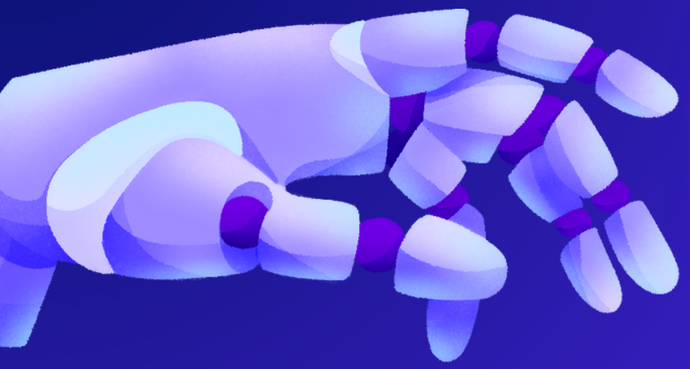
QUANTUM COMPUTING HACKATHON

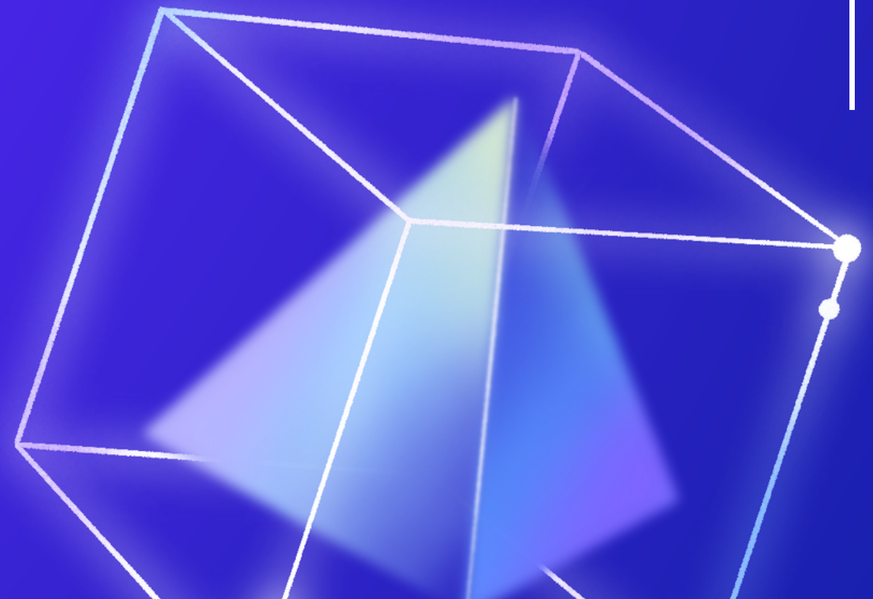
KIDNEY TRANSPLANT PROBLEM

By Quattro

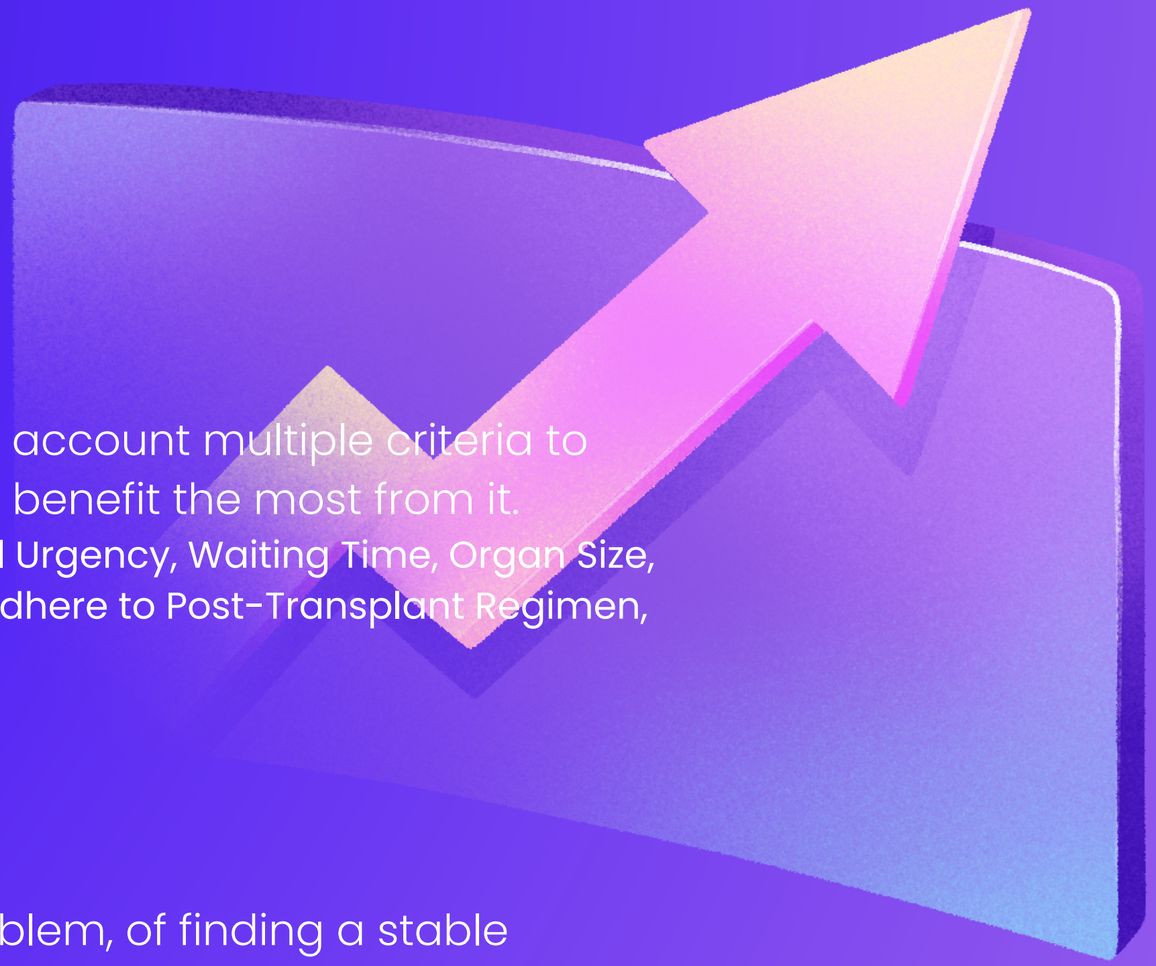


QUATTRO

- Laura Matínez [SomeOneTwo#4308]
- Guillermo Mijares [gmijares#8131]
- Camilo Perez [camilo_peruz]
- Carlo Peña [carlo_62036]



INTRODUCTION



Organ allocation is a complex process that takes into account multiple criteria to ensure that each organ goes to the recipient who will benefit the most from it.
For example: Blood Type Compatibility, Tissue Match, Medical Urgency, Waiting Time, Organ Size, Recipient Health, Geographic Location, Recipient's Ability to Adhere to Post-Transplant Regimen, Ethical and Policy Considerations, Psychosocial Factors

---

Stable Marriage Problem

The Stable Marriage Problem (SMP) describes the problem, of finding a stable matching between two equally sized sets of elements given an ordering of preferences (constrains) for each element.
A Quantum Annealing Approach for Solving Hard Variants of the Stable Marriage Problem is selected to apply in the allocation of a donor’s kidney given a list of possible recipients.

PROBLEM DEFINITION

Let D and R be a list of donors and recipients, respectively.
Objective function: maximize the number of successful transplants.
This is achieved by matching kidney donors and recipients in a way that maximizes the number of people receiving compatible kidneys.

Compatibility constraints:

- Blood group compatibility
- Immunological conmpatibility between donor and recipient (Human Leukocyte Antigens - HLA).
- Antibody compatibility (Panel Reactive Antibodies Coefficient - CPRA).
- Kidney size compatibility and age
- Immunosupression considerations.
- Psychological evaluation

MATHEMATICAL EXPRESSION

Suppose there are *n* kidney donors and *m* kidney recipients. Let's define the following binary variables, such that

$x_{ij} = 1$ means that donor i is assigned to recipient j,

$x_{ij} = 0$ otherwise.

- **Objective function.**

The objective function aims to maximize the compatibility between donors and recipients, which can be expressed as the sum of individual compatibilities as follows,

$$max \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Compatibility_{ij} \cdot x_{ij}$$

where $Compatibility_{ij}$ is a measure of compatibility between donor i and recipient j, which corresponds exactly to element j in the i-th list of *results*.

(i.e. $Compatibility_{ij}$ = *results[i][j]*)

Then,

$$H_{objective} = -\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Compatibility_{ij} \cdot x_{ij}$$

MATHEMATICAL EXPRESSION

- **Unique assignment constraint (Each donor is assigned to one recipient, and vice versa).**

$$\sum_{j=0}^{m-1} x_{ij} = 1$$

$\forall i \in 0, \ldots, n-1$ and

$$\sum_{i=0}^{n-1} x_{ij} = 1$$

$\forall j \in 0, \ldots, m-1.$

Therefore,

$$H_{uniqueness} = \sum_{i=0}^{n-1}(\sum_{j=0}^{m-1} x_{ij} - 1)^2 + \sum_{j=0}^{m-1}(\sum_{i=0}^{n-1} x_{ij} - 1)^2$$

So that,

$$H_{total} = -\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Compatibility_{ij} \cdot x_{ij} + \sum_{i=0}^{n-1}(\sum_{j=0}^{m-1} x_{ij} - 1)^2 + \sum_{j=0}^{m-1}(\sum_{i=0}^{n-1} x_{ij} - 1)^2$$

```
[ ]  from openqaoa.backends import create_device

     for optimizer in ["Nelder-Mead", "BFGS", "COBYLA", "SLSQP"]:
         for k in range(1,30):
             q = QAOA()
```

15 min

Long processing time

120 min

```
[ ]  for optimizer in ["COBYLA", "SLSQP"]:
         for k in range(1,30):
             q = QAOA()

             device = create_device(location='local', name='vectorized')
             q.set_device(device)
```

```
[ ]  from openqaoa.backends import create_device

     q = QAOA()

     qiskit_device = create_device(location='local', name='vectorized')
     q.set_device(qiskit_device)

     # circuit properties
     q.set_circuit_properties(p=13, param_type='standard', init_type='rand', mixer_hamiltonian='x')

     # backend properties (already set by default)
     q.set_backend_properties(prepend_state=None, append_state=None)

     # classical optimizer properties
     q.set_classical_optimizer(method='SLSQP',
                               maxiter=500, tol=0.001,
                                 optimization_progress=True, cost_progress=True, parameter_log=True)
```

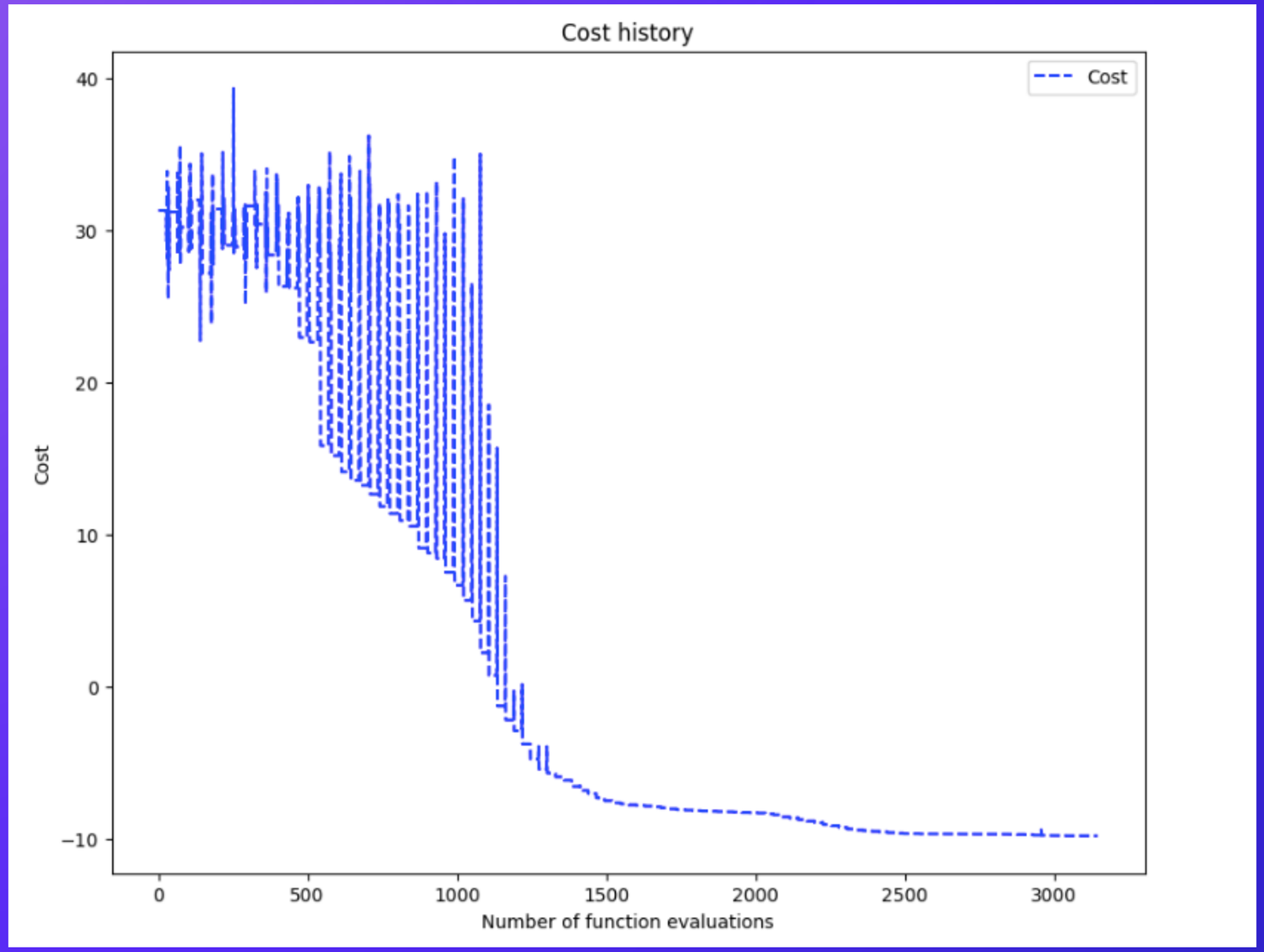
```
     q.compile(qubo_instance)
     q.optimize()
```

```
     resultados = q.result
     resultados.plot_cost()
```

```
[19] print(parejas_solucion)

     [[0, 1], [1, 2], [2, 3], [3, 0]]
```

COSTE


Cost history
Cost
40
30
20
10
0
−10
Cost
0
500
1000
1500
2000
2500
3000
Number of function evaluations

RESULTS AND ACHIEVEMENTS

01

- Brute Force calculations were more time consuming than applying directly the Q Circuit.
- For real life applications, there should be upgraded technology with much more qbits availability.

02

- It was possible to calculate an optimized allocation matrix, with one kidney per recipient, with the highest compatibility

THANK YOU!