Introduction to QUDO, Tensor QUDO and HOBO formulations: Qudits, Equivalences, Knapsack Problem, Traveling Salesman Problem and Combinatorial Games

Alejandro Mata Ali

Instituto Tecnológico de Castilla y León, Burgos, Spain

In this paper, we present a brief review and introduction to Quadratic Unconstrained D-ary Optimization (QUDO), Tensor Quadratic Unconstrained D-ary Optimization (T-QUDO) and Higher-Order Unconstrained Binary Optimization (HOBO) formulations for combinatorial optimization problems. We also show their equivalences. To help their understanding, we make some examples for the knapsack problem, traveling salesman problem and different combinatorial games. The games chosen to exemplify are: Hashiwokakero, N-Queens, Kakuro, Inshi no heya, and Peg Solitaire. Although some of these games have already been formulated in a QUBO formulation, we are going to approach them with more general formulations, allowing their execution in new quantum or quantum-inspired optimization algorithms. This can be an easier way to introduce these more complicated formulations for harder problems.

Contents

Alejandro Mata Ali: alejandro.mata@itcl.es

1 Introduction

Combinatorial optimization has been one of the most studied and applied fields in recent years due to its application into industry. It involves determining a combination of elements $\vec{x} = (x_0, x_1, \ldots, x_{n-1})$ which minimizes a function $C(\vec{x})$, known as the *cost function*. However, some of the most interesting problems are NP-Hard, without an efficient classical algorithm to solve them exactly. Interesting examples are the knapsack problem [1] and the traveling salesman problem [2], which are combinatorial optimization problems with use cases and easy to understand.

For this reason, one of the points of interest in recent quantum computing is combinatorial optimization. In this context, the most popular algorithm is *Quantum Approximate Optimization Algorithm* (QAOA) [3]. With this algorithm, an approximate solution to the minimization problem of a certain problem Hamiltonian can be obtained. A commonly used Hamiltonian is the *Ising Hamiltonian* given its easy implementation, which consists of interactions of pairs of variables by means of Z operators. For the treatment of classical combinatorial problems, the most commonly used formulation is the *Quadratic Unconstrained Binary Optimization* (QUBO) [4], since it can be translated directly into an Ising Hamiltonian. The cost function of the QUBO problem is

$$C(\vec{x}) = \sum_{i \leq j}^{n-1} Q_{ij} x_i x_j, \tag{1}$$

being x_i the i-th binary variable of the solution and Q the matrix that defines the problem.

A straightforward generalization is the *Quadratic Unconstrained D-ary Optimization* (QUDO), also called *Unconstrained Quadratic Programming* (UQP). In this case, the variables are positive integers instead of binary. Its equation is

$$C(\vec{x}) = \sum_{i \leq j}^{n-1} Q_{ij} x_i x_j + \sum_{i=0}^{n-1} D_i x_i, \tag{2}$$

being $\vec{D}$ the vector that defines the linear part of the problem, because in this cause $x_i^2 = x_i$ is not satisfied as in the binary case. This formulation requires the use of qudits, a technology currently in study [5, 6]. We can also generalize it even more with the *Tensor Quadratic Unconstrained D-ary Optimization* (T-QUDO), with the cost function

$$C(\vec{x}) = \sum_{i \leq j}^{n-1} Q_{i,j,x_i,x_j}, \tag{3}$$

being Q the cost tensor which depends on two variables values and which ones they are. This formulation has not been used in the state-of-the-art in quantum optimization with algorithms such as QAOA yet. Another possible generalization is to increase the order of the interactions. That is, instead of having a quadratic function, we have a function of order m

$$C(\vec{x}) = \sum_{i_0, i_1, \ldots, i_{m-1}=0}^{n-1} Q_{i_0, i_1, \ldots, i_{m-1}} x_{i_0} x_{i_1} \ldots x_{i_{m-1}}, \tag{4}$$

being Q the cost tensor which depends on which variables we evaluate, but not on their values, and the x_i are again the binary variables. This is the *Higher-Order Binary Optimization* (HOBO) formulation. This formulation has previously been studied [7] and is applicable to quantum devices [8, 9].

All of these formulations are convenient to approach different combinatorial problems, but they may be more difficult to understand, formulate, and implement than the QUBO formulation. One particular way to introduce oneself into them is solving easier problems, maybe toy models or academic ones, and obtaining knowledge from them. The most interesting and easy case are the mathematical combinatorial games.

Mathematical games have always been of great interest due to their underlying mathematical properties [10]. This has led to their study to learn more about them and possible solving strategies. They are also useful for testing new ideas [11, 12], technologies [13–16] and optimization algorithms [17, 18]. Combinatorial games can be defined as those games in which there is only one player who knows all the information about the system, they are deterministic and the goal is to determine a combination x that solves the problem and wins the game. This combination can be a configuration, in static games, or a combination of actions (sequence), in dynamic games. Previous work has presented QUBO formulations for Takuzu and N-Queens games [19], indicating the susceptibility of this kind of problems to being formulated in this way.

In this work, we present a brief introduction to QUDO, T-QUDO and HOBO formulations, exploring their limitations and uses. To understand them better, we create the formulations for the knapsack problem, the traveling salesman problem, the hashiwokakero, the N-Queens, the kakuro, the inshi no heya and the peg solitaire. This serves as examples of how to use these formulations. We optimize the number of variables needed and simplify the interactions.

2   Quadratic Unconstrained D-ary Optimization (QUDO)

As we have presented in the introduction, the Quadratic Unconstrained D-ary Optimization (QUDO) formulation consists in expressing a problem as an unconstrained combinatorial optimization problem with a cost function

$$C(\vec{x}) = \sum_{i \leq j}^{n-1} Q_{ij} x_i x_j + \sum_{i=0}^{n-1} D_i x_i, \tag{5}$$

where $\vec{x}$ are the positive integer variables to determine, and Q and $\vec{D}$ are the matrix and vector which define the problem. If we want, we can transform this problem into a QUBO formulation, simply by the substitution of the QUDO variables x_i with a set of binary variables y_{ik}

$$x_i = \sum_k 2^k y_{ik}. \tag{6}$$

So we can binarize the QUDO variables, introducing the new QUBO variables into Eq. (5), we can recover the QUBO cost function in Eq. 1. This implies that the main use of the QUDO formalism is the reduction of the needed resources, changing from qubits to qudits, because it is not more resilient than the QUBO formulation.

In this case, the first thing to note is that we cannot take advantage of certain properties of the QUBO formulation, such as $x_i^2 = x_i$. This forces us to introduce a linear term in the cost function to maintain its generality to the second order. The creation of the cost function does not have too much difficulty, as long as it is well adapted to this quadratic formalism, a fundamental condition for using this formalism as opposed to others. However, there are many constraints easily implementable in QUBO formulations that are not possible in QUDO, due to its higher dimensionality. We show several interesting cases:

1. Counting constraint: in the QUBO formulation, if we want only a certain number N of variables in $\mathcal{X}$ to take the value 1, $\sum_{i \in \mathcal{X}} x_i = N$ we impose a restriction term $(N - \sum_{i \in \mathcal{X}} x_i)^2$. In the weighted case, $\sum_{i \in \mathcal{X}} a_i x_i = W$, we simply multiply each variable x_i by its factor a_i, so the term is $(W - \sum_{i \in \mathcal{X}} a_i x_i)^2$. In the QUDO formulation, we can require that the sum of the values of the variables results in a particular value N, which is achieved with the same term. The same in the weighted case.

   If we want to determine that there are N variables with non-zero values, i.e. $\sum_{i \in \mathcal{X}} H(x_i) = N$, being H the Heaviside step function, we need to have other type of terms, not compatible with the QUDO formulation. This is due to the fact that we do not know previously the value of the other variables. In the QUBO case, we know that if N variables have non-zero values, they sum N, but in this case, they can have several different values.

2. Inequality counting constraint: if we want the weighted sum of a set of variables to be less than a certain amount Q, in the QUBO formulation we use slack variables $\vec{s}$ in a way that $\sum_{i \in \mathcal{X}} a_i x_i \leq Q$ becomes $\sum_{i \in \mathcal{X}} a_i x_i + \sum_j 2^j s_j = Q$, requiring only $\log_2(Q)$ slack variables (assuming $a_i \in \mathbb{Z}$). This is, the term is

$$\left(Q - \sum_{i=0} a_i x_i - \sum_{j=0}^{\log_2(Q)-1} 2^j s_j \right)^2 . \tag{7}$$

   Of course, you can avoid using slack variables with the unbalanced penalization [20], but we will keep them the rest of the paper for simplicity.

   In the QUDO case, we can impose the same restriction with a similar term. However, if the slack variables have d possible values, we only need $\log_d(Q)$ slack variables and the constraint is $\sum_{i \in \mathcal{X}} a_i x_i + \sum_j d^j s_j = Q$. The term is

$$\left(Q - \sum_{i=0} a_i x_i - \sum_{j=0}^{\log_d(Q)-1} d^j s_j \right)^2 . \tag{8}$$

   However, if we want to do it for the number of non-zero variables, we have the same problem as before.

3. Non-coincidence constraint: if two variables x_i and x_j cannot be non-zero at same time, in both QUBO and QUDO formulations, we impose a term $x_i x_j$. If we want to impose that they cannot be $x_i = a$ and $x_j = b$ at same time, i.e. $x_i = a \implies x_j \neq b$ and $x_j = b \implies x_i \neq a$, in QUBO we use

$$(1 - (-1)^a (x_i - a))(1 - (-1)^b (x_j - b)), \tag{9}$$

   where the $(-1)^a$ and $(-1)^b$ factors guarantee that their terms are $+1$ if $x_i \neq a$ and $x_j \neq b$ for all possible values of a and b. In the QUDO case, we cannot impose in general this constraint, because it depends on the value of the sum when $x_j \neq b$, which can be different values if x_j is not binary. This makes that the sign of the term depends on if the value of x_j is higher or lower than b.

4. Non-equality constraint: If we want them to satisfy $x_i \neq a \implies x_j = b$ and $x_j \neq b \implies x_i = a$, in the QUBO case we need a term

$$(-1)^{a+b}(x_i - a)(x_j - b). \tag{10}$$

However, in the QUDO case, due to the same reason as before, the sign of the term is defined by the value of the x_i and x_j respect to a and b, preventing this constraint to be implementable.

5. Implication constraint: if we have a restriction $x_i = a \implies x_j = b$, in the QUBO case we had a term

$$(1 - (-1)^a(x_i - a))(-1)^b(x_j - b), \tag{11}$$

so if $x_i = a$, the first term equals 1, else equals 0, and the second term is 0 if $x_j = b$ and 1 else. However, in the QUDO case there is the same problem with the sign of this term, making this constraint not implementable.

We can see that in the QUDO formalism, the restrictions are hard to implement with respect to the QUBO case. However, this does not mean that this formulation is useless. We show a use case in the knapsack problem, reducing the amount of resources and variables, and in the Hashiwokakero, which can be approximately formulated in this way.

Before checking the examples, we can ask how the QUDO formulation can be implemented in the QAOA formulation. The answer is simple. We need to implement the qudit version of the QAOA [6], making use of the concepts presented in [5] to create the cost hamiltonian gates. In this case, if we want a direct implementation, we do not need to implement the hamiltonian in an Ising-inspired way, like in the QUBO case. We can simply implement the exponential of the cost function operator, via qudit phase gates

$$P(\theta) = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & e^{i\theta} & 0 & \cdots & 0 \\ 0 & 0 & e^{i2\theta} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & e^{i(d-1)\theta} \end{pmatrix}, \quad P(\theta)_{kl} = \delta_{k,l}e^{i\theta k}, \tag{12}$$

for the linear terms, being a single qudit gate applied in j-th qudit with $\theta = \gamma D_j$,

$$P2(\theta) = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & e^{i\theta} & 0 & \cdots & 0 \\ 0 & 0 & e^{i2^2\theta} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & e^{i(d-1)^2\theta} \end{pmatrix}, \quad P2(\theta)_{kl} = \delta_{k,l}e^{i\theta k^2}, \tag{13}$$

for the single variable quadratic terms, being a single qudit gate applied in j-th qudit with $\theta = \gamma Q_{jj}$, and

$$PP(\theta)_{l,m,n,o} = \delta_{l,n}\delta_{m,o}e^{i\theta lm}, \tag{14}$$

for the other quadratic terms, being a two-qudits gate applied in j-th and k-th qudits with $\theta = \gamma Q_{jk}$.

2.1 Knapsack Problem

The knapsack problem [1] consists in choosing a set of elements to take into a knapsack, with the highest possible total value V and without exceeding the total capacity Q. We have N classes of objects, with a value v_i and weight w_i for the i-th class, which can be selected up to c_i times. In the QUBO formulation, the problem is expressed with a set of

variables x_{ij}, which indicate if we select the j-th object in the i-th class. This way, the problem is

$$\text{maximize } V(x) = \sum_{i=0}^{N-1} \sum_{j=0}^{c_i-1} v_i x_{ij},$$

$$\text{subject to } W(x) = \sum_{i=0}^{N-1} \sum_{j=0}^{c_i-1} w_i x_{ij} \leq Q. \tag{15}$$

In a QUBO cost function we use is

$$C(x) = -\sum_{i=0}^{N-1} \sum_{j=0}^{c_i-1} v_i x_{ij} + \lambda \left(Q - \sum_{i=0}^{N-1} \sum_{j=0}^{c_i-1} w_i x_{ij} - \sum_{k=0}^{\log_2(Q)-1} 2^k s_k \right)^2, \tag{16}$$

being s_k the slack variables. This requires $\sum_{i=0}^{N-1} c_i + \log_2(Q) = N \langle c \rangle + \log_2(Q)$ variables, $\langle c \rangle$ being the mean of $\vec{c}$. We can optimize the formulation with a condensed formulation, if all c_i satisfy $c_i = 2^{\xi_i}$ for some $\xi_i \in \mathbb{N}$,

$$\text{maximize } V(x) = \sum_{i=0}^{N-1} \sum_{j=0}^{\log_2(c_i)-1} v_i 2^j x_{ij},$$

$$\text{subject to } W(x) = \sum_{i=0}^{N-1} \sum_{j=0}^{\log_2(c_i)-1} w_i 2^j x_{ij} \leq Q, \tag{17}$$

being $x_{ij} = 1$ if we choose 2^j objects from the i-th class. In this case, the QUBO formulation is

$$C(x) = -\sum_{i=0}^{N-1} \sum_{j=0}^{\log_2(c_i)-1} v_i 2^j x_{ij} + \lambda \left(Q - \sum_{i=0}^{N-1} \sum_{j=0}^{\log_2(c_i)-1} w_i 2^j x_{ij} - \sum_{k=0}^{\log_2(Q)-1} 2^k s_k \right)^2, \tag{18}$$

requiring only $N \langle \log_2(c) \rangle + \log_2(Q)$ variables. We can relax the $c_i = 2^{\xi_i}$ condition mixing both formulations, with some extra variables.

In the QUDO case, if we have qudits with the correct dimension $c_i + 1$ for the main variables, and qudits of dimension d for the slack variables, we can express the problem as

$$\text{maximize } V(x) = \sum_{i=0}^{N-1} v_i x_i,$$

$$\text{subject to } W(x) = \sum_{i=0}^{N-1} w_i x_i \leq Q, \tag{19}$$

being $x_i \in [0, c_i]$ the number of times we choose an element of the i-th class.

The cost function in the QUDO formulation is

$$C(x) = -\sum_{i=0}^{N-1} v_i x_i + \lambda \left(Q - \sum_{i=0}^{N-1} w_i x_i - \sum_{k=0}^{\log_d(Q)-1} d^k s_k \right)^2, \tag{20}$$

requiring $N + \log_d(Q)$ variables. This is the opposite process described before, but now we have to get the QUDO from the QUBO. We can relax the dimensionality condition of the

qudits mixing formulations. If we have qudits of dimension d exclusively, and all $c_i = d^{\xi_i}$, the problem can be expressed with the cost function

$$C(x) = -\sum_{i=0}^{N-1}\sum_{j=0}^{\log_d(c_i)-1} v_i d^j x_{ij} + \lambda \left(Q - \sum_{i=0}^{N-1}\sum_{j=0}^{\log_d(c_i)-1} w_i d^j x_{ij} - \sum_{k=0}^{\log_d(Q)-1} d^k s_k \right)^2 , \quad (21)$$

being $x_{ij} \in [0, d-1]$ the number of times we choose a group of d^j elements of the i-th class. This requires $N \langle \log_d(c) \rangle + \log_d(Q)$ variables. We can also relax the $c_i = d^{\xi_i}$ condition mixing formulations. This formulation needs less variables than the QUBO one to give the same cost function.

2.2 Hashiwokakero

In this game, we have a grid of $N \times N$. In the set of vertexes $\mathcal{V}$ there are some nodes with numbers that indicate the number of edges that must connect them to their vertical and horizontal neighbor nodes. The node in position (i, j) has a number D_{ij}. An example of the problem is shown in Fig. 1 a. Also, each node can be connected to each other with zero, one, or two edges, and edges cannot cross each other. Finally, it can never be regions of vertexes isolated from others. That is, all cells must be connected to each other. A solution is shown in Fig. 1 b. We need to determine the connections between nodes to ensure that every node has its corresponding number of connections.
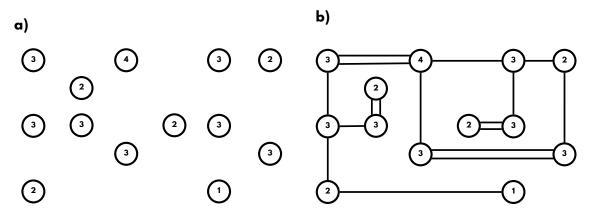

Figure 1: Hashiwokakero with a 5×6 grid. a) Problem instance, b) Solution.

As the variables to describe a solution naturally have three possible values, it is susceptible to be treated with a QUDO formulation. To properly formulate the problem in a simple way, we will redefine it in the following way. We have a set of N nodes, with D_i the objective number of connections of the i-th node with other ones, and $\mathcal{C}_i$ is the set of nodes which can be connected to the i-th node. In this case, the variable $x_{ij} \in \{0, 1, 2\}$ indicates the number of connections between the i-th node and the j-th node. The connections between (i, j) and (j, i) are the same, so $x_{ij} = x_{ji}$. In addition, each node cannot connect to itself, so all the $x_{ii} = 0$. This implies that we can avoid half of the variables. However, to simplify the expressions, we will keep all the variables, knowing that we only need to perform those substitutions in the equation. Finally, we only need one variable for every possible connection, avoiding impossible connections. As every node can have at most four connections and half of them are the same, we only need at most $2N$ variables with dimension three.

The cost function consists of two terms

$$C(\vec{x}) = \sum_{i=0}^{N-1} \left(D_i - \sum_{j=0|j\in\mathcal{C}_i}^{N-1} x_{ij} \right)^2 + \sum_{i,j=0}^{N-1} \left(x_{ij} \sum_{k,l\in\mathcal{X}_{ij}} x_{kl} \right), \tag{22}$$

being $\mathcal{X}_{ij}$ the edges that cross the edge that joins the nodes i and j.

With the first term, if the number of connections from i-th node to the other nodes with which it can be connected is exactly D_i, its value will be zero, and higher in the other cases. The second term imposes the non-cross constraint; we only need a term that forbids two edges that cross to activate at same time. The non-isolation constraint is not implementable in the QUDO formalism. We can generalize it to the case in which every node i can be connected to other j up to n_{ij} times simply by changing the dimension of the variables, $x_{ij} \in \{0, 1, \ldots, n_{ij} - 1\}$. Also, we can generalize the shape of the grid and the connectivity simply by changing the elements of $\mathcal{C}$.

3 Tensor Quadratic Unconstrained D-ary Optimization (T-QUDO)

We have seen the main advantages of the QUDO formulation in terms of reducing the required resources on resolutions. However, we also have seen a lot of limitations in constraint terms. The Tensor Quadratic Unconstrained D-ary Optimization (T-QUDO) is a generalization of QUDO which can overcome some of these limitations. Its cost function is

$$C(\vec{x}) = \sum_{i\leq j}^{n-1} Q_{i,j,x_i,x_j}, \tag{23}$$

where we can easily check that if $Q_{i,j,x_i,x_j} = Q_{ij}x_ix_j \,\forall i \neq j$ and $Q_{i,i,x_i,x_i} = D_ix_i + Q_{ii}x_i^2$, we recover Eq. (5). In this way, we can impose all the things we can impose in the QUDO formulation because we can use it as a part of the definition of the terms and more general things.

1. Counting constraint: if we want to determine that there are N variables with non-zero values, i.e. $\sum_{i\in\mathcal{X}} H(x_i) = N$, being H the Heaviside step function, we need a term

$$(N - \sum_{i\in\mathcal{X}} H(x_i))^2 = N^2 + \sum_{i,j\in\mathcal{X}} H(x_i)H(x_j) - 2N \sum_{i\in\mathcal{X}} H(x_i), \tag{24}$$

which is a T-QUDO expression if $Q_{i,j,x_i,x_j} = 2H(x_i)H(x_j)$ if $i \neq j$ (the 2 is for the symmetry) and $Q_{i,i,x_i,x_i} = H(x_i)^2 - 2NH(x_i)$ else. If we want the inequality expression, we can easily add slack variables.

2. Non-coincidence constraint: if we want to impose that $x_i = a \implies x_j \neq b$ in T-QUDO we use a term

$$Q_{i,j,a,b} = \lambda. \tag{25}$$

3. Non-equality constraint: if we want them to satisfy $x_i \neq a \implies x_j = b$, we use a term

$$Q_{i,j,\bar{a},\bar{b}} = \lambda, \tag{26}$$

being $\bar{c}$ all possible values for that variable but c.

4. Implication constraint: if we have a restriction $x_i = a \implies x_j = b$, we use a term

$$Q_{i,j,a,\bar{b}} = \lambda. \tag{27}$$

We can also implement the opposite implication, $x_i \neq a \implies x_j \neq b$, with a term

$$Q_{i,j,\bar{a},b} = \lambda. \tag{28}$$

As we can observe, we can easily implement all the restrictions that we were unable to implement in the QUDO formulation.

The last important problem of the formulation is the implementation in the QAOA. In this case, we need to implement the exponential of the cost hamiltonian. To do this, we must use controlled focus phase gates. We define a focus phase gate as

$$FP(\theta, b)_{lm} = \delta_{lm} \left(\sum_{c \neq b} \delta_{lc} + \delta_{lb} e^{i\theta} \right). \tag{29}$$

To impose the $Q_{i,j,a,b}$ term, we need to make a controlled focus phase gate $FP(\theta, b)$, with control in the i-th qudit (activated with the state $|a\rangle$) and target in the j-th qudit. We can also impose it with an auxiliary qubit in state $|1\rangle$, with a double controlled phase gate, controlled by the i-th and j-th qudits in states $|a\rangle$ and $|b\rangle$ and target the auxiliary qubit.

We will show the interest of this formulation with the Traveling Salesman Problem, due to its applications, and with the N-queens problem, because it is not feasible with the QUDO formulation. The T-QUDO of the Knapsack problem is not needed, due to the fact that involves the same variables as in the QUDO case.

3.1 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) [2] consists in finding a path between the V nodes of a graph with the lowest possible cost, without repeating any node, and returning to the starting point. We choose as variables $x_t \in [0, V-1]$ being the node chosen in the timestep t, giving a path $\vec{x} = (x_0, x_1, \ldots, x_{V-1})$. This implies the need for V variables, which can be reduced to $V - 1$ fixing the first node of the path due to the symmetries. If the path between the i-th and j-th nodes has a cost E_{ij}, the total cost of the path is

$$C(\vec{x}) = \sum_{t=0}^{V-2} E_{x_t, x_{t+1}} + E_{x_{V-1}, x_0}. \tag{30}$$

If two nodes are not connected, its cost will be $E_{ij} = \lambda$. We can check that this is a particular case of the T-QUDO formulation without the indexes indicating the position of the variables. To impose the restriction of the non-repetition, we can use the prime numbers decomposition. If we have the list of the first V prime numbers $\vec{p} = (1, 2, 3, 5, 7, \ldots)$, we know that every feasible path satisfies

$$P_V = \prod_{t=0}^{V-1} p_{x_t} = \prod_{i=0}^{V-1} p_i, \tag{31}$$

and due to the uniqueness of the prime decomposition of P_V, each prime number must appear only once in the product, so every non-correct path has a different product result. If we apply logarithms on both sides

$$\log_2(P_V) = \sum_{t=0}^{V-1} \log_2(p_{x_t}) = \sum_{i=0}^{V-1} \log_2(p_i), \tag{32}$$

so this sum has this value only for the correct paths. This way, the restriction term must be

$$C_r = \left(\sum_{i=0}^{V-1} \log_2(p_i) - \sum_{t=0}^{V-1} \log_2(p_{x_t}) \right)^2 = \left(\sum_{i=0}^{V-1} \log_2 \left(\frac{p_i}{p_{x_t}} \right) \right)^2 = \left(\sum_{i=0}^{V-1} \log_2 \left(\frac{p_{x_t}}{p_i} \right) \right)^2, \quad (33)$$

which increases the cost of the paths which repeat nodes. We can easily see that this is a T-QUDO expression. With both terms, the T-QUDO formulation of the TSP is

$$C(\vec{x}) = \sum_{t=0}^{V-2} E_{x_t,x_{t+1}} + E_{x_{V-1},x_0} + \lambda \left(\sum_{i=0}^{V-1} \log_2 \left(\frac{p_{x_t}}{p_i} \right) \right)^2. \quad (34)$$

We can easily transform this cost function for the TSP generalization with a cost that changes with the timestep. In this case, the cost of the travel from i-th vertex to j-th vertex at t-th timestep is $E_{t,i,j}$, so the T-QUDO cost function is

$$C(\vec{x}) = \sum_{t=0}^{V-2} E_{t,x_t,x_{t+1}} + E_{V-1,x_{V-1},x_0} + \lambda \left(\sum_{i=0}^{V-1} \log_2 \left(\frac{p_{x_t}}{p_i} \right) \right)^2. \quad (35)$$

The non-repetition also can be implemented comparing the value of each pair of variables, with terms

$$C_r = \lambda \sum_{t=0}^{V-1} \sum_{t'>t}^{V-1} \delta_{x_t,x_{t'}}, \quad (36)$$

being the total cost function

$$C(\vec{x}) = \sum_{t=0}^{V-2} E_{t,x_t,x_{t+1}} + E_{V-1,x_{V-1},x_0} + \lambda \sum_{t=0}^{V-1} \sum_{t'>t}^{V-1} \delta_{x_t,x_{t'}}. \quad (37)$$

We can check that this is also a T-QUDO expression.

3.2 N-Queens

The N-Queens problem consists in putting N queens on a chessboard of $N \times N$, in a way that no queen threatens others. A solution is shown in Fig. 2. The constraints can be expressed as

- Rows condition: there can only be one queen per row.

- Columns condition: there can only be one queen per column.

- Diagonal condition: there cannot be two queens on the same diagonal.

This problem has previously been formulated in a QUBO formulation with more generalizations [19]. However, if we want to reduce the amount of resources needed, we can use as variables x_i as the position where the i-th row queen is, we can formulate the T-QUDO formulation as

$$C(\vec{x}) = \sum_{i<j}^{N-1} C_{i,j,x_i,x_j}, \quad (38)$$

being i the row of the first queen and j the row of the second queen. This requires N variables instead of N^2.

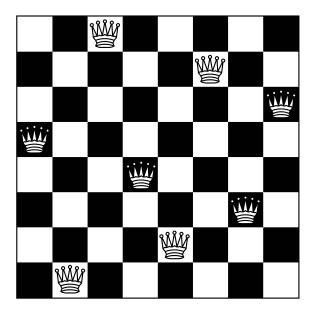The cost tensor C must increase the cost of illegal configurations. The non-zero elements are:

Figure 2: N-Queens solution with a 8×8 chessboard.

1. Column condition: if two queens are from different rows, their column must be different. So, if $i \neq j$, then $x_i \neq x_j$:

$$C_{i,j,a,a} = 1, \ \forall a \in [0, N-1] \wedge i \neq j. \tag{39}$$

2. Diagonal condition: if two queens are from different rows, separated by k positions, their column cannot be the same separated by k positions in the same direction. So, if $j = i + k$, then $x_j \neq x_i + k$ nor $x_j \neq x_i - k$:

$$C_{i,i+k,a,a\pm k} = 1, \ \forall i, k, a. \tag{40}$$

All other terms of the tensor are zero. This way, the final expression is

$$C(\vec{x}) = \sum_{i<j}^{N-1} \left(\delta_{x_i, x_j} + \delta_{x_i, x_j + (j-i)} + \delta_{x_i, x_j - (j-i)} \right). \tag{41}$$

3.3 Kakuro

This game consists of a board with $N \times N$ cells, some of them not available (black cells) and others available (white cells) to have a number from 1 to M. Each portion of a row or column has a number that tells the sum of the numbers of that portion of a row/column. Additionally, each number can appear only once in each portion of a row and column. An example of a solution is shown in Fig. 3.

It seems natural to use as variables x_{ij} as the number in the cell of the i-th row and j-th column. The k-th portion of the i-th row is defined by the cells in $\mathcal{R}_{ik}$ and the k-th portion of the j-th column is defined by the cells in $\mathcal{C}_{ik}$. The portion $\mathcal{R}_{ik}$ has an associated sum number S_{ik}^r and the portion $\mathcal{C}_{jk}$ has an associated sum number S_{jk}^c. If we have qudits of dimension M, we need only as variables the white cells, at most N^2. Otherwise, we can use more variables with the encoding previously shown.

With these definitions, we can define the T-QUDO formulation. First, we define the term that implements the sum constraint. We know that $\sum_{i',j' \in \mathcal{R}_{ik}} x_{i'j'} = S_{ik}^r \ \forall i, k$, and
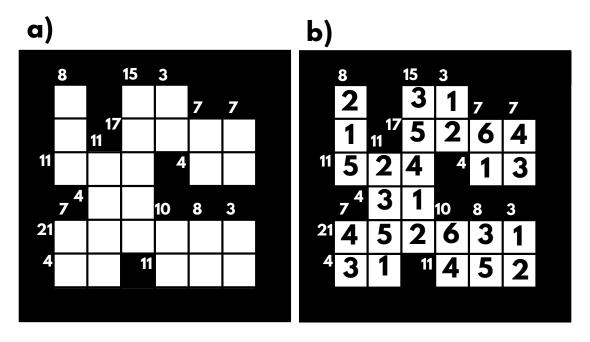
Figure 3: a) Kakuro problem for a 6×6 board, b) solution for a 6×6 Kakuro.

$\sum_{i',j' \in \mathcal{C}_{jk}} x_{i'j'} = S^c_{jk} \; \forall j, k$, so the constraint term is

$$C_s = \sum_{i=0}^{N-1} \sum_k \left(\sum_{i',j' \in \mathcal{R}_{ik}} x_{i'j'} - S^r_{ik} \right)^2 + \sum_{j=0}^{N-1} \sum_k \left(\sum_{i',j' \in \mathcal{C}_{jk}} x_{i'j'} - S^c_{jk} \right)^2. \qquad (42)$$

This is a QUDO term. The second restriction, the non-repetition one, is a QUDO term with compares each pair of variables in each region. Their non-zero terms are

$$\begin{aligned}
Q_{(i',j'),(l',m'),a,a} &= 1 \; \forall i', j' \in \mathcal{R}_{ik}, \forall l', m' \neq i', j' \in \mathcal{R}_{ik}, \forall i, k, a, \\
Q_{(i',j'),(l',m'),a,a} &= 1 \; \forall i', j' \in \mathcal{C}_{jk}, \forall l', m' \neq i', j' \in \mathcal{C}_{jk}, \forall j, k, a.
\end{aligned} \qquad (43)$$

This makes the non-repetition term

$$C_r = \sum_{i=0}^{N-1} \sum_k \sum_{i',j' \in \mathcal{R}_{ik}} \sum_{l',m' \neq i',j' \in \mathcal{R}_{ik}} \delta_{x_{i',j'}, x_{l',m'}} + \sum_{j=0}^{N-1} \sum_k \sum_{i',j' \in \mathcal{C}_{jk}} \sum_{l',m' \neq i',j' \in \mathcal{C}_{jk}} \delta_{x_{i',j'}, x_{l',m'}}. \qquad (44)$$

The total cost function is

$$C(x) = \sum_{i=0}^{N-1} \sum_k \left(\sum_{i',j' \in \mathcal{R}_{ik}} x_{i'j'} - S^r_{ik} \right)^2 + \sum_{j=0}^{N-1} \sum_k \left(\sum_{i',j' \in \mathcal{C}_{jk}} x_{i'j'} - S^c_{jk} \right)^2 + $$
$$+ \sum_{i=0}^{N-1} \sum_k \sum_{i',j' \in \mathcal{R}_{ik}} \sum_{l',m' \neq i',j' \in \mathcal{R}_{ik}} \delta_{x_{i',j'}, x_{l',m'}} + \sum_{j=0}^{N-1} \sum_k \sum_{i',j' \in \mathcal{C}_{jk}} \sum_{l',m' \neq i',j' \in \mathcal{C}_{jk}} \delta_{x_{i',j'}, x_{l',m'}}. \qquad (45)$$

We can check that this is a T-QUDO expression.

3.4 Inshi No Heya

The inshi no heya is a game similar to kakuro, but in this case the $N \times N$ board is divided into regions (originally rectangular, but we can generalize it to arbitrary shapes). We need

to give a number to each cell in the board, without repeating them in the same row or column. The numbers placed in the k-th region $\mathcal{R}_k$ must sum the value S_k. An example is shown in Fig. 4.
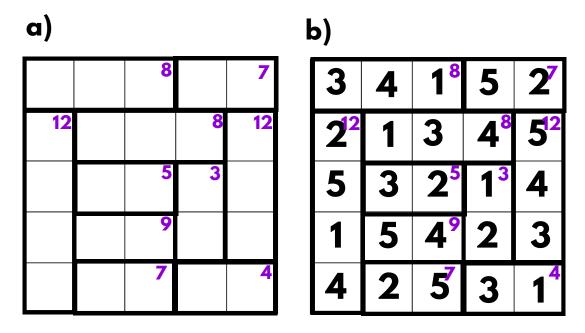

Figure 4: a) Ishin no heya problem for a 5×5 board, b) Solution for the 5×5 Ishin no heya.

We use variables x_{ij} as the value in the cell of the i-th row and j-th column. The first constraint is the sum of all the cells in a region. This is implemented with a QUDO term

$$C_s = \sum_k \left(\sum_{i',j' \in \mathcal{R}_k} x_{i'j'} - S_k \right)^2 . \tag{46}$$

The repetition condition is implemented with a T-QUDO term

$$C_r = \sum_{i,j=0}^{N-1,N-1} \left(\sum_{i'>i}^{N-1} \delta_{x_{i,j},x_{i',j}} + \sum_{j'>j}^{N-1} \delta_{x_{i,j},x_{i,j'}} \right) . \tag{47}$$

The total cost function is

$$C(x) = \sum_k \left(\sum_{i',j' \in \mathcal{R}_k} x_{i'j'} - S_k \right)^2 + \sum_{i,j=0}^{N-1,N-1} \left(\sum_{i'>i}^{N-1} \delta_{x_{i,j},x_{i',j}} + \sum_{j'>j}^{N-1} \delta_{x_{i,j},x_{i,j'}} \right) . \tag{48}$$

4 Higher-Order Binary Optimization (HOBO)

We have seen the formalisms to work with qudits, but they are not directly implementable in qubit devices. Also, these formalisms allows us to work only with interactions between pairs of variables, but not more complex ones. These are good reasons to explore higher-order formalisms. The Higher-Order Binary Optimization (HOBO) is the simplest one that we can formulate, with a m-order cost function

$$C(\vec{x}) = \sum_{i_0,i_1,\dots,i_{m-1}=0}^{n-1} Q_{i_0,i_1,\dots,i_{m-1}} x_{i_0} x_{i_1} \dots x_{i_{m-1}}, \tag{49}$$

with its lower order terms implicit by the $x_i^2 = x_i$ property due to the binary variables. Due to the fact that QUBO is a particular case, HOBO can implement everything implementable in QUBO, implying also everything implementable in QUDO. So, the main advantage of this formulation is the additional possible interactions. It has been previously studied and implemented in [7–9]. We will see its potential with the popular game Peg Solitaire.

The interesting thing is that this formulation is compatible with the T-QUDO formulation. In the second QAOA implementation of the proposed T-QUDO, we use two qudit controls to apply a phase gate to an ancilla qubit. However, we can change each of the qudit controls to a set of qubit controls, which creates the binary representation of the states of the qudit. In this way, if we had a qudit of dimension d_0 and another qudit of dimension d_1 which controls the phase gate, with control states $|a\rangle$ and $|b\rangle$, we substituted them with $\log_2(d_0)$ and $\log_2(d_1)$ qubits, respectively, with control states the binary representation of $|a\rangle$ and $|b\rangle$. This means that we can interpret it as a higher-order interaction between the qubits. In this way, we can interpret that the T-QUDO and the HOBO are equivalent.

4.1 Peg Solitaire

Peg solitaire consists on a board with a certain shape, with a set of balls in each cell but one, as shown in Fig. 5 a. In each turn, the player can make a movement. Each movement consists in moving a ball (the player chooses it in each turn) from its cell to an empty cell, crossing a cell with another ball. The movement involves two adjacent cells in the same direction as the moving ball cell. This means that if the ball moved starts in the cell (i, j), there is a ball in the cell $(i, j+1)$ and moves to the right, its new position is $(i, j+2)$. An example is shown in Fig. 5 b. It can only move in directions with existing balls, and after movement, the crossed ball disappears from the board, as shown in Fig. 5 c. The objective is to make a set of moves that end in having a unique ball in the starting empty cell and no other balls.
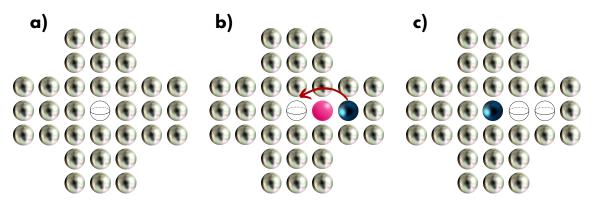

Figure 5: a) Initial configuration of the peg solitaire, b) movement of the green ball across the red ball, c) result from the movement.

We present a HOBO formulation, so our variables will be x_{ijt}, which is the state of the cell in row i and column j at the timestep t. If the cell is empty, its value is zero, else it is value is one. For simplicity, we will work with a square board, but the process and HOBO are easily generalizable to other shapes. If we have a square board of $N \times N$ dimensions, we have N^2 cells and $N^2 - 1$ balls. If we consider the starting configuration as step zero, we need $T = N^2 - 1$ time steps to remove the $N^2 - 2$ balls. So we need $(N^2 - 1)N^2 = N^4 - N^2$ variables. However, we will not optimize the first and last steps,

so they are only $(N^2 - 3)N^2 = N^4 - 3N^2$.

Moreover, we need more auxiliary variables $a_{ijtk_1k_2}$, which indicate that in the timestep t we move the ball in (i,j) in the vertical direction k_1 (0 no movement, 1 down, 2 up) and k_2 (0 no movement, 1 left, 2 right). Obviously, there are no variables a_{ijt00}, because they imply no movement, and neither diagonal movement variables. This means that there are only a_{ijt01}, a_{ijt02}, a_{ijt10} and a_{ijt20}. For simplicity, we say that the others are always zero. The number of a variables is four times higher than the number of x variables, resulting in a combined total of $5N^4 - 12N^2$ variables. We can optimize a little bit more the number of variables taking into account the forbidden moves in the first timesteps.

First, we need to impose the initialization and finalization conditions. If the empty cell at the beginning is (I, J), we have that

$$x_{ij0} = 1 - \delta_{i,I}\delta_{j,J}, \qquad x_{ij,T-1} = \delta_{i,I}\delta_{j,J}. \tag{50}$$

The configuration of each step t depends on the configuration of the previous step.

The first restriction is the number of balls per timestep, because one ball disappears each turn. So, in step t there must be $N^2 - 1 - t$ balls. This means the first term of the HOBO is

$$C_q = \sum_{t=0}^{T-1} \left(\left(N^2 - 1 - t \right) - \sum_{i,j=0}^{N-1} x_{ijt} \right)^2. \tag{51}$$

This is a second-order QUBO term.

The second restriction is the continuity. Only three cells have changed between timesteps. The number of coincidences must be exactly $N^2 - 3$, so its term is

$$C_c = \sum_{t=1}^{T-1} \left(\left(N^2 - 3 \right) - \sum_{i,j=0}^{N-1} (x_{ijt}x_{ij,t-1} + (1 - x_{ijt})(1 - x_{ijt-1})) \right)^2. \tag{52}$$

This is a 4-order HOBO term. The first summand takes into account when the cell remains in one, and the second when it remains in zero.

The third is the action constraint. We can take only one action per timestep. We can impose it with a term

$$C_a = \sum_{t=0}^{T-2} \left(1 - \sum_{ijk_1k_2} a_{ijtk_1k_2} \right)^2. \tag{53}$$

This is a second-order QUBO term.

The final restriction is the action effect. This means, if we have $a_{ijtk_1k_2} = 1$, then

$$\begin{aligned} x_{ijt} = x_{i+b_{k_1},j+b_{k_2},t} &= 1, \\ x_{i+2b_{k_1},j+2b_{k_2},t} &= 0, \\ x_{ijt} = x_{i+b_{k_1},j+b_{k_2},t+1} &= 0, \\ x_{i+2b_{k_1},j+2b_{k_2},t+1} &= 1, \end{aligned} \tag{54}$$

being the displacement vector $\vec{b} = (0, -1, 1)$.

To implement this, we use a term

$$\begin{aligned} C_m = \sum_{ijtk_1k_2} a_{ijtk_1k_2} &\left[(x_{ijt}x_{i+b_{k_1},j+b_{k_2}}(1 - x_{i+2b_{k_1},j+2b_{k_2},t}) - 1)^2 + \right. \\ &\left. + ((1 - x_{ijt})(1 - x_{i+b_{k_1},j+b_{k_2},t+1})x_{i+2b_{k_1},j+2b_{k_2},t+1} - 1)^2 \right]. \end{aligned} \tag{55}$$

This is a 7-order HOBO interaction.

The global HOBO formulation is a 7-order interaction cost function

$$C(x,a) = C_q + C_c + C_a + C_m =$$

$$= \sum_{t=0}^{T-1} \left(\left(N^2 - 1 - t\right) - \sum_{i,j=0}^{N-1} x_{ijt} \right)^2 +$$

$$+ \sum_{t=1}^{T-1} \left(\left(N^2 - 3\right) - \sum_{i,j=0}^{N-1} \left(x_{ijt}x_{ij,t-1} + (1 - x_{ijt})(1 - x_{ijt-1})\right) \right)^2 +$$

$$+ \sum_{t=0}^{T-2} \left(1 - \sum_{i,j=0}^{N-1,N-1} \sum_{k_1,k_2=0}^{2,2} a_{ijtk_1k_2} \right)^2 +$$

(56)

$$+ \sum_{i,j,t=0}^{N-1,N-1,T-2} \sum_{k_1,k_2=0}^{2,2} a_{ijtk_1k_2} \left[\left(x_{ijt}x_{i+b_{k_1},j+b_{k_2}}(1 - x_{i+2b_{k_1},j+2b_{k_2},t}) - 1\right)^2 +$$

$$+ \left((1 - x_{ijt})(1 - x_{i+b_{k_1},j+b_{k_2},t+1})x_{i+2b_{k_1},j+2b_{k_2},t+1} - 1\right)^2 \right].$$

The correct combination of movements which solves the problem has a cost of zero.

To generalize to arbitrary shape boards, with M total cells in a set $\mathcal{C}$, we need $O(M^2)$ variables, $T = M - 1$ timesteps and the formulation is

$$C(x,a) = C_q + C_c + C_a + C_m =$$

$$= \sum_{t=0}^{M-1} \left((M - 1 - t) - \sum_{i,j \in \mathcal{C}} x_{ijt} \right)^2 +$$

$$+ \sum_{t=1}^{M-1} \left((M - 3) - \sum_{i,j \in \mathcal{C}} \left(x_{ijt}x_{ij,t-1} + (1 - x_{ijt})(1 - x_{ijt-1})\right) \right)^2 +$$

(57)

$$+ \sum_{t=0}^{M-2} \left(1 - \sum_{i,j \in \mathcal{C}} \sum_{k_1,k_2=0}^{2,2} a_{ijtk_1k_2} \right)^2 +$$

$$+ \sum_{i,j \in \mathcal{C}} \sum_{t=0}^{M-2} \sum_{k_1,k_2=0}^{2,2} a_{ijtk_1k_2} \left[\left(x_{ijt}x_{i+b_{k_1},j+b_{k_2}}(1 - x_{i+2b_{k_1},j+2b_{k_2},t}) - 1\right)^2 +$$

$$+ \left((1 - x_{ijt})(1 - x_{i+b_{k_1},j+b_{k_2},t+1})x_{i+2b_{k_1},j+2b_{k_2},t+1} - 1\right)^2 \right].$$

5 Conclusions

We have presented and studied the QUDO, T-QUDO and HOBO formulations, with their advantages and limitations. We also applied them for the knapsack problem, the traveling salesman problem, and five interesting combinatorial games. As we show, many problems can be formulated in an easier and more efficient way with these more complex formalisms. However, it may be difficult if we do not choose the correct formulation. Moreover, we can reformulate QUDO into QUBO and implement the T-QUDO as a HOBO problem, allowing their application in current algorithms without any modification. Moreover, we show that they are implementable in the QAOA with small modifications.

Future lines of research may include a deeper analysis of the details of implementation, optimization of interaction types, the design of efficient hardware architectures, and the application to other problems.

Acknowledgment

References

[1] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations.* John Wiley & Sons, Inc., 1990.

[2] David L. Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. 2007.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014. URL https://arxiv.org/abs/1411.4028.

[4] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models, 2019. URL https://arxiv.org/abs/1811.11538.

[5] Yuchen Wang, Zixuan Hu, Barry C. Sanders, and Sabre Kais. Qudits and high-dimensional quantum computing. *Frontiers in Physics*, 8, November 2020. ISSN 2296-424X. DOI: 10.3389/fphy.2020.589504. URL http://dx.doi.org/10.3389/fphy.2020.589504.

[6] Yannick Deller, Sebastian Schmitt, Maciej Lewenstein, Steve Lenk, Marika Federer, Fred Jendrzejewski, Philipp Hauke, and Valentin Kasper. Quantum approximate optimization algorithm for qudit systems. *Physical Review A*, 107(6), June 2023. ISSN 2469-9934. DOI: 10.1103/physreva.107.062410. URL http://dx.doi.org/10.1103/PhysRevA.107.062410.

[7] Krzysztof Domino, Akash Kundu, Özlem Salehi, and Krzysztof Krawiec. Quadratic and higher-order unconstrained binary optimization of railway rescheduling for quantum computing. *Quantum Information Processing*, 21(9):337, Sep 2022. ISSN 1573-1332. DOI: 10.1007/s11128-022-03670-y. URL https://doi.org/10.1007/s11128-022-03670-y.

[8] Zoe Verchere, Sourour Elloumi, and Andrea Simonetto. Optimizing Variational Circuits for Higher-Order Binary Optimization . In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 19–25, Los Alamitos, CA, USA, September 2023. IEEE Computer Society. DOI: 10.1109/QCE57702.2023.00011. URL https://doi.ieeecomputersociety.org/10.1109/QCE57702.2023.00011.

[9] Daniel Goldsmith and Joe Day-Evans. Beyond qubo and hobo formulations, solving the travelling salesman problem on a quantum boson sampler, 2024. URL https://arxiv.org/abs/2406.14252.

[10] Daniel Andersson. Hashiwokakero is np-complete. *Information Processing Letters*, 109(19):1145–1146, 2009. ISSN 0020-0190. DOI: https://doi.org/10.1016/j.ipl.2009.07.017. URL https://www.sciencedirect.com/science/article/pii/S002001900900235X.

[11] Omid Moghimi and Amin Amini. A novel approach for solving the n-queen problem using a non-sequential conflict resolution algorithm. *Electronics*, 13(20), 2024. ISSN 2079-9292. DOI: 10.3390/electronics13204065. URL https://www.mdpi.com/2079-9292/13/20/4065.

[12] Jordan Bell and Brett Stevens. A survey of known results and research areas for n-queens. *Discrete Mathematics*, 309(1):1–31, 2009. ISSN 0012-365X. DOI: https://doi.org/10.1016/j.disc.2007.12.043. URL https://www.sciencedirect.com/science/article/pii/S0012365X07010394.

[13] Pascal Fua and Krzysztof Lis. Comparing python, go, and c++ on the n-queens problem, 2020. URL `https://arxiv.org/abs/2001.02491`.

[14] Santhosh G S, Piyush Joshi, Ayan Barui, and Prasanta K. Panigrahi. A quantum approach to solve n-queens problem, 2023. URL `https://arxiv.org/abs/2312.16312`.

[15] Bouneb Zine El Abidine. An incremental approach to the n-queen problem with polynomial time. *Journal of King Saud University - Computer and Information Sciences*, 35(3):1–7, 2023. ISSN 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2023.02.002. URL `https://www.sciencedirect.com/science/article/pii/S1319157823000319`.

[16] Tao Zhang, Wei Shu, and Min-You Wu. Optimization of n-queens solvers on graphics processors. In Olivier Temam, Pen-Chung Yew, and Binyu Zang, editors, *Advanced Parallel Processing Technologies*, pages 142–156, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-24151-2.

[17] Leandro C. Coelho, Gilbert Laporte, Arinei Lindbeck, and Thibaut Vidal. Benchmark instances and branch-and-cut algorithm for the hashiwokakero puzzle, 2019. URL `https://arxiv.org/abs/1905.00973`.

[18] Christopher Jefferson, Angela Miguel, Ian Miguel, and S. Armagan Tarim. Modelling and solving english peg solitaire. *Computers & Operations Research*, 33(10):2935–2959, 2006. ISSN 0305-0548. DOI: https://doi.org/10.1016/j.cor.2005.01.018. URL `https://www.sciencedirect.com/science/article/pii/S0305054805000195`. Part Special Issue: Constraint Programming.

[19] Alejandro Mata Ali and Edgar Mencia. A qubo formulation for the generalized linkedin queens and takuzu/tango game, 2024. URL `https://arxiv.org/abs/2410.06429`.

[20] J A Montañez-Barrera, Dennis Willsch, A Maldonado-Romo, and Kristel Michielsen. Unbalanced penalization: a new approach to encode inequality constraints of combinatorial problems for quantum optimization algorithms. *Quantum Science and Technology*, 9(2):025022, April 2024. ISSN 2058-9565. DOI: 10.1088/2058-9565/ad35e4. URL `http://dx.doi.org/10.1088/2058-9565/ad35e4`.