An example of Algorithm 1:

Given the benchmark quantum circuit "rd32-v0_66" shown in Fig. 1 (in which single qubit quantum logical gate is omitted) and a linear nearest neighbor architecture shown in Fig. 2, calculate lower bound of qubit allocation cost function according to Algorithm1.

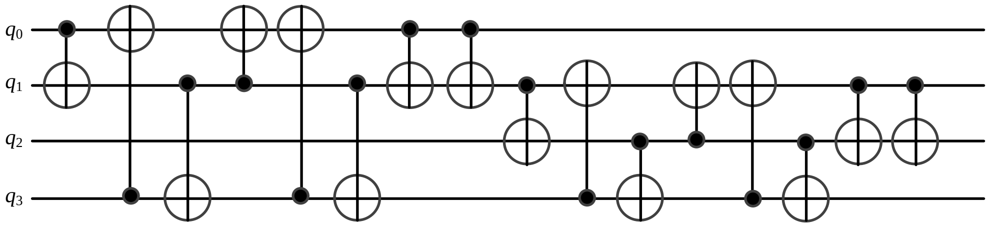

Fig1. Quantum circuit "rd32-v0_66"

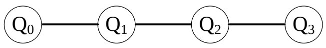

Fig2. A linear nearest neighbor architecture with four physical qubits

The logical qubit interaction frequency matrix **F** of quantum circuit "rd32-v0_66" is:

0	4	0	2
4	0	4	4
0	4	0	2
2	4	2	0

The physical qubit coupling distance matrix **R** of architecture as shown in Fig. 1 is:

0	0	1	2
0	0	0	1
1	0	0	0
2	1	0	0

Based on F and R, the original cost matrix **C** is:

0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	4	8	0	*	0	4	4	0	*	0	8	4	0	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
*	0	2	4	0	*	0	2	2	0	*	0	4	2	0	*
*	0	4	8	0	*	0	4	4	0	*	0	8	4	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	4	8	0	*	0	4	4	0	*	0	8	4	0	*
*	0	4	8	0	*	0	4	4	0	*	0	8	4	0	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
*	0	4	8	0	*	0	4	4	0	*	0	8	4	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	2	4	0	*	0	2	2	0	*	0	4	2	0	*
*	0	2	4	0	*	0	2	2	0	*	0	4	2	0	*
*	0	4	8	0	*	0	4	4	0	*	0	8	4	0	*
*	0	2	4	0	*	0	2	2	0	*	0	4	2	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0

The original leader matrix L is:

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Using cost matrix C and leader matrix L as input parameters, Algorithm 1 is called, initially, the lower bound *low_bound* is set to 0, and the upper bound *up_bound* is set to infinity. The low bound is calculated as follows:

1. First iteration

1.1 Step 1 of Algorithm 1: Update cost matrix **C** with non-zero leader matrix elements.

Since the leading elements are all zero at the beginning, the cost matrix remains unchanged.

1.2 Step 2 of Algorithm 1: Divide the value of complementary elements in cost matrix equally.

Due to the symmetry of the matrix F and R, the values of the two complementary elements are equal at the beginning, so the cost matrix remains unchanged.

1.3 Step 3 of Algorithm 1: Apply Hungarian algorithm to each sub-matrix $\mathbf{C}_{ij}$ of cost matrix **C**, and update leader matrix as well as upper bound of cost function.

1.3.1 On sub-matrix $\mathbf{C}_{00}$

Apply Hungarian algorithm to $\mathbf{C}_{00}$, $\mathbf{C}_{00}$ becomes:

0	*	*	*
*	0	2	6
*	2	0	0
*	0	0	2

The solution returned by Hungarian algorithm is {0, 1, 3, 2}, and the value returned by Hungarian algorithm is 2. Then, **L**[0][0]=**L**[0][0]+2=2.

Using {0, 1, 3, 2} as input, compute cost function value which is equal to 12;

Because the current upper bound is greater than 12, update the upper bound of cost function, i.e. *up_bound*=12.

1.3.2 On sub-matrix $\mathbf{C}_{01}$

Apply Hungarian algorithm to $\mathbf{C}_{01}$, $\mathbf{C}_{01}$ becomes:

*	0	*	*
0	*	0	4
0	*	0	0
0	*	0	2

The solution from Hungarian algorithm is {1, 2, 3, 0}, and the value returned by Hungarian algorithm is 0. Then, **L**[0][1]=**L**[0][1]+0=0.

Using {1, 2, 3, 0} as input, compute cost function value which is equal to 16;

Because the current upper bound is equal to 12 which is less than 16, keep the upper bound of cost function unchanged.

1.3.3 On sub-matrix $\mathbf{C}_{02}$

....

1.3.16 On sub-matrix $\mathbf{C}_{33}$

Apply Hungarian algorithm to $\mathbf{C}_{33}$, $\mathbf{C}_{33}$ becomes:

0	0	0	*

4	2	0	*
0	0	0	*
*	*	*	0

The solution returned by Hungarian algorithm is {0 ,2 ,1 ,3},and the value returned by Hungarian algorithm is 6. Then, **L**[3][3]=**L**[3][3]+6=6.

Using {0, 2, 1, 3} as input, compute `cost function value which is equal to 20;

Because the current upper bound is equal to 12 which is less than 20, keep the upper bound of cost function unchanged.

1.4 Step 4 of Algorithm 1: Apply Hungarian algorithm to leader matrix **L**, and update lower bound as well as upper bound of cost function.

Apply Hungarian algorithm to **L**, **L** becomes:

0	*	*	*
*	0	2	6
*	2	0	0
*	0	0	2

The solution from Hungarian algorithm is {3, 2, 0, 1}, and the value returned by Hungarian algorithm is 10. Then, increase lower bound by 10, *low_bound*=*low_bound*+10=10.

Using {3, 2, 0, 1}as input, compute cost function value which is equal to 12.

Because the current upper bound is equal to 12, keep the upper bound unchanged.

1.5 Step 5 of Algorithm 1. None of the conditions to finish the loop are satisfied, jump to Step 1 of Algorithm 1. The first iteration ends and the second iteration begins. At this point, the cost matrix **C** is:

0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	2	6	0	*	0	4	4	0	*	0	6	2	0	*
*	2	0	0	0	*	0	0	0	0	*	0	0	0	2	*
*	0	0	2	0	*	0	2	2	0	*	0	2	0	0	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
*	2	0	0	0	*	0	0	0	0	*	0	0	0	2	*
*	0	2	6	0	*	0	4	4	0	*	0	6	2	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	0	2	0	*	0	2	2	0	*	0	2	0	0	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
*	0	2	4	0	*	0	2	2	0	*	0	4	2	0	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0

The leader matrix **L** is:

0	0	0	0
6	0	0	6
0	0	0	0
2	0	0	2

2. Second iteration

2.1 Step 1 of Algorithm 1: Update cost matrix **C** with non-zero leader matrix elements. Distribute the non-zero **L**[i][j] to allowed elements of $\mathbf{C}_{ij}$ in uniform. The cost matrix and leader matrix become:

0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	2	6	0	*	0	4	4	0	*	0	6	2	0	*
*	2	0	0	0	*	0	0	0	0	*	0	0	0	2	*
*	0	0	2	0	*	0	2	2	0	*	0	2	0	0	*
*	2	2	2	0	*	0	0	0	0	*	0	2	2	2	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	2	2	2	0	*	0	0	0	0	*	0	2	2	2	*
*	2	2	2	0	*	0	0	0	0	*	0	2	2	2	*
*	2	0	0	0	*	0	0	0	0	*	0	0	0	2	*
*	0	2	6	0	*	0	4	4	0	*	0	6	2	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	0	2	0	*	0	2	2	0	*	0	2	0	0	*
*	1	1	1	0	*	0	0	0	0	*	0	1	1	1	*
*	1	3	5	0	*	0	2	2	0	*	0	5	3	1	*
*	0	0	0	0	*	0	0	0	0	*	0	0	0	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

2.2 Step 2 of Algorithm 1: Divide the value of complementary elements in cost matrix equally.

0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	1	4	1	*	0	3	3	0	*	1	4	1	0	*
*	1	0	0	1	*	0	0	0	0	*	1	0	0	1	*
*	0	0	1	1	*	0	1	2	0	*	0	1	0	0	*
*	1	3	4	0	*	0	1	1	0	*	0	4	3	1	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	1	3	4	0	*	0	1	1	0	*	0	4	3	1	*
*	1	2	4	1	*	0	2	1	0	*	1	4	2	1	*
*	1	0	0	1	*	0	0	0	0	*	1	0	0	1	*
*	0	1	4	1	*	0	3	3	0	*	1	4	1	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0
*	0	0	1	0	*	0	1	1	0	*	0	1	0	0	*
*	0	1	2	0	*	0	0	0	0	*	0	2	2	1	*
*	0	2	3	1	*	0	2	2	0	*	1	3	1	0	*
*	0	1	1	0	*	0	0	0	0	*	0	1	1	0	*
0	*	*	*	*	0	*	*	*	*	0	*	*	*	*	0

2.3 Step 3 of Algorithm 1. Similar to the first iteration.

2.4 Step 4 of Algorithm 1. Similar to the first iteration.

2.5 Step 5 of Algorithm 1. Similar to the first iteration.

3. Third iteration

3.1 Step 3 of Algorithm 1. Similar to the second iteration.

....

3.4 Step 4 of Algorithm 1: Apply Hungarian algorithm to leader matrix **L**, and update lower bound as well as upper bound of cost function.

Apply Hungarian algorithm to **L**, **L** becomes:

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{matrix}$$

The solution from Hungarian algorithm is {0, 1, 3, 2}, and the value returned by Hungarian algorithm is 1. Then, increase lower bound by 1, *low_bound*=*low_bound*+1=12.

Using {0, 1, 3, 2}as input, compute cost function value which is equal to 12.

Because the current upper bound is equal to 12, keep the upper bound unchanged.

1.5 Step 5 of Algorithm 1. At this point the upper bound is equal to the lower bound, in this case, algorithm 1 not only obtains the lower bound, but also finds the optimal value. Algorithm 1 exits and returns optimal value 12 as well as optimal solution {0, 1, 3, 2}, which means assign q_0 to Q_0, q_1 to Q_1, q_2 to Q_3, and q_3 to Q_2, respectively.