910): -2054.0, (1624, 1628): -178.0, (1578, 1581): -4108.0, (1696, 1700): 558.0, (947, 949): -293.5, (1713, 1716): -1024.0, (856, 984): 33.0, (1648, 1653): -2054.0, (705, 708): 2817.0, (1771, 1774): -1430.0, (1643, 1647): 1288.0, (1122, 1125): -95.0, (1729, 1735): -92.0, (1650, 1655): -283.0, (1883, 1887): -178.0, (1609, 1615): -214.0, (953, 958): -13.16667, (984, 990): -13.16667, (1243, 1246): 80.5, (848, 852): 33.0, (1203, 1207): -1390.5, (1458, 1460): -4.0, (873, 876): -498.0, (1331, 1334): -2446.0}

[1] Roth, Stefan, and Michael J. Black. *Fields of experts: A framework for learning image priors.* In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 2, pp. 860-867. IEEE, 2005.

[2] n Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 2, pp. 860-867. IEEE, 2005.
[3] Ishikawa, Hiroshi. *Transformation of general binary MRF minimization to the first-order case.* IEEE transactions on pattern analysis and machine intelligence 33.6 (2011): 1234-1249.
[4] Ishikawa, Hiroshi. *Higher-order clique reduction in binary graph cut*. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.