D:wave

PRACTICAL QUANTUM COMPUTING

CATHERINE POTTS, PHD
SENIOR SALES ENGINEER

LEAP™: THE REAL-TIME QUANTUM CLOUD SERVICE

D-WAVE™ ADVANTAGE™

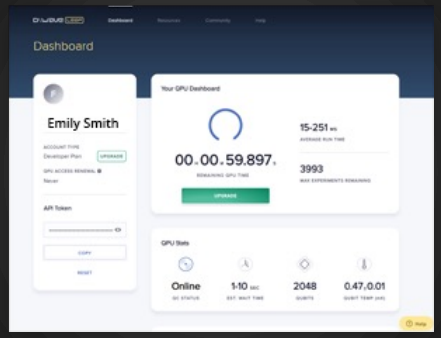
CLOUD-BASED

Leap

Cloud access to the
D-Wave quantum computer

REAL-TIME ACCESS

USER INTERFACE



- Real-time cloud access
- Programming model
- Open-source tools and templates
- Interactive tools and visualization
- Community support
- Online training and resources

BUILT FOR IN-PRODUCTION APPLICATIONS

ADVANTAGE ACCESS TODAY:

- Businesses, developers, and new users have access to the most connected and powerful quantum computer built for business

SEAMLESS TRANSITION:

- Ocean™ tools use Advantage out of the box
- Existing programs will switch over with feature-based solver selection
- Hybrid solvers are adapted to the new technology and will continue improving with ongoing algorithm development

SOFTWARE DEVELOPMENT KIT

SUITE OF OPEN-SOURCE PYTHON TOOLS ON THE D-WAVE GITHUB REPOSITORY

OPTIMIZATION

Portfolio optimization

Traffic flow

New application

CONSTRAINT SATISFACTION

Scheduling

Circuit fault detection

New application

APPLICATIONS

OCEAN SOFTWARE

Graph mapping

Constraint compilation

New mapping method

MAPPING METHODS

Problem Suitable for QPU: Quadratic Model (BQM, DQM, CQM)

UNIFORM SAMPLER API

Simulated annealing

D-Wave API

Hybrid sampler

New sampler

SAMPLERS

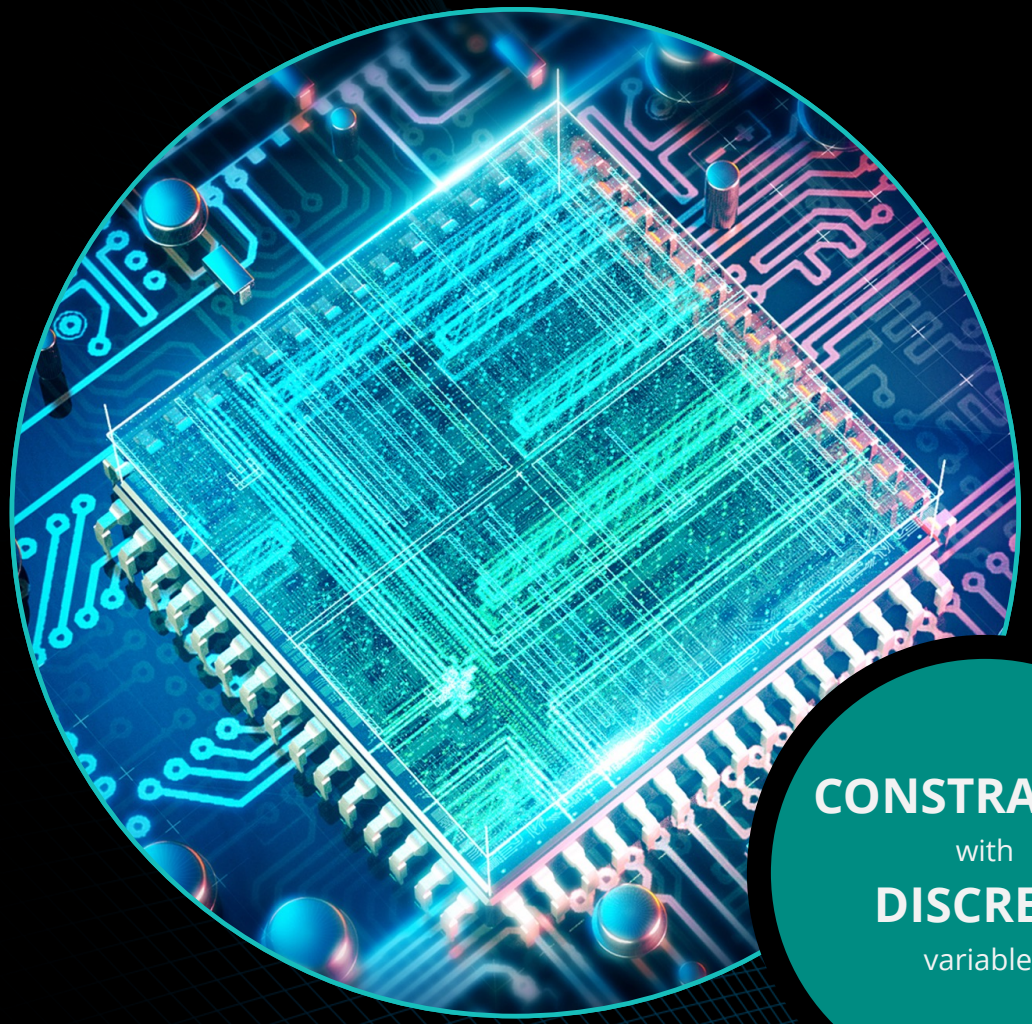
CPUs and GPUs

QPUs

COMPUTE RESOURCES

BUILDING QUANTUM APPLICATIONS

DISTRIBUTED COMPUTING



CONSTRAINED
with
DISCRETE
variables

GOAL: Efficiently assign tasks to computers in a distributed environment

CHALLENGES:

- Some tasks are related and require information passing
- Each computer is assigned the same number of tasks

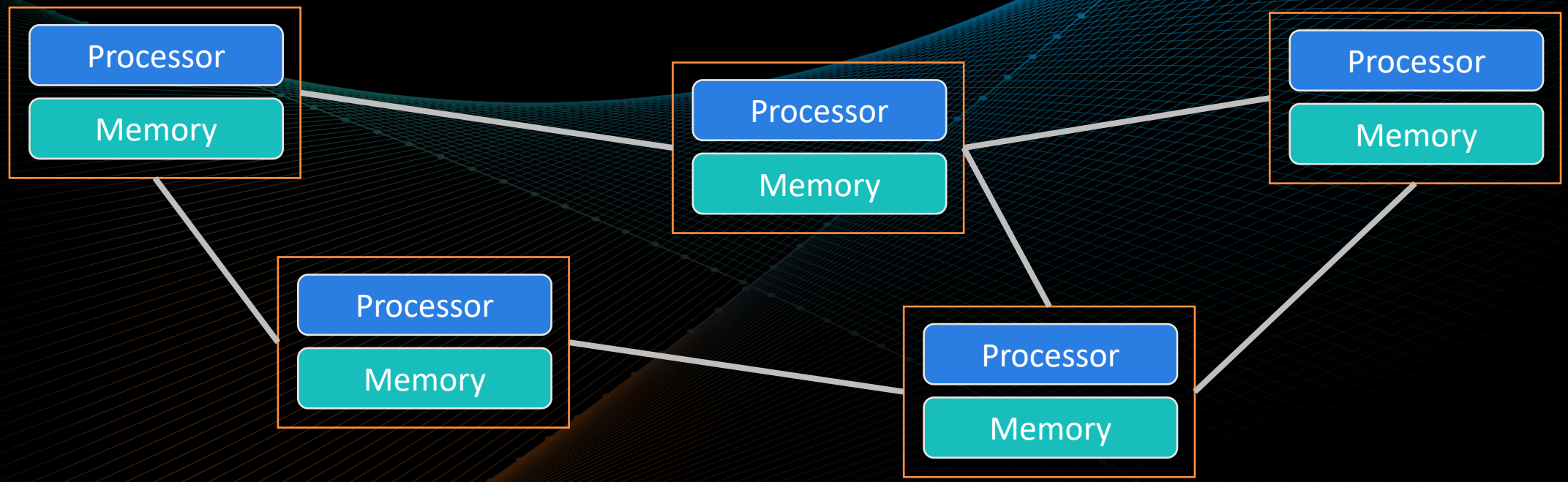
This is a challenging, NP-hard problem.

DISTRIBUTED COMPUTING

We have a network of computers and a set of operations to be performed.

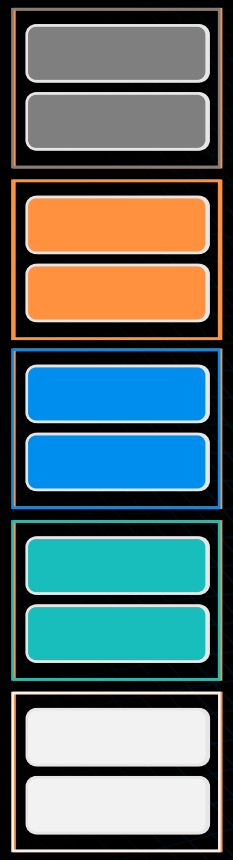
- The operations will be distributed across the computers
- The computers will need to pass messages to share inputs/outputs

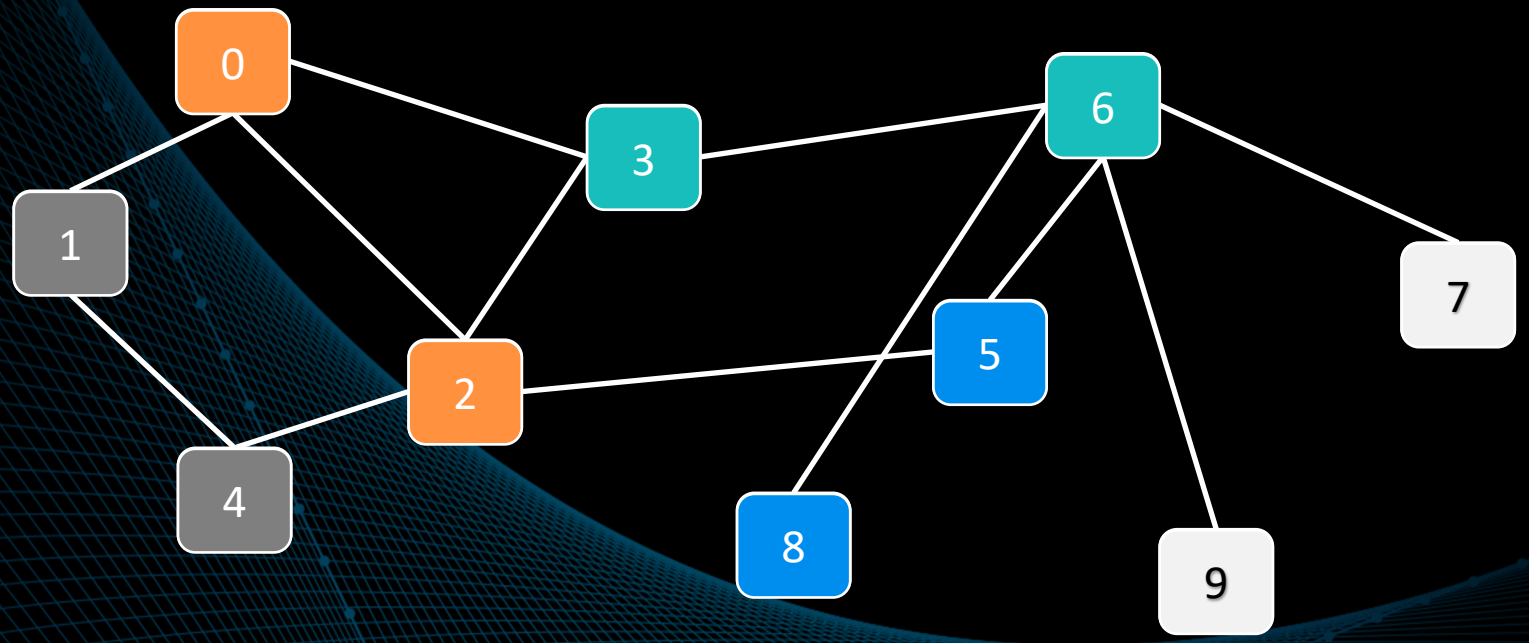
How can we minimize the message passing that's required, and evenly balance the workload?



GRAPH MODEL

Operations									
0	1	2	3	4	5	6	7	8	9





GRAPH MODEL

Graph:

- **Nodes:** operations
- **Edges:** message needs to be passed

Goal:

- Partition into k sets of nodes
- k = number of computers
- Partitions should be equal size
- Minimize number of edges between partitions

This is known as the graph k-partitioning problem.

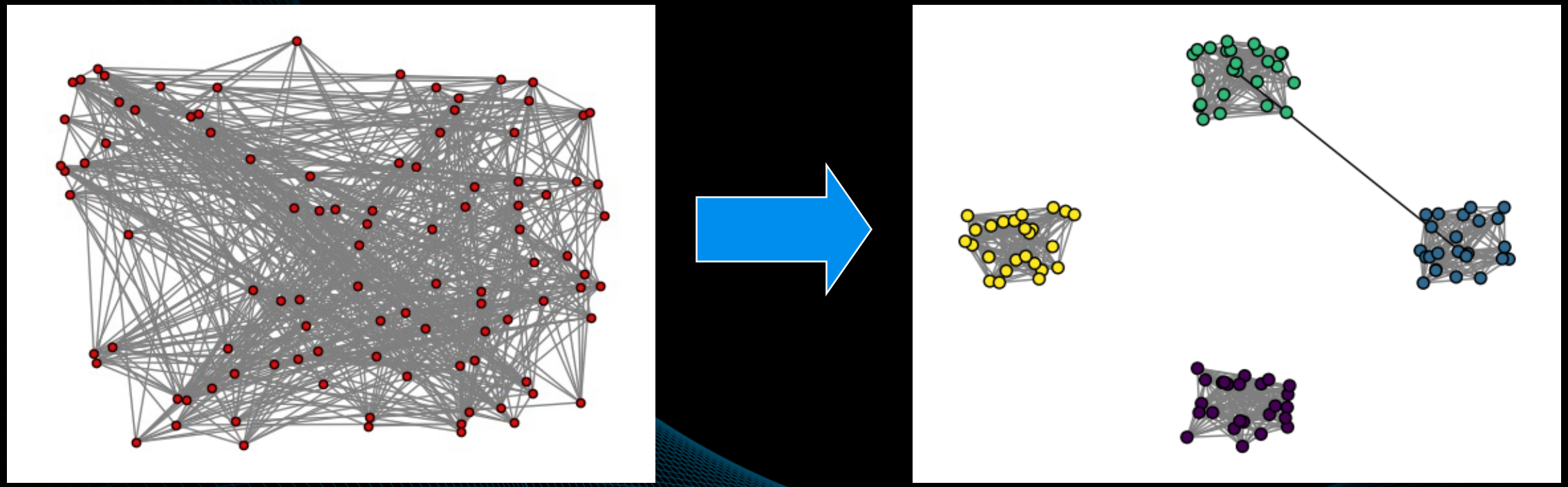
GRAPH MODEL POLL

Questions:

1. Why would having very different sized partitions be inefficient?
   - (a) Many more messages are passed between computers
   - (b) Memory access increases significantly across all computers
   - (c) Some computers have to complete many more tasks

2. What would happen if we maximized the number of edges between partitions?
   - (a) The system would overheat
   - (b) Inefficient ordering of tasks per computer
   - (c) Computers would have to wait for transmitted messages before finishing tasks

EXAMPLE INPUT / OUTPUT



PROBLEM FORMULATION

Input:

- Graph of nodes (*operations*) and edges (*messages*)
- k: number of partitions (*computers*)

Objective:

- Minimize the number of edges between partitions

Constraints:

- Partitions are equal sizes
- Each node is assigned to one partition

VARIABLES

We will use binary variables for this problem.

$$v_{i,p} = \begin{cases} 1, & \text{if node } i \text{ is in partition } p; \\ 0, & \text{otherwise.} \end{cases}$$

- Binary variables instead of integer allow us to build a constraint that the partitions have equal sizes.
- This is a standard method of encoding discrete variables using a set of binary variables

CONSTRAINTS

	Operations									
	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	1	0
3	0	0	0	1	0	0	1	0	0	0
4	0	0	0	0	0	0	0	1	0	1

- Each computer should have an equal number of tasks
- Each operation should be assigned to one computer

CONSTRAINTS

Constraints:

- Partitions are equal sizes

$$\sum_{i \in G} v_{i,p} = \frac{|G|}{k}, \qquad \text{for each partition } p$$

- Each node is assigned to one partition

$$\sum_{p \in [k]} v_{i,p} = 1, \qquad \text{for each node } i$$

OBJECTIVE

Objective Requirements:

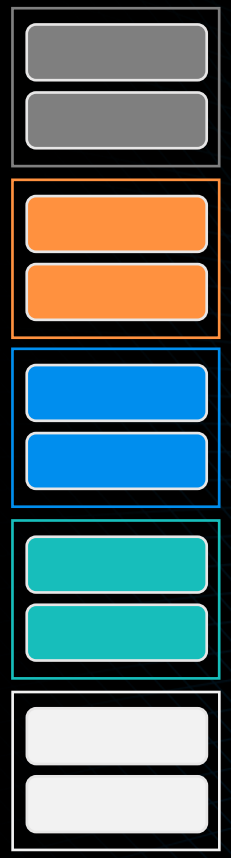
- Minimization expression
- Linear and quadratic terms

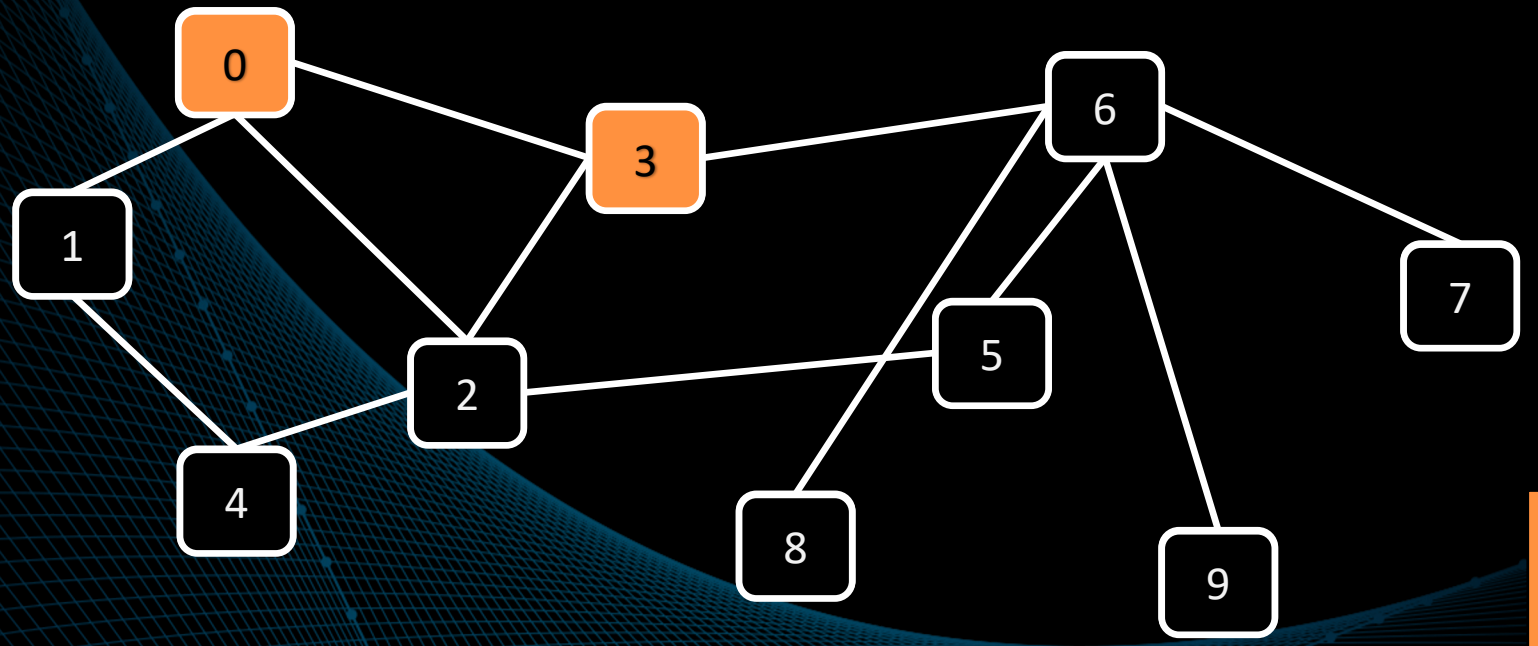
General Form:

$$\mathrm{Obj}(a_i, b_{ij}; q_i) = \sum_i a_i q_i + \sum_{i<j} b_{ij} q_i q_j$$

GRAPH MODEL

Operations									
0	1	2	3	4	5	6	7	8	9

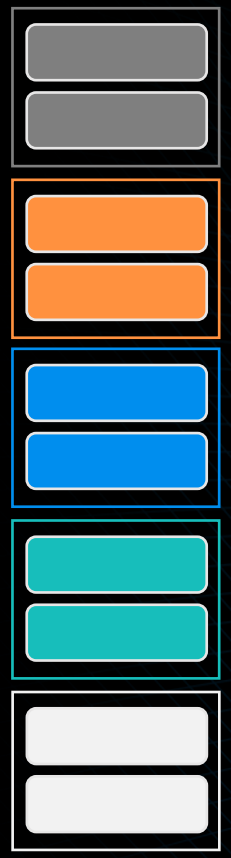


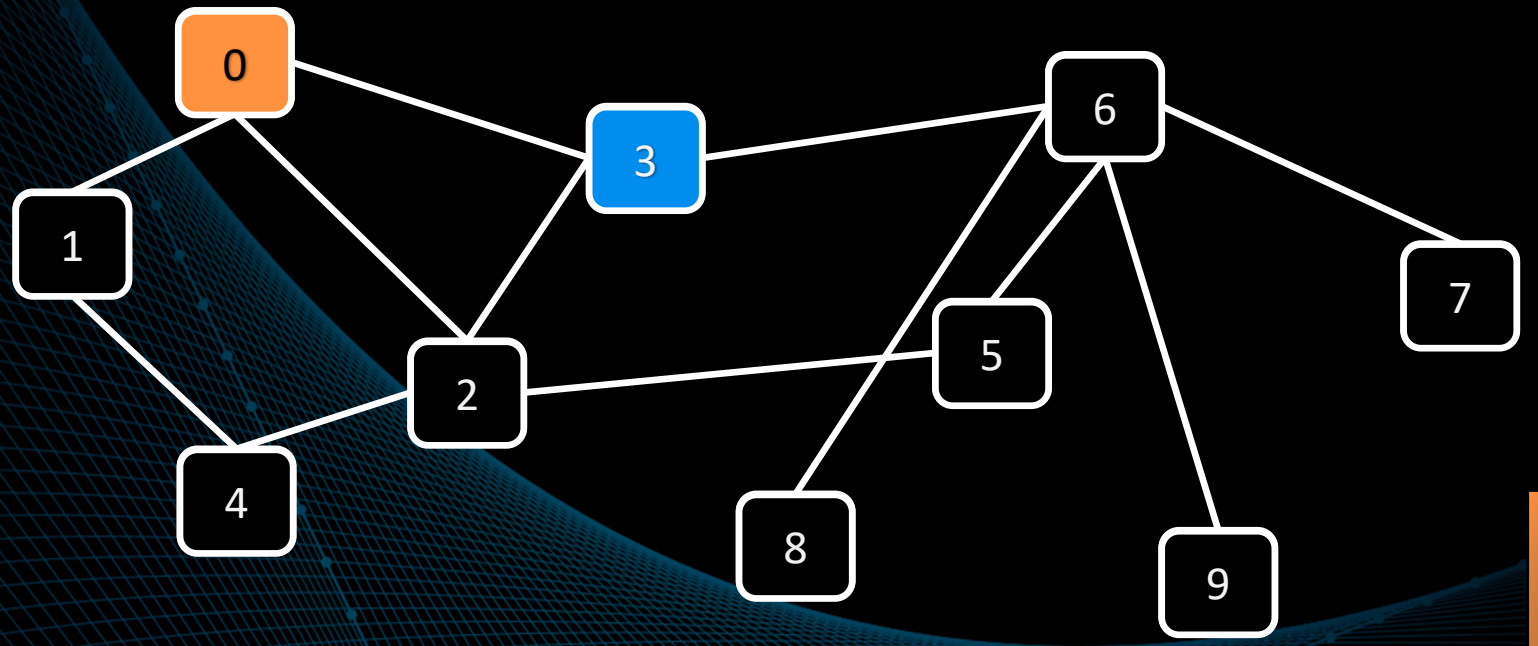


GOOD

GRAPH MODEL

Operations									
0	1	2	3	4	5	6	7	8	9





BAD

OBJECTIVE

Operation i	Operation j	
Orange	Orange	GOOD
Orange	Not Orange	BAD
Not Orange	Orange	BAD
Not Orange	Not Orange	GOOD

OBJECTIVE

$v_{i,p}$	$v_{j,p}$	
1	1	GOOD
1	0	BAD
0	1	BAD
0	0	GOOD

OBJECTIVE

$v_{i,p}$	$v_{j,p}$	Value
1	1	0
1	0	1
0	1	1
0	0	0

OBJECTIVE

We need our values to fit the general form:

$$a_i v_i + a_j v_j + b_{i,j} v_i v_j + C$$

v_i	v_j	Value	Equation
0	0	0	~~$a_i \cdot 0 + a_j \cdot 0 + b_{i,j} \cdot 0 \cdot 0 + C$~~ → $0 = C$
0	1	1	~~$a_i \cdot 0 + a_j \cdot 1 + b_{i,j} \cdot 0 \cdot 1 + C$~~ → $1 = a_j + C$ → $1 = a_j$
1	0	1	~~$a_i \cdot 1 + a_j \cdot 0 + b_{i,j} \cdot 1 \cdot 0 + C$~~ → $1 = a_i + C$ → $1 = a_i$
1	1	0	~~$a_i \cdot 1 + a_j \cdot 1 + b_{i,j} \cdot 1 \cdot 1 + C$~~ → $0 = a_i + a_j + b_{i,j} + C$ → $-2 = b_{i,j}$

OBJECTIVE

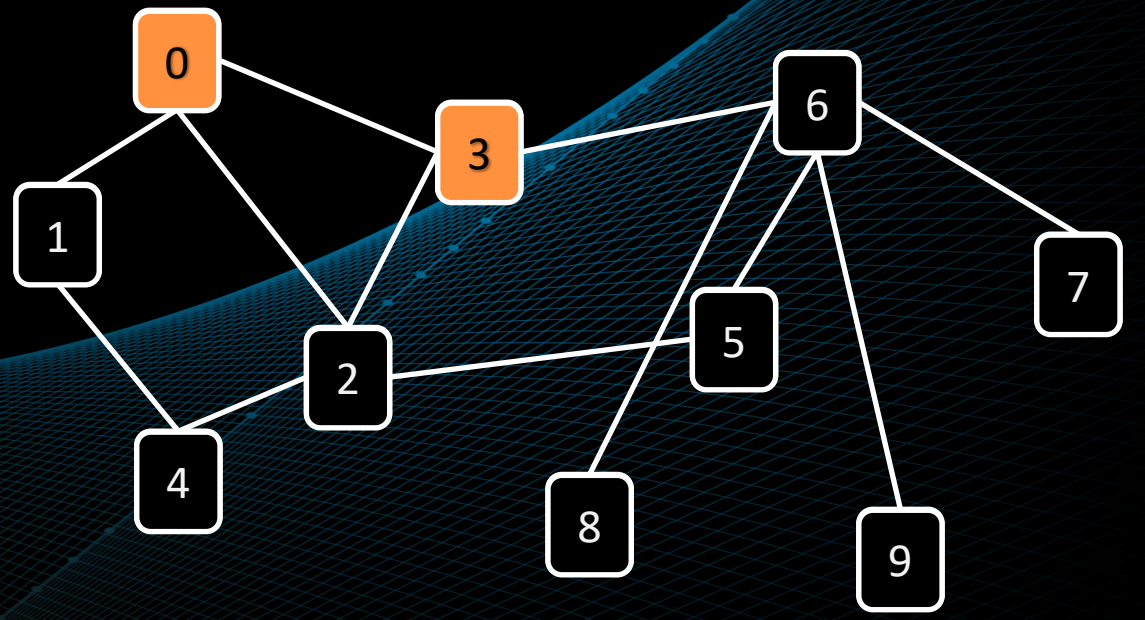
For one pair of operations and one computer:

$$\min(v_{i,p} + v_{j,p} - 2v_{i,p}v_{j,p})$$

For all pairs:

$$\min \sum_{(i,j)\in E} \sum_{p\in[k]} (v_{i,p} + v_{j,p} - 2v_{i,p}v_{j,p})$$

We only look at the pairs with a message (edge).



PROBLEM FORMULATION

Input:

- Graph of nodes (*operations*) and edges (*messages*)
- k: number of partitions (*computers*)

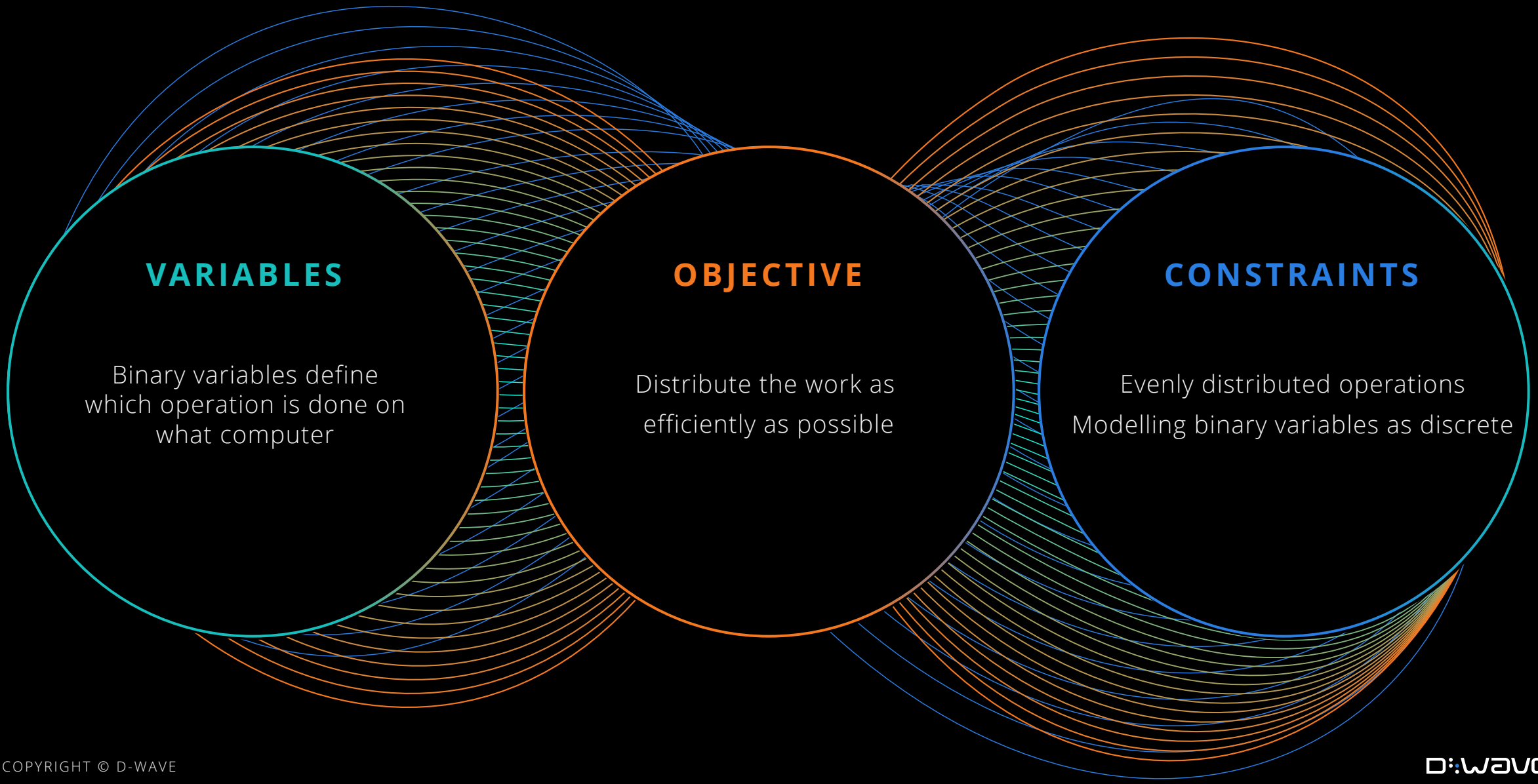
Objective:

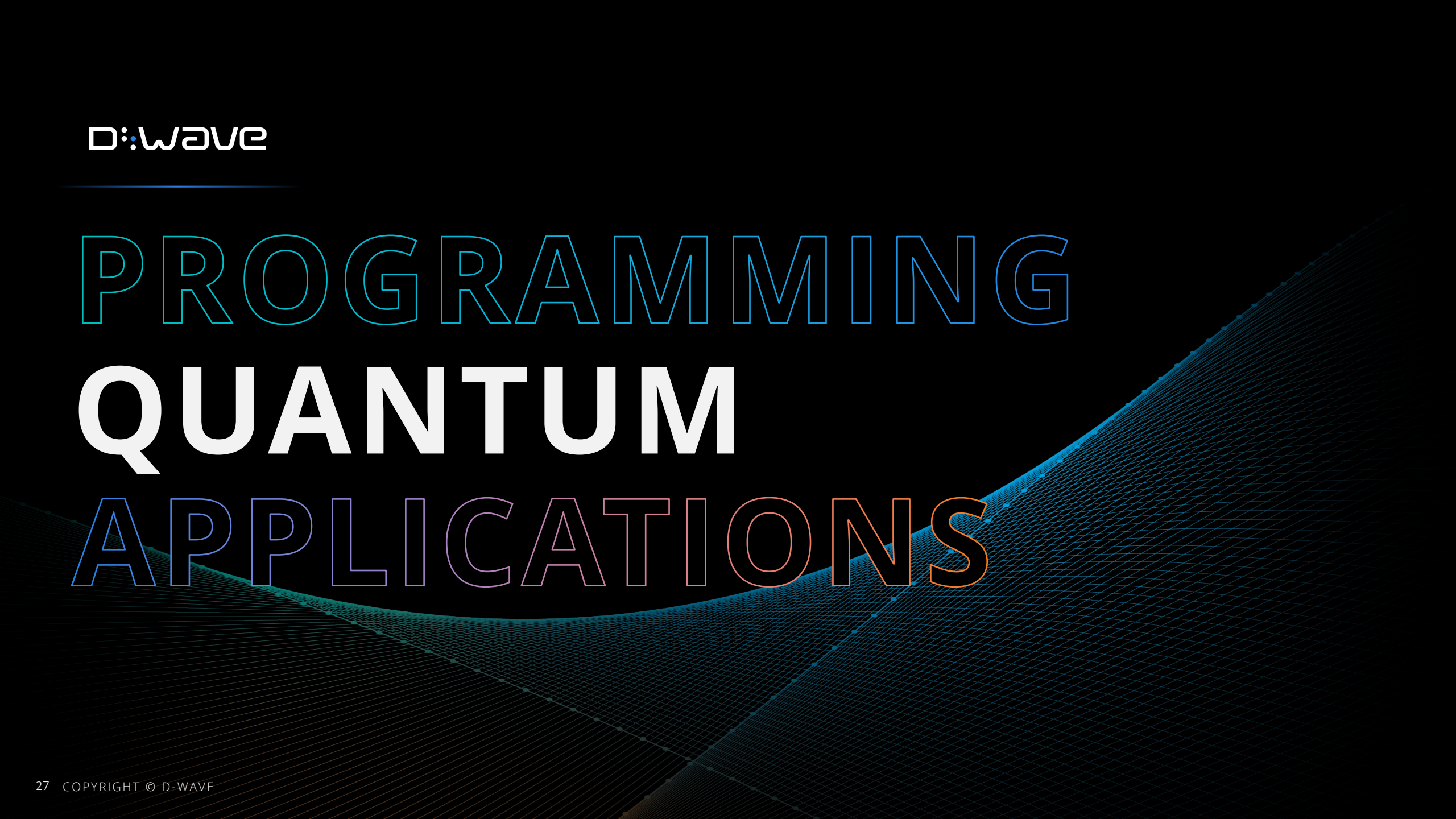
$$\min \sum_{(i,j)\in E} \sum_{p\in[k]} (v_{i,p} + v_{j,p} - 2v_{i,p}v_{j,p})$$

Constraints:

- Partitions are equal sizes: $\sum_{i\in G} v_{i,p} = \frac{|G|}{k}, \quad \text{for each partition } p$
- Each node is assigned to one partition: $\sum_{p\in[k]} v_{i,p} = 1, \quad \text{for each node } i$

KEY COMPONENTS





PROGRAMMING ESSENTIALS

Steps to solving a problem with D-Wave's technology:

1. Build problem with LEAP and Ocean tools
2. Connect to D-Wave solver
3. Sample problem using the solver
4. Interpret results

OCEAN: SOFTWARE DEVELOPMENT KIT

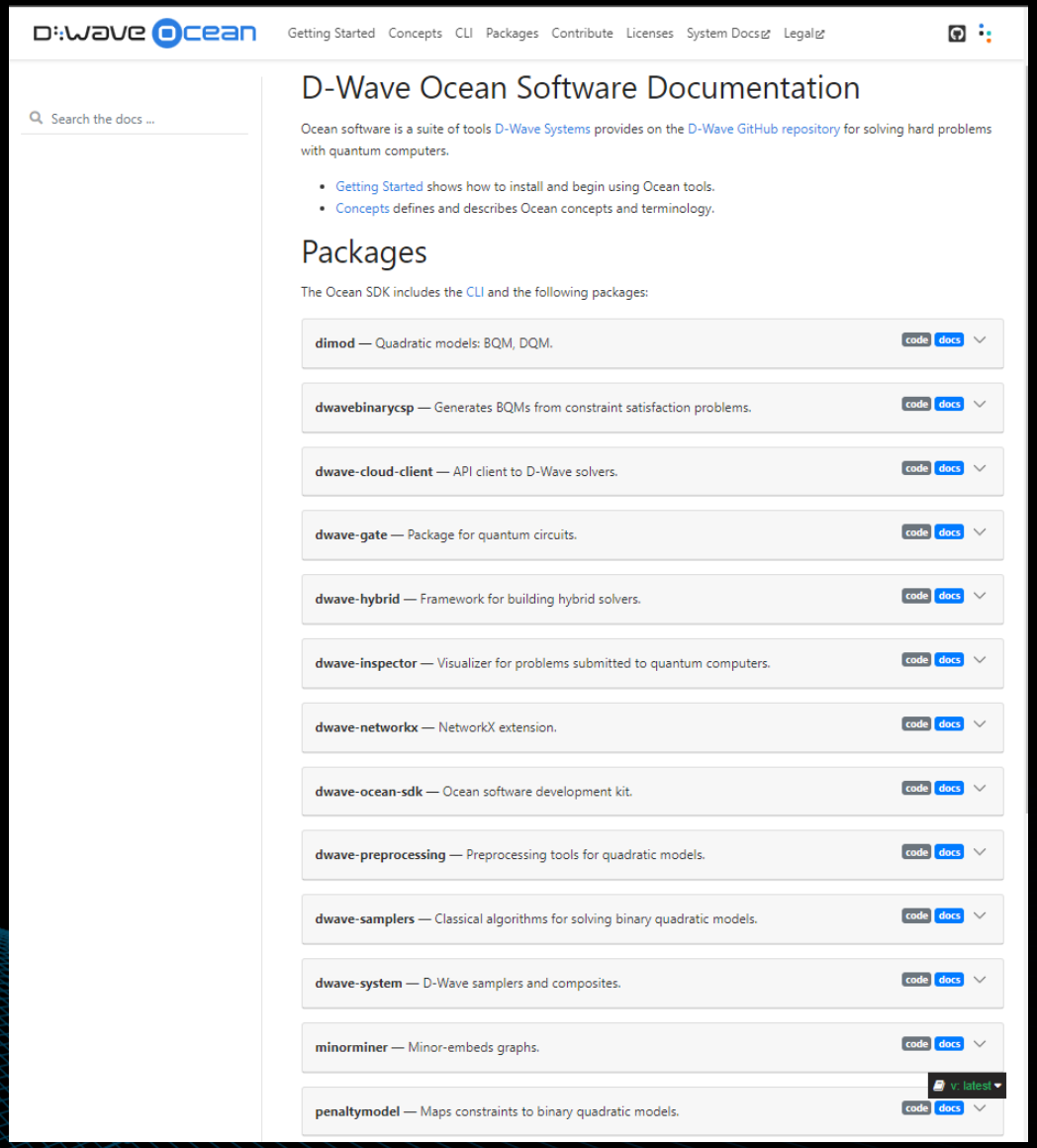
- D-Wave's open-source Python SDK
- Package of tools for
  - Accessing D-Wave's systems
  - Preprocessing data/models
  - Postprocessing solutions
  - Building quadratic models

Ocean SDK

https://github.com/dwavesystems

Ocean Documentation

https://docs.ocean.dwavesys.com/en/latest/index.html



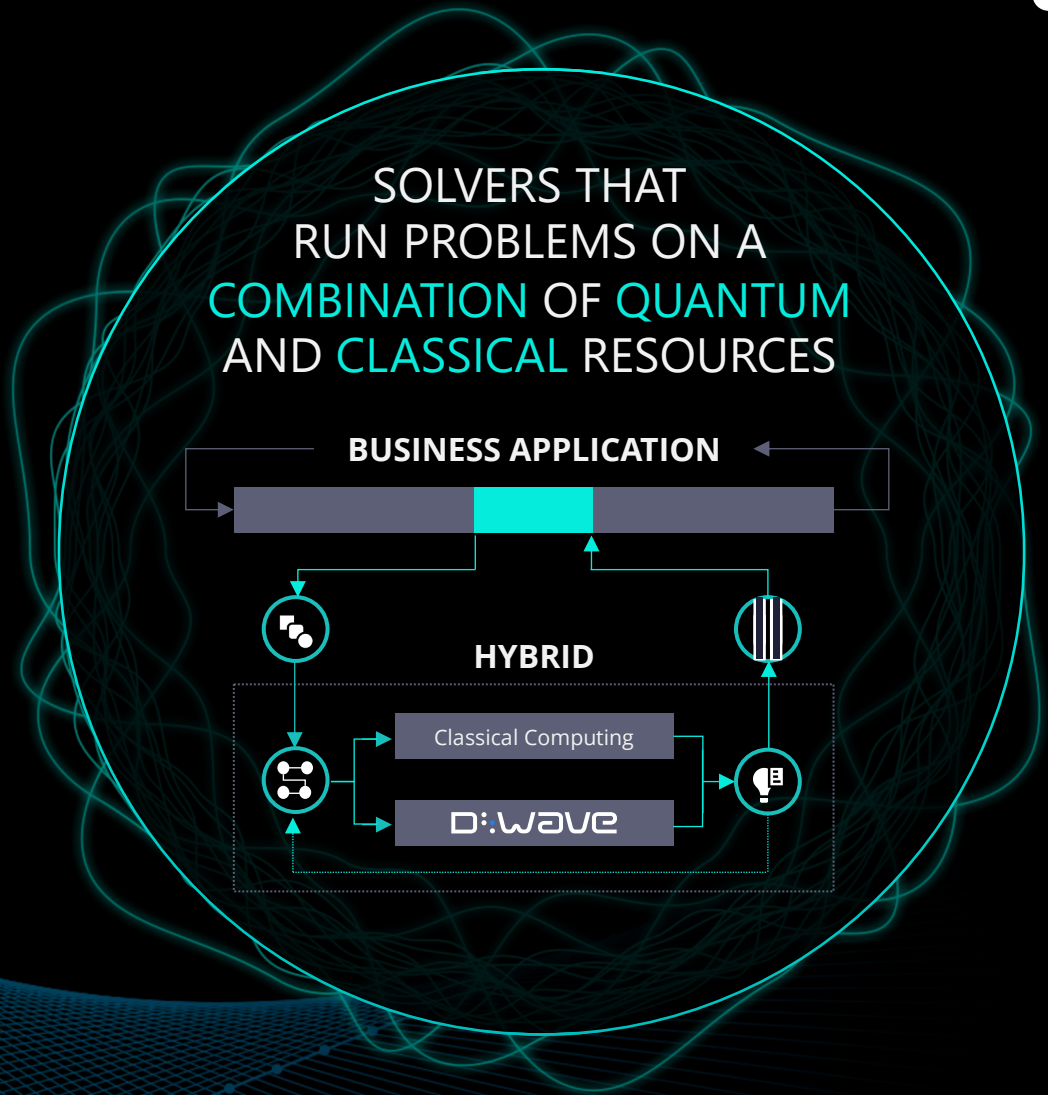
POWERFUL HYBRID SOLVERS

CONSTRAINED QUADRATIC MODEL SOLVER

- More native representation of problem
- Unlocks larger application problems
- Inequality and equality constraints
- Hard and weighted constraints
- Binary, integer, and real/continuous variables

BINARY QUADRATIC MODEL SOLVER

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



PROGRAMMING ESSENTIALS

Defining variables:

```
25    from dimod import Binary, ConstrainedQuadraticModel, quicksum
```

```
115       # Set up the partitions
116       partitions = range(k)
```

```
122       # Add binary variables, one for each node and each partition in the graph
123       print("\nAdding variables....")
124       v = [[Binary(f'v_{i},{p}') for p in partitions] for i in G.nodes]
```

PROGRAMMING ESSENTIALS

Setting up a model:

```
25    from dimod import Binary, ConstrainedQuadraticModel, quicksum
```

```
118       # Initialize the CQM object
119       print("\nBuilding constrained quadratic model...")
120       cqm = ConstrainedQuadraticModel()
```

PROGRAMMING ESSENTIALS

Defining objective function:

```
138     # Objective: minimize edges between partitions
139     print("\nAdding objective...")
140     min_edges = []
141     for i,j in G.edges:
142         for p in partitions:
143             min_edges.append(v[i][p]+v[j][p]-2*v[i][p]*v[j][p])
144     cqm.set_objective(sum(min_edges))
```

Objective:

$$\min \sum_{(i,j)\in E} \sum_{p\in[k]} (v_{i,p} + v_{j,p} - 2v_{i,p}v_{j,p})$$

PROGRAMMING ESSENTIALS

Defining assignment constraint:

```
126     # One-hot constraint: each node is assigned to exactly one partition
127     print("\nAdding one-hot constraints...")
128     for i in G.nodes:
129         # print("\nAdding one-hot for node", i)
130         cqm.add_discrete([f'v_{i},{p}' for p in partitions],
131                          label=f"one-hot-node-{i}")
```

Constraint:

Each node is assigned to one partition: $\sum_{p\in[k]} v_{i,p} = 1, \quad \text{for each node } i$

PROGRAMMING ESSENTIALS

Defining partition constraint:

```
25    from dimod import Binary, ConstrainedQuadraticModel, quicksum
```

```
133     # Constraint: Partitions have equal size
134     print("\nAdding partition size constraint...")
135     for p in partitions:
136         # print("\nAdding partition size constraint for partition", p)
137         cqm.add_constraint(quicksum(v[i][p] for i in G.nodes) == G.number_of_nodes()/k,
138                            label=f'partition-size-{p}')
```

Constraint:

Partitions are equal sizes: $\sum_{i \in G} v_{i,p} = \frac{|G|}{k}$, for each partition p

PROGRAMMING ESSENTIALS

Connecting to D-Wave hybrid solver:

```
26    from dwave.system import LeapHybridCQMSampler
```

```
293       # Initialize the CQM solver
294       print("\nOptimizing on LeapHybridCQMSampler...")
295       sampler = LeapHybridCQMSampler()
```

PROGRAMMING ESSENTIALS

Sampling the problem:

```
163     # Solve the CQM problem using the solver
164     sampleset = sampler.sample_cqm(cqm, label='Example - Graph Partitioning')
```

	v_0,0	v_0,1	v_0,2	v_0,3	v_1,0	v_1,1	v_1,2	...	v_99,3	energy	num_oc.
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	6.0	1
1	0.0	0.0	1.0	0.0	0.0	0.0	1.0	...	0.0	6.0	1
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	6.0	1
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	6.0	1
4	0.0	0.0	1.0	0.0	0.0	0.0	1.0	...	0.0	6.0	1

PROGRAMMING ESSENTIALS

Interpret results:

```
166     feasible_sampleset = sampleset.filter(lambda row: row.is_feasible)
167
168     # Return the first feasible solution
169     if not len(feasible_sampleset):
170         print("\nNo feasible solution found.\n")
171         return None
```

```
189     partitions = defaultdict(list)
190     soln = [-1]*G.number_of_nodes()
191
192     for node in G.nodes:
193         for p in range(k):
194             if sample[f'v_{node},{p}'] == 1:
195                 partitions[p].append(node)
196                 soln[node] = p
```

D-WAVE SOLVERS

Hybrid Solvers:

- Combines classical optimization and quantum annealing
- Supports binary, discrete, integer, and continuous variables
- ConstrainedQuadraticModel

Pure Quantum Solver:

- Only uses quantum annealing
- Supports binary or Ising variables
- Quadratic Unconstrained Binary Optimization (QUBO)

PROBLEM FORMULATION

Input:

- Graph of nodes (*operations*) and edges (*messages*)
- k: number of partitions (*computers*)

Objective:

$$\min \sum_{(i,j)\in E} \sum_{p\in[k]} (v_{i,p} + v_{j,p} - 2v_{i,p}v_{j,p})$$

Constraints:

- Partitions are equal sizes: $\sum_{i\in G} v_{i,p} = \frac{|G|}{k}$, for each partition p
- Each node is assigned to one partition: $\sum_{p\in[k]} v_{i,p} = 1$, for each node i

PROBLEM FORMULATION POLL

Questions:

1. For only two computers, how can we reduce the number of variables in our problem?
    - (a) By saying computers can't talk to each other
    - (b) By changing each variable to represent a binary assignment between the two computers
    - (c) By assigning some tasks to one computer by hand

2. If we change each variable to represent a binary assignment, which constraint is no longer necessary?
    - (a) Partitions are equal size
    - (b) Each node is assigned to one partition
    - (c) Both are unnecessary

PROBLEM FORMULATION

Input:

- Graph of nodes (*operations*) and edges (*messages*)
- k=2: number of partitions (*computers*)

Objective:

$$\min \sum_{(i,j)\in E} (v_i + v_j - 2v_i v_j)$$

Constraints:

- Partitions are equal sizes: $\sum_{i\in G} v_i = \frac{|G|}{2}$
- ~~Each node is assigned to one partition:~~

PROBLEM FORMULATION

Combine objective and constraints:

$$\min \sum_{(i,j)\in E} (v_i + v_j - 2v_i v_j) \; + \gamma \left(\sum_{i \in G} v_i - \frac{|G|}{2} \right)^2$$

Simplify:

$$\min \sum_{(i,j)\in E} (v_i + v_j - 2v_i v_j) \; + \gamma \sum_{i \in G} v_i^2 + \gamma \sum_{i \in G} \sum_{j>i} 2v_i v_j - \gamma |G| \sum_{i \in G} v_i + \gamma \frac{|G|^2}{4}$$

Simplify:

$$\min \sum_{(i,j)\in E} (v_i + v_j - 2v_i v_j) \; + 2\gamma \sum_{i \in G} \sum_{j>i} v_i v_j + (1 - |G|)\gamma \sum_{i \in G} v_i$$

QUBO FORMULATION

Input:

- Graph of nodes (*operations*) and edges (*messages*)
- k=2: number of partitions (*computers*)

Objective:

$$\min \sum_{(i,j)\in E} (v_i + v_j - 2v_i v_j) + 2\gamma \sum_{i\in G} \sum_{j>i} v_i v_j + (1 - |G|)\gamma \sum_{i\in G} v_i$$

PROGRAMMING ESSENTIALS

Steps to solving a QUBO problem with D-Wave's technology:

1. Build QUBO dictionary
2. Connect to D-Wave sampler
3. Sample problem using the solver
4. Interpret results

May need embedding!

Set parameters!

WHERE DO WE GO NEXT?

CONTACT ME: Request additional information.
cpotts@dwavesys.com

LEAP FREE SIGN-UP: Sign up for D-Wave Leap today to explore and get started.
https://cloud.dwavesys.com/leap/signup

EVENTS AND WEBINARS: Check out our upcoming events and webinars.
https://www.dwavesys.com/learn/events-and-webinars/

ADDITIONAL RESOURCES: Explore resources for developers. Videos, whitepapers, and more.
https://www.dwavesys.com/learn/resource-library

D-WAVE CAREERS: Watch out for new career opportunities at D-Wave.
https://www.dwavesys.com/company/careers