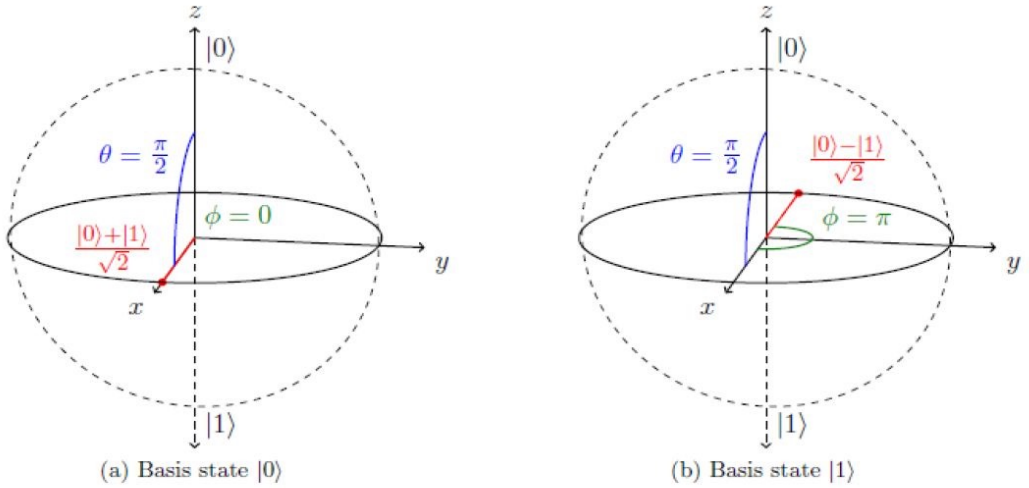
VICTORLAMP

煤油灯科技



《与当下工程师们讲量子计算》之05：D-Wave量子计算机编程

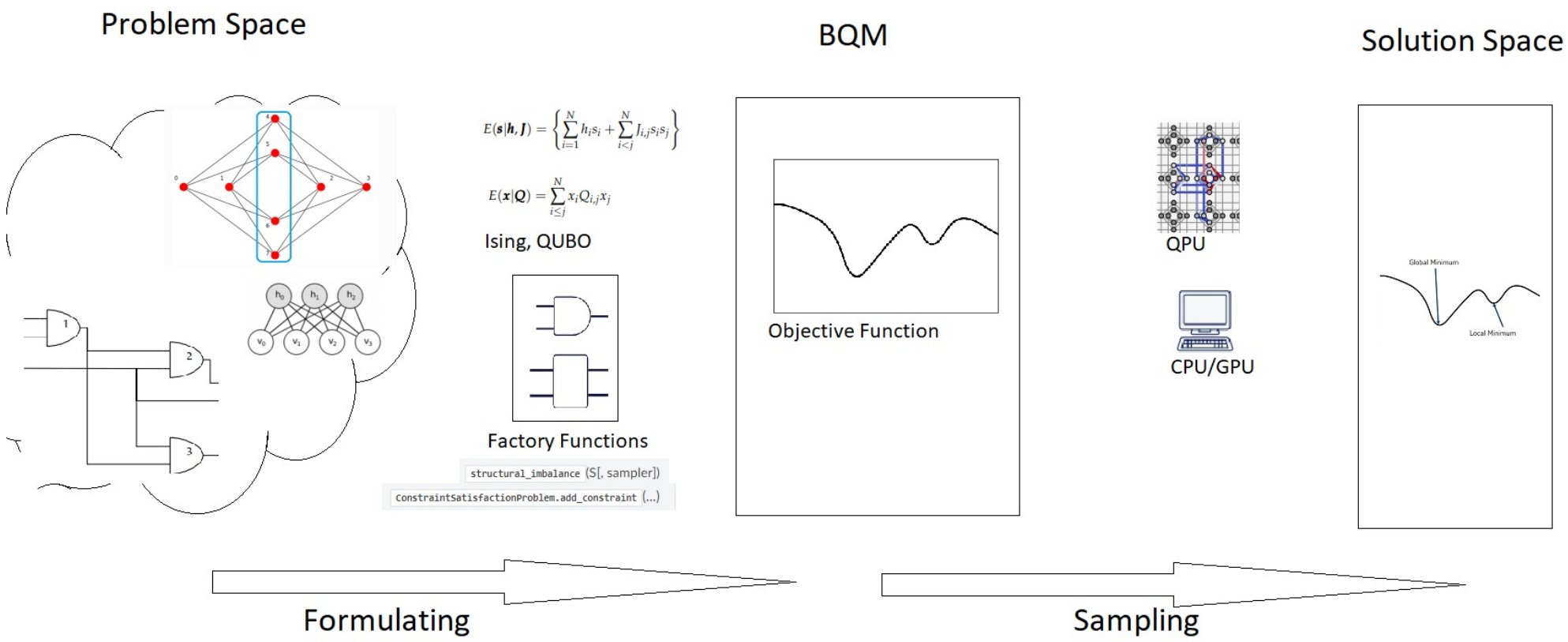
煤油灯科技公司 http://www.victorlamp.com

与当下工程师们计量子计算

多媒体课程：《与当下工程师们讲量子计算》

e休，爱编程的葫芦娃：exiu@victorlamp.com

这一节，我们讲解如何使用D-Wave量子计算机解决实际的问题?



两步骤:

第1步、把实际问题建模成Ising模型或者QUBO模型。

第2步、把模型映射（embedding）到量子计算机阵列D-Wave 2000Q Chimera graph 或D-Wave Advantage Pegasus graph 进行运算。

两个步骤+编程实践

第1步、把实际问题建模成Ising模型或者QUBO模型。

第2步、把模型映射（embedding）到量子计算机阵列D-Wave 2000Q Chimera graph 或D-Wave Advantage Pegasus graph 进行运算。

D-Wave量子计算机是用来解决离散最优化（或者叫整数规划）问题

Discrete optimization（离散优化问题），also known as combinatorial optimization（又称为整数规划），is the optimization of objective functions defined over search spaces consisting of a finite set of objects.

离散优化问题有哪些，比如：指派问题（Assignment problem）、旅行商问题（TravelingSalesmanProblem，TSP）、车间调度问题、运输问题、背包问题、设施选址问题、图划分问题（Graph Partitioning Problem）、图着色问题、聚类问题，等等。

Ising模型和QUBO模型都是用于解决离散优化的范畴。

- 定义在-1/+1符号串上的Ising问题
- 在0/1符号串上定义的二次无约束二进制优化（QUBO）问题

Any problem in the NP complexity class may be translated into this form.

《Ising formulations of many NP problems》February 2014，Andrew Lucas，Harvard University

很多的NP难问题都可以建模转化成 Ising模型， Andrew Lucas这篇文章里面介绍了如何把NP问题转化成Ising方程。

NP难问题用传统计算机没法解决，转化成Ising模型可以用量子计算机求解。

什么是Ising优化模型?

Ising问题是定义在一系列取值为-1/+1变量上的，Ising目标函数包含N个变量s=[s1，…，sN]，其中$s_i \in \{+1, -1\}$ 。

目标函数是一个二次方程

$$\min\left(\sum_i h_i s_i + \sum_i \sum_{j>i} J_{i,j} s_i s_j\right)$$

每个变量s_i对应于一个物理伊辛自旋，可以是+1或−1状态，每个自旋上都有一个局部应用场，使其倾向于+1或+1状态。符号和大小(系数） 即，局部场的值用hi表示。

自旋i和j之间也可能存在耦合，因此系统更希望自旋对定义的两组中的任何一组，位于$s_i = s_j$ （铁磁耦合）或$s_i = - s_j$ （反铁磁耦合)，这种偏好的符号和大小表示为$J_{i,j}$ 。

Ernst Ising其人

恩斯特·伊辛（Ernst Ising）于1900年出生于德国科隆。恩斯特·伊辛（Ernst Ising）的父亲是商人古斯塔夫·伊辛（Gustav Ising），母亲是谢克拉·洛维（TheklaLöwe）。
小时候放学后，他就去哥廷根大学和汉堡大学学习物理数学。

1922年，他开始在Wilhelm Lenz的指导下研究铁磁性。

1924年，在汉堡大学获得物理学博士学位。他的博士论文研究了Wilhelm Lenz建议的问题，线性磁矩链的特殊情况，这些磁矩只能有"上"和"下"两种情况，而且它们只与最近邻的磁矩发生耦合。

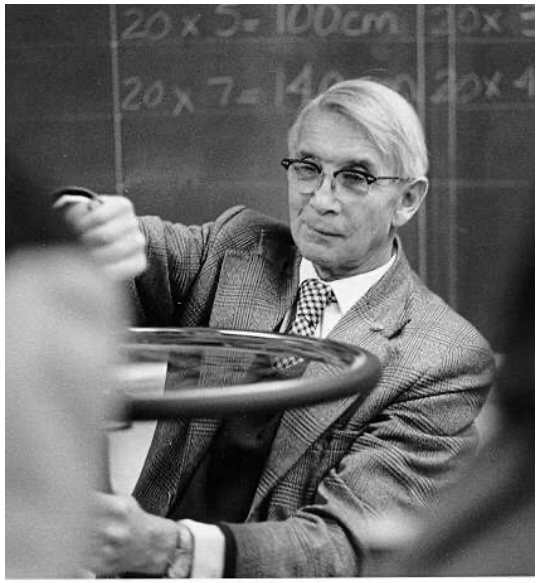
1930年，他与经济学家Johanna Ehmer博士结婚。

1931年希特勒上台，作为一个年轻的德国犹太科学家，伊辛被禁止教学和研究。

1934年，他在波茨坦附近的卡普斯犹太学校（这个学校招收被公立学校赶出的犹太学生）找到了一个职位，刚开始是老师，后来担任校长。恩斯特（Ernst）和他的妻子约翰娜·伊辛（Johanna Ising）博士，住在爱因斯坦家族着名的夏季住所附近的卡普斯（Caputh）。

1938年，Caputh的学校被纳粹摧毁。1939年，Ising逃到了卢森堡，在那里，Ising以作牧羊人和铁路工人维持生计。在德国游击队占领卢森堡后，恩斯特·伊辛被迫为军队服务。

1947年，伊辛一家移民到美国。虽然他在伊利诺伊州皮奥里亚州布拉德利大学（Bradley University）担任物理学教授，但从未再次发表过文章。 Ising于1998年在皮奥里亚的家中死亡，就在他98岁生日之后的一天。



BIOGRAPHY OF ERNST ISING
(1900 - 1998)
born May 10, 1900, Cologne , Germany.
died May 11, 1998, Peoria , Ill., USA.

什么是QUBO优化模型?

QUBO（Quadratic Unconstrained Binary Optimization），是D-Wave公司的专有名词， QUBO与Ising模型是等价的，可以互相转换。

QUBO 模型具有下面形式。二进制变量x_i和 x_j 取值 {0, 1}。

目标函数是一个二次方程

$$\min\left(\sum_i a_i x_i + \sum_i \sum_{j>i} b_{i,j} x_i x_j + c\right)$$

QUBO到Ising：要将QUBO模型转换为Ising模型，我们使用以下替换。

$$x_i \mapsto \frac{s_i + 1}{2}$$

Ising到QUBO：要将Ising模型转换为QUBO模型，我们使用以下替换。

$$s_i \mapsto 2x_i - 1$$

什么是BQM（Binary Quadratic Model，二进制二次方程模型）？

二元二次模型（BQM）问题由伊辛模型和二次无约束二元优化（QUBO）模型组成。

BQM只是一个融合的概念，包括Ising和QUBO，也是D-Wave专有名词。

融合之后，模型方程：BQM采用以下形式。系数a_i和$b_{i,j}$是我们用来定义问题的常数，常数项c也是。

目标函数是一个二进制二次方程 $$\min\left(\sum_i a_i v_i + \sum_i \sum_{j>i} b_{i,j} v_i v_j + c\right)$$

因为我们已经习惯于计算机的0、1表示方法，本课程接下来的讲解，我们主要使用QUBO来进行讲解，对于Ising模型是等价的。

QUBO建模应用示例1：2-变量模型实例

我们从一个简单的问题开始。我们有两个变量，设为v_1和v_2，我们需要一个表示约束$v_1 = v_2$的BQM。

第一步：建立约束满足真值表

限制条件： $v_1 = v_2$

意思也就是我的目标是要看看两个变量是否相等。

QUBO		Ising			
x_1	x_2	s_1	s_2	Equal	
0	0	-1	-1	Yes	0
0	1	-1	+1	No	1
1	0	+1	-1	No	1
1	1	+1	+1	Yes	0

结果中，我们将：

Yes表示为0

No表示为1

QUBO建模应用示例1：2-变量模型实例

第二步：建立一个方程组

QUBO方程: $a_1 x_1 + a_2 x_2 + b_{1,2} x_1 x_2 + c$,
因为是2-变量的二次方程，只能是这种形式。

根据上一页的真值表代入上面方程。

QUBO
$a_1 \cdot 0 + a_2 \cdot 0 + b_{1,2} \cdot 0 \cdot 0 + c = 0$
$a_1 \cdot 0 + a_2 \cdot 1 + b_{1,2} \cdot 0 \cdot 1 + c = 1$
$a_1 \cdot 1 + a_2 \cdot 0 + b_{1,2} \cdot 1 \cdot 0 + c = 1$
$a_1 \cdot 1 + a_2 \cdot 1 + b_{1,2} \cdot 1 \cdot 1 + c = 0$

第三步：求解方程组

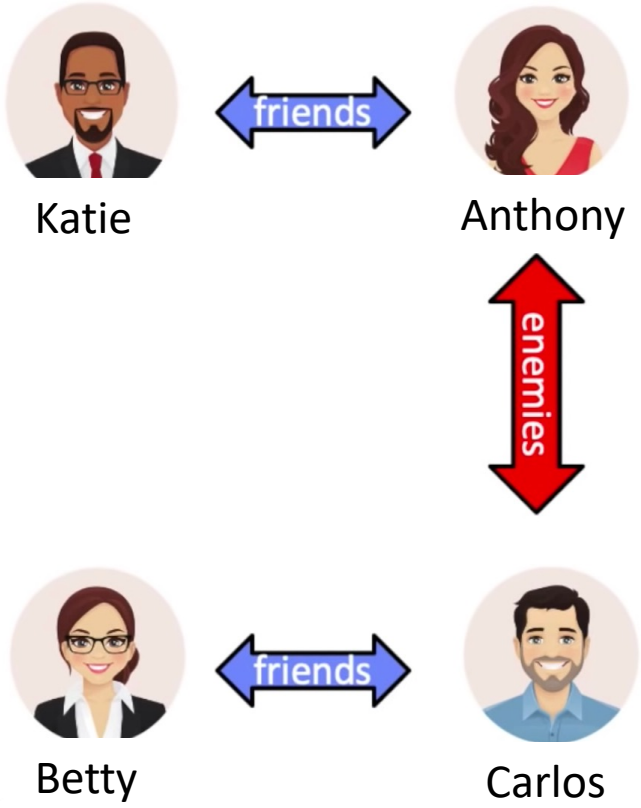
求解方程系数。

QUBO
$a_1 = 1$
$a_2 = 1$
$b_{1,2} = -2$
$c = 0$

最后，求得QUBO方程为: $x_1 + x_2 - 2 x_1 x_2$

这样就已经把实际问题转化成了一个QUBO模型方程。

QUBO建模应用示例2： Multiple Constraints多条件变量模型实例



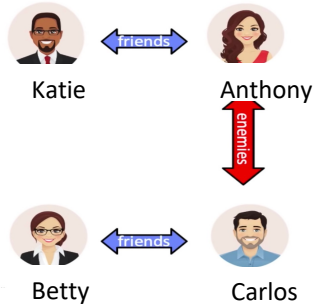
社交网络问题：

假设我们被要求将这些人分成两组。也许他们被分成两个团队，或者被分成小组进行一个项目。

目标要求是敌人不能在同一组里面，朋友要在一个组里。怎么分？有哪几种分法？

我们如何把这个问题映射到QUBO模型？

QUBO建模应用示例2： Multiple Constraints多条件变量模型实例



第一步、 问题建模

第一步是思考如何使用二进制变量来表示问题的答案。当我们考虑我们的问题时，我们要寻找的答案是每个人应该被分配到哪一组。如果我们称这些组为“A”和“B”，那么答案应该是这样的：

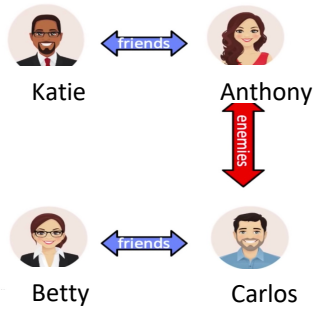
- Katie in Group A
- Anthony in Group A
- Carlos in Group B
- Betty in Group B

	Group A	Group B
QUBO	0	1
Ising	-1	+1

Katie	Anthony	Good/Bad?
A	A	Good
A	B	Bad
B	A	Bad
B	B	Good

Katie和 Anthony分组真值表

QUBO建模应用示例2： Multiple Constraints多条件变量模型实例



第二步、 建立QUBO模型，写出两两关系的QUBO方程，然后联合求解

2.1 朋友关系

对于我们的朋友关系，我们在前一节（两个变量的小问题）中看到，我们可以用以下BQM来表示这种关系（x1 = x2）。

QUBO 方程为: $x_1 + x_2 - 2\,x_1\,x_2$

	QUBO
Katie and Anthony	$x_A + x_K - 2x_A x_K$
Carlos and Betty	$x_B + x_C - 2x_B x_C$

2.2 敌人关系

对于我们的敌人关系，我们需要弄清楚什么是BQM模型的关系。我们使用的方法与我们在前面部分中两个变量BQM求解方法相同。

1）建立限制条件的真值表:

QUBO		Ising		
x_1	x_2	s_1	s_2	Different?
0	0	-1	-1	1
0	1	-1	+1	0
1	0	+1	-1	0
1	1	+1	+1	1

2）建立QUBO方程组

QUBO
$a_1 \cdot 0 + a_2 \cdot 0 + b_{1,2} \cdot 0 \cdot 0 + c = 1$
$a_1 \cdot 0 + a_2 \cdot 1 + b_{1,2} \cdot 0 \cdot 1 + c = 0$
$a_1 \cdot 1 + a_2 \cdot 0 + b_{1,2} \cdot 1 \cdot 0 + c = 0$
$a_1 \cdot 1 + a_2 \cdot 1 + b_{1,2} \cdot 1 \cdot 1 + c = 1$

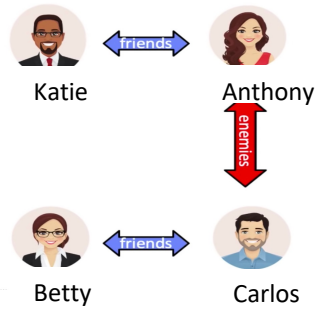
3）求解方程组

QUBO
$a_1 = -1$
$a_2 = -1$
$b_{1,2} = 2$
$c = 1$

求得敌人关系的QUBO方程:

$- x_1 - x_2 + 2\,x_1\,x_2 + 1$

QUBO建模应用示例2： Multiple Constraints多条件变量模型实例



2.3 合并前面朋友关系和敌人关系得出的三个方程式为一个方程

现在，我们为每个关系都有了一个QUBO，我们把它们加在一起，就得到了整个问题的QUBO。

$$(x_A + x_K - 2x_A x_K) + (x_B + x_C - 2x_B x_C) + (-x_A - x_C + 2x_A x_C + 1)$$

化简为如下：

$$x_B + x_K + 2x_A x_C - 2x_A x_K - 2x_B x_C + 1$$

朋友关系

	QUBO
Katie and Anthony	$x_A + x_K - 2x_A x_K$
Carlos and Betty	$x_B + x_C - 2x_B x_C$

\+ 敌人关系

QUBO: $-x_1 - x_2 + 2x_1 x_2 + 1$

也就是：$-x_A - x_C + 2x_A x_C + 1$

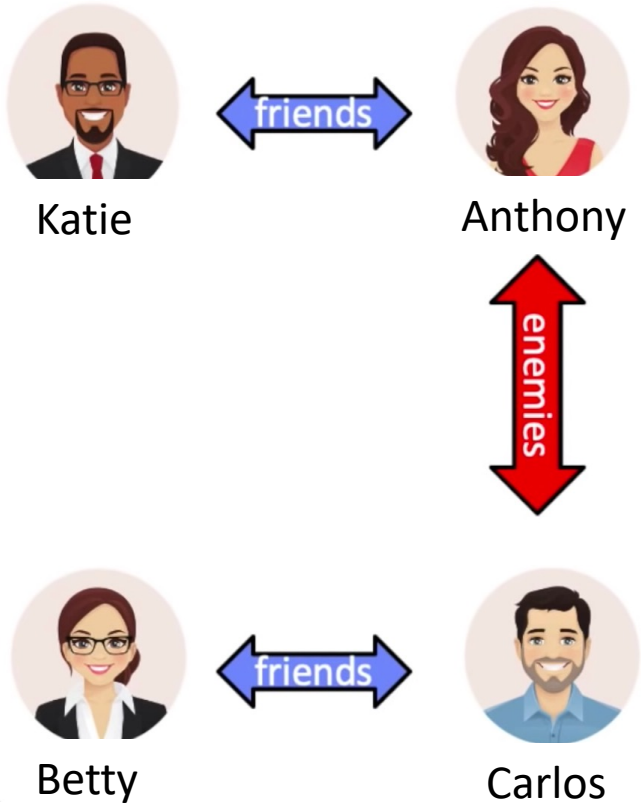
Anthony	Betty	Carlos	Katie	Value
A	A	A	A	1
A	A	A	B	2
A	A	B	A	1
A	A	B	B	2
A	B	A	A	2
A	B	A	B	3
A	B	B	A	0
A	B	B	B	1
B	A	A	A	1
B	A	A	B	0
B	A	B	A	3
B	A	B	B	2
B	B	A	A	2
B	B	A	B	1
B	B	B	A	2
B	B	B	B	1

找到了有两个目标结果为0的组合。

QUBO建模应用示例2： Multiple Constraints多条件变量模型实例

找到了有两个目标结果为0的组合。

Anthony	Betty	Carlos	Katie	Value
A	A	A	A	1
A	A	A	B	2
A	A	B	A	1
A	A	B	B	2
A	B	A	A	2
A	B	A	B	3
A	B	B	A	0
A	B	B	B	1
B	A	A	A	1
B	A	A	B	0
B	A	B	A	3
B	A	B	B	2
B	B	A	A	2
B	B	A	B	1
B	B	B	A	2
B	B	B	B	1



经过验证，找到的答案确实是符合条件的。

前面两个例子都是比较简单，用手工可以算出来，主要是为了让大家理解这个思路。如果变量的数量增加到一定的规模，如100个变量，手工是不可能算出来的，传统的计算机也算不出来，量子计算机可以在很短的时间内得出结果。
D-Wave量子计算机提供了相应的工具和模板给开发者来完成前面的步骤。后续章节，我们会用D-wave工具和模板来求解一个实际的问题。

QUBO建模：整个过程的回顾总结BQM Development Process

第一步：写下目标和约束条件

问题的目标是我们寻找最小化或最大化的东西。问题中的约束是我们必须遵循的规则。

例如，以前的社交网络问题可以用以下描述：

目标：无

约束1：朋友在同一团队中

限制2：敌人在不同团队中

第二步：将目标和约束转化为二进制数学表达式

第二步是考虑转化为二进制数学变量表达式。对于你的问题，你在寻找什么答案？你能问什么“Yes”或“No”的问题来给你答案？一旦我们用这些术语来思考我们的问题，我们就可以为每个问题分配一个二进制变量。

接下来，我们就可以使用这些二进制变量将目标和约束转化为数学表达式。

QUBO建模：整个过程的回顾总结BQM Development Process

第三步：将数学表达式转换为QUBO

不同类型的表达需要不同的处理策略。如果您在步骤二中使用了真值表方法，您可能不需要进行任何调整！下面是一些常见的转换。

1）平方项的处理：我们的QUBO和Ising模型没有空间容纳平方二元变量。事实证明，我们实际上并不需要它们。如QUBO中的二进制变量是0和1。请注意，0^2=0和1^2=1，因此我们可以用x替换任何项x^2。

2）从最大化变换到最小化。如果我们的目标函数是一个最大化函数（例如，你可能是利润最大化），我们可以通过将整个表达式乘以-1将其转化为一个最小化。

$$\arg\max(3v_1 + 2v_1v_2) = \arg\min(-3v_1 - 2v_1v_2)$$

3）从相等变换到最小化。如果你有一个等式约束，我们可以将它转换成一个最小化表达式，方法是将所有参数和常数移到等式的一侧，并将等式的非零侧平方。这将给我们留下一个表达式，它满足于它的最小值（0），而不满足于任何更大的值（0）。

Example:

Original constraint:

$$v_1 + v_2 = 1$$

Move all arguments and constants to one side:

$$v_1 + v_2 - 1 = 0$$

Square the expression:

$$(v_1 + v_2 - 1)^2$$

变成了求这个平方的最小值

QUBO建模：整个过程的回顾总结BQM Development Process

第四步：组合多个方程式

一旦我们将所有组件（目标和约束）写成QUBO表达式，我们就可以通过将所有组件的方程相加来生成最终的QUBO 。

在这一步中，如果需要，我们可以选择对组件的方程进行称重加权。该加权使用拉格朗日参数进行。

通常，我们会为每个约束指定单独的拉格朗日参数，以对有效的解决方案（满足所有约束的解决方案）进行优先级排序。

$$QUBO = \min(objective + \gamma(constraint))$$

QUBO建模：整个过程的回顾总结BQM Development Process

γ 参数如何选取?
例如：假设我们有两个公司可以完成相同的工作，每个公司的价格不同。公司1可以以12美元完成这项工作，而公司2可以以8美元完成这项工作。我们希望以最小的成本完成这项工作。

$$\min\left((12x_1 + 8x_2) + \gamma(x_1 + x_2 - 1)^2\right)$$

如果我们选择γ=1，我们可以看到以下QUBO值。

x_1	x_2	Value
0	0	1
0	1	8
1	0	12
1	1	21

两个公司都没有选上，显然这不是我们想要的结果。

如果我们选择γ=20

x_1	x_2	Value
0	0	20
0	1	8
1	0	12
1	1	40

调整γ = 20，现在可以选取唯一的结果。

请注意，您可能需要尝试几个不同的值，以确定特定BQM的最佳拉格朗日参数值。一个好的起始值是将γ设置为目标函数值的最佳估计值。如果您发现返回的解决方案不满足您的约束，您可能需要增加γ。另一方面，如果您的约束都得到了满足，但您的解决方案并不接近最优，那么您可能需要减少γ。
一般来说，选项结果太少了，增大γ值，选项结果太多了，减小γ的值。

补充： 把QUBO模型方程用矩阵表示，便于通过软件工具输入到量子计算机处理

QUBO模型采用以下形式：

$$\min\left(\sum_i a_i x_i + \sum_i \sum_{j>i} b_{i,j} x_i x_j + c\right)$$

二进制变量x_i和x_j可以从{ 0, 1 }取值。
当我们准备使用D-Wave Ocean软件工具包运行QUBO时，一种方法是将QUBO存储为Ocean SDK的矩阵格式。我们可以通过在对角线上放置线性项，在非对角线位置放置二次项，将QUBO映射到矩阵。

举例：我们映射方程 $x_1 + 2x_2 + 3x_3 + 12x_1x_2 + 13x_1x_3 + 14x_2x_4$ 成下面的矩阵，在这个矩阵中，我们将系数a_i放置在矩阵位置（i，i），将系数$b_{i,j}$放置在矩阵位置（i，j）。

$$\begin{bmatrix} 1 & 12 & 13 & 0 \\ 0 & 2 & 0 & 14 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

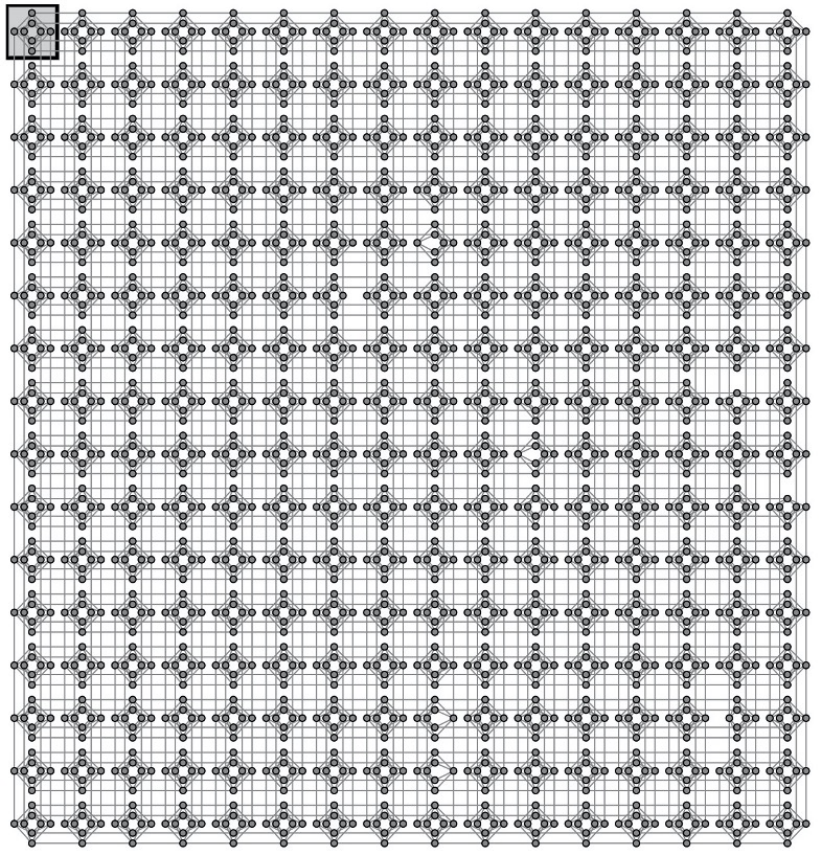
编程的时候，我们是把这些参数通过一个矩阵表示，输入到量子计算机运行。

两个步骤+编程实践

第1步、把实际问题建模成Ising模型或者QUBO模型。

第2步、把模型映射（embedding）到量子计算机阵列D-Wave 2000Q Chimera graph 或D-Wave Advantage Pegasus graph 进行运算。

D-Wave 2000Q Chimera graph

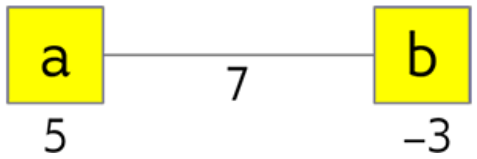


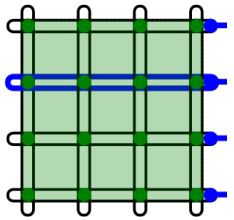
把模型映射（embedding）到量子计算机阵列D-Wave 2000Q Chimera graph

把QUBO方程映射成Chimera图
图由节点和边的集合组成，这些节点和边可以分别用来表示目标函数的变量以及它们之间的联系。
例如，要表示一个二次方程，

$$H(a,b)=5a+7ab-3b$$

你需要两个节点a及b, 偏差为5和−3，以及它们之间强度为7的边缘，如下图所示。





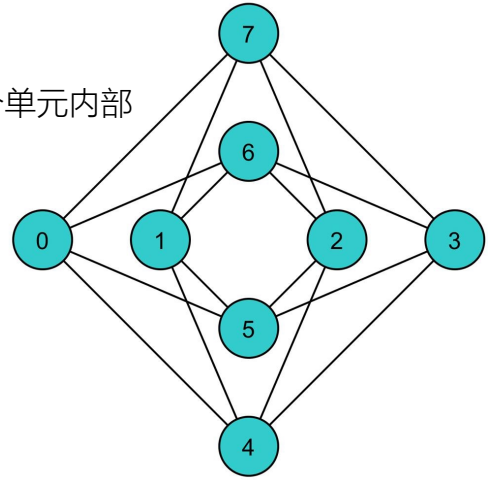
此图形表示意味着您可以将目标函数映射到QPU：

- 表示目标函数变量的节点，例如$\mathrm{s_i}$（Ising）或$\mathrm{q_i}$（QUBO）被映射到QPU上的量子位。（变量--》量子位）
- 表示目标函数二次系数的边，例如$\mathrm{J}_{i,j}$（Ising）及$b_{i,j}$（QUBO）被映射到耦合器。（系数--》耦合器）

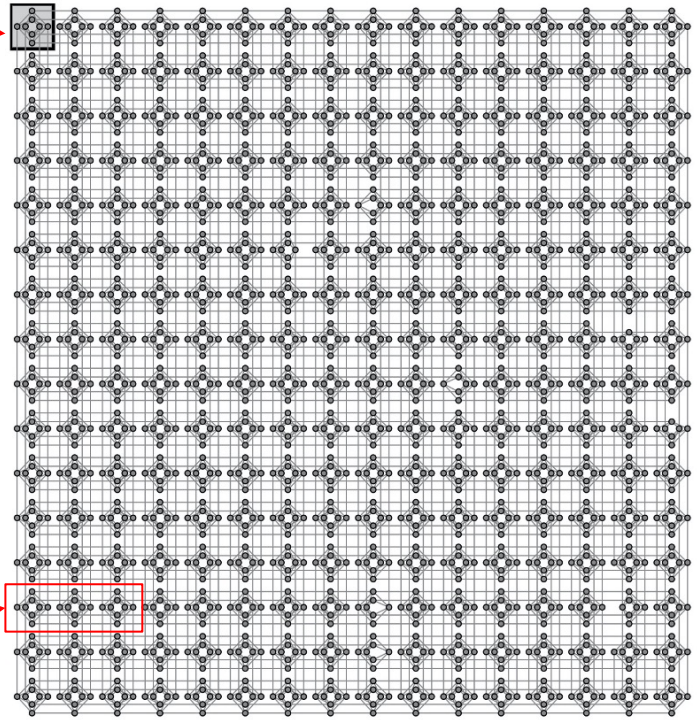
将问题方程中的变量映射到QPU上的量子位的过程称为次要嵌入（Minor Embedding）。

量子计算机阵列D-Wave 2000Q Chimera graph

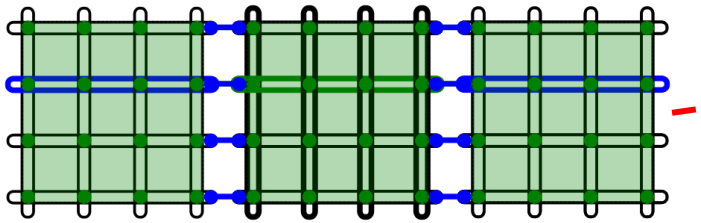
一个单元内部



Chimera graph内部结构



多个单元连接



把QUBO模型映射到量子计算机示例： traveling salesperson problem

优化旅行商在一系列城市的路线时，需要遵从一个约束，迫使旅行商在旅行的每个步骤恰好在一个城市：将旅行商同时停留在两个或多个地方的解决方案是无效的。

精确一个真约束是一个布尔可满足性问题，给定一组变量，当精确一个变量为真时（等于1）。
本章研究具有唯一真值约束的三个变量问题。

Fig. The traveling salesperson problem is an optimization problem that can be solved using exactly-one-true constraints. Map data © 2017 GeoBasis-DE/BKG (© 2009), Google.

把QUBO模型映射到量子计算机示例： Formulating an Objective Function

建立目标方程

对于少量二进制变量的问题（在本例中为三个：a, b, 及c），您可以将所有可能的排列列成表格，并确定一个变量为1，另两个变量为0时的状态。您可以使用真值表来执行此操作：

a	b	c	Exactly One is True
0	0	0	FALSE
1	0	0	TRUE
0	1	0	TRUE
1	1	0	FALSE
0	0	1	TRUE
1	0	1	FALSE
0	1	1	FALSE
1	1	1	FALSE

a、b、c 三个地点作为例子

请注意，真值表中满足约束的三种状态的共同点是变量之和等于1，因此可以将约束数学表示为方程

$$a+b+c=1$$

$$E(a,b,c)=(a+b+c-1)^2.$$

展开平方项，同时记住，对于值为0或1的二元变量，变量的平方就是它本身，X^2=X以显式形式显示目标函数的二次项和线性项。

$$\begin{aligned}E(a,b,c) &= a^2+ab+ac-a+ba+b^2+bc-b+ca+cb+c^2-c-a-b-c+1\\ &= a^2+b^2+c^2+2ab+2ac+2bc-2a-2b-2c+1\\ &= 2ab+2ac+2bc-a-b-c+1,\end{aligned}$$

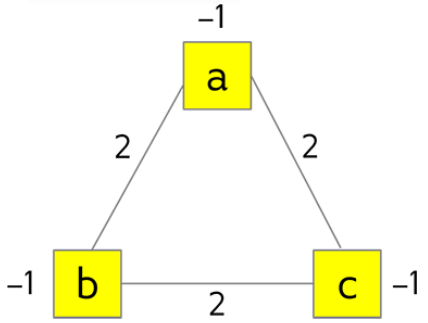
（这一步用到了 $X^2=X$）

请注意，这个目标公式与三个变量的QUBO格式相匹配。

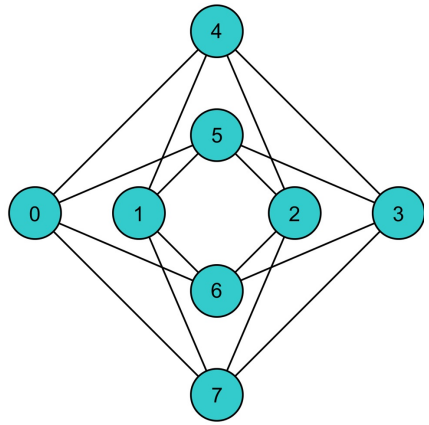
把QUBO模型映射到量子计算机示例： Embedding a graph into Chimera by using a chain

QUBO方程：$E(a, b, c) = 2ab + 2ac + 2bc - a - b - c + 1,$

转化成图：

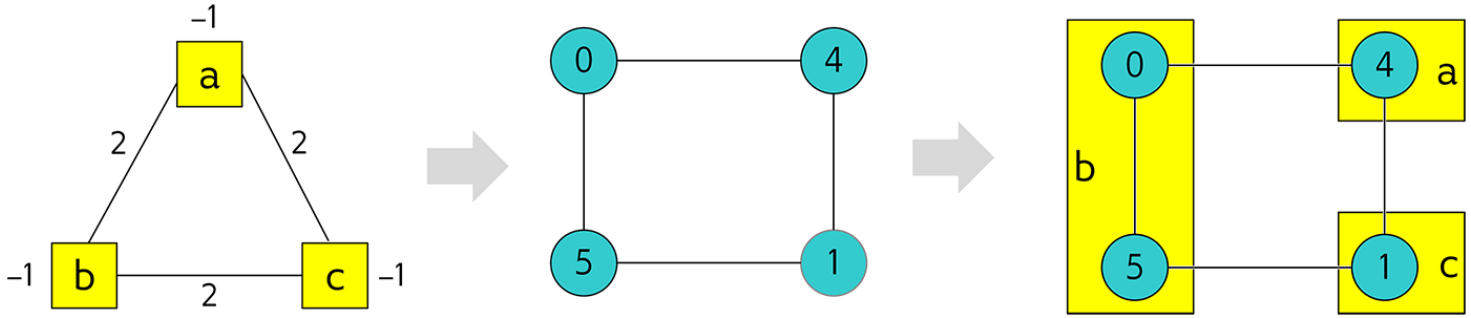


Triangular graph for an exactly-one-true constraint with its biased nodes and edges



Chimera unit cell.

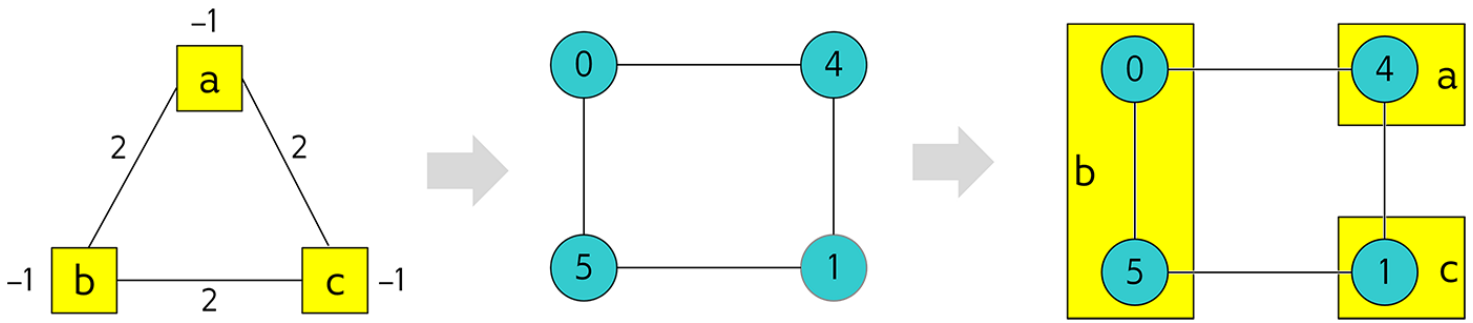
使用链将三角形图嵌入到Chimera中。用多个相邻的Qubits表示一个二进制位。



链的概念是用两个或多个量子位的联合在一起表示一个变量比如这里的b实际上表示的是一个变量，用两个量子位0和5表示变量b, 它们之间的连接器强度必须设置为足够负。

把QUBO模型映射到量子计算机示例：重新设置每个节点和边的权重

完成映射之后，怎么设置每个节点和边的权重系数？



将量子位0和5连接起来表示变量b需要在它们之间添加强负耦合强度。步骤：

1）在量子位元0和5之间平均地分配来自变量b的−1。现在这两个量子位的偏差是−0.5.

2）为量子比特0和5之间的链选择强负耦合强度。这个例子任意选择−3因为它比周围耦合器的值更强大。

3）补偿在第2步中的−3，通过向量子比特0和5的每个偏压添加-(-3/2)=1.5。现在这些量子位的偏压是1。

Edge	Quadratic Coefficient	Coupler	Strength
(a,b)	2	(0,4)	2
(a,c)	2	(1,4)	2
(b,c)	2	(1,5)	2
		(0,5)	-3

比周边任何一个都小，取负号，所以得-3。

边的权重系数

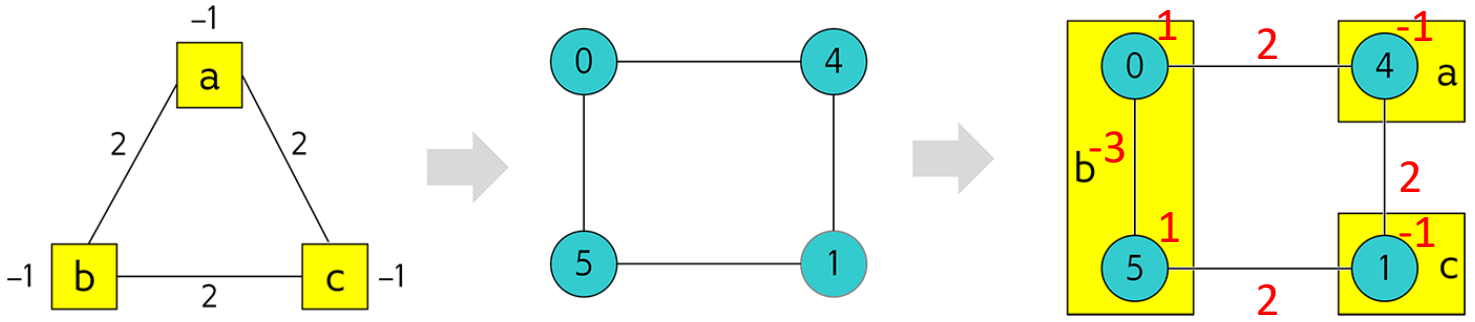
Node	Linear Coefficient（原本参数）	Qubits	Bias（计算出来的权重）
a	-1	4	-1
b	-1	0, 5	1, 1
c	-1	1	-1

节点系数补偿边的负向权重。

节点权重系数

把QUBO模型映射到量子计算机示例：设置量子计算机，然后运行出来结果， unembedded

现在我们就可以按照这些参数设置量子计算机，然后运行，出来结果看看。



提交运行：将该问题提交给D-Wave 2000Q系统进行1000次退火的结果。

Energy	Qubit				Occurrences
	0	5	4	1	
-1.0	0	0	1	0	206
-1.0	0	0	0	1	526
-1.0	1	1	0	0	267
0.0	1	1	0	1	1

基态能量是−1，不是零。因为从目标函数中减去常数+1。

$$E(a,b,c) = 2ab + 2ac + 2bc - a - b - c + 1,$$

Unembedded反映射求出结果，三种结果。

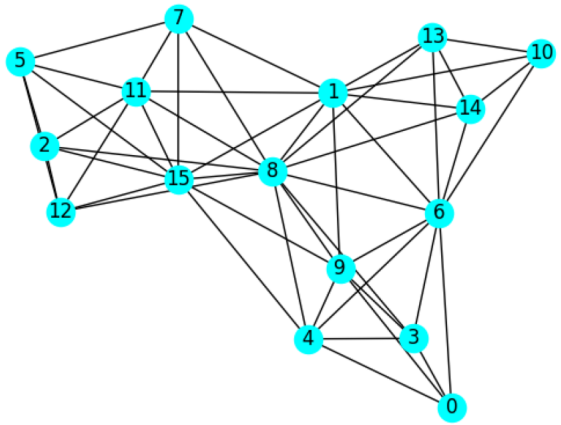
对于这个单链的简单例子，解除嵌入包括将量子位4，1映射到变量a,c 量子位元0，5到变量b. 上表中的结果解除嵌入后：

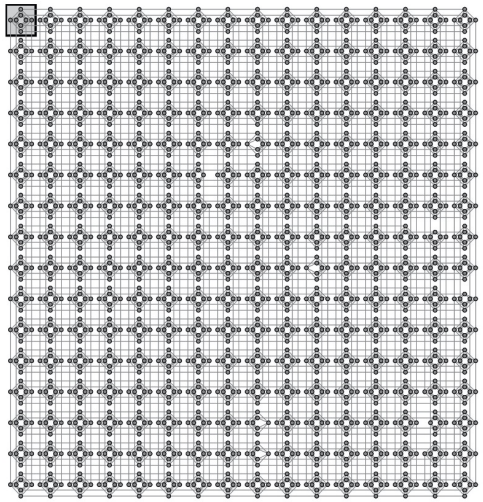
- 第1行：解决方案(a,b,c)=（1,0,0）能量−1，发现206次。
- 第2行：解决方案(a,b,c)=（0,0,1）能量−1，发现526次。
- 第3行：解决方案(a,b,c)=（0,1,0）能量−1，发现267次。

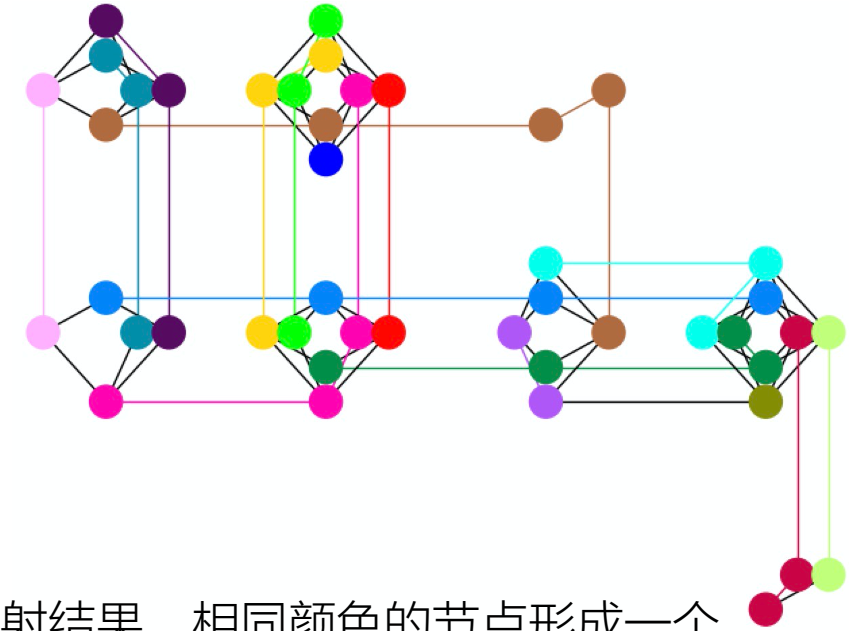
手动次要嵌入问题通常仅适用于变量很少或重复结构映射到QPU拓扑单元的问题。在这两种情况下，您都只有一个或多个单元。当面临一个随机的100变量问题，你不太可能手动嵌入。

把QUBO模型映射到量子计算机示例： N = 16 random geometric graph

考虑一个问题，我们随机地将N = 16个物体放在一个正方形中，并且只有当它们之间的距离小于某个半径时，物体之间才用边相连，我们选择这个半径为0.5，正方形边长的一半。（距离小于正方形的边长的一半）
我们的目标是将图分成两个子集（也称为分区），我们希望它们的大小相等，并最小化两个分区之间的链接数量。这个问题称为图划分问题，一个典型的NP难问题。

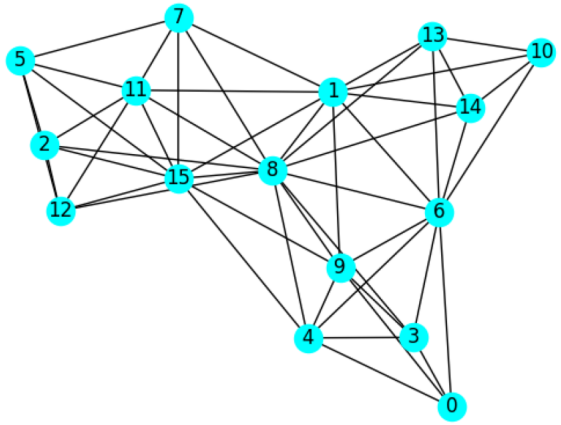






映射结果，相同颜色的节点形成一个Chain，组合一起表示一个比特位。

把QUBO模型映射到量子计算机示例： What is a chain break?



用于图分区的QUBO具有以下形式：

$$\min \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j) + \gamma \left(-N \cdot \sum_i^N x_i + \sum_i^N x_i + \sum_i^N \sum_{j>i}^N 2x_i x_j \right)$$

a broken chain就是属于同一个Chain里的Qubits的值不是完全一致。

在上一节中，当我们讨论链时，我们说成员量子比特是链接在一起的。这意味着它们在数学上被限制为具有相同的值：0或1（QUBO）；或-1或1（伊辛）。为了实现这一点，使用了参数链强度，这使链能够与QUBO中第一项的相对强度和约束条件竞争。

如果链强度参数太小，每个链中的量子位将不会具有相同的值，并且所需的量子位之间的关系将无法按预期工作。链内量子位取不同值的链称为断链，问题与程序员想要解决的图不太一样。

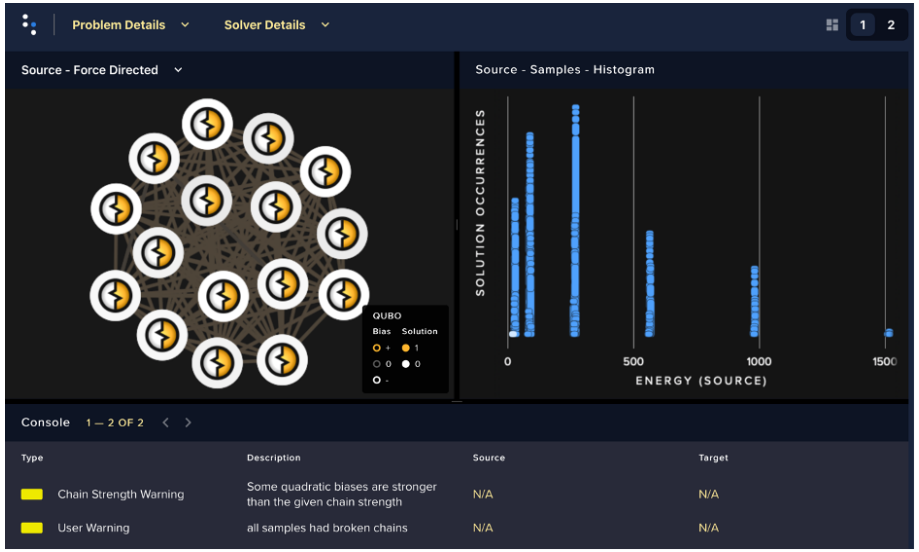
相反，如果链强度参数太大，嵌入项将主导图划分问题的细节，我们可能会失去研究预期问题的能力。显然，必须仔细选择链条强度参数：不要太小，也不要太大。

把QUBO模型映射到量子计算机示例： What is a chain break?

把前面所描述的16节点随机图问题映射成QUBO图之后，程序中保存在下面程序第四行的Q变量中，提交给量子计算机运行。Python程序如下：

```
import dwave.inspector

dwave_sampler = FixedEmbeddingComposite(DWaveSampler(solver={'qpu': True}),
    embedding)

bqm = dimod.BinaryQuadraticModel.from_qubo(Q, offset=offset)

sampleset = dwave_sampler.sample(bqm, num_reads=1000)

dwave.inspector.show(sampleset)
```

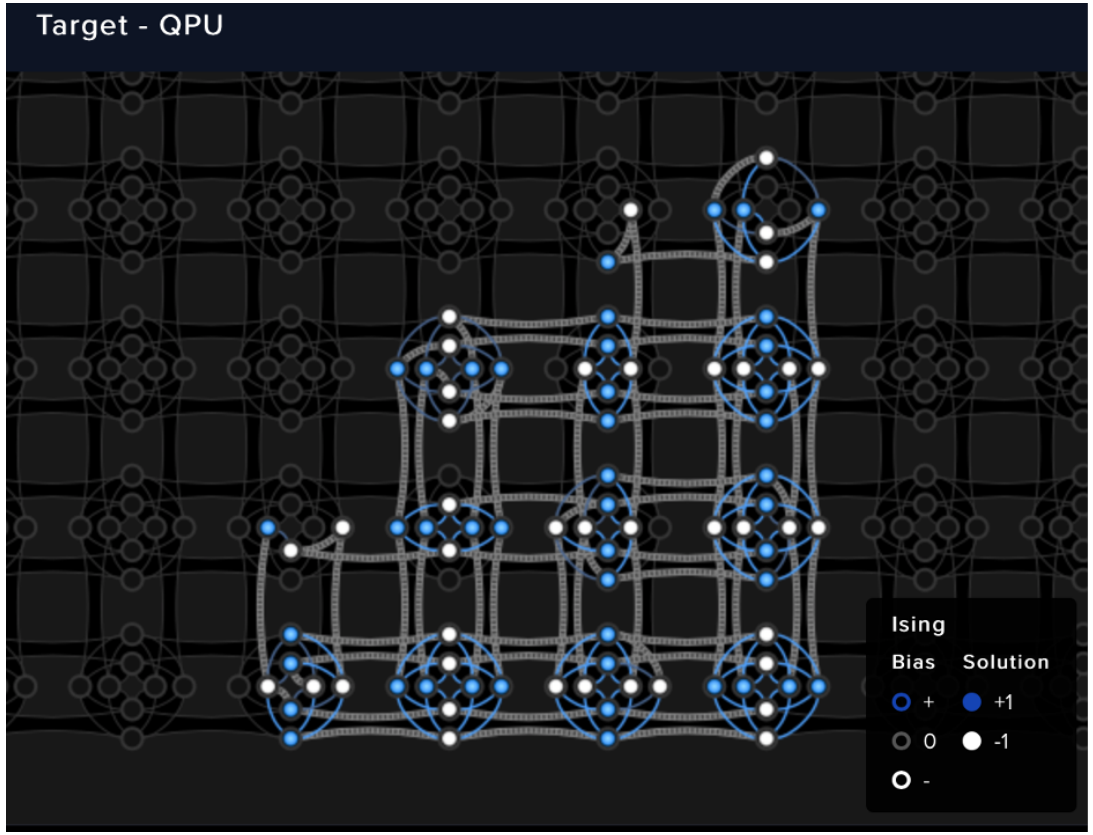
函数chain_stength参数，使用缺省值是1。



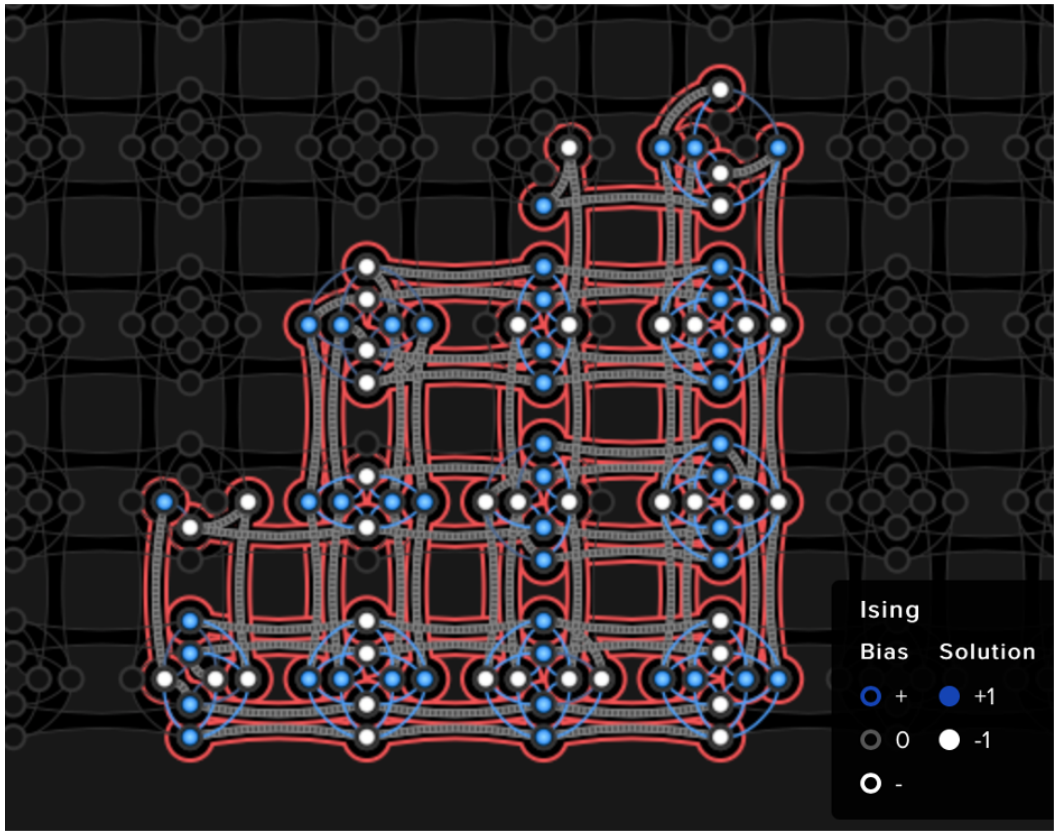
- 分区大小相同（每个分区中有8个节点）。
- 分区之间有19个链接。
- 有一种有效的解决方案（1000个样品的0.1%）。
- 问题中的所有链条都断了。问题变量上的白色和橙色之字形符号表示断链。

把QUBO模型映射到量子计算机示例： What is a chain break?

使用问题Inspector显示嵌入的链



请注意，灰色链连接着多组量子位。我们希望每条链包含相同颜色的量子位，即全部0或全部1。如前所述，如果一条链连接不同颜色的量子位，它就会断开。



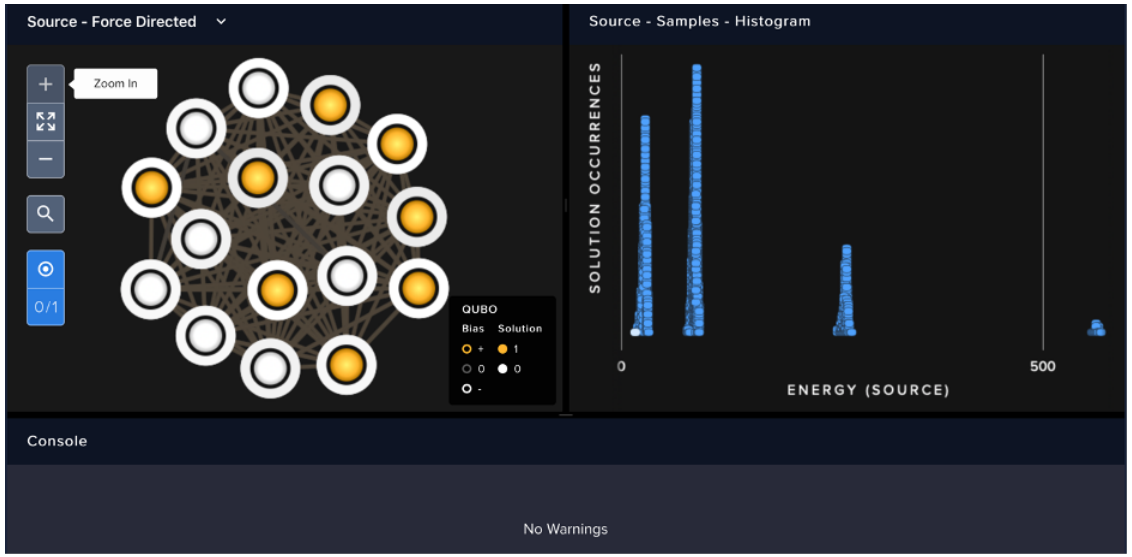
通过启用问题inspector （屏幕左侧工具栏上）中的另一个选项以红色显示断链，我们可以看到所有链都断了。

把QUBO模型映射到量子计算机示例： Solve chain break?

增加链条强度
我们如何知道链条强度参数的合理值？一个好的第一个估计是将其设置为接近问题QUBO中最大绝对值的值。在这个图分区问题中，QUBO条目的范围从-896到大约120。因此，链强度参数的第一个好猜测是1000。

我们更改代码中的链强度参数：就是在原来代码的第四行，加上chain_stength参数, chain_stength参数缺省值是1，现在我们把它改成1000.

```
sampleset = dwave_sampler.sample(bqm,num_reads=1000,
    chain_strength=chain_strength_value)
```



- 没有断链
- 没有警告
- 我们确认分区之间的链接数是正确的

两个步骤+编程实践

第1步、把实际问题建模成Ising模型或者QUBO模型。

第2步、把模型映射（embedding）到量子计算机阵列D-Wave 2000Q Chimera graph 或D-Wave Advantage Pegasus graph 进行运算。

第3步、 D-Wave编程实践，使用D-Wave提供的Ocean package软件工具和模板完成前面两步骤的工作。

D-Wave量子计算机软件编程工具平台Ocean Software

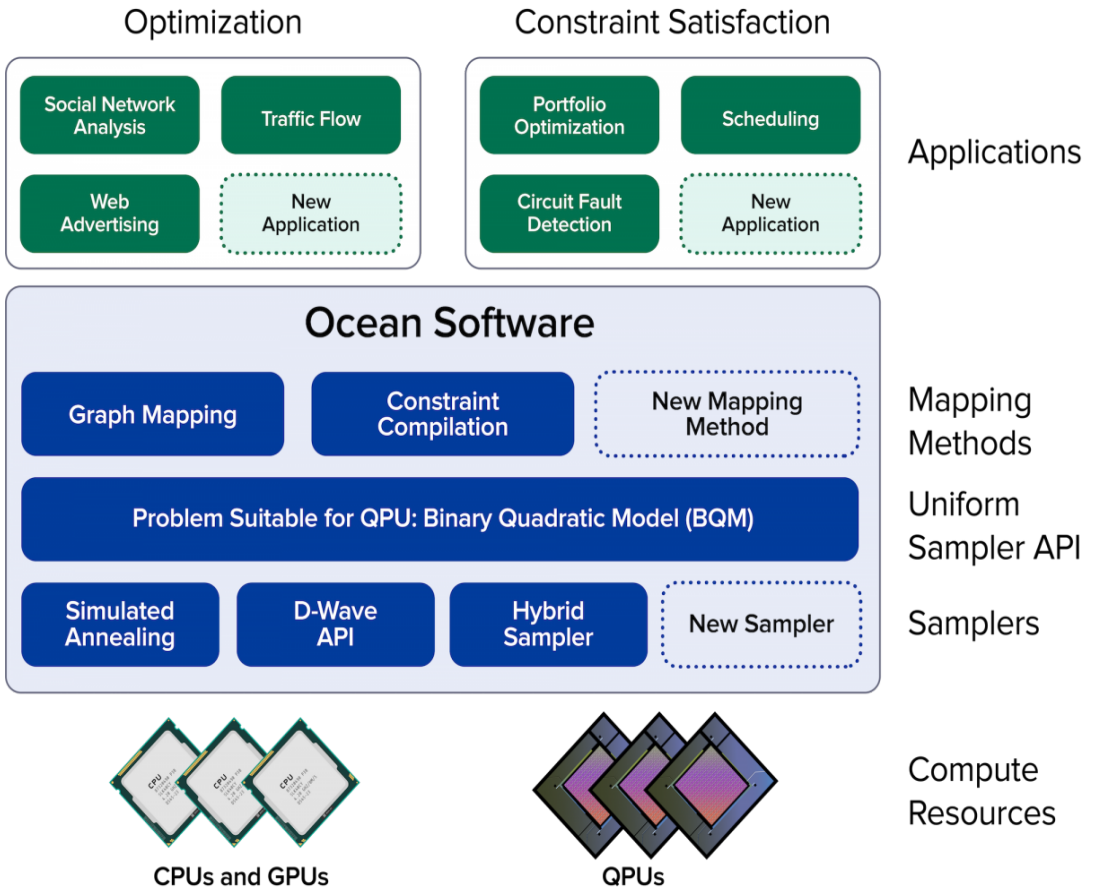
•Application： 原始问题（“问题空间”）及其上下文，包括应用程序数据和明确定义的目标。例如，芯片制造中的电路故障诊断应用需要识别电路中故障逻辑门的最小集合。

•Mapping Methods： 将应用程序目标和数据转换为适合量子计算形式的问题的工具。他们还接收解决方案采样结果，并将其转换回应用层的解决方案。例如，dwave_networkx帮助将结构不平衡性分析结果映射到BQM中。

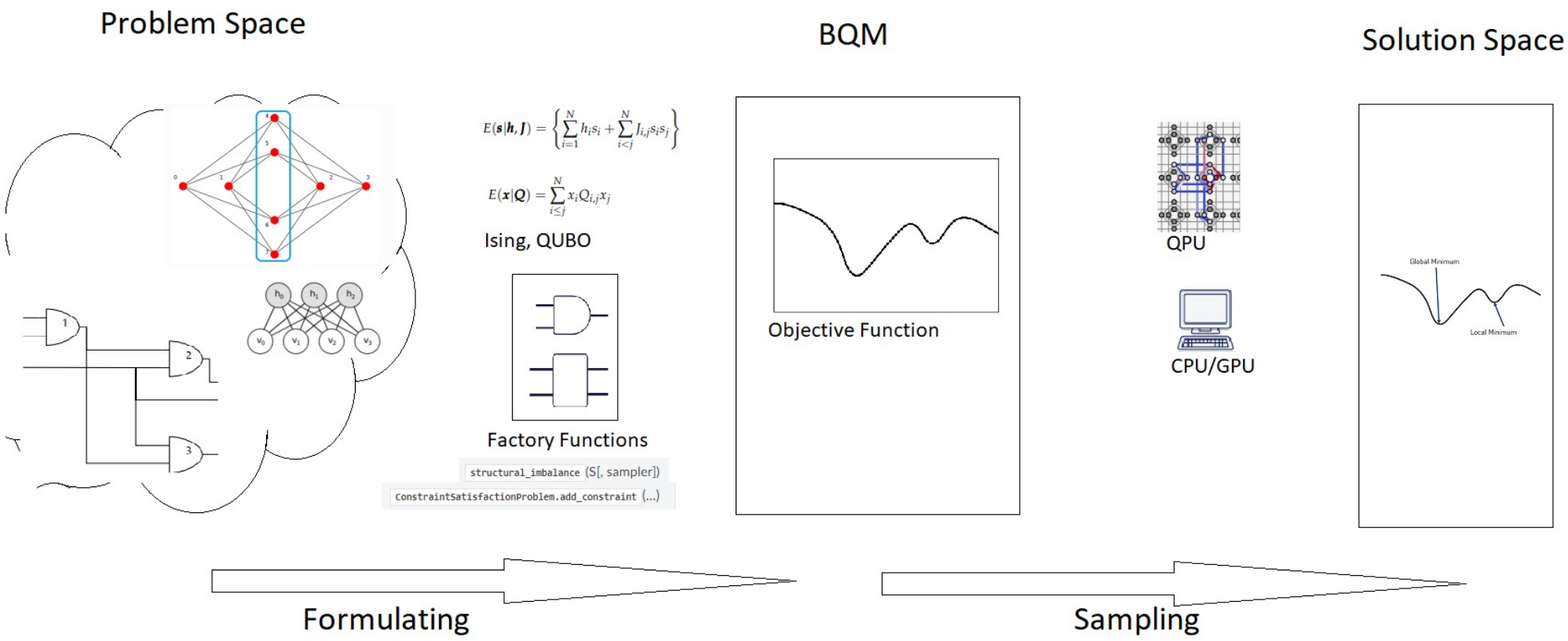
•Uniform Sampler API:： 抽象层，把问题转换成适应具体Sampler要求的格式。

•Samplers： 以BQM形式接收问题并返回解决方案采样结构的工具。Ocean使用D-Wave QPU和经典计算资源实现了几个采样器。您可以使用Ocean工具自定义D-Wave Samplers 、创建自己的或使用现有的经典Samplers 。

•Compute Resources： 执行问题的量子计算机硬件，包括量子计算机部分和经典计算机部分，这可能是一个D-Wave QPU，但也可能是笔记本电脑的CPU。



D-Wave量子计算机软件编程工具平台Ocean Software

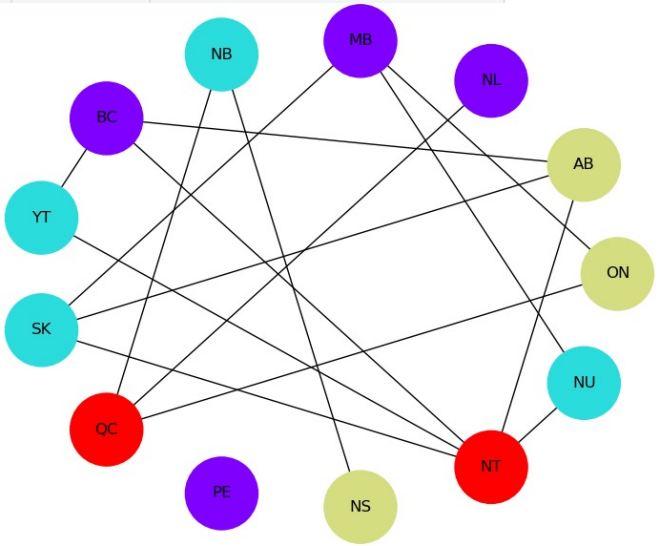


煤油灯科技
VICTORLAMP

D-Wave量子计算机软件编程工具平台Ocean Software：地图着色问题

Code	Province	Code	Province
AB	Alberta	BC	British Columbia
MB	Manitoba	NB	New Brunswick
NL	Newfoundland and Labrador	NS	Nova Scotia
NT	Northwest Territories	NU	Nunavut
ON	Ontario	PE	Prince Edward Island
QC	Quebec	SK	Saskatchewan
YT	Yukon		

Coloring a map of Canada with four colors.
四种颜色，相邻的区域颜色不能相同。



具体程序代码见下一页.....

利用量子计算机运行结果

D-Wave量子计算机软件编程工具平台Ocean Software: Map Coloring Full Code

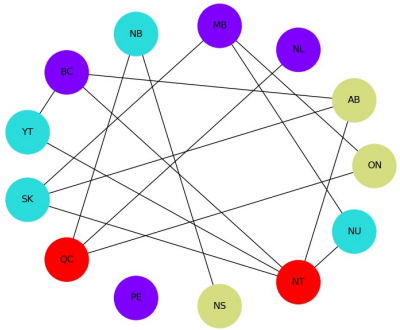
```
import dwavebinarycsp
from dwave.system import DWaveSampler,
EmbeddingComposite
import networkx as nx
import matplotlib.pyplot as plt

# Represent the map as the nodes and edges of a graph
provinces = ['AB', 'BC', 'MB', 'NB', 'NL', 'NS', 'NT', 'NU',
'ON', 'PE', 'QC', 'SK', 'YT']
neighbors = [('AB', 'BC'), ('AB', 'NT'), ('AB', 'SK'), ('BC',
'NT'), ('BC', 'YT'), ('MB', 'NU'),
          ('MB', 'ON'), ('MB', 'SK'), ('NB', 'NS'), ('NB', 'QC'),
('NL', 'QC'), ('NT', 'NU'),
          ('NT', 'SK'), ('NT', 'YT'), ('ON', 'QC')]

# Function for the constraint that two nodes with a
# shared edge not both select one color
def not_both_1(v, u):
    return not (v and u)
```

```
# Function that plots a returned sample
def plot_map(sample):
    G = nx.Graph()
    G.add_nodes_from(provinces)
    G.add_edges_from(neighbors)
    # Translate from binary to integer color representation
    color_map = {}
    for province in provinces:
        for i in range(colors):
            if sample[province+str(i)]:
                color_map[province] = i
    # Plot the sample with color-coded nodes
    node_colors = [color_map.get(node) for node in G.nodes()]
    nx.draw_circular(G, with_labels=True,
node_color=node_colors, node_size=3000,
cmap=plt.cm.rainbow)
    plt.show()
```

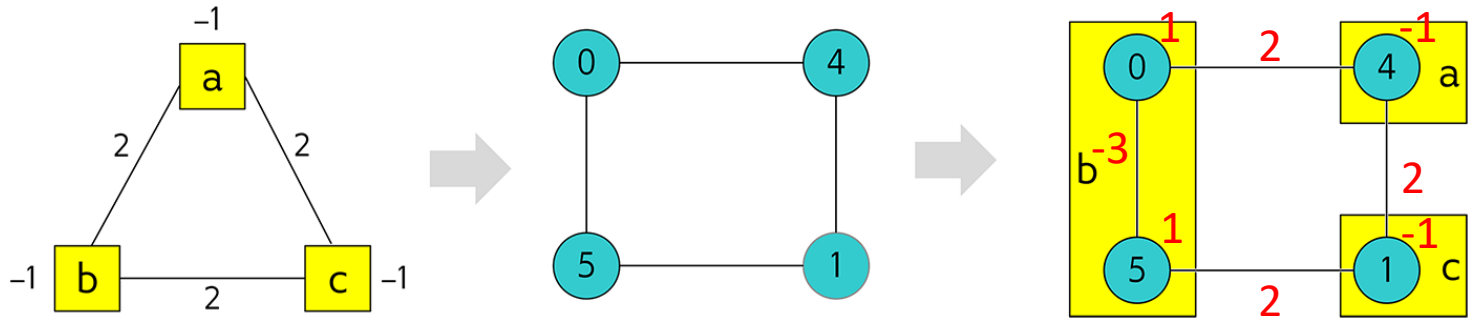
D-Wave量子计算机软件编程工具平台Ocean Software : Map Coloring Full Code

```
# Valid configurations for the constraint that each node select a single
# color
one_color_configurations = {(0, 0, 0, 1), (0, 0, 1, 0), (0, 1, 0, 0), (1, 0, 0, 0)}
colors = len(one_color_configurations)

# Create a binary constraint satisfaction problem
csp =
dwavebinarycsp.ConstraintSatisfactionProblem(dwavebinarycsp.BINARY)

# Add constraint that each node (province) select a single color
# for province in provinces:
    variables = [province+str(i) for i in range(colors)]
    csp.add_constraint(one_color_configurations, variables)

# Add constraint that each pair of nodes with a shared edge not both
# select one color
for neighbor in neighbors:
    v, u = neighbor
    for i in range(colors):
        variables = [v+str(i), u+str(i)]
        csp.add_constraint(not_both_1, variables)
```

```
# Convert the binary constraint satisfaction problem to a
# binary quadratic model
bqm = dwavebinarycsp.stitch(csp)

# Set up a solver using the local system' s default D-Wave
# Cloud Client configuration file and sample 1000 times
sampler = EmbeddingComposite(DWaveSampler())
sampleset = sampler.sample(bqm, num_reads=1000)

# Plot the lowest-energy sample if it meets the constraints
sample = sampleset.first.sample
if not csp.check(sample):
    print("Failed to color map")
else:
    plot_map(sample)
```



D-Wave量子计算机软件编程工具平台Ocean Software：朋友分组问题



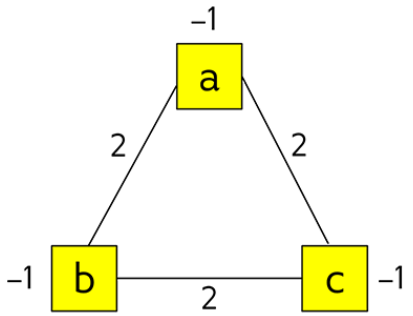

```
from dwave.system import DWaveSampler

sampler_manual = DWaveSampler(solver={'topology__type':
'chimera'})

qubit_biases = {(0, 0): 1, (1, 1): -1, (4, 4): -1, (5, 5): 1}
coupler_strengths = {(0, 4): 2, (0, 5): -3, (1, 4): 2, (1, 5): 2}

Q = {**qubit_biases, **coupler_strengths}

sampleset = sampler_manual.sample_qubo(Q, num_reads=5000)

print(sampleset)
```

量子计算机运行结果

```
>>> print(sampleset)
   0  1  4  5 energy num_oc.
0  0  0  1  0   -1.0    1274
1  0  1  0  0   -1.0    1726
2  1  0  0  1   -1.0    1997
3  1  1  0  1    0.0       1
4  0  1  1  0    0.0       2
['BINARY', 5 rows, 5000 samples, 4 variables]
```

D-Wave量子计算机软件编程工具平台Ocean Software：TSP问题

也可以由软件自动完成模型映射。




```
from dwave.system import DWaveSampler, EmbeddingComposite

sampler_auto =
EmbeddingComposite(DWaveSampler(solver={'topology__type':
'chimera'}))

linear = {('a', 'a'): -1, ('b', 'b'): -1, ('c', 'c'): -1}
quadratic = {('a', 'b'): 2, ('b', 'c'): 2, ('a', 'c'): 2}

Q = {**linear, **quadratic}

sampleset = sampler_auto.sample_qubo(Q, num_reads=5000)

print(sampleset)
```

量子计算机运行结果

```
>>> print(sampleset)
   a  b  c energy num_oc. chain_b.
0  0  1  0   -1.0    1465      0.0
1  1  0  0   -1.0    1838      0.0
2  0  0  1   -1.0    1695      0.0
3  0  1  1    0.0       1      0.0
4  1  0  1    0.0       1 0.333333
['BINARY', 5 rows, 5000 samples, 3 variables]
```

D-Wave量子计算机软件编程工具平台Ocean Software：总结

1、构建QUBO/BQM模型： D = { key1: value1, key2: value2, ... }.

2、通过交互构建Sampler/Solver

Example. To use the D-Wave QPU, we might use the following lines of code.

```
from dwave.system.samplers import DWaveSampler, EmbeddingComposite

sampler = EmbeddingComposite(DWaveSampler(solver={'qpu': True}))
```

True参数表示使用物理硬件系统

3、调用 Sampler，比如调用Sampler的sample_qubo(.....)函数

4、 Sampler运行之后返回 SampleSet对象。
SampleSet.record: The full set of samples
SampleSet.first: The sample with the lowest energy.
SampleSet.data: The complete information about the solutions and sampler.

步骤看起来非常清晰、简单，具体到实际问题处理，还是需要很多的技巧。下面这个网址有很多的例子可以参考，https://github.com/dwave-examples，多看例子和代码，也就会用了。

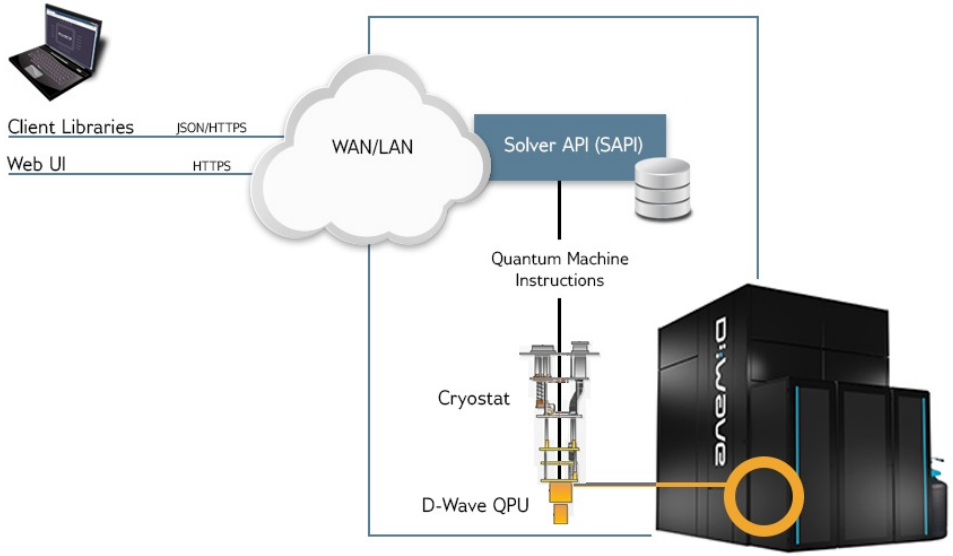
使用D-Wave Leap Quantum Cloud Service云平台进行量子计算编程

Leap™ Quantum Cloud Service

Leap于2018年推出，是D-Wave系统公司的量子云服务。

了解D-Wave量子计算机可以解决的问题类型，在系统上运行交互式演示和编码示例，贡献您的编码想法，并加入我们的用户社区中持续的沟通交流。

在此处注册Leap：https://cloud.dwavesys.com/leap.



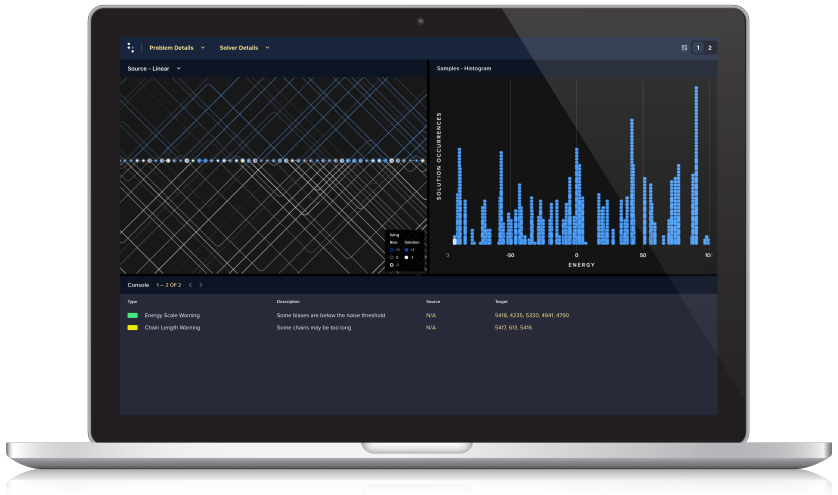
使用D-Wave Leap Quantum Cloud Service云平台进行量子计算编程

Ocean SDK

D-Wave基于Python的开源软件开发工具包（SDK）Ocean使量子计算机的应用程序开发快速高效，并促进了协作项目。请参阅GitHub上的Ocean SDK以访问Ocean SDK，并参阅Ocean文档以获取相关文档。

IDE

D-Wave的Leap integrated development environment（IDE）是开始编写量子应用程序或学习使用海洋工具的最快方法。所有Leap帐户都可以在浏览器中运行此基于云的IDE。它提供了可重用/一次性工作区（开发人员环境预先配置了Ocean和其他标准库以及D-Wave扩展），用于从您自己的GitHub存储库或一组代码示例中运行代码，然后您可以修改这些代码示例。



使用D-Wave Leap Quantum Cloud Service云平台进行量子计算编程

参考资料文档:

D-Wave System Documentation https://docs.dwavesys.com/docs/latest/index.html

D-Wave Leap Quantum Cloud Service https://docs.dwavesys.com/docs/latest/leap.html

D-Wave Ocean Software https://docs.dwavesys.com/docs/latest/ocean.html

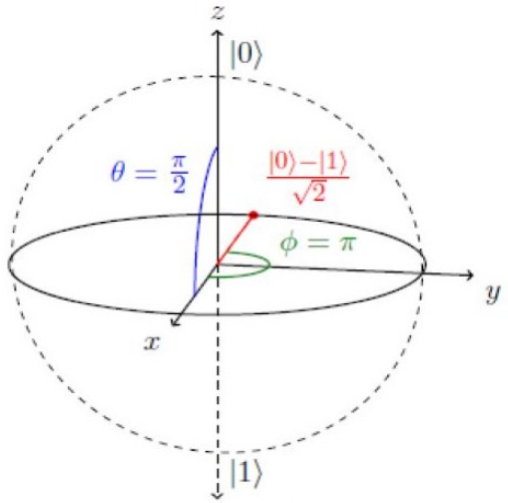
Leap用户注册网址（中国大陆暂未开放）:

Sign up for Leap here: https://cloud.dwavesys.com/leap.

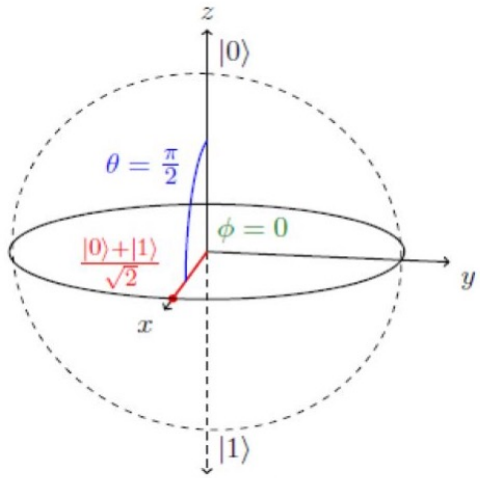
GitHub资源

https://github.com/dwave-examples

e休，爱编程的葫芦娃
Email：exiu@victorlamp.com



(b) Basis state $|1\rangle$



(a) Basis state $|0\rangle$

VICTORLAMP

Shenzhen VictorLamp Technologies CO. Ltd.
http://www.victorlamp.com