RecSys 2025
Prague

A Hands-on Dive Into Quantum Computing for Recommender Systems

Part 2: QUBO Formulation and Quantum Annealer Architecture

POLITECNICO
MILANO 1863

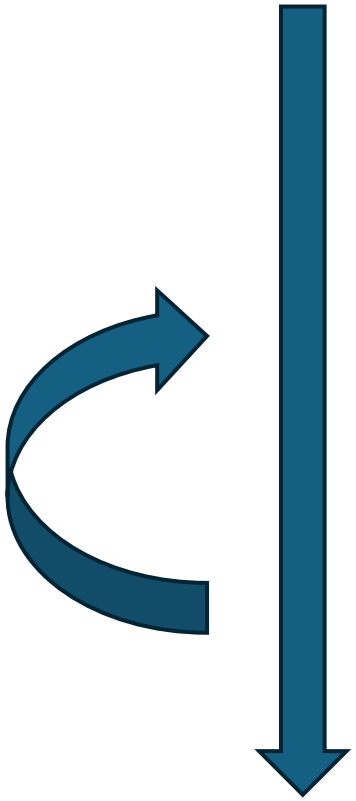
Maurizio Ferrari Dacrema, Politecnico di Milano

@Maurizio_fd

Outline

- **QUBO Formulation for optimization problems**
- **QUBO Formulation of feature selection and clustering**
- **Quantum Annealer architecture and minor-embedding**

Using a Quantum Annealer in practice



1. Formulate the problem as QUBO
2. Compile it (minor-embedding)
3. Run the annealing process
4. Measure the final state of the qubits
5. Repeat as needed (sampling of solutions)
6. Apply postprocessing on a traditional computer if needed

QUBO Formulation

QUBO Formulation

- **Quadratic:** The problem must be at most quadratic
- **Unconstrained:** No hard constraints (soft constraints with a penalty term)
- **Binary:** Variables are binary (real and categorical require a workaround)
- **Optimization:** Represents optimization problems (NP-Complete, NP-Hard)

QUBO Formulation

$$\min_x y = x^T Q x$$

$$y = \sum_i q_{i,i} x_i + \sum_{i>j} q_{i,j}\, x_i x_j$$

Linear
diagonal of Q

Quadratic
off-diagonal of Q

Where:

- n is the problem size
- $x \in \{0,1\}^n$ are the binary problem variables
- $Q \in R^{n \times n}$ is the matrix of coefficients (upper triangular or symmetric)

Notice that $x_i^2 = x_i$

QUBO Formulation

- Advantages

  The formulation is simple and very general, so it can be used for NP-complete and many NP-hard optimization problems.

  An effective QUBO solver can impact many tasks.

- Disadvantages

  The formulation hides the specific structure of the problem, which is usually what one leverages to develop highly effective solvers for particular classes of problems or structures.

Simple QUBO example $a \neq b$

Suppose we want to find the value of two variables a, b such that the Boolean function $a \neq b$ is True.

a	b	$a \neq b$
0	0	0
0	1	1
1	0	1
1	1	0

Simple QUBO example $a \neq b$

First, we need to transform this into a minimization problem. We define our cost *y* so that the lowest value, 0, corresponds to the variable assignments that satisfy the Boolean function. We set 1 to all the others.

a	b	$a \neq b$	**y**
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Simple QUBO example $a \neq b$

Then, we need to describe y as a function $y(a, b)$ of our problem variables a, b. In this example we can proceed iteratively starting from the simplest case.

a	b	$a \neq b$	**y**
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

$y(a, b) = ?$

Simple QUBO example $a \neq b$

Then, we need to describe y as a function $y(a, b)$ of our problem variables a, b. In this example we can proceed iteratively starting from the simplest case.

a	b	$a \neq b$	**y**
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

$y(a, b) = ?$

1

Simple QUBO example $a \neq b$

Then, we need to describe y as a function $y(a, b)$ of our problem variables a, b. In this example we can proceed iteratively starting from the simplest case.

a	b	$a \neq b$	y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

$y(a, b) = ?$

1

$1 - \mathrm{a} - \mathrm{b}$

Simple QUBO example $a \neq b$

Then, we need to describe y as a function $y(a, b)$ of our problem variables a, b. In this example we can proceed iteratively starting from the simplest case.

a	b	$a \neq b$	y	$y(a,b) = ?$
0	0	0	1	1
0	1	1	0	$1 - \mathrm{a} - \mathrm{b}$
1	0	1	0	
1	1	0	1	$1 - \mathrm{a} - \mathrm{b} + 2\mathrm{ab}$

The final formulation is: $y(a, b) = 1 - \mathrm{a} - \mathrm{b} + 2\mathrm{ab}$

Simple QUBO example $a == b$

This is another example in which we want to find the values of a, b such that $a == b$. Again, we represent it as a minimization problem:

a	b	$a == b$
0	0	1
0	1	0
1	0	0
1	1	1

Simple QUBO example $a == b$

This is another example in which we want to find the values of a, b such that $a == b$. Again, we represent it as a minimization problem:

a	b	$a == b$	y
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0

Simple QUBO example $a == b$

This is another example in which we want to find the values of a, b such that $a == b$. Again, we represent it as a minimization problem:

a	b	$a == b$	y	$y(a,b) = ?$
0	0	1	0	0
0	1	0	1	$\mathrm{a} + \mathrm{b}$
1	0	0	1	
1	1	1	0	$\mathrm{a} + \mathrm{b} - 2\mathrm{ab}$

The final formulation is: $y(a,b) = \mathrm{a} + \mathrm{b} - 2\mathrm{ab}$

What if the problem is not quadratic?

- Sometimes you can reformulate the problem in an equivalent quadratic formulation or approximate the non-quadratic functions into quadratic ones. Beware of introducing too many approximations.

- If the problem is binary but with higher degree polynomials, it is easily possible to convert it into one that is quadratic:

Term	QUBO
$x_1 x_2 x_3$	$x_1 + x_2 + x_3 - 2$
x_1^n, n>2	x_1

What if the variables are not Boolean?

Categorical variables: If your problem has variables which are defined on a set of values, say $x \in \{A, B, C, D, E\}$, you can use one hot-encoded binary variable per value. A constraint is needed to ensure exactly one is selected.

Continuous variables: Can be approximated with binary expansion:

$$r_i = \sum_{e=-2}^{+2} 2^e x_i = \frac{1}{4}x_1 + \frac{1}{2}x_2 + x_3 + 2x_4 + 4x_5$$

How to add constraints in QUBO

QUBO is a formulation for “unconstrained” problems, meaning we cannot set a hard constraint on the variables.

Workaround: introduce a penalty that will be a positive number for the states that violate the constraints and zero otherwise. Note that the solver might still find unfeasible solutions, you should always check their feasibility.

$$\min_x y = xQx^T + \gamma \cdot \text{penalty}(x)$$

Common penalty terms

Constraint	Penalty
$x_1 + x_2 \leq 1$	$x_1 x_2$
$x_1 + x_2 \geq 1$	$1 - x_1 - x_2 + x_1 x_2$
$x_1 + x_2 = 1$	$1 - x_1 - x_2 + 2x_1 x_2$
$x_1 \leq x_2$	$x_1 - x_1 x_2$
$x_1 = x_2$	$x_1 + x_2 - 2x_1 x_2$
$x_1 + x_2 + x_3 \leq 1$	$x_1 x_2 + x_1 x_3 + x_2 x_3$

Common penalty terms

Constraint	Penalty
$x_1 + x_2 \leq 1$	$x_1 x_2$
$x_1 + x_2 \geq 1$	$1 - x_1 - x_2 + x_1 x_2$
$x_1 + x_2 = 1$	$1 - x_1 - x_2 + 2x_1 x_2$
$x_1 \leq x_2$	$x_1 - x_1 x_2$
$x_1 = x_2$	$x_1 + x_2 - 2x_1 x_2$
$x_1 + x_2 + x_3 \leq 1$	$x_1 x_2 + x_1 x_3 + x_2 x_3$

Looks familiar?

Penalty weight

The penalty weight γ is a hyperparameter, you can change it and see how it affects the results. Different constraints may require different penalties.

If the penalty is too low it may lead to the selection of unfeasible solutions, if it is too high it may overwhelm the problem and lead to worse effectiveness.

A good starting point is the expected best value of the cost function.

From a general problem to QUBO

A general procedure to transform a generic (linear or quadratic) optimization problem in binary variables and real coefficients as QUBO:

$$\min_x y = x^T C x$$
$$\text{subject to } Ax = b, x \in \{0,1\}^n$$

Simply convert the constraints into quadratic penalties:

$$\min_x y = x^T C x + \gamma (Ax - b)^T (Ax - b)$$

Highly constrained problems

Adding constraints to a QUBO is easy in principle.

However, if the problem is highly constrained only a small portion of the sampled solutions may be feasible!

Highly constrained problems may not be a good fit for quantum annealing.

List of QUBO formulations

The following are two references which contain the formulation of many NP-Complete and NP-Hard optimization problems both as QUBO or as the equivalent Ising formulation:

- *Lucas, A. (2014). Ising formulations of many NP problems. Frontiers in Physics, 2, 5.*
- *Glover, F., Kochenberger, G., & Du, Y. (2019). Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. 4OR, 17(4), 335-371.*

Feature Selection and Clustering

Feature selection

A good feature selection may help you reduce the computational cost of a learning algorithm trained on the data and may improve its effectiveness by removing noise.

We can write feature selection as a quadratic optimization problem.

Each feature will be associated to a binary variable, if $x_i = 1$ then feature i will be selected, otherwise it will not.

M. Ferrari Dacrema, F. Moroni, R. Nembrini, N. Ferro, G. Faggioli, and P. Cremonesi. Towards Feature Selection for Ranking and Classification Exploiting Quantum Annealers. In Proceedings of SIGIR 2022.

Underlying idea

We want to select a subset of features that are:

- As informative as possible on what we want to predict or classify
- Not overlapping, avoid those that are redundant

This can be done by selecting the subset that maximizes a certain information measure. Finding the optimal subset is a combinatorial problem and becomes intractable but several heuristic methods are available.

QUBO Correlation

A simple strategy is to select features based on their correlations. The idea being to select features correlated with the target label and discarding those that are correlated with each other:

$$Q_{ij} = \begin{cases} -r(f_i, t) & if\ i = j \\ r(f_i, f_j) & if\ i \neq j \end{cases}$$

Where $r(\cdot,\cdot)$ is the Pearson Correlation, but other ones can be used.

MIQUBO: MI Feature Selection as QUBO

MIQUBO is a similar approach that uses Mutual Information (MI), which is based on the Shannon Entropy of the two features:

$$Q_{ij} = \begin{cases} -MI(f_i, t) & if\ i = j \\ -MI\left(f_i, t \,\middle|\, f_j\right) & if\ i \neq j \end{cases}$$

Where $MI(f_i, t)$ is the MI between feature f_i and target variable, $MI\left(f_i, t \,\middle|\, f_j\right)$ is the conditional MI between feature f_i and the target, given f_j .

Restricting the number of features

Since with MI all terms of the Q matrix are negative, the best solution is always obtained selecting **all the features.**

Solution: Introduce a penalty term to force the selection of exactly *k* features.

$$\min_x y = x^T Q x + \gamma \left(\sum_{i=1}^{n} x_i - k \right)^2$$

Clustering with Binary Partitioning

One of the simplest forms of clustering is to partition a graph in two clusters.

- Nodes **within** a cluster are **highly** connected
- Nodes in **different** clusters are **sparsely** connected
- A penalty balances the cluster sizes

Hayato Ushijima-Mwesigwa, Christian F. A. Negre, and Susan M. Mniszewski. 2017. Graph Partitioning using Quantum Annealing on the D-Wave System. In Proceedings of PMES'17

Clustering with Binary Partitioning

Consider a graph $G = (V, E)$ which we want to partition in two sets.

The objective is to minimize the number of *cut edges* (edges between nodes in different sets) and balance the set size.

Each node will be associated to a binary variable, if True the node will belong to one set, otherwise to the other. The problem will have $|V|$ variables:

$$x_i = \begin{cases} 1 & \text{if node } i \text{ is in the first set} \\ 0 & \text{otherwise} \end{cases}$$

Clustering with Binary Partitioning

Since we want to minimize the number of edges traversing the *cut*, we can write the optimization problem based on the couples of nodes connected by an edge. If the two nodes belong to different sets, we increase our cost function:

x_i	x_j	$cut(x_i, x_j)$
0	0	0
0	1	1
1	0	1
1	1	0

$$\min_x \sum_{i,j \in E} x_i + x_j - 2\, x_i x_j$$

Clustering with Binary Partitioning

Since we want to minimize the number of edges traversing the *cut*, we can write the optimization problem based on the couples of nodes connected by an edge. If the two nodes belong to different sets, we increase our cost function:

x_i	x_j	$cut(x_i, x_j)$
0	0	0
0	1	1
1	0	1
1	1	0

$$\min_x \sum_{i,j \in E} x_i + x_j - 2\, x_i x_j$$

Looks familiar?

Clustering with Binary Partitioning

The second requirement of Binary Partitioning is that the nodes are split evenly in the two sets, so we can add a penalty term that will be zero when the nodes are split evenly. The final optimization problem becomes:

$$\min_{x} \sum_{i,j \in E} x_i + x_j - 2\, x_i x_j + \gamma \left(\sum_{i \in N} x_i - \frac{|N|}{2} \right)^2$$

Clustering with k-medoids

K-medoids is similar to k-means, but ensures that the centroid of a cluster is among the data points. The idea is as follows, each binary variable represents a data point, if the variable is True, then the data point is a medoid (the center of a cluster), otherwise it is not.

The optimization problem has two components:

- One that selects points that are **distant**
- One that selects points that are **central**

Clustering with k-medoids

$$\min_x \left(\sum_{i,j}^{N} x_i x_j \left(\gamma - \frac{1}{2} \alpha \Delta_{ij} \right) - 2\gamma k \sum_{i}^{N} x_i \right) + \left(\sum_{i,j}^{N} \gamma x_i x_j + \sum_{i}^{N} x_i \left(\sum_{j}^{N} \beta \Delta_{ij} - 2\gamma k \right) \right)$$

Distant Points

Central Points

Where:

- N is the number of data points, k is the number of medoids
- α, β, γ are weights that can be chosen heuristically
- Δ_{ij} is a function of the distance between point i and j

Clustering with k-medoids

Note that this method only identifies which data points are the medoids.

It is possible to allocate a data point to a cluster by choosing the cluster associated to the medoid that has the lowest distance, Δ_{ij}, from it.

The method requires to compute the distances between all couples of points and is not iterative.

Community Detection

Community detection can be used as another tool to create cluster of users.

Idea: instead of building a single model that should provide good recommendations for all users, split them in communities and use a different model for each of them.

Challenge: finding the optimal communities is computationally expensive

Community Detection

The task can be transformed into a QUBO by maximizing the **modularity:**

$$Q = \frac{1}{2|E|}\sum_{i,j} \mathcal{M}_{ij}\,\delta(c_i, c_j) \qquad\qquad \mathcal{M}_{ij} = A_{ij} - P_{ij}$$

Nembrini, R., Carugno, C., Ferrari Dacrema, M., & Cremonesi, P. (2022). Towards recommender systems with community detection and quantum computing. In ACM RecSys 2022.

Community Detection

The task can be transformed into a QUBO by maximizing the **modularity:**

$$Q = \frac{1}{2|E|}\sum_{i,j}\mathcal{M}_{ij}\boxed{\delta(c_i, c_j)} \qquad \mathcal{M}_{ij} = A_{ij} - P_{ij}$$

Do nodes i, j belong to the same community?

Nembrini, R., Carugno, C., Ferrari Dacrema, M., & Cremonesi, P. (2022). Towards recommender systems with community detection and quantum computing. In ACM RecSys 2022.

Community Detection

The task can be transformed into a QUBO by maximizing the **modularity:**

$$Q = \frac{1}{2|E|} \sum_{i,j} \mathcal{M}_{ij} \delta(c_i, c_j)$$

Do nodes i, j belong to the same community?

$$\mathcal{M}_{ij} = A_{ij} - P_{ij}$$

Adjacency Matrix

Probability Matrix Null Model

Nembrini, R., Carugno, C., Ferrari Dacrema, M., & Cremonesi, P. (2022). Towards recommender systems with community detection and quantum computing. In ACM RecSys 2022.

Iterative Community Detection

The model requires one-hot-encoding of the user-community.

Mitigation: Iterative process where the users are split in two.

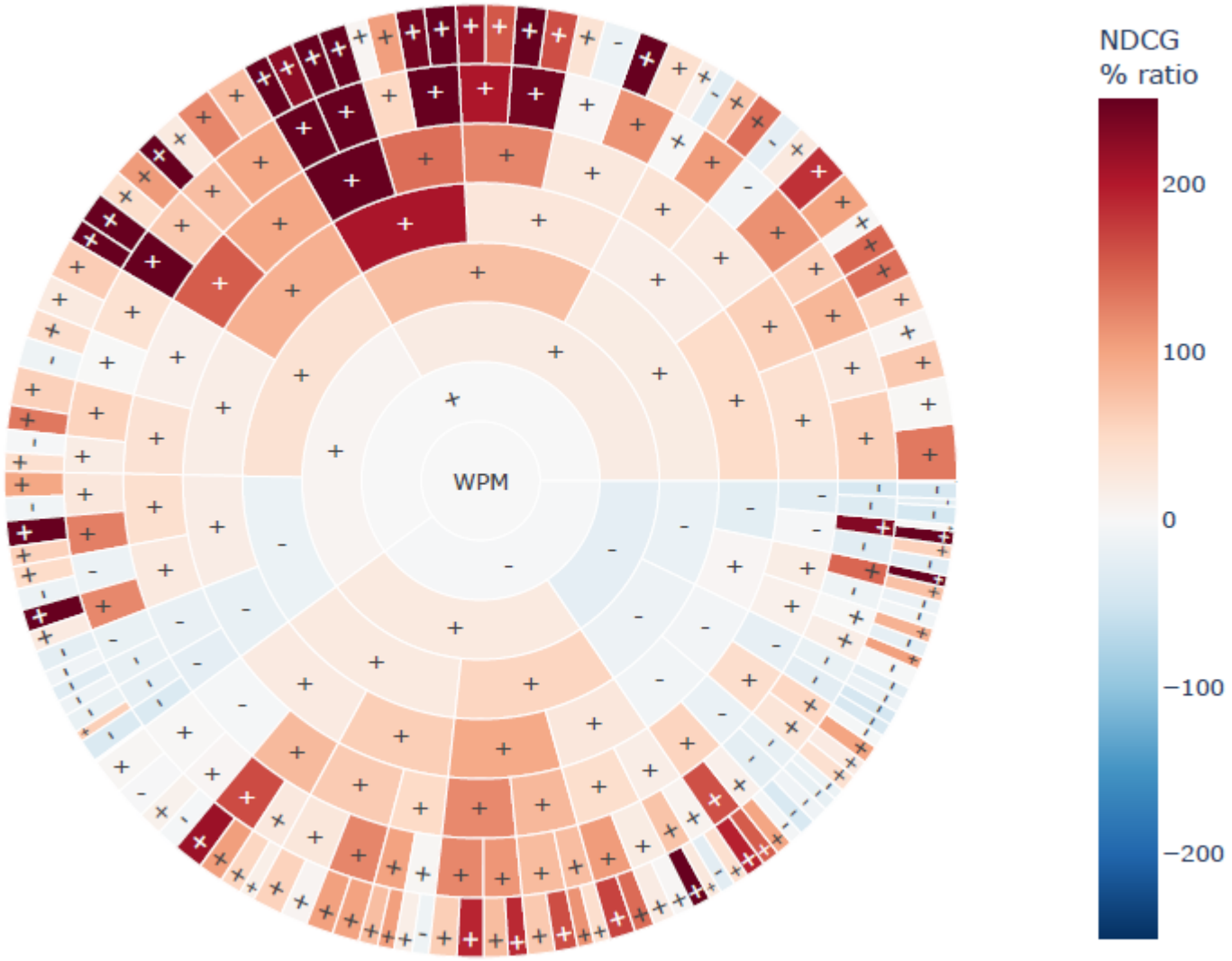
At each iteration, a simple TopPopular model is trained on the community.

Results are compared vs a TopPopular trained on all users.

Applied on 8 datasets.

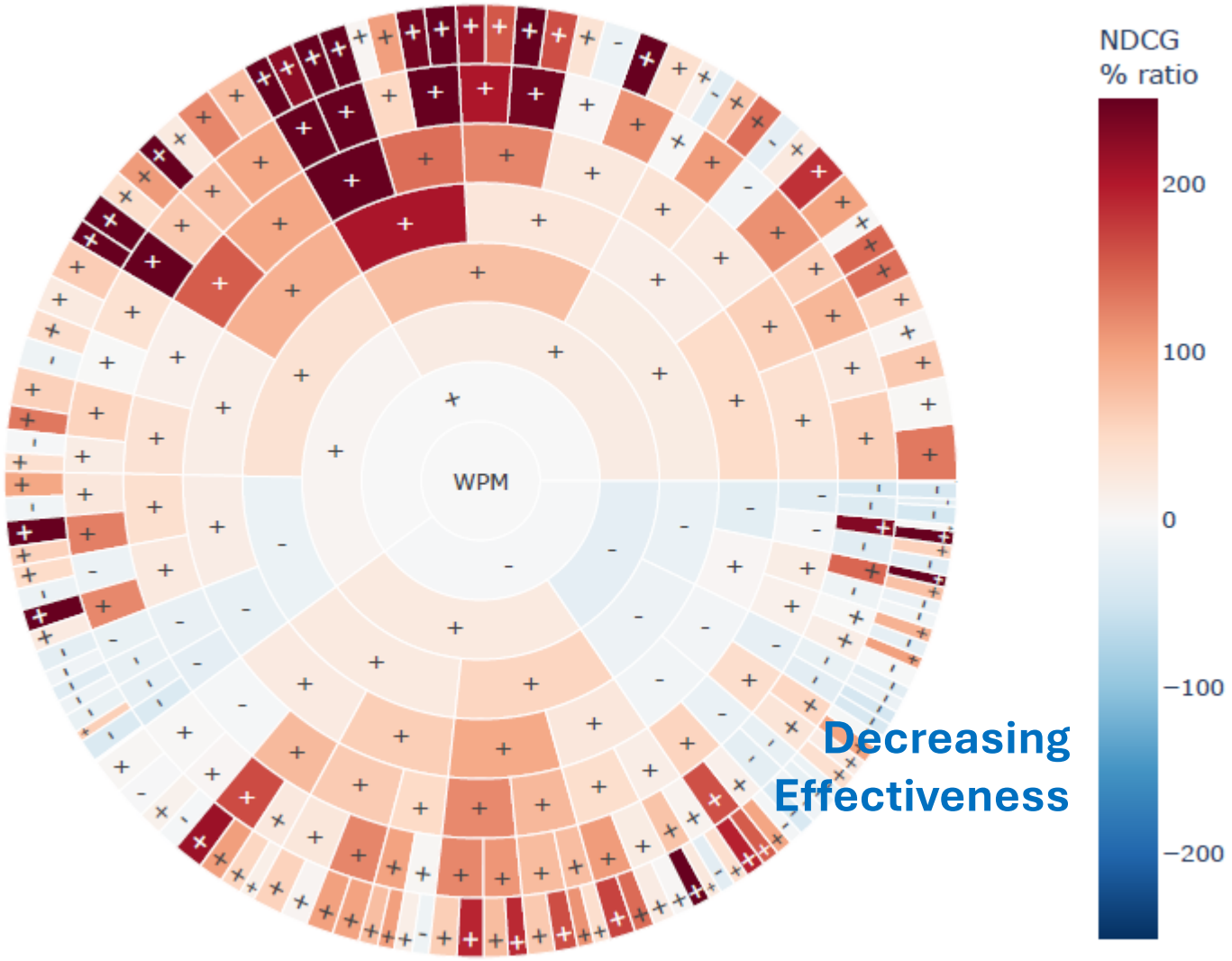
Result Overview (ML1M)

	N	C	Bipartite Modularity				Weighted Projection Modularity			
			Precision	MAP	NDCG	I Cov.	Precision	MAP	NDCG	I Cov.
Baseline	-	-	0.1072	0.0586	0.1109	0.0234	0.1072	0.0586	0.1109	0.0234
D-Wave Leap Hybrid	1	2	0.1169	0.0673	0.1228	0.0348	0.1223	0.0734	0.1295	0.0337
	2	4	0.1288	0.0753	0.1389	0.0531	0.1299	0.0775	0.1390	0.0500
	3	8	0.1401	0.0840	0.1499	0.0788	0.1471	0.0885	0.1528	0.0688
	4	16	0.1515	0.0903	0.1600	0.1154	0.1561	0.0948	0.1671	0.0917
	5	32	0.1579	0.0952	0.1662	0.1561	0.1657	0.0998	0.1772	0.1254
	6	64	**0.1603**	**0.0957**	**0.1684**	0.2004	0.1703	**0.1011**	0.1821	0.1476
	7	128	0.1548	0.0895	0.1621	0.2320	**0.1724**	0.1000	**0.1837**	0.1808
	8	256	0.1468	0.0810	0.1532	0.2599	-	-	-	-
	9	512	0.1305	0.0684	0.1357	0.2552	-	-	-	-
Quantum Annealing	7	128	0.1554	0.0895	0.1616	0.2354	-	-	-	-
	8	256	0.1469	0.0810	0.1513	0.2658	-	-	-	-
	9	512	0.1309	0.0688	0.1351	0.2627	-	-	-	-



(b) Weighted Projection Modularity

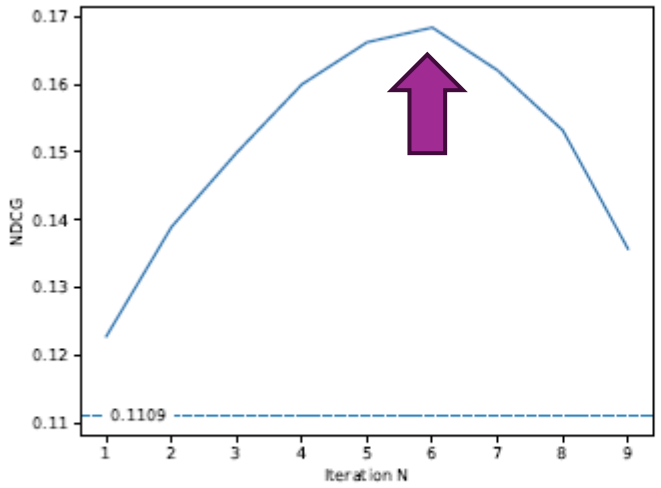
Increasing Effectiveness



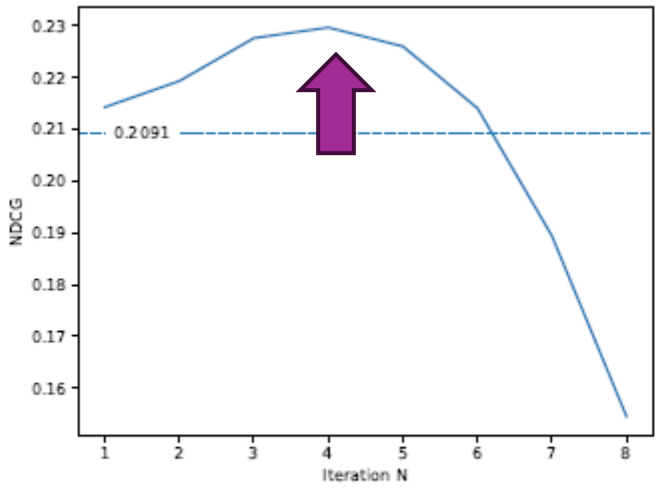
Decreasing Effectiveness

Decreasing Effectiveness

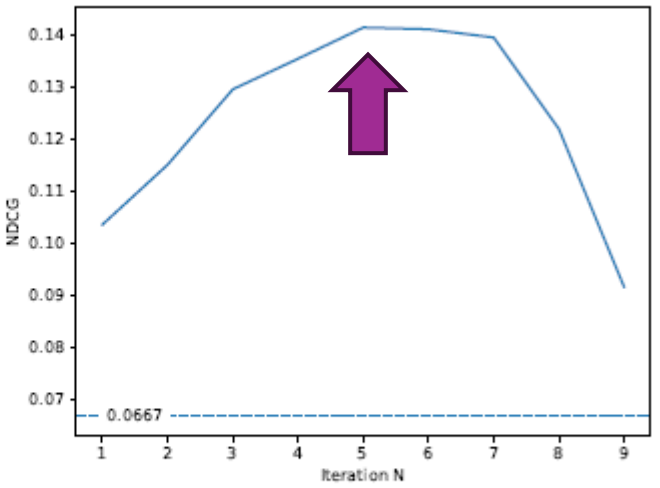
(b) Weighted Projection Modularity



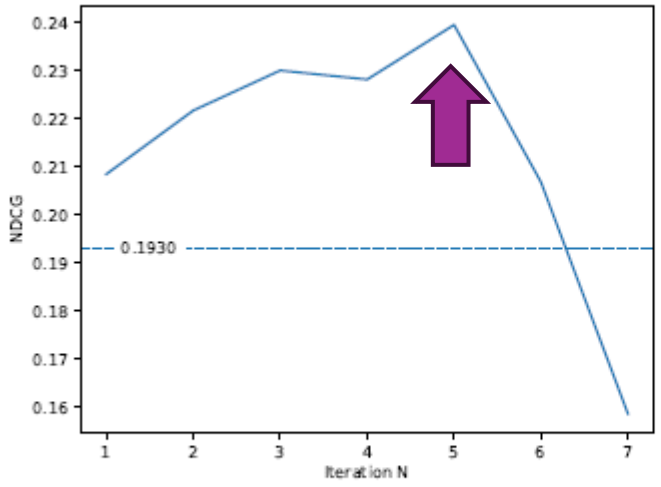
(b) MovieLens1M



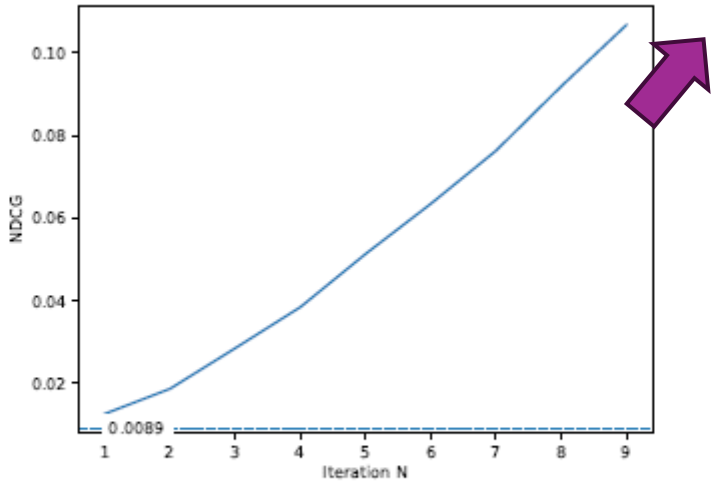
(c) MovieLens Hetrec 2011



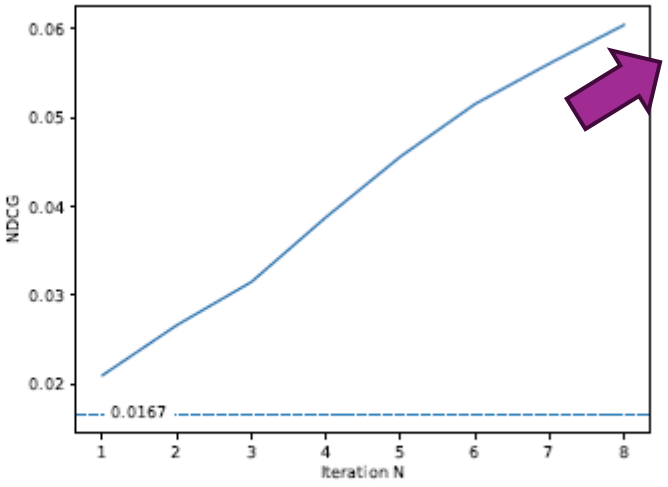
(d) LastFM Hetrec 2011



(f) Frappe



(g) CiteULike-a



(h) CiteULike-t

Quantum Annealer Architecture

D-Wave quantum annealer

Current quantum annealers (D-Wave) are meta-heuristic devices that use quantum annealing to minimize QUBO problems. I will call them Quantum Processing Unit (QPU).

They do this by creating a physical system that is equivalent to the QUBO problem and then guide this physical system towards a state of minimal energy. The final state of the qubits will correspond to a solution of the QUBO.

D-Wave quantum annealer

The leading technology is **superconducting electronics**. In the D-Wave QPU the qubit is a superconducting ring, the state is the direction of the current.

Magnetic fields can be used to favor the flow in one or the other direction (**bias**) and control how easily the current can change its rotation, as well as control the strength of the interaction between qubits (**quadratic terms, coupling**).

Problem connectivity

The QUBO problem formulation allows quadratic terms between all existing problem variables, so it can be “fully connected”.

$$y = \sum_i q_{i,i} x_i + \sum_{i>j} q_{i,j}\, x_i x_j$$

It is not feasible in practice to build a fully connected structure in a planar circuit. Each physical qubit is connected according to a certain topology.

QPU topology

The D-Wave QPU is based on a physical lattice of qubits and couplers.

Over the years different architectures have been introduced: Chimera, Pegasus and Zephyr.

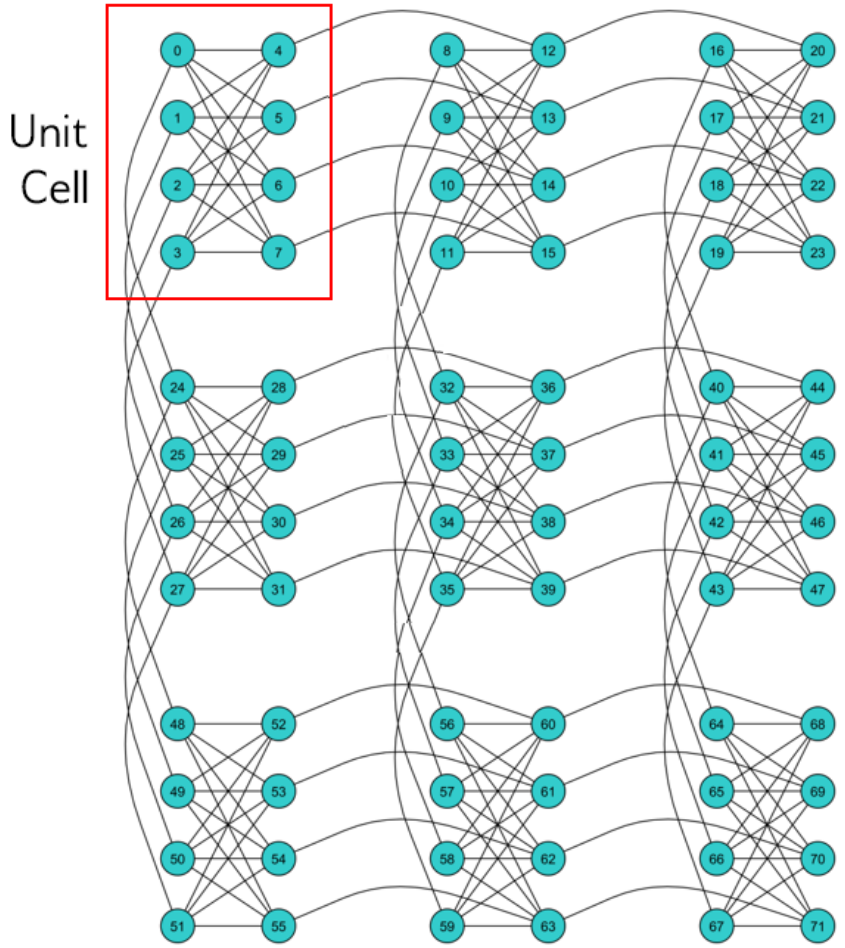
The Advantage QPU uses the Pegasus topology and contains over 5000 qubits.

Each is coupled to a maximum of 15 other qubits.

Chimera topology

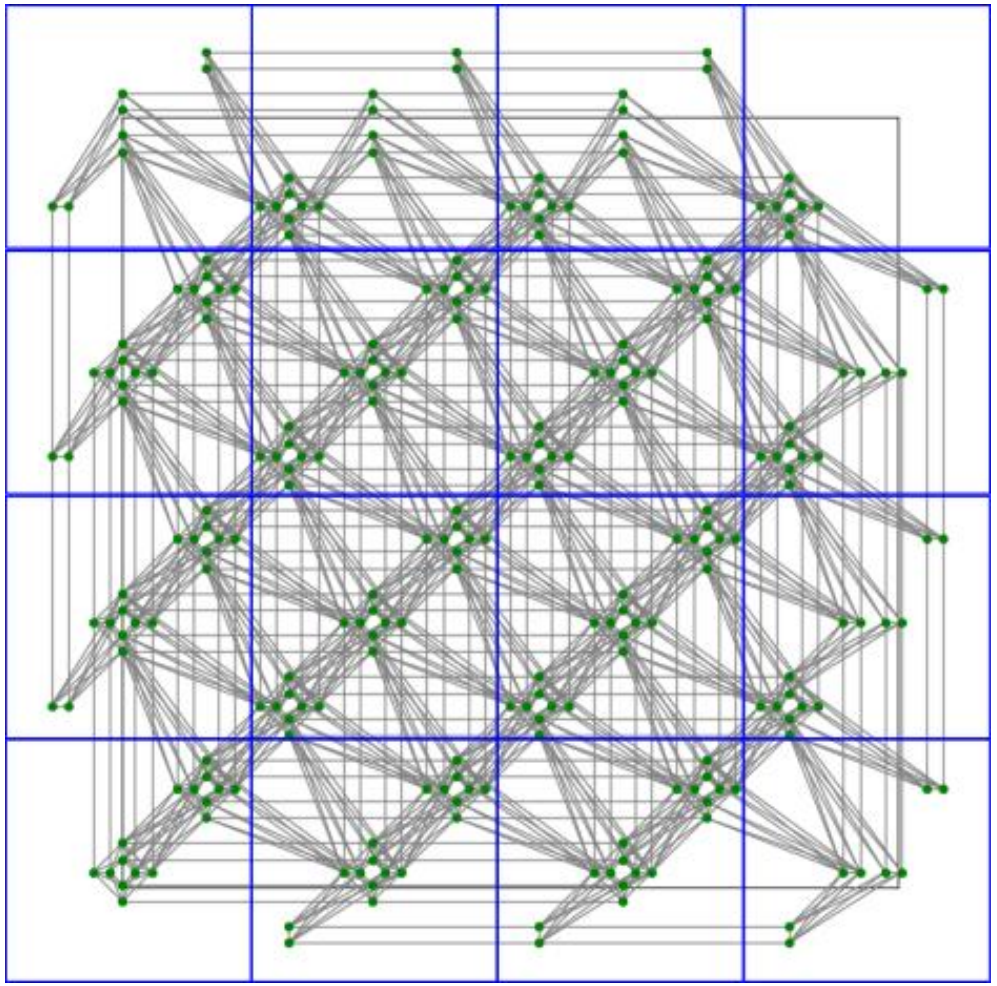
The Chimera architecture comprises sets of connected *unit cells*, each with 8 qubits.

Each qubit has 6 couplers.



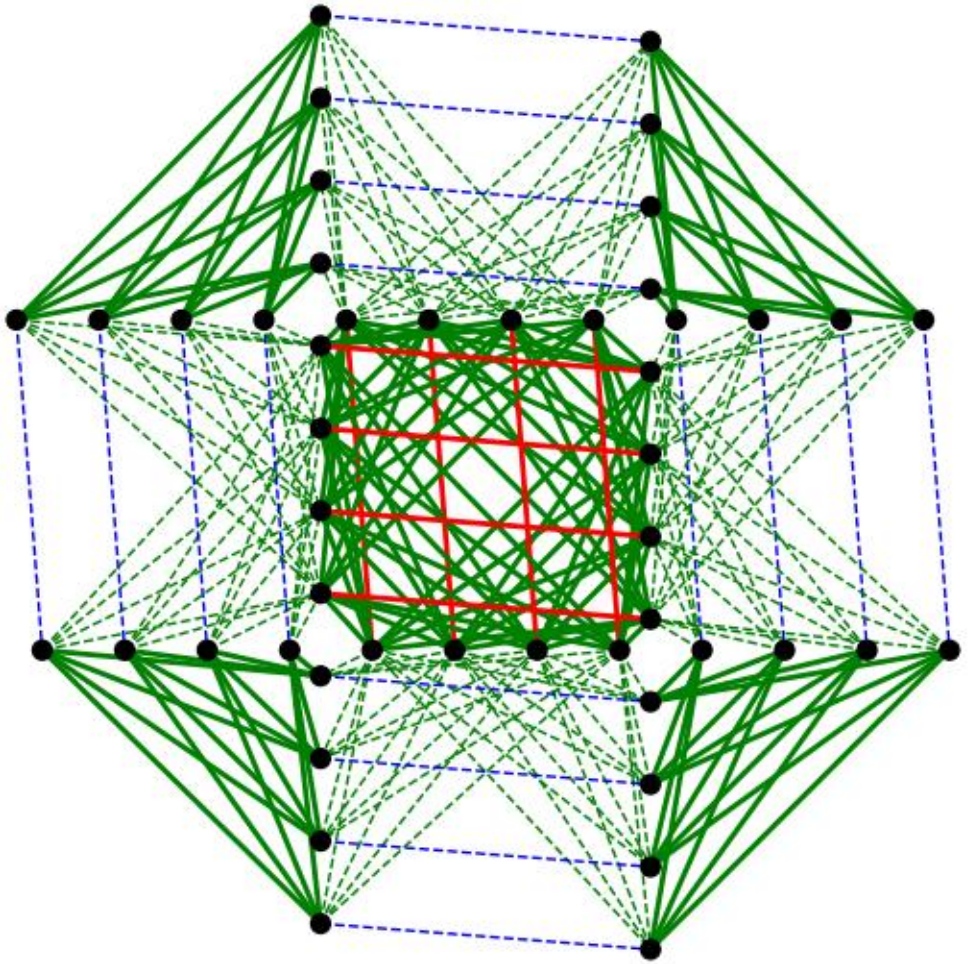
Pegasus topology

The Pegasus topology is based on the same basic unit cell as Chimera, each qubit is connected to at most 15 others



Zephyr topology

The Zephyr topology is the most recent topology, which increases the connectivity to 20.



Minor embedding

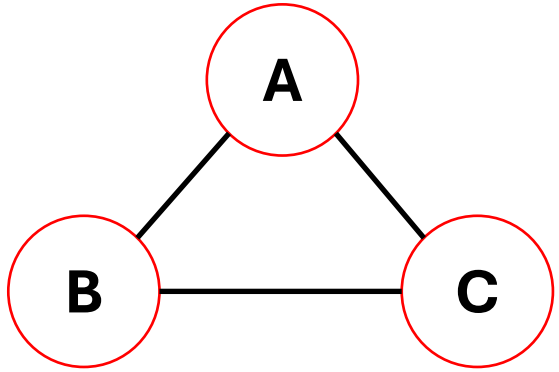
In order to solve a problem using the QPU we have to ensure its structure fits in the QPU topology. The process of transforming the original problem in its abstract formulation to one that can fit on the QPU is called *minor embedding*.

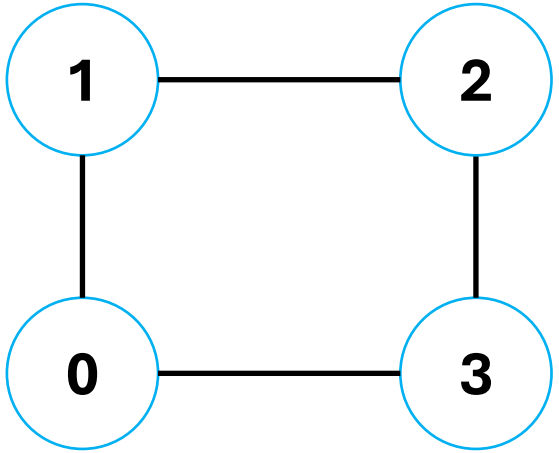
Minor embedding is itself a NP-Hard problem.

Heuristic algorithms in polynomial time are available.

Simple embedding example

Say that we want to embed a problem with a triangular graph (variables A, B and C) into a square topology.

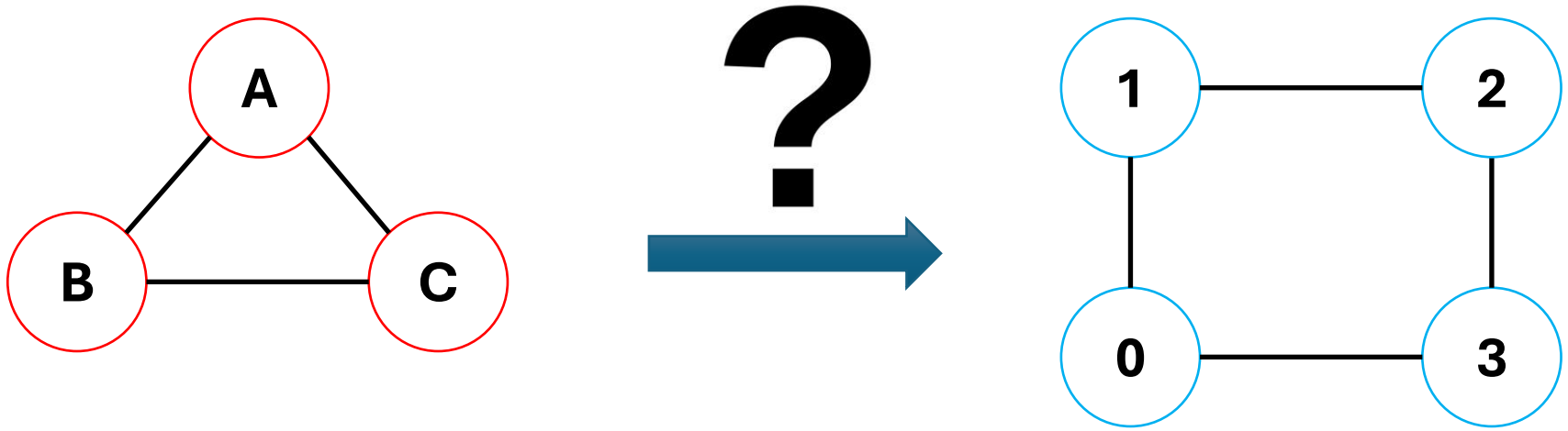




Simple embedding example

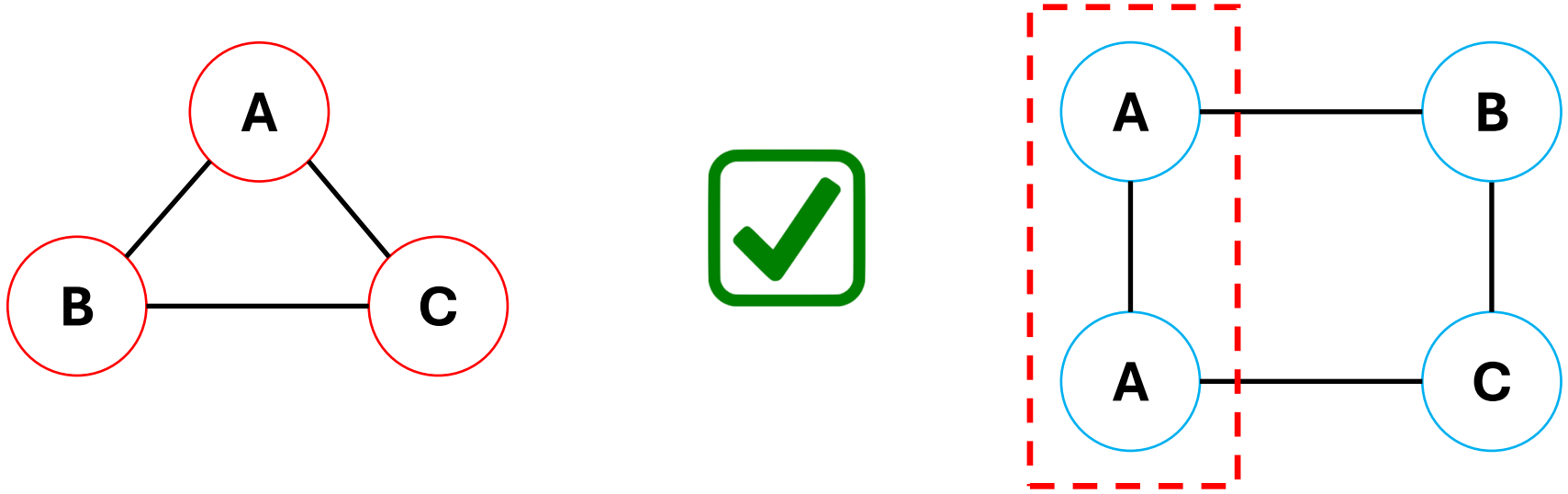
Say that we want to embed a problem with a triangular graph (variables A, B and C) into a square topology.

How do you make them fit?



Simple embedding example

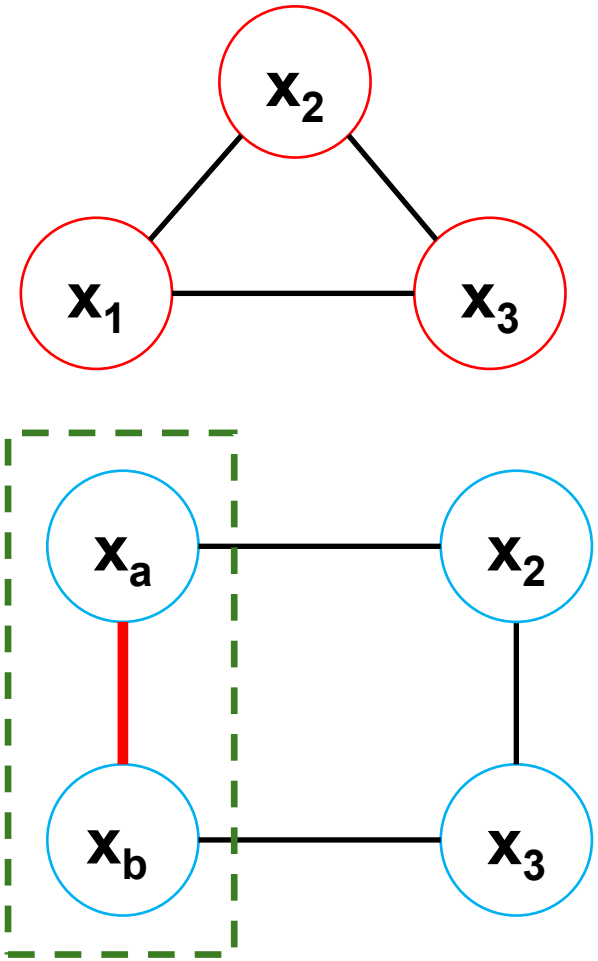
You duplicate one of your problem variables, in this example A, and represent it with two physical qubits. This will create a *chain* of qubits that represent the same logical variable.



QUBO with embedding

Consider the problem with triangular topology

$$y = q_{1,2}x_1x_2 + q_{1,3}x_1\ x_3 + q_{2,3}x_2x_3$$



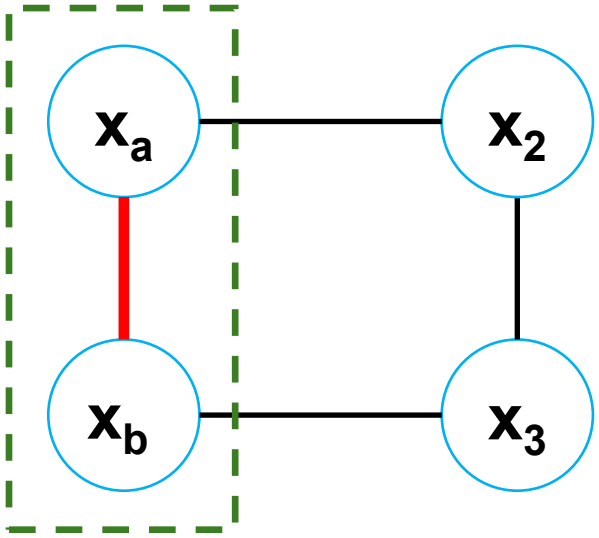
What happens when a chain is created?

x_1 is represented by two qubits, x_a and x_b, which should be equal

QUBO with embedding

Original QUBO $y = q_{1,2}x_1x_2 + q_{1,3}x_1\ x_3 + q_{2,3}x_2x_3$

Embedded QUBO $y' =$



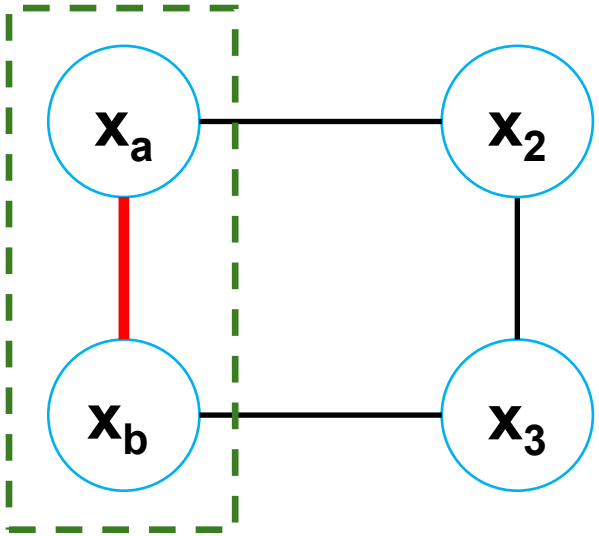
QUBO with embedding

Original QUBO

$$y = q_{1,2}x_1x_2 + q_{1,3}x_1\ x_3 + q_{2,3}x_2x_3$$

Embedded QUBO

$$y' = q_{1,2}x_ax_2$$



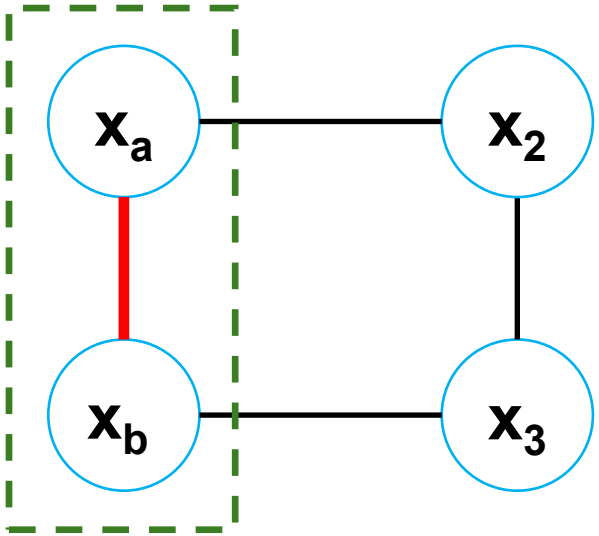
QUBO with embedding

Original QUBO

$$y = q_{1,2}x_1x_2 + q_{1,3}x_1\ x_3 + q_{2,3}x_2x_3$$

Embedded QUBO

$$y' = q_{1,2}x_ax_2 + q_{1,3}x_bx_3$$



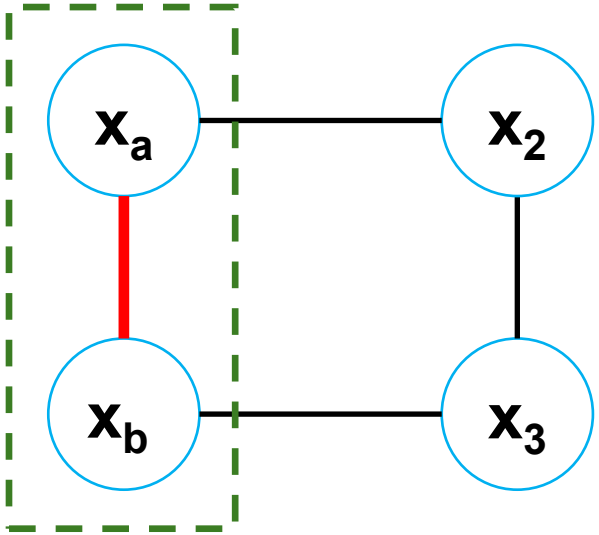
QUBO with embedding

Original QUBO

$$y = q_{1,2}x_1x_2 + q_{1,3}x_1\ x_3 + q_{2,3}x_2x_3$$

Embedded QUBO

$$y' = q_{1,2}x_ax_2 + q_{1,3}x_bx_3 + q_{2,3}x_2x_3$$



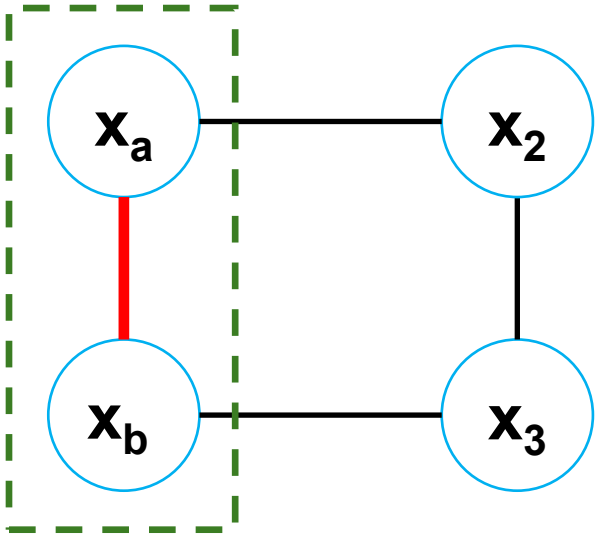
QUBO with embedding

Original QUBO

$$y = q_{1,2}x_1x_2 + q_{1,3}x_1\ x_3 + q_{2,3}x_2x_3$$

Embedded QUBO

$$y' = q_{1,2}x_ax_2 + q_{1,3}x_bx_3 + q_{2,3}x_2x_3 + \gamma c$$



$$c = -1 + x_a + x_b - 2x_ax_b$$

Equality constraint

Problems with chains

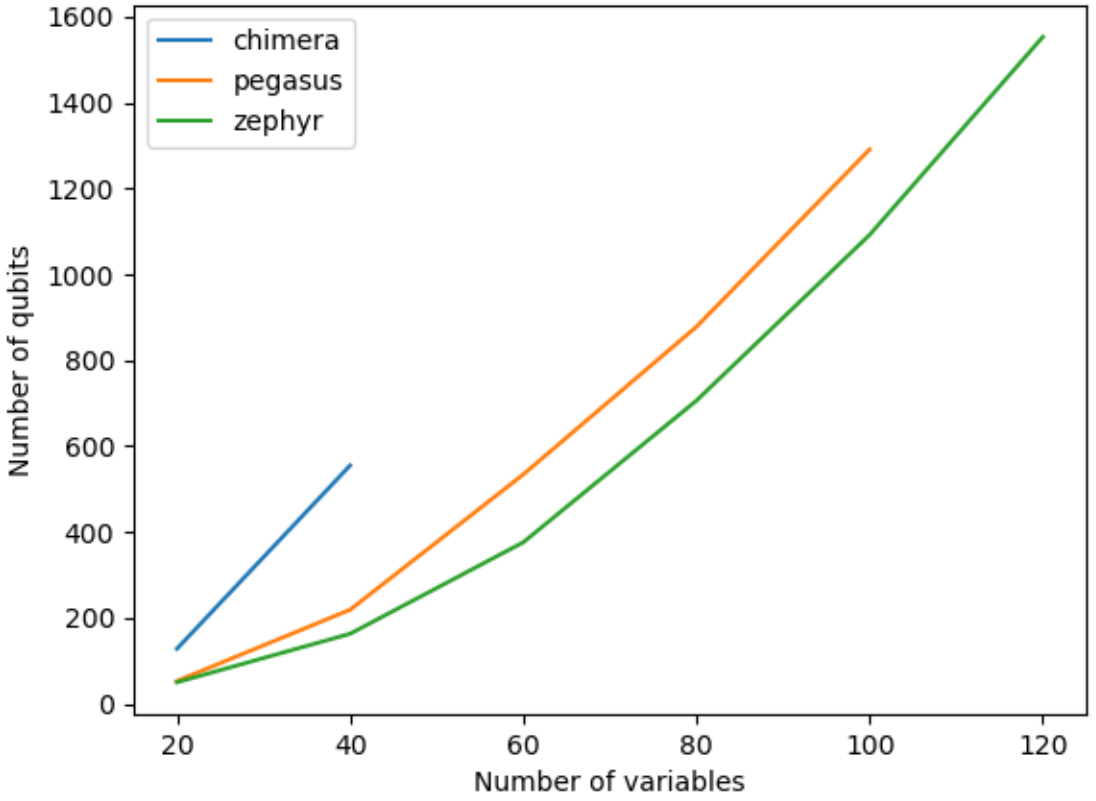
For the results to be consistent all qubits belonging to the same chain must be equal. Sometimes this may not be the case.

If the problem requires chains, some qubits will be used to represent the same logical variables, meaning the number of “free” variables you can use on the QPU will be smaller.

D-Wave Advantage has 5000 qubits but supports only 160 variables for fully connected problems.

Problems with chains

This plot shows the maximum number of variables that a fully connected problem can have to be embedded in a QPU of at most 1000 qubits.



Open research directions

Develop new or improved QUBO formulations:

- Computational cost of computing Q
- Number of variables required (categorical, binary expansion)
- Dense Q matrix makes minor embedding more challenging
- The characteristics of Q will make the problem more or less difficult to solve

Develop algorithms for minor embedding

Develop ways to split large problems into subproblems

Thank you!

Any questions?