

WOMANIUM | QUANTUM

wiSER

WISER QUANTUM PROJECT 2: QUANTUM FOR PORTFOLIO OPTIMIZATION

Prepared by: Hao Mack Yang Li

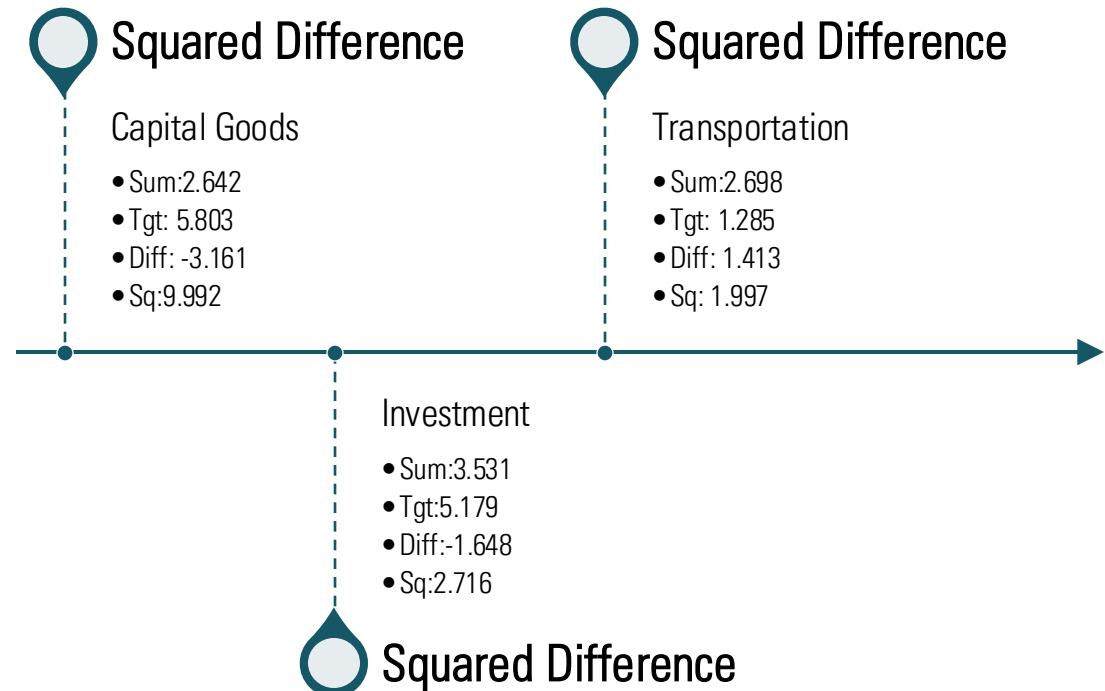
AGENDA

1. PROBLEM STATEMENT (0:10)
2. OUR SOLUTION (0:45)
3. RESULTS & IMPACT (1:40)
4. FUTURE SCOPE (2:40)



PROBLEM STATEMENT

- We are implementing the *simplified OneOpto* optimization model.
- The objective function is a binary quadratic model.
 - Entities include securities, set of risk buckets, set of characteristics, and a set of guardrails.
 - You are to minimize the sum of the squared difference between the target and the current portfolio contribution across all baskets and characteristics.

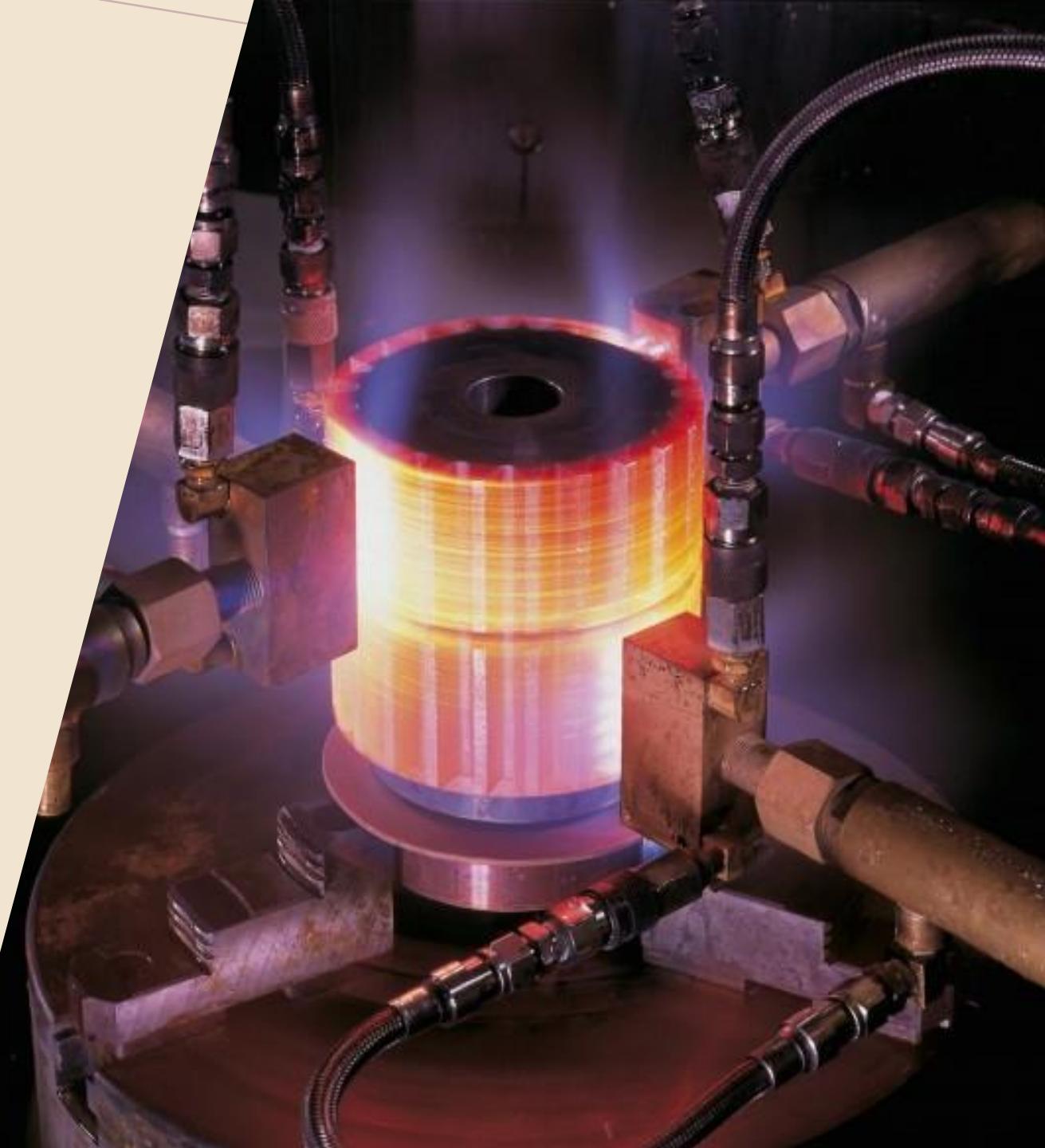


isin	ccy	assetId	krd1y	oac
1055BJ00	USD	001055BJ0	0.033056371152557	4.80459
1084AS13	USD	001084AS1	0.048588823367606	7.14734
108WAM29	USD	00108WAM2	0.020727166709757	5.19781
108WAP59	USD	00108WAP5	0.040987951949345	6.11096
108WAR16	USD	00108WAR1	0.045538368463281	6.64802
108WAT71	USD	00108WAT7	0.047172649143919	7.13612
115AAR05	USD	00115AAR0	0.044188432993148	7.35766
130HCG83	USD	00130HCG8	0.024547043236489	5.59428
206RGQ92	USD	00206RGQ9	0.037274651529775	4.59514
206RJY99	USD	00206RJY9	0.026847895926904	5.82149
206RKH48	USD	00206RKH4	0.022725597954055	6.49705
206RMM15	USD	00206RMM1	0.026742189942927	7.77357
206RMT67	USD	00206RMT6	0.044475996888843	7.17003
217GAB95	USD	00217GAB9	0.031740965502626	6.33841
2824BQ25	USD	002824BQ2	0.014141440456291	5.31202
287YBX67	USD	00287YBX6	0.029437115786959	4.50616
287YDT38	USD	00287YDT3	0.042026161568062	5.36954
287YDU01	USD	00287YDU0	0.042827438212052	7.34068
440FAA21	USD	00440FAA2	0.069281666971303	4.35851
440KAC71	USD	00440KAC7	0.038005490017179	5.89682
440KAD54	USD	00440KAD5	0.040186212153237	7.84911
510RAD52	USD	00510RAD5	0.021561454838955	5.57076
724PAD15	USD	00724PAD1	0.021774314167194	4.82374
724PAG46	USD	00724PAG4	0.042603342186904	7.42196
7589AD66	USD	007589AD6	0.021723106291976	5.13961
774MAX39	USD	00774MAX3	0.031546980528688	6.24501
774MAY12	USD	00774MAY1	0.033705815187094	7.39746
774MBE49	USD	00774MBE4	0.050479148292221	4.89841
774MBH79	USD	00774MBH7	0.044679594035597	7.10586
774MBK09	USD	00774MBK0	0.057178420415543	4.24982
774MBM64	USD	00774MBM6	0.043469427149448	7.62829
7903BF39	USD	007903BF3	0.035519519576115	6.32303
7944AH47	USD	007944AH4	0.046454659294363	5.91679
7944AK75	USD	007944AK7	0.048165952205359	7.35274
8252AP33	USD	008252AP3	0.031082400946197	4.97131
8252AR98	USD	008252AR9	0.046842924387641	7.41820
846UAM36	USD	00846UAM3	0.020873647356666	5.12527
846UAN19	USD	00846UAN1	0.022869837084699	5.77288
846UAR23	USD	00846UAR2	0.041781376205977	7.69947
8513AA19	USD	008513AA1	0.028263543016454	5.31682
8513AC74	USD	008513AC7	0.027409420598708	7.41484
8513AD57	USD	008513AD5	0.042727195598018	6.45272

WHY DOES IT MATTER

- Numerous trades occur during the day at very high rates, think about high frequency trading.
- The optimization problem becomes **computationally infeasible** classically.
 - The brute force approach is exponential to the number of securities.
 - Local optima and multiple optima can cause optimizers to fail.
 - Even GUROBI may fail to find a global optima within ten minutes, especially with hundreds of risk buckets and tens of characteristics.
- The optimization problem is of a **matrix** form, and is well suited for quantum algorithms, such as QAOA.

A QUANTUM ANNEALING APPROACH



THE VARIABLES

INPUT VARIABLES

- Let C denote the set of securities.
 - Each element c is internally indexed.
 - Each element is expressed as a quintuple:
 - Market price p , Trade range (min, max, inventory), Minimum increment δ
- Let L denote the set of risk buckets
 - Can be a rating or an industry.
- Let J denote the set of characteristics
 - Can be any numeric field in the dataframe.

CONSTRAINT VARIABLES

- There is a guardrail and target K^{target} , K^{low} , K^{up} for each l in L and j in J .
 - There are thus $|L||J|$ such targets, guardrails, and quadratic terms in the objective.
- Let $\mathbf{K}[l]$ denote the subset of bonds belonging to risk bucket l .
- Total bonds N , residual cash guardrail R .
- Output is the \mathbf{y} , the included bonds.

THE EQUATION

REDUCED DATASET

- $|C| = 31$
- $L = \{\text{'Capital Goods'}, \text{'Investment'}, \text{'Transportation'}\}$
- $J = \{\text{'PMV'}\}$
- $|K| = 3$
 - $K_{\text{Capital Goods, PMV}} = [4.758, 5.803, 6.847]$
 - $K_{\text{Investment, PMV}} = [4.134, 5.178, 6.223]$
 - $K_{\text{Transportation, PMV}} = [0.240, 1.284, 2.329]$

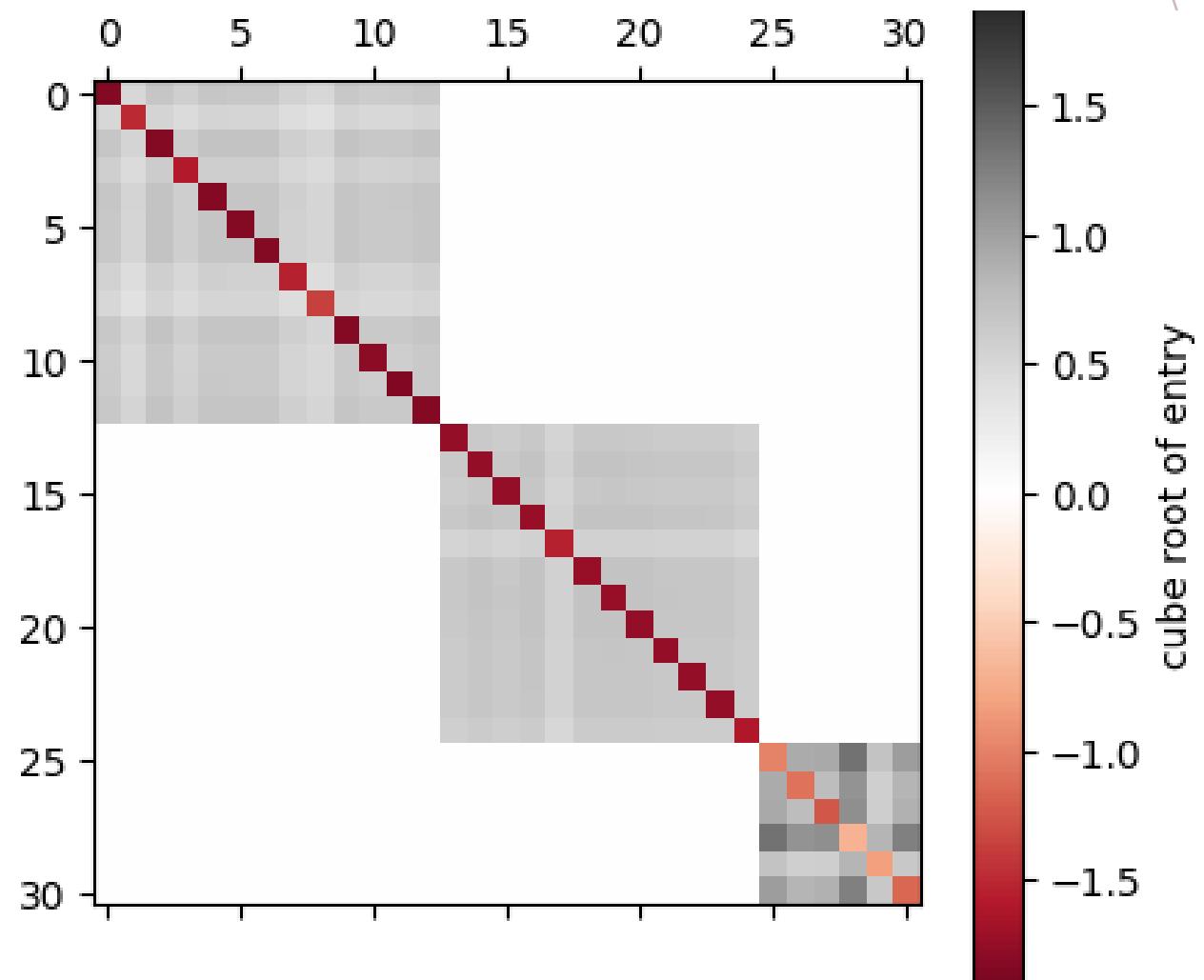
OPTIMIZATION AND CONSTRAINTS

- x_c is a fixed multiple of y_c depending on the minimum and maximum trade amounts.
- ϱ_j and $\beta_{j,c}$ are contribution weights.
- $\min_y \sum_l \sum_j \varrho_j (\sum_{c \in K[l]} \beta_{j,c} x_c - K^{\text{target}})^2$
- $\sum_c y_c \leq N, \sum_c p_c \delta_c \beta_{j,c} x_c \in R$
- $\forall j \forall l \sum_{c \in K[l]} p_c \delta_c \beta_{j,c} x_c \in K_{l,j}$

OUR D-WAVE SOLUTION

We implemented our solution in Python, using the *dwave-system* package.

- The optimization equation is converted to expanded form, and then to the format compatible with D-WAVE.
- Each element of \mathbf{y} is named after the bond.
 - Sorted by the risk bucket, so block-diagonal if the buckets do not overlap.
- To the right is the QAOA matrix.
 - Red entries negative, gray entries positive.
 - The off-diagonal entries have much lower magnitudes (order of 0.2 to 1.5) against the diagonal entries (order of -5).



OUR RESULTS

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** D-WAVE Experiment
- Description:** We use the D-WAVE sampler to perform quantum annealing. The result will be exported as a CSV file.
- Code:**

```
from dwave.samplers import PathIntegralAnnealingSampler
sampler = PathIntegralAnnealingSampler()
sampleset = sampler.sample(bqm, num_reads=READS)
df_s = sampleset.to_pandas_dataframe().sort_values(by="energy")
df_s.to_csv("dwave_result.csv")
df_s
```
- Execution:** [9] 2m 31.4s
- Output:** A table showing the results of the quantum annealing. The columns are labeled with problem instances: 020002BJ9, 026874DS3, 081437AT2, 097023CJ2, 13645RAD6, 13645RBF0, 14448CBC7, 15135BAW1, ..., 46080, 46079, 38893, 38825, 48912, ..., 40283, 49672, 58158. The values represent the state of each qubit (0 or 1).
- Log:**

```
Solution #1 (1792): ['438516CM6', '45687VAB2', '75513EAD3', '760759BC3', '020002BJ9', '21871XAS8', '444859BR2', '444859BV3', '444859CA8', '655844CR7']  
17.500 0.034 0.594 ('14448CBC7', 5.0, 30.0, 30.0, 0.033939846355, 22.473301347308, 1, 'Cap...  
17.500 0.033 0.583 ('21871XAS8', 5.0, 30.0, 30.0, 0.033333333333, 26.975300311131, 1, 'In...  
17.500 0.033 0.571 ('24422EXP9', 5.0, 30.0, 30.0, 0.032622048628, 16.111012630144, 1, 'Cap...  
17.500 0.032 0.557 ('36166NAK9', 5.0, 30.0, 30.0, 0.031825968182, 14.941828851305, 1, 'Cap...  
17.500 0.032 0.554 ('438516CM6', 5.0, 30.0, 30.0, 0.031667951458, 13.522903255546, 1, 'Cap...  
17.500 0.033 0.578 ('444859BV3', 5.0, 30.0, 30.0, 0.033043461247, 27.751655496693, 1, 'In...  
17.500 0.033 0.580 ('444859BY7', 5.0, 30.0, 30.0, 0.033169141002, 30.60969003408, 1, 'Insu...  
17.500 0.032 0.563 ('444859CA8', 5.0, 30.0, 30.0, 0.032189269112, 19.3556173304, 1, 'Insu...  
17.500 0.032 0.561 ('539830CD9', 5.0, 30.0, 30.0, 0.032078431567, 15.855019693051, 1, 'Cap...  
17.500 0.032 0.561 ('760759BC3', 5.0, 30.0, 30.0, 0.032050068416, 18.718284601554, 1, 'Cap...  
Energy: 23.153  
Bonds: 10  
Flow: 5.704  
17.500 0.032 0.554 ('438516CM6', 5.0, 30.0, 30.0, 0.031667951458, 13.522903255546  
7.000 0.034 0.236 ('45687VAB2', 5.0, 9.0, 9.0, 0.033698227268, 20.092890519524, 1  
16.000 0.028 0.452 ('75513EAD3', 5.0, 27.0, 27.0, 0.028236246188, 9.403495008104,  
17.500 0.032 0.561 ('760759BC3', 5.0, 30.0, 30.0, 0.032050068416, 18.718284601554  
Bonds[I_Capital_Goods][fund_enriched.pmv]: Σ=1.803 Δ=-4.001 Δ²=16.005 (3.087, 5.803, 8  
17.500 0.026 0.461 ('020002BJ9', 5.0, 30.0, 30.0, 0.02633418857, 11.747405182296,  
17.500 0.033 0.583 ('21871XAS8', 5.0, 30.0, 30.0, 0.033333333333, 26.975300311131  
11.500 0.026 0.300 ('444859BR2', 5.0, 18.0, 18.0, 0.026085891521, 19.713755048306  
17.500 0.033 0.578 ('444859BV3', 5.0, 30.0, 30.0, 0.033043461247, 27.751655496693  
17.500 0.032 0.563 ('444859CA8', 5.0, 30.0, 30.0, 0.032189269112, 19.3556173304, 1  
Bonds[I_Insurance][fund_enriched.pmv]: Σ=2.486 Δ=-2.693 Δ²=7.253 (2.463, 5.179, 7.895)
```

SUMMARY

We are able to successfully implement the minimization of the objective function while enforcing the guardrail conditions.

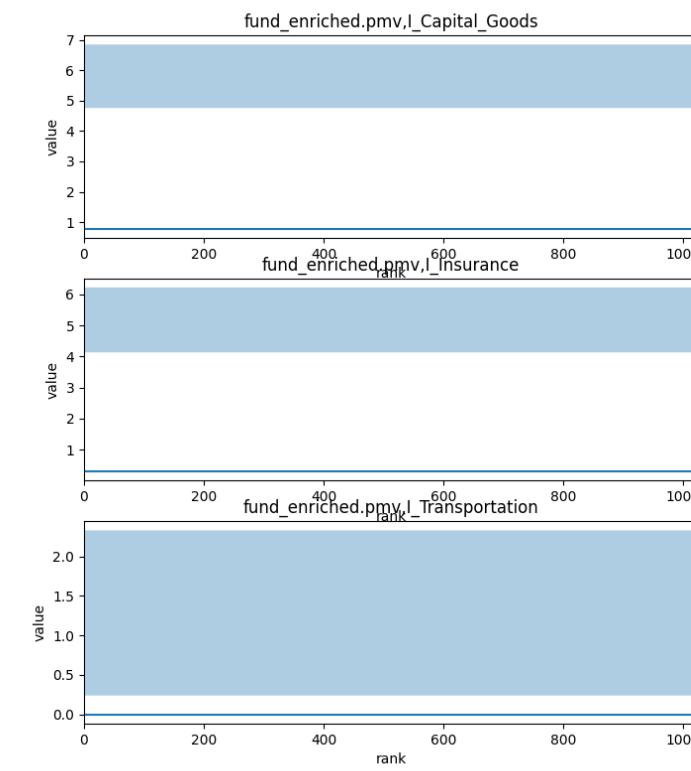
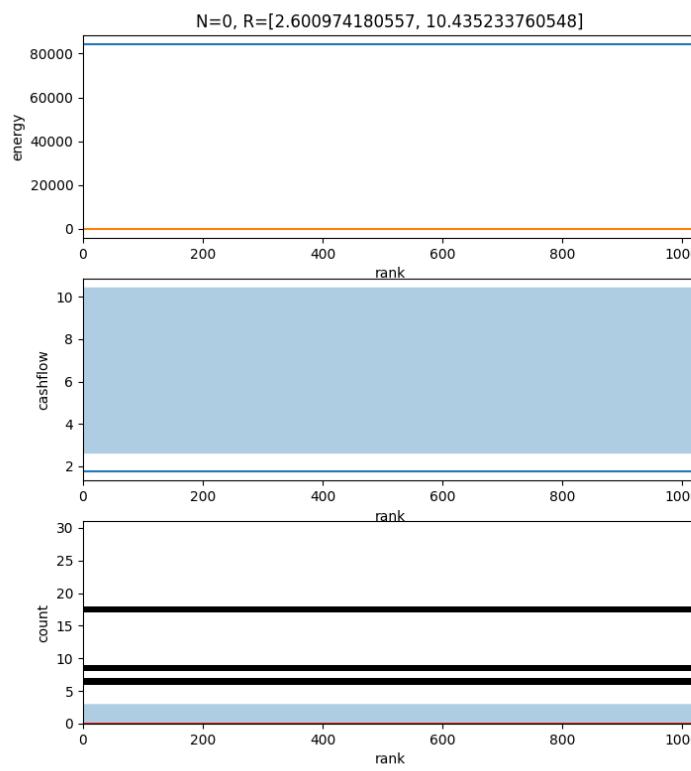
- Unfortunately, we are not able to gain access to the actual quantum resource via LEAP.
- The simulated annealer can provide a rapidly convergent solution. Even with 65536 runs, it only takes three minutes on the 31-bond dataset and three guardrails.

The next slide will show the animation of the samples, with the best one by energy listed *first*.

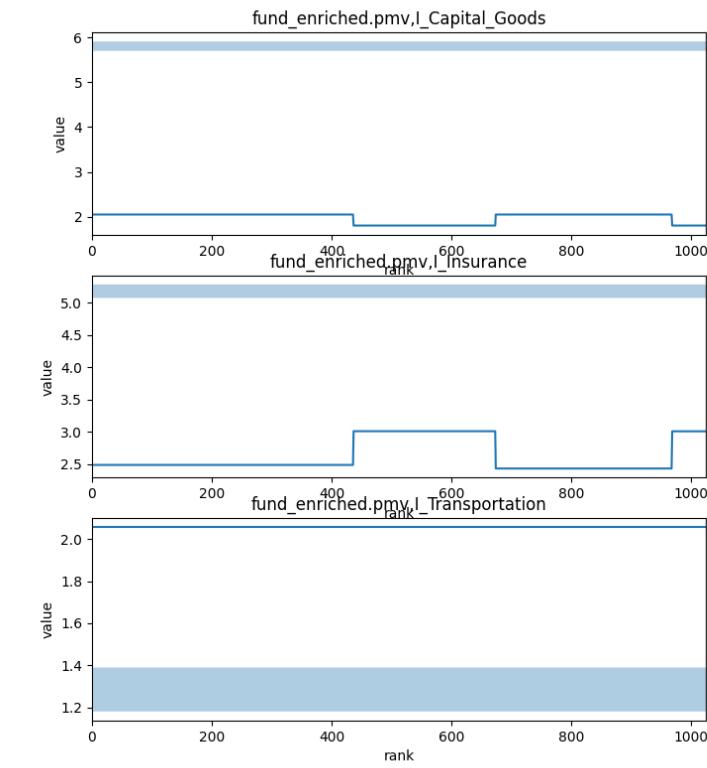
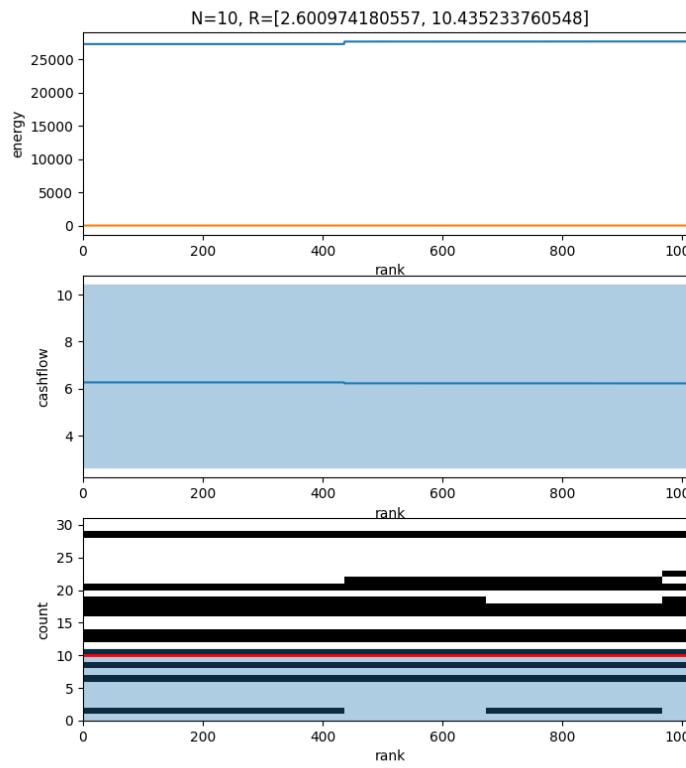
- D-WAVE energy, QAOA energy
- Cashflow
- Bond selection (by internal index) overlaid with total bonds
- Values and guardrail (assume target is midpoint)

```
Solution #1 {-48}: ['081437AT2', '097023CJ2', '14448CBC7', '24422EWZ8', '438516CM6', '443201AC2', '45687VAB2', '539830CD9', '75513EAD3', '760759BC3', '026874DS3', '151358AW1', '21871XAS8', '444859BR2', '444859BV3', '444859BY7', '444859CA8', '56501RAN6', '91324PEJ7', '13645RBF0', '314353AA1'] 17.500 0.026 0.461 ('026874DS3', 5.0, 5.0, 30.0, 30.0, 0.02633418597, 11.747405182296, 1, 'Insurance', [1]) 17.500 0.032 0.563 ('026874DS3', 5.0, 30.0, 30.0, 0.028510851146, 13.316192340935, 1, 'Capital_Goods', [1]) 9.000 0.027 0.246 ('097023CJ2', 5.0, 13.0, 13.0, 0.027298851045, 34.461503917596, 1, 'Capital_Goods', [1]) 17.500 0.034 0.594 ('14448BC7', 5.0, 30.0, 30.0, 0.033939846355, 22.473301347308, 1, 'Capital_Goods', [1]) 17.500 0.028 0.487 ('151358AW1', 5.0, 30.0, 30.0, 0.027845453268, 21.626486622322, 1, 'Insurance', [1]) 17.500 0.033 0.583 ('21871XAS8', 5.0, 30.0, 30.0, 0.033333333333, 26.975300311131, 1, 'Insurance', [1]) 11.000 0.033 0.360 ('24422EWZ8', 5.0, 17.0, 17.0, 0.032714779285, 7.4190547577, 1, 'Capital_Goods', [1]) 17.500 0.033 0.571 ('24422EXP9', 5.0, 30.0, 30.0, 0.032622048628, 16.111012630144, 1, 'Capital_Goods', [1]) 17.500 0.032 0.557 ('36166NAK9', 5.0, 30.0, 30.0, 0.031825968182, 14.941828851305, 1, 'Capital_Goods', [1]) 17.500 0.032 0.554 ('438516CM6', 5.0, 30.0, 30.0, 0.0316677951458, 13.522903255546, 1, 'Capital_Goods', [1]) 10.000 0.032 0.319 ('443201AC2', 5.0, 15.0, 15.0, 0.031871754859, 14.15453485858, 1, 'Capital_Goods', [1]) 11.500 0.026 0.300 ('444859BR2', 5.0, 18.0, 18.0, 0.026085891521, 19.713755048306, 1, 'Insurance', [1]) 17.500 0.033 0.578 ('444859BV3', 5.0, 30.0, 30.0, 0.033043461247, 27.75165549693, 1, 'Insurance', [1]) 17.500 0.033 0.580 ('444859BY7', 5.0, 30.0, 30.0, 0.033169141002, 30.60969003408, 1, 'Insurance', [1]) 17.500 0.032 0.563 ('444859CA8', 5.0, 30.0, 0.033698227268, 20.092890519524, 1, 'Capital_Goods', [1]) 17.500 0.032 0.561 ('539830CD9', 5.0, 30.0, 30.0, 0.032078431567, 15.855019693051, 1, 'Capital_Goods', [1]) 17.500 0.030 0.522 ('56501RAN6', 5.0, 30.0, 30.0, 0.029814829851, 13.086051978345, 1, 'Insurance', [1]) 8.500 0.034 0.285 ('655844CT3', 5.0, 12.0, 12.0, 0.033512025872, 0.033512025872, 1, 'Transportation', [1]) 17.500 0.030 0.517 ('759351AP4', 5.0, 30.0, 30.0, 0.029540913811, 11.974870385817, 1, 'Insurance', [1]) 17.500 0.026 0.462 ('760759BA7', 5.0, 30.0, 30.0, 0.026425255721, 15.38801820894, 1, 'Capital_Goods', [1]) Energy: -1.228 Bonds: 22 Flow: 10.399 17.500 0.029 0.499 ('081437AT2', 5.0, 30.0, 30.0, 0.028510851146, 13.316192340035, 1, 'Capital_Goods', [1]) 9.000 0.027 0.246 ('097023CJ2', 5.0, 30.0, 13.0, 0.027298851645, 34.461503917596, 1, 'Capital_Goods', [1]) 17.500 0.032 0.533 0.594 ('14448BC7', 5.0, 30.0, 30.0, 0.033939846355, 22.473301347308, 1, 'Capital_Goods', [1]) 11.000 0.033 0.360 ('24422EWZ8', 5.0, 17.0, 17.0, 0.032714779285, 7.4190547577, 1, 'Capital_Goods', [1]) 17.500 0.032 0.532 0.554 ('36166NAK9', 5.0, 30.0, 30.0, 0.0316677951458, 13.522903255546, 1, 'Capital_Goods', [1]) 10.000 0.032 0.302 0.339 ('438516CM6', 5.0, 15.0, 15.0, 0.031871754859, 14.15453485858, 1, 'Capital_Goods', [1]) 7.000 0.034 0.304 0.339 ('443201AC2', 5.0, 15.0, 15.0, 0.031871754859, 14.15453485858, 1, 'Capital_Goods', [1]) 17.500 0.032 0.532 0.561 ('444859BV3', 5.0, 30.0, 30.0, 0.033698227268, 20.092890519524, 1, 'Capital_Goods', [1]) 16.000 0.028 0.045 0.452 ('539830CD9', 5.0, 30.0, 30.0, 0.032078431567, 15.855019693051, 1, 'Capital_Goods', [1]) 17.500 0.032 0.532 0.561 ('760759BC3', 5.0, 30.0, 30.0, 0.032050668416, 18.718284601554, 1, 'Capital_Goods', [1]) 17.500 0.032 0.532 0.561 ('760759BA7', 5.0, 30.0, 30.0, 0.032050668416, 18.718284601554, 1, 'Capital_Goods', [1]) Bonds[I_Capital_Goods][fund_enriched.pmv]: ε=4.381 Δ=-1.422 Δ²=2.022 (4.759, 5.803, 6.848) 17.500 0.026 0.461 ('026874DS3', 5.0, 30.0, 0.0333418597, 11.747405182296, 1, 'Insurance', [1]) 17.500 0.032 0.563 0.461 ('026874DS3', 5.0, 30.0, 0.032168634542, 19.30639960602, 1, 'Insurance', [1]) 17.500 0.028 0.487 ('151358AW1', 5.0, 30.0, 30.0, 0.027845453268, 21.626486622322, 1, 'Insurance', [1]) 17.500 0.033 0.583 ('21871XAS8', 5.0, 30.0, 30.0, 0.033333333333, 26.975300311131, 1, 'Insurance', [1]) 11.500 0.026 0.390 ('444859BR2', 5.0, 18.0, 18.0, 0.026085891521, 19.713755048306, 1, 'Insurance', [1]) 17.500 0.033 0.578 ('444859BV3', 5.0, 30.0, 30.0, 0.032034361247, 27.75165549693, 1, 'Insurance', [1]) 17.500 0.033 0.580 ('444859BY7', 5.0, 30.0, 30.0, 0.033169141002, 30.60969003408, 1, 'Insurance', [1]) 17.500 0.032 0.563 ('444859CA8', 5.0, 30.0, 30.0, 0.032189269112, 19.3556173304, 1, 'Insurance', [1]) 17.500 0.030 0.522 ('56501RAN6', 5.0, 30.0, 30.0, 0.029814829851, 13.086051978345, 1, 'Insurance', [1]) 13.000 0.031 0.408 ('91324PEJ7', 5.0, 21.0, 21.0, 0.031401384436, 11.756014097259, 1, 'Insurance', [1]) Bonds[I_Insurance][fund_enriched.pmv]: ε=5.046 Δ=-0.133 Δ²=0.018 (4.134, 5.179, 6.224) 23.000 0.028 0.641 ('13645RBF0', 5.0, 41.0, 41.0, 0.027884438059, 0.027884438059, 1, 'Transportation', [1]) 32.500 0.021 0.695 ('314353AA1', 5.0, 60.0, 60.0, 0.021396294681, 0.021396294681, 1, 'Transportation', [1]) Bonds[I_Transportation][fund_enriched.pmv]: ε=1.337 Δ=0.052 Δ²=0.003 (0.240, 1.285, 2.329)
```

CHANGING BOND LIMIT (30M)

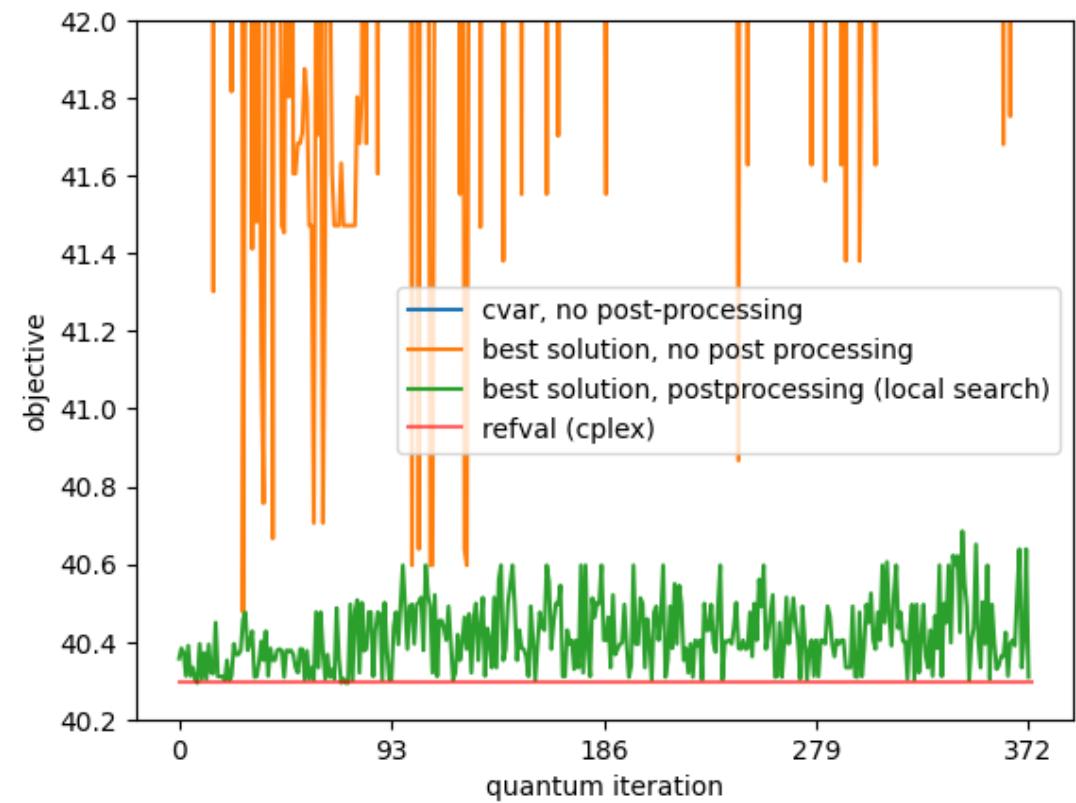


CHANGING GUARDRAILS (2H)



SCALABILITY AND CLASSICAL

- The reference run uses a real Qiskit run with or without postprocessing.
- In the small dataset, the reference optimal value has an energy of 40.3.
 - The actual Qiskit quantum run without postprocessing considerably misses the goal.
 - With postprocessing, it misses the goal by 0.15 units on average.
- D-WAVE energy is separate, but correlated with QAOA energy; D-WAVE energy factors in guardrails as well.



```
implementation.py X
implementation.py > SimplifiedOneOpto
  65  class SimplifiedOneOpto():
  66      def __init__(self,
124          self.matrix = matrix
125          self.constantterm = constantterm
126          self.results = None
127
128      def run(self, num_reads):
129          # Convert to binary quadratic model
130          bqm = BinaryQuadraticModel(vartype=dimod.BINARY)
131          for i in range(len(C)):
132              bqm.add_variable(C[i][CID])
133              bqm.add_linear(C[i][CID], self.matrix[i][i])
134          for i1 in range(len(C)):
135              for i2 in range(i1 + 1, len(C)):
136                  if self.matrix[i1][i2]:
137                      bqm.add_quadratic(C[i1][CID], C[i2][CID], 2 * self.matrix[i1][i2])
138
139          # Add master linear inequality constraints
140          sy = []
141          sx = []
142          for i in range(len(self.C)):
143              sy.append((self.C[i][CID], 1))
144              sx.append((self.C[i][CID], self.C[i][PMV] * self.C[i][DELTA] * x(self.C[i])))
145
146          lm = np.sum(abs(self.matrix), axis=1) # Hard restraint
147          add_linear_inequality_constraint(sy, ub = self.N, lagrange_multiplier=np.dot(lm, self.R[0]), constraint_type='leq')
148          add_linear_inequality_constraint(sx, lb = self.R[0], ub = self.R[1], lagrange_multiplier=1)
149
150          # Set and characteristic based inequality constraints
151          for j in range(len(self.J)):
152              for l in range(len(self.L)):
153                  if C[1][j][l] == 0:
154
155                      # Set based inequality constraints
156                      if C[1][j][l] == 0:
157                          syl = [(j, 1)]
158                          sxl = [(l, 1)]
159
160                          add_linear_inequality_constraint(syl, lb = -1, constraint_type='leq')
161                          add_linear_inequality_constraint(sxl, lb = self.R[0], constraint_type='leq')
162
163                          # Characteristic based inequality constraints
164                          if C[1][j][l] == 0:
165                              sampler = SimulatedAnnealingSampler()
166                              sampler()
167
168                              df = pd.DataFrame()
169                              df['energy'] = sampler.sample_qubo(bqm, num_reads=num_reads)
170
171                              df = df.groupby('energy').size().reset_index()
172                              df = df.sort_values(by="energy")
173
174                              sv = df['energy'].value_counts()
175
176                              if sv.max() / sv.sum() < 0.01:
177
178                                  if C[1][j][l] == 0:
179                                      if C[1][j][l] == 0:
180
181                                          if C[1][j][l] == 0:
182
183                                              if C[1][j][l] == 0:
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
```

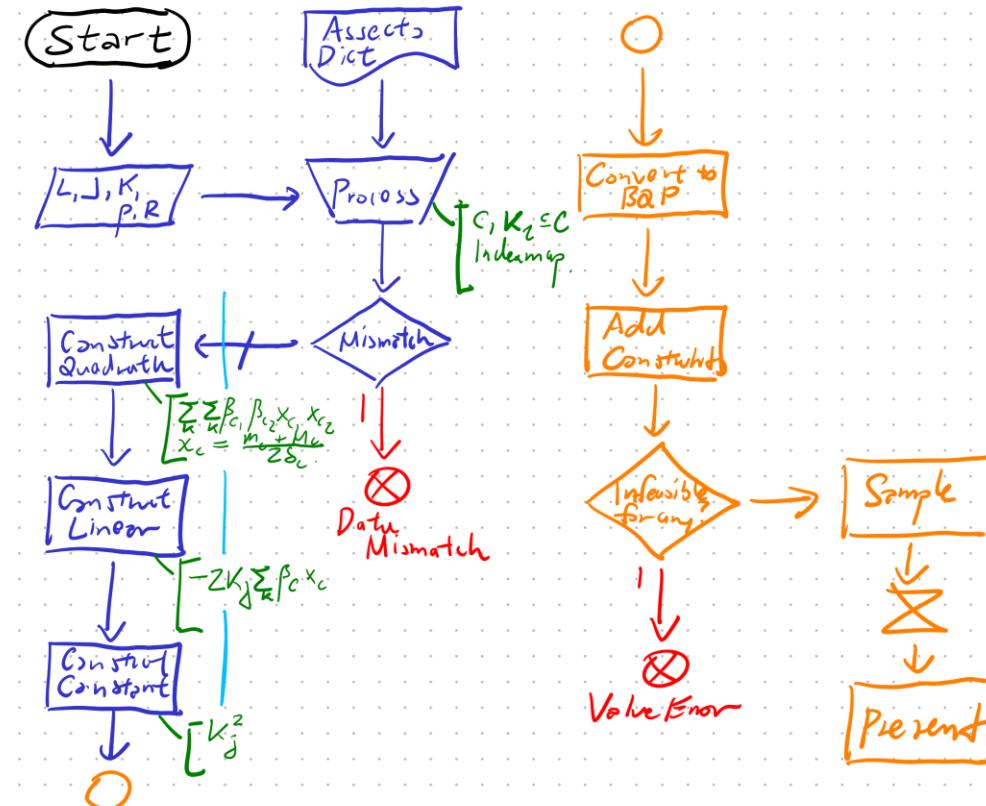
FUTURE ROADMAP

- Software engineering
- Scalability
- Different datasets
- D-WAVE

CODE MAINTENANCE

Much time spent on coding instead visits the theory by providing test examples to affirm or refute the correctness of the program, and to refactor code to adhere to software engineering principles.

- Object-oriented programming
 - A dedicated C and K object.
 - Architecture to repeatedly run the procedure while varying some of the values.
- Exception handling
 - Prevents harder to trace exceptions.
- Code reusability
 - Allows easy inspection, avoids code smells.

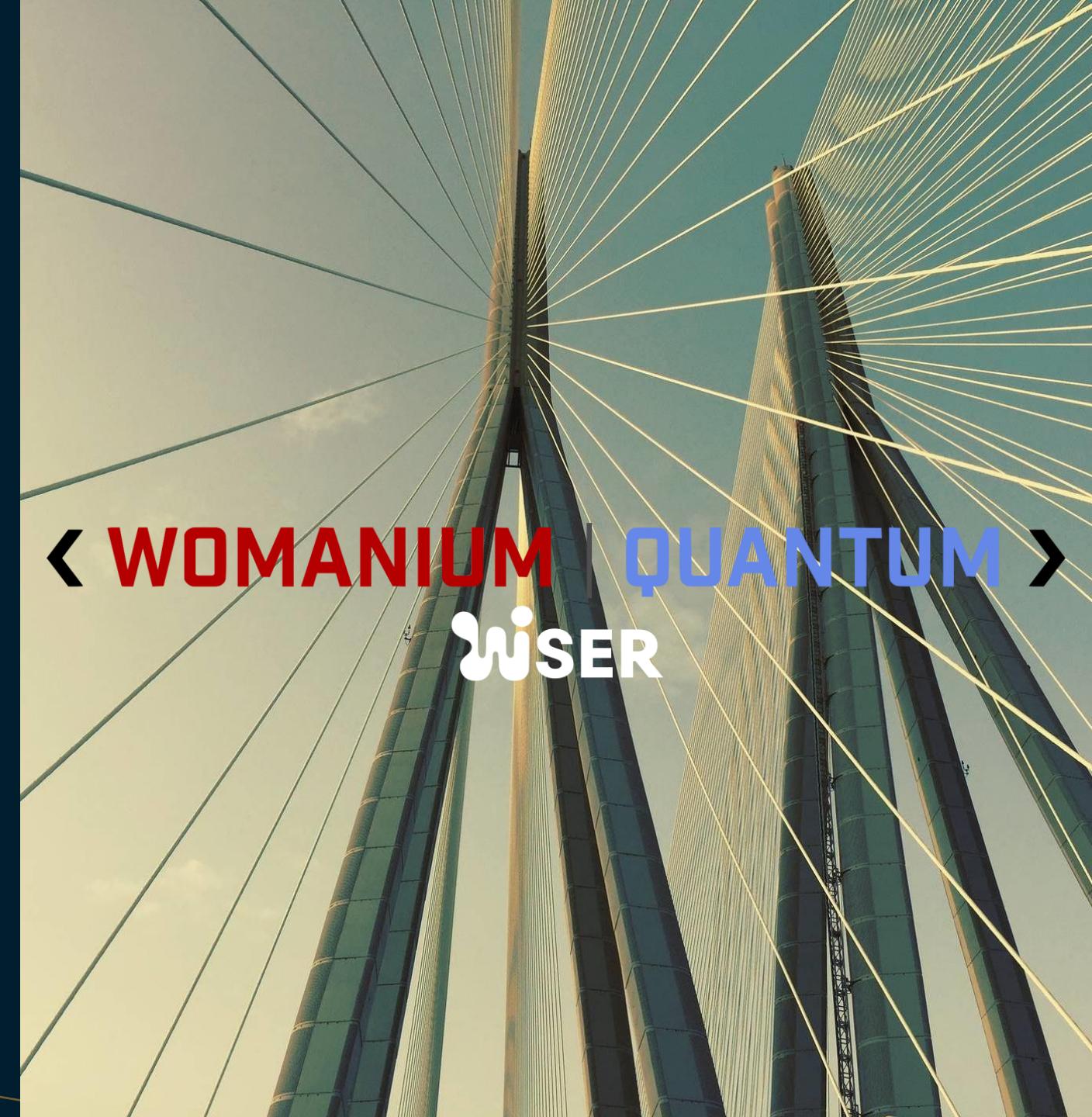


LIMITATIONS

- To conclude this presentation, we will address the limitations encountered during the submission period.
 - We did not get our LEAP application approved within the timeframe, so we cannot measure our progress using a real quantum annealer, which can support up to thousands of variables.
 - We mostly discussed the smallest dataset.
 - Though software engineering best practices have been follows and we are able to see results, we need to scrutinize our code again to resolve any anomalies, such as
 - Negative values of the optimization function.
 - The constant bias between different quantum and classical optimization process.
 - Potentially confusing the variables and guardrails they officially provided versus my implementation.
 - It is important to time manage, and not crunch on time.

THANK YOU

COMMENT BELOW FOR ANY QUESTIONS!



< **WOMANIUM** | **QUANTUM** >
WiSER