1 3 Nodes Problem in 9 Qubits

1.1 Representation of the solution

We represent our solution as a 3×3 matrix of binary variables: $(x_{i,t})$. If $x_{i,t}$ is 1, the group was at location i at time t, and else it was not.

1.2 Cost Function and Constraints

The cost of a certain solution is given by $\sum_{i,j} \sum_{t=0,1} w_{ij} x_{i,t} x_{j,t+1}$. To ensure the group only visits each place once, and only one at a time, there has to be exactly one entry 1 per row and per column. These constraints can be represented by the cost functions $\sum_i (1 - \sum_t x_{i,t})^2$ and $\sum_t (1 - \sum_i x_{i,t})^2$. Our final cost function is the sum of the previous. We weight the constraints with a factor of 2. We generate the cost Hamiltonian by replacing x_{it} by the operator $1/2(1-\sigma^Z_{i,t})$.

1.3 Mixing and Initial Hamiltonian

As a mixing Hamiltonian for QAOA and an initial Hamiltonian for AQO, respectively, we chose $H = \sum_{i,t} \sigma_i^X$

1.4 Scaling of representation

Our representation is scalable and needs n^2 qubits to solve the TSP with n nodes.

1.5 Results

The probability distributions of results for QAOA and AQO can be found on the powerpoint. AQO has much better fidelity as it has an over 70 % chance of outputting the minimizing solution. On the contrary, QAOA is fairly good at sorting solutions that do not fulfill the constraints but the chance of it outputting the minimizing solution is only around 10%. Thus, AQO is clearly superior to QAOA.

2 5 Nodes Problem in 4 Qubits

2.1 Representation of the solution

Overall we choose the same principle for the representation of our solution as before. This time we have a 5×5 matrix of binary variables: $(x_{i,t})$. If $x_{i,t}$ is 1, the group was at location i at time t, and else it was not. Again, we ensure the group only visits each place once, and only one at a time, by only allowing exactly one entry 1 per row and per column. As we now have the starting and ending node already given we know that $x_{0,0} = 1$ and $x_{4,4} = 1$. Knowing that there can only be one "1" in each row and collumn, this problem can basically be traced back to the 3 node problem. In order to break down the problem to less Qubits we notice, that the 3×3 matrix is fully determined by the upper 2×2 matrix in the 3×3 matrix. Our thought process can be seen on the powerpoint.

2.2 Cost Function and Constraints

The cost of a certain solution is given by $\sum_{i,j} \sum_{t=0,1} w_{ij} x_{i,t} x_{j,t+1}$. We penalize solutions using edges that do not exist by setting $w_{ij} = 1.2$ if the edge i, j does not exist. To ensure the group only visits each place once, and only one at a time, there has to be exactly one entry 1 per row and per column. These constraints can be represented by the cost functions $\sum_i (1 - \sum_t x_{i,t})^2$ and $\sum_t (1 - \sum_i x_{i,t})^2$. Our final cost function is the sum of the previous. We weight the constraints with a factor of 2. We generate the cost Hamiltonian by replacing x_{it} by the operator $1/2(1-\sigma^Z_{i,t})$ For $x_{i,t}$ which are not represented by our four qubits we use the constraints to infer their value and then plug the pauli-matrices into the resulting linear combination.

2.3 Mixing and Initial Hamiltonian

As a mixing Hamiltonian for QAOA and an initial Hamiltonian for AQO, respectively, we

chose $H = \sum_{i,t} \sigma_i^X$

2.4 Scaling of representation

Our representation is scalable and needs $(n-3)^2$ qubits to solve the TSP with n nodes.

2.5 Results

The probability distributions of results for QAOA and AQO can be found on the powerpoint. AQO has much better fidelity as it has an over 99 % chance of outputting the minimizing solution. This is an extremly good result for a complex problem. On the contrary, QAOA is fairly good at sorting solutions that do not fulfill the constraints but the chance of it outputting the minimizing solution is only around 18%. Thus, AQO is clearly superior to QAOA also for this problem.