Discrete Quadratic Models

Copyright © D-Wave Systems Inc.

Hybrid Solvers

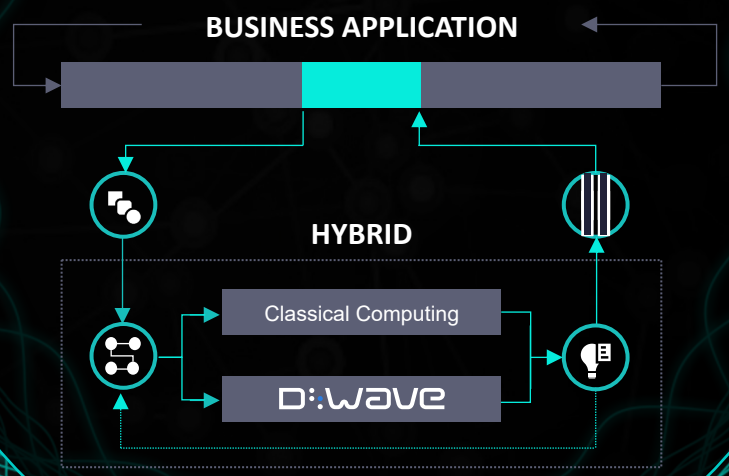
Binary quadratic model solvers

- Up to 1,000,000 variables
- Enables enterprise-scale problem solving
- Accepts problems with binary variables

Discrete quadratic model solvers

- Expands into new problem types
- Enables optimization with option selection: e.g., Choose one of 11, 19, 29
- Accepts discrete multi-level variables

SOLVERS THAT RUN PROBLEMS ON A COMBINATION OF QUANTUM AND CLASSICAL RESOURCES



Copyright © D-Wave Systems Inc.

Review: Binary Variables

Our QUBO and Ising models both use binary variables.

- **QUBO:** $x = 0$ or $x = 1$
- **Ising:** $x = -1$ or $x = +1$

Copyright © D-Wave Systems Inc.

Introducing Discrete Variables

A discrete quadratic model uses discrete, class-based variables.

- **Problem 1:** $x = 0, 1, \text{or } 2$
- **Problem 2:** $x = \text{red}, \text{blue}, \text{green}, \text{or purple}$
- **Problem 3:** $x = \text{True or False}$
  - Note that a binary model is also a discrete model

Copyright © D-Wave Systems Inc.

The DQM Hybrid Solver

Parameters:

- Maximum number of variables: 3,000
- Max number of cases per variable: 10,000
- Max quadratic biases: 3B

Copyright © D-Wave Systems Inc.

Problem Requirements for the DQM Solver

Variables:

- All variables can take values from some finite, discrete set

Constraints:

- Every variable must take exactly one value
  - “1 in N constraint”

Copyright © D-Wave Systems Inc.

Example

Copyright © D-Wave Systems Inc.

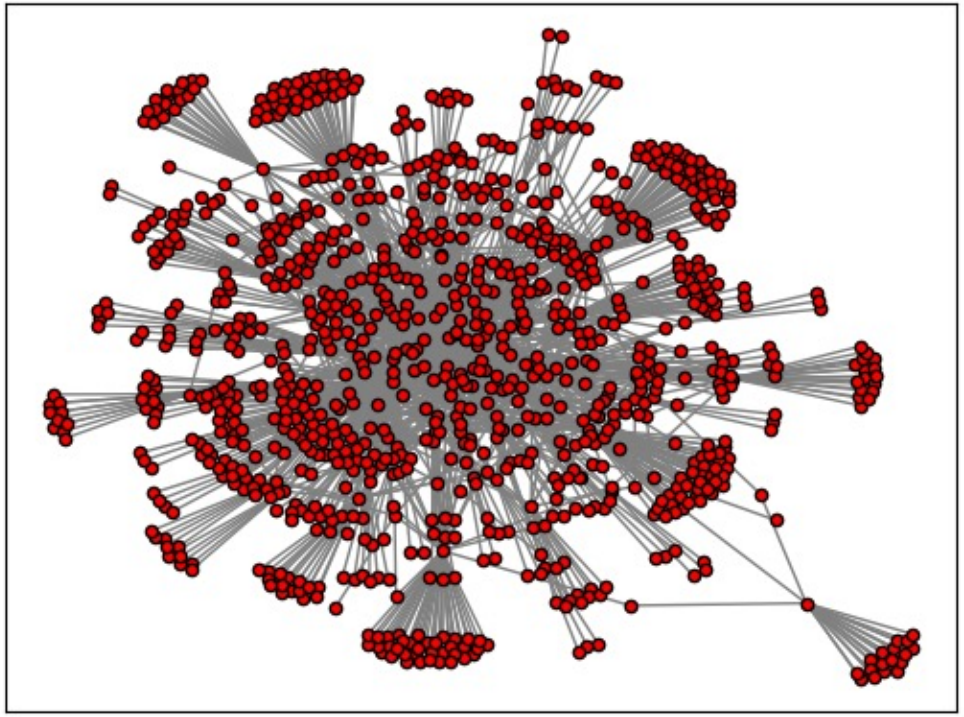
Problem Set Up – Immunization Strategy

What do we have:

- A network of people
  - Each person is a node in the graph
  - An edge between two people means that they visit the same places

What do we want:

- A small subset of the people that splits the network into two large groups.
  - This group can be vaccinated in order to prevent transmission from one large group to the other



Copyright © D-Wave Systems Inc.

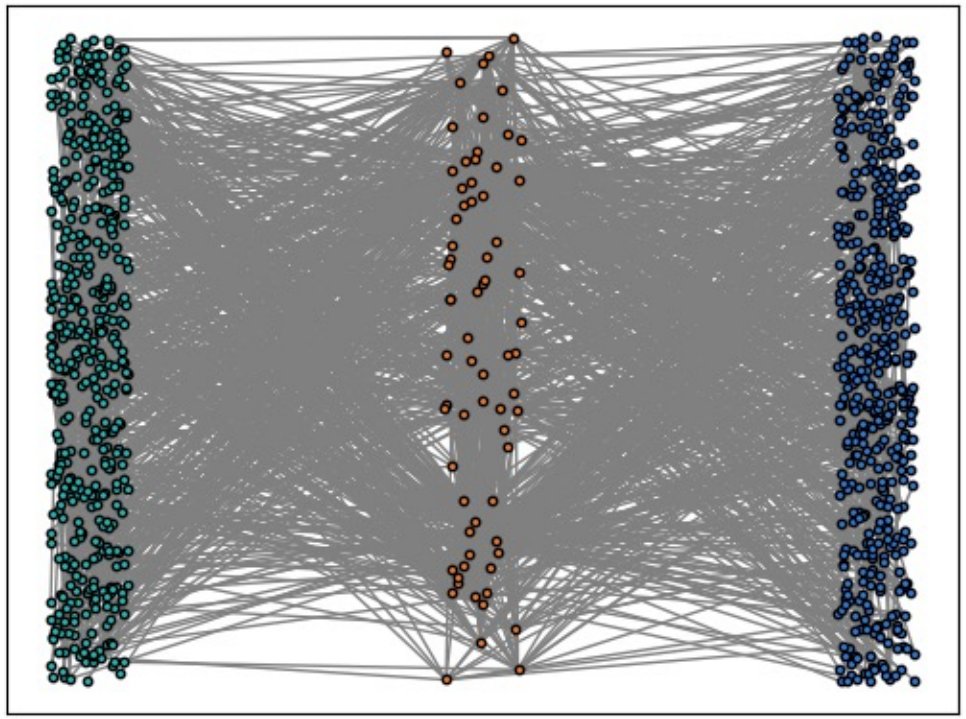
Problem Set Up – Immunization Strategy

What do we have:

- A network of people
  - Each person is a node in the graph
  - An edge between two people means that they visit the same places

What do we want:

- A small subset of the people that splits the network into two large groups.
  - This group can be vaccinated in order to prevent transmission from one large group to the other



Copyright © D-Wave Systems Inc.

Problem Set Up

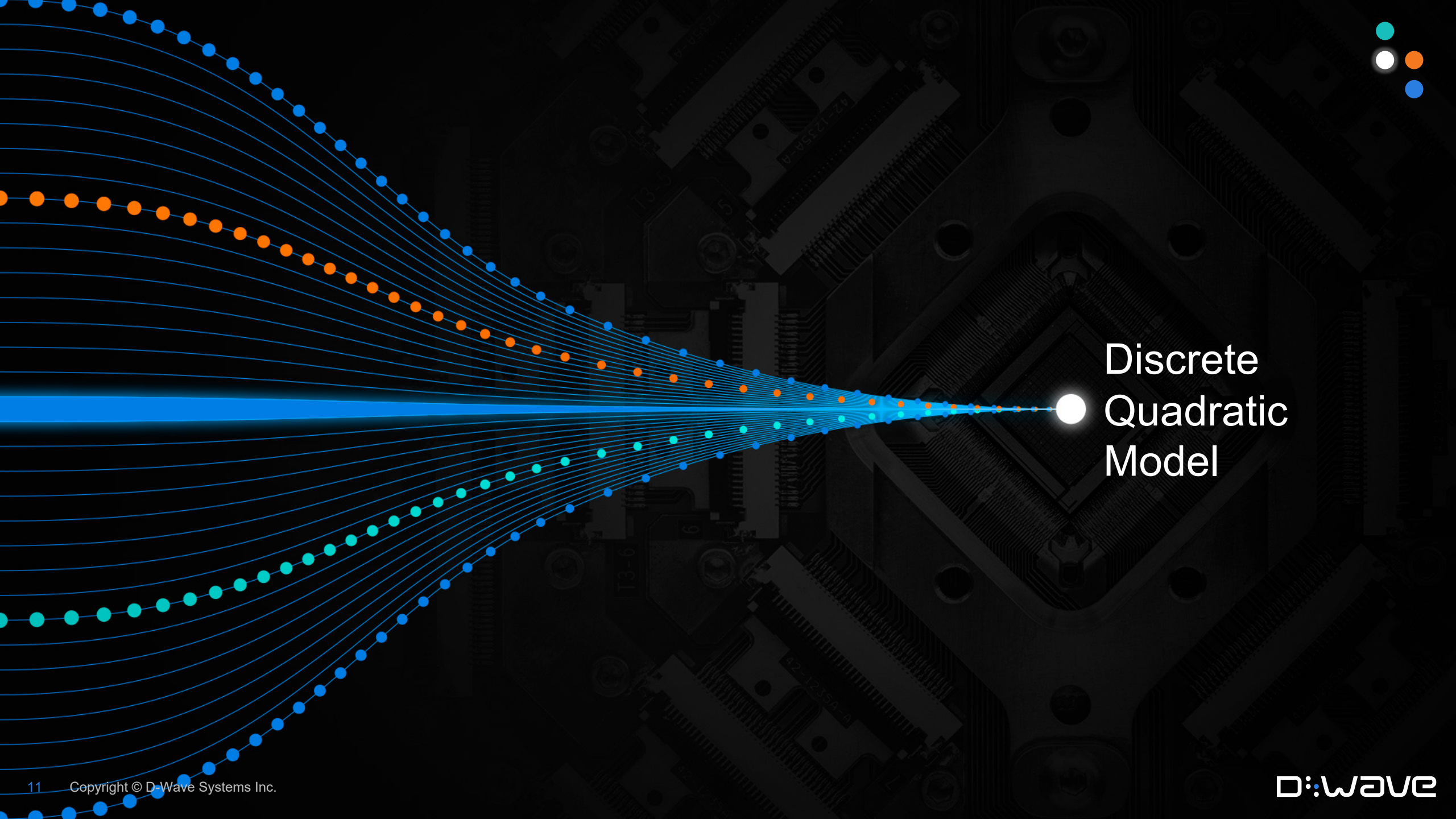
Objective:

- Minimize the number of people vaccinated (in the separator)

Constraints:

- Remaining population split into two large groups of approximately the same size
- No edges between these two large groups
- Each person in the network is assigned to one of the large groups or the separator

Copyright © D-Wave Systems Inc.



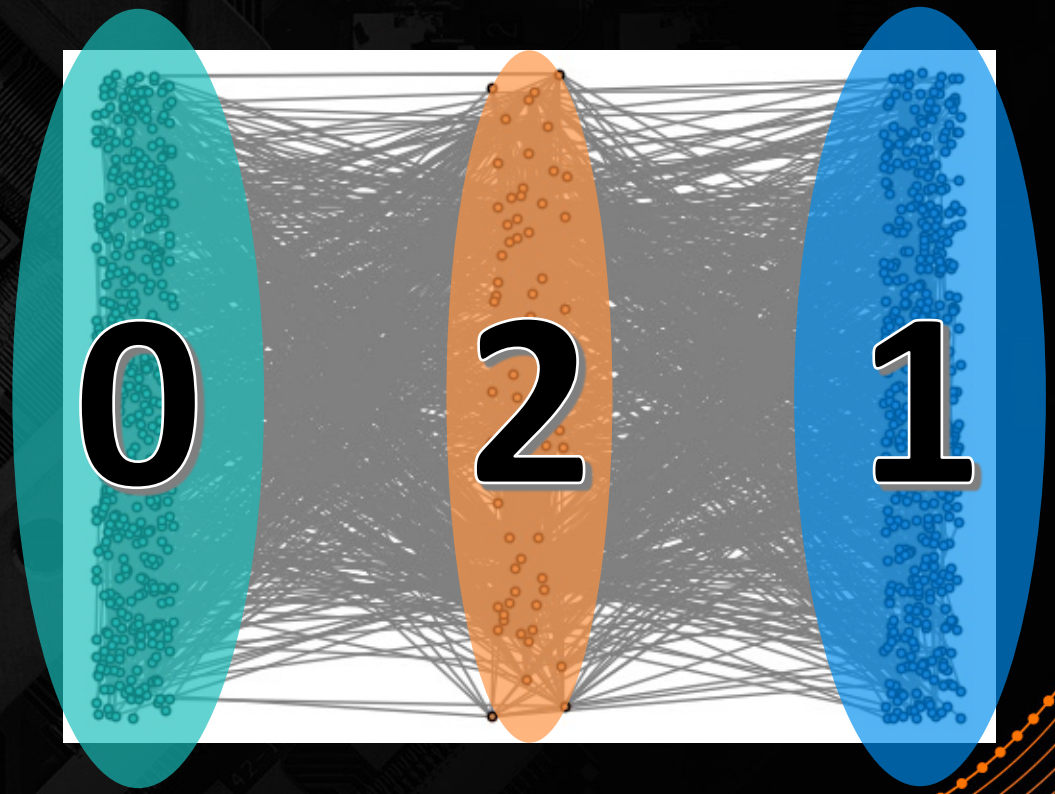
Copyright © D-Wave Systems Inc.

The Discrete Variables

Variables: Each variable represents a person in the network

Cases: Each person's variable has 3 cases:

- 0: One of the large groups
- 1: The other large group
- 2: The separator



Copyright © D-Wave Systems Inc.

General Strategy for Working with Discrete Variables

<u>The trick:</u> think about each discrete variable with 3 cases as 3 distinct binary variables.

$$x_{i,0} \quad x_{i,1} \quad x_{i,2}$$

These binary variables can be defined as:

$$x_{i,j} = \begin{cases} 1, & \text{if } x_i = j; \\ 0, & \text{otherwise.} \end{cases}$$

This corresponds to “variable i, case j”.

Copyright © D-Wave Systems Inc.

Objective: Minimize the size of the separator

<u>Recall:</u> The separator set is case 2 for our variables.

We need to count how many variables have $x_{i,2} = 1$.

To count the people assigned to the separator:

$$\sum_{i=0}^{N} x_{i,2}$$

Copyright © D-Wave Systems Inc.

Constraint: Each person is assigned to one of the sets

Each discrete variable gets one value of 0, 1, or 2.

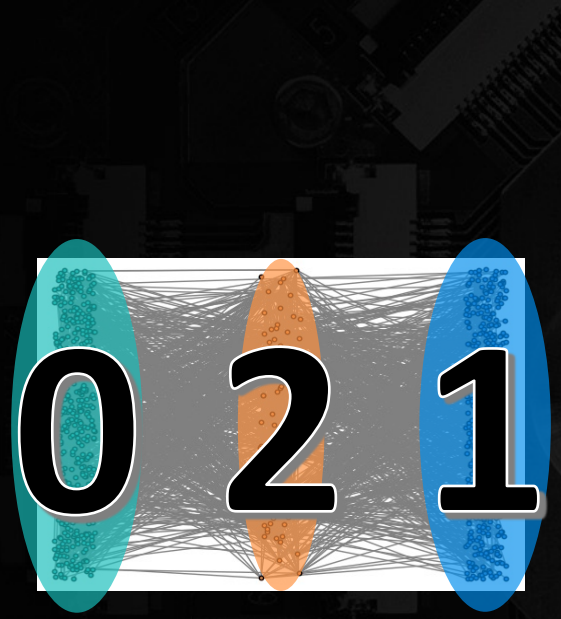
→ Each person gets assigned to set 0, 1, or 2

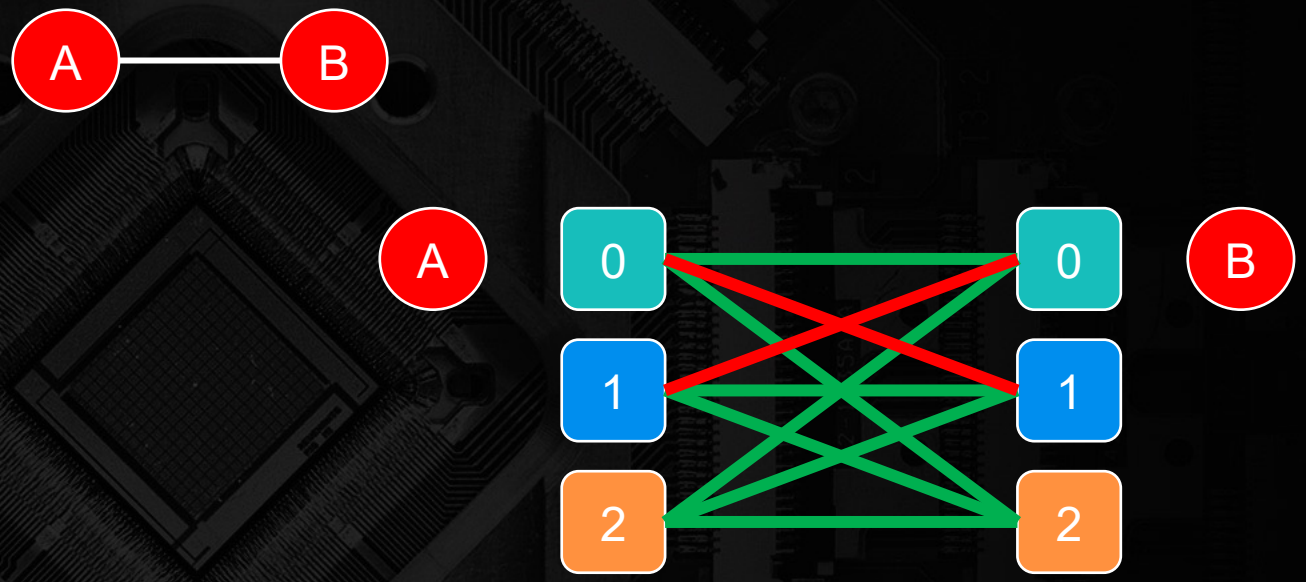
→ Each person gets assigned to either one of the large groups or the separator

Copyright © D-Wave Systems Inc.

Constraint: No edges between large groups

Let's look at one edge and penalize the bad cases.





Copyright © D-Wave Systems Inc.

Constraint: No edges between large groups

Let's look at one edge and penalize the bad cases.

A — B

	0	1	2
0	✓	✗	✓
1	✗	✓	✓
2	✓	✓	✓

Penalty Function for (A, B): $x_{A,0}x_{B,1} + x_{A,1}x_{B,0}$

Penalty Function for everyone:

$$\sum_{(i,j)\in E} (x_{i,0}x_{j,1} + x_{i,1}x_{j,0})$$

Copyright © D-Wave Systems Inc.

Constraint: The two large groups have the same size

What do we want in terms of our binary variables?

$$\sum_{i=0}^{N} x_{i,0} = \sum_{i=0}^{N} x_{i,1}$$

We need to make this "QUBO appropriate".

- Move everything to one side: $\sum_{i=0}^{N} x_{i,0} - \sum_{i=0}^{N} x_{i,1} = 0$
- Square the expression: $\left(\sum_{i=0}^{N} x_{i,0} - \sum_{i=0}^{N} x_{i,1}\right)^2$
- Multiply out:

$$\sum_{i=0}^{N} (x_{i,0}^2 - 2x_{i,0}x_{i,1} + x_{i,1}^2) + 2\sum_{i=0}^{N}\sum_{j>i} (x_{i,0}x_{j,0} - x_{i,1}x_{j,0} - x_{i,0}x_{j,1} + x_{i,1}x_{j,1})$$

Copyright © D-Wave Systems Inc.

Constraint: The two large groups have the same size

What do we want in terms of our binary variables?

$$\sum_{i=0}^{N} x_{i,0} = \sum_{i=0}^{N} x_{i,1}$$

We need to make this "QUBO appropriate".

- Move everything to one side: $\sum_{i=0}^{N} x_{i,0} - \sum_{i=0}^{N} x_{i,1} = 0$
- Square the expression: $\left(\sum_{i=0}^{N} x_{i,0} - \sum_{i=0}^{N} x_{i,1}\right)^2$
- Multiply out:

$$\sum_{i=0}^{N} \left(x_{i,0}^2 - 2x_{i,0}x_{i,1} + x_{i,1}^2\right) + 2\sum_{i=0}^{N}\sum_{j>i} \left(x_{i,0}x_{j,0} - x_{i,1}x_{j,0} - x_{i,0}x_{j,1} + x_{i,1}x_{j,1}\right)$$

become linear

Copyright © D-Wave Systems Inc.

Constraint: The two large groups have the same size

What do we want in terms of our binary variables?

$$\sum_{i=0}^{N} x_{i,0} = \sum_{i=0}^{N} x_{i,1}$$

We need to make this "QUBO appropriate".

- Move everything to one side: $\sum_{i=0}^{N} x_{i,0} - \sum_{i=0}^{N} x_{i,1} = 0$
- Square the expression: $\left(\sum_{i=0}^{N} x_{i,0} - \sum_{i=0}^{N} x_{i,1}\right)^2$
- Multiply out:

$$\sum_{i=0}^{N} (x_{i,0}^2 - 2x_{i,0}x_{i,1} + x_{i,1}^2) + 2\sum_{i=0}^{N}\sum_{j>i} (x_{i,0}x_{j,0} - x_{i,1}x_{j,0} - x_{i,0}x_{j,1} + x_{i,1}x_{j,1})$$

becomes zero

Copyright © D-Wave Systems Inc.

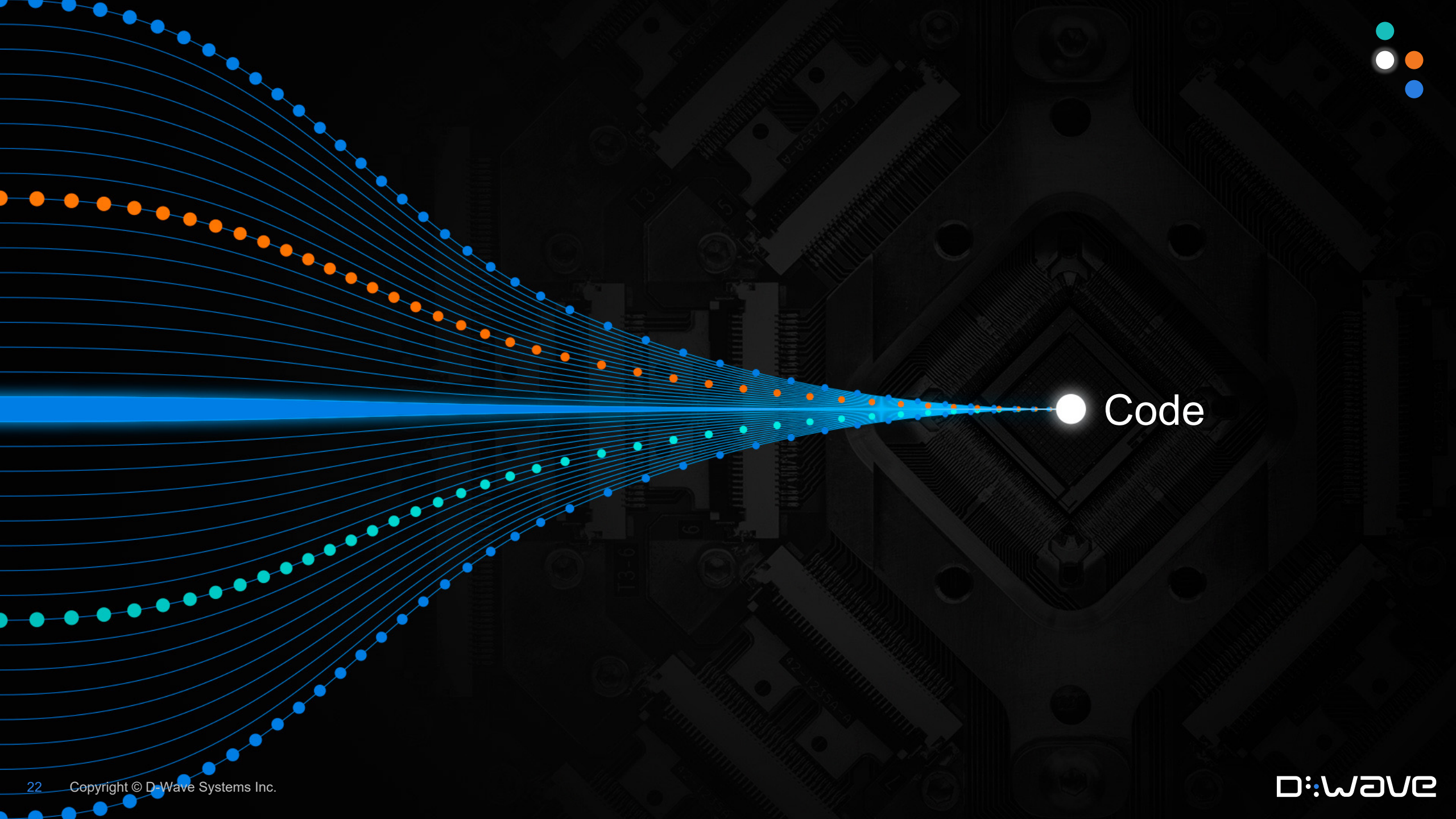
The Discrete Quadratic Model

Putting this all together builds the DQM.

DQM = Objective + Constraint1 + Constraint2

We can weight the objective versus constraints using Lagrange parameters.

Copyright © D-Wave Systems Inc.



Copyright © D-Wave Systems Inc.

Using the DQM Solver

Formulating a DQM follows this general flow.

1. Initialize the DQM object
2. Add variables with classes
3. Add linear biases
4. Add quadratic biases
5. Sample the DQM

Copyright © D-Wave Systems Inc.

Using the DQM Solver

Initialize	`dqm=DiscreteQuadraticModel()`
Add Variables	`dqm.add_variable(cases,label)`
Linear Biases	• `dqm.set_linear_case(u,3,17)` • `dqm.set_linear(u,[0,-1,3])`
Quadratic Biases	• `dqm.set_quadratic_case(u,2,v,1,-3)` • `dqm.set_quadratic(u,v,{(0,2):1.5})`
Sample	`sampler = LeapHybridDQMSampler()` `sampleset = sampler.sample_dqm(dqm)`

Copyright © D-Wave Systems Inc.

The Code: Setting up the DQM

```python
def build_dqm(G):
    """ Build the DQM for the problem instance."""

    # Two groups (cases 0, 1) and one separator group (case 2)
    num_groups = 3

    # Lagrange parameter on constraints
    gamma_1 = 1
    gamma_2 = 100

    # Initialize the DQM object
    print("\nBuilding DQM...")
    dqm = DiscreteQuadraticModel()

    # Build the DQM starting by adding variables
    for name in G.nodes():
        dqm.add_variable(num_groups, label=name)
```


Copyright © D-Wave Systems Inc.

Problem Formulation Summary

Objective:

$$\min \sum_{i=0}^{N} x_{i,2}$$

Constraint 1:

$$\sum_{i=0}^{N} (x_{i,0} + x_{i,1}) + \sum_{i=0}^{N} \sum_{j>i} (2x_{i,0}x_{j,0} - 2x_{i,1}x_{j,0} - 2x_{i,0}x_{j,1} + 2x_{i,1}x_{j,1})$$

Constraint 2:

$$\sum_{(i,j)\in E} (x_{i,0}x_{j,1} + x_{i,1}x_{j,0})$$

Copyright © D-Wave Systems Inc.

The Code: Adding in the Objective

Objective:

$$\min \sum_{i=0}^{N} x_{i,2}$$

```
# Add objective to DQM
for name in G.nodes():
    dqm.set_linear_case(name, 2, 1)
```


Copyright © D-Wave Systems Inc.

The Code: Adding in Constraint 1

Constraint 1:

$$\sum_{i=0}^{N}(x_{i,0}+x_{i,1})+\sum_{i=0}^{N}\sum_{j>i}(2x_{i,0}x_{j,0}-2x_{i,1}x_{j,0}-2x_{i,0}x_{j,1}+2x_{i,1}x_{j,1})$$

```
# Add constraint to DQM: |G1|=|G2|
all_nodes_ordered = list(G.nodes())
for i in range(len(all_nodes_ordered)):
    dqm.set_linear_case(all_nodes_ordered[i], 0, gamma_1)
    dqm.set_linear_case(all_nodes_ordered[i], 1, gamma_1)
    for j in range(i+1, len(all_nodes_ordered)):
        dqm.set_quadratic_case(all_nodes_ordered[i], 0, all_nodes_ordered[j], 0, 2*gamma_1)
        dqm.set_quadratic_case(all_nodes_ordered[i], 1, all_nodes_ordered[j], 1, 2*gamma_1)
        dqm.set_quadratic_case(all_nodes_ordered[i], 0, all_nodes_ordered[j], 1, -2*gamma_1)
        dqm.set_quadratic_case(all_nodes_ordered[i], 1, all_nodes_ordered[j], 0, -2*gamma_1)
```


Copyright © D-Wave Systems Inc.

The Code: Adding in Constraint 2

Constraint 2:

$$\sum_{(i,j)\in E} (x_{i,0}x_{j,1} + x_{i,1}x_{j,0})$$

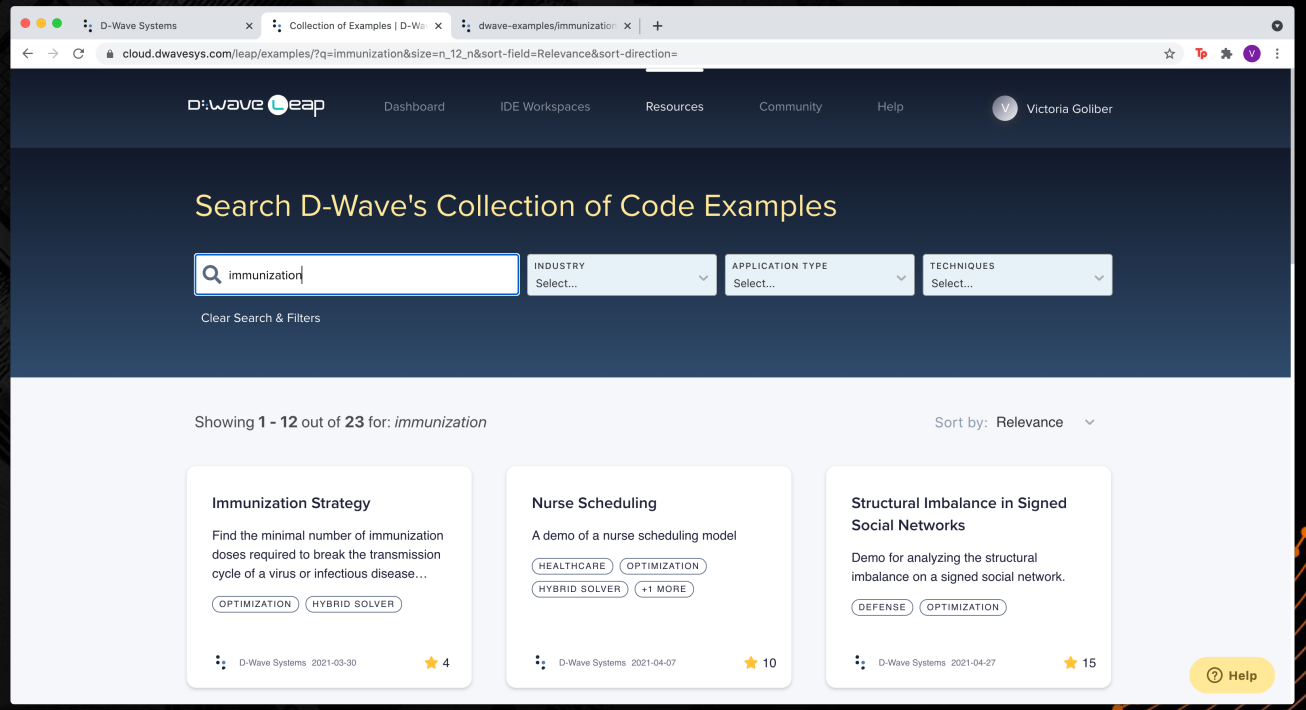
```
# Add constraint to DQM: e(G1, G2) = 0
for a, b in G.edges():
    if a != b:
        dqm.set_quadratic_case(a, 0, b, 1, gamma_2)
        dqm.set_quadratic_case(a, 1, b, 0, gamma_2)
```


Copyright © D-Wave Systems Inc.

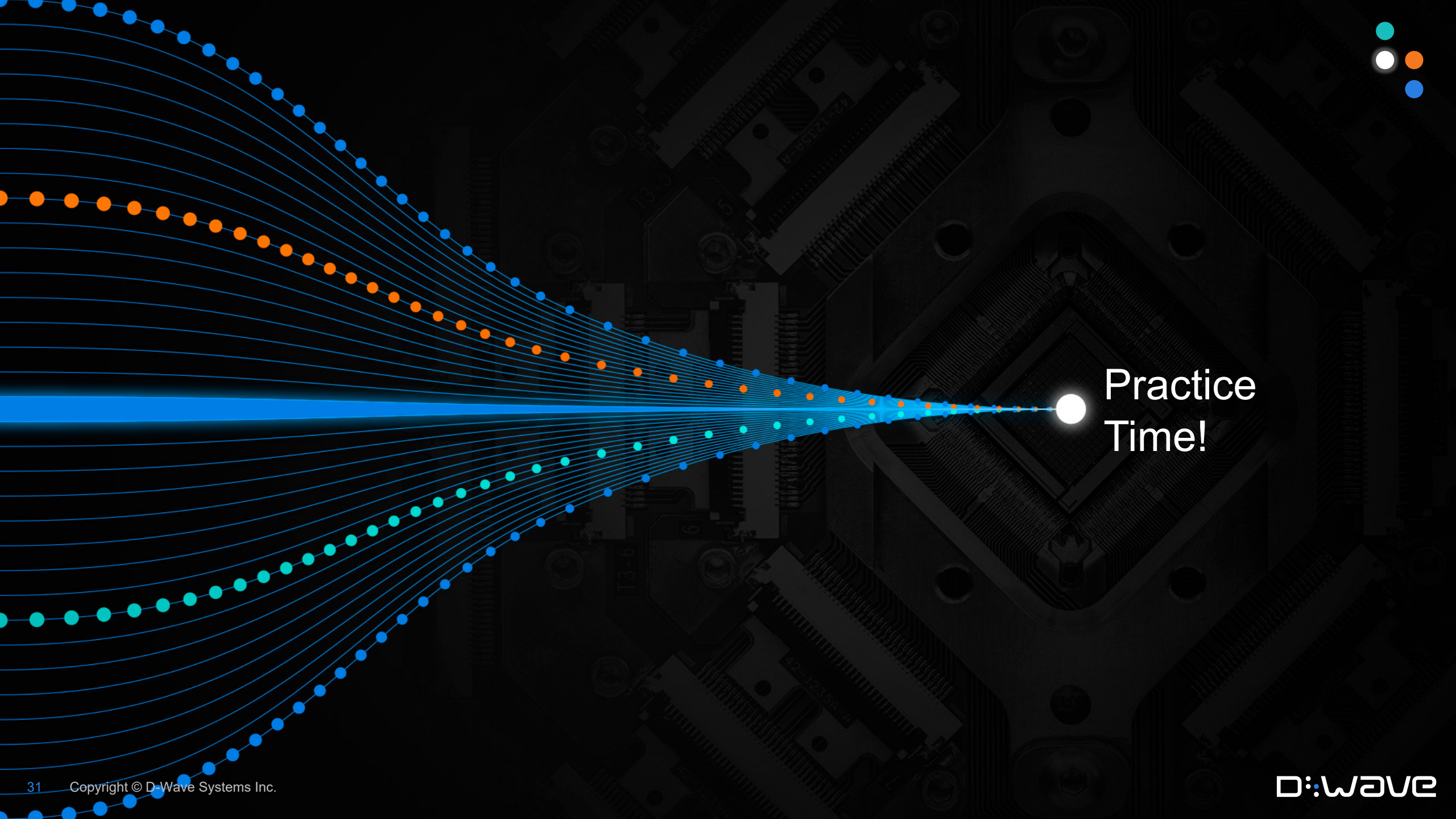
Running the Code

Go to the Collection of Code Examples and find the “Immunization Strategy” example.

- Take a look at the code and find the build_dqm function
- Identify the initialization, objective, and constraints
- Run the demo in the Leap IDE and explore the results
  - There are different types of networks available to run in the demo
  - Look at the README to see the full list of options available



Copyright © D-Wave Systems Inc.



Copyright © D-Wave Systems Inc.

Example Problem: Schedules

We have 4 employees and 4 shifts.

- Each employee has a ranking for the shifts from most preferred (1) to least preferred (4)
- We need each employee to be assigned to exactly one shift

What is the best way to schedule the employees?

	Shift 1	Shift 2	Shift 3	Shift 4
Anna	1	2	3	4
Bill	3	2	1	4
Chris	4	2	3	1
Diane	4	1	2	3

Copyright © D-Wave Systems Inc.

Building our DQM

1. Initialize the DQM object
2. Add variables with cases
3. Add linear biases
4. ~~Add quadratic biases~~

```python
from dimod import DiscreteQuadraticModel
from dwave.system import LeapHybridDQMSampler

# Problem information
employees = ["Anna", "Bill", "Chris", "Diane"]
shifts = [1, 2, 3, 4]
num_shifts = len(shifts)

# Initialize the DQM object
dqm = DiscreteQuadraticModel()

# Build the DQM starting by adding variables
for name in employees:
    dqm.add_variable(num_shifts, label=name)

# Use linear weights to assign employee preferences
dqm.set_linear("Anna", [1,2,3,4])
dqm.set_linear("Bill", [3,2,1,4])
dqm.set_linear("Chris", [4,2,3,1])
dqm.set_linear("Diane", [4,1,2,3])
```


Copyright © D-Wave Systems Inc.

Optimizing Employee Schedules

Navigate to github.com/dwave-training/employee-scheduling.

1. Open the file `scheduling_preferences.py` and look at how we're building up our discrete quadratic model
2. Open the file `scheduling_addemployees.py` and follow the instructions on the README to add additional employees
3. Open the file `scheduling_restrictions.py` and follow the instructions on the README to build in the restrictions
4. Challenge Problem! Modify `scheduling_restrictions.py` so that only two employees are scheduled per shift

Copyright © D-Wave Systems Inc.