QUANTUM COMBINATORIAL DESIGN

Gopsill, James (1,2);
Johns, Guy (2);
Hicks, Ben (1)

1: Design Manufacturing Futures Lab, School of Civil, Aerospace and Mechanical Engineering, University of Bristol, UK;
2: Centre for Modelling and Simulation, Bristol, UK

ABSTRACT

Combinatorial Design such as configuration design, design optioneering, component selection, and generative design, is common across engineering. Generating solutions for a combinatorial design task often involves the application of classical computing solvers that can either map or navigate design spaces. However, it has been observed that classical computing resource power-law scales with many design space models. This observation suggests classical computing may not be capable of modelling our future design space needs.

To meet future design space modelling needs, this paper examines quantum computing and the characteristics that enables its resources to scale polynomially with design space size. The paper then continues to present a combinatorial design problem that is subsequently represented, constrained and solved by quantum computing. The results of which are the derivation of an initial set of circuits that represent design space constraints. The study shows the game-changing possibilities of quantum computing as an engineering design tool and is the start of an exciting new journey for design research.

Keywords: Constraint modelling, Robust design, Simulation, Quantum Computing, Design Space

Contact:
Gopsill, James
University of Bristol
Mechanical Engineering
United Kingdom
james.gopsill@bristol.ac.uk

Cite this article: Gopsill, J., Johns, G., Hicks, B. (2021) 'Quantum Combinatorial Design', in *Proceedings of the International Conference on Engineering Design (ICED21),* Gothenburg, Sweden, 16-20 August 2021. DOI:10.1017/pds.2021.512

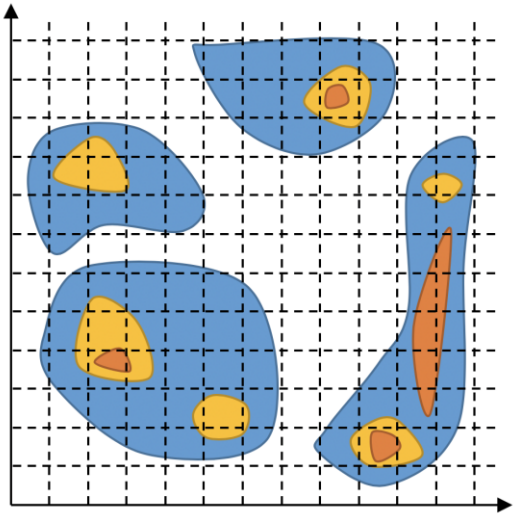
1 INTRODUCTION

Configuration design, design optioneering, component selection, and generative design are common Engineering Design tasks. Examples include developing structures that satisfy construction requirements, deriving mechanisms that have desired motion profiles and generating toolpaths that account for manufacturing constraints (Feng and Kusiak, 1995; Gopsill, Shindler, et al., 2017; Mathias et al., 2017; Medland and Mullineux, 1993). Together, they represent forms of combinatorial design.

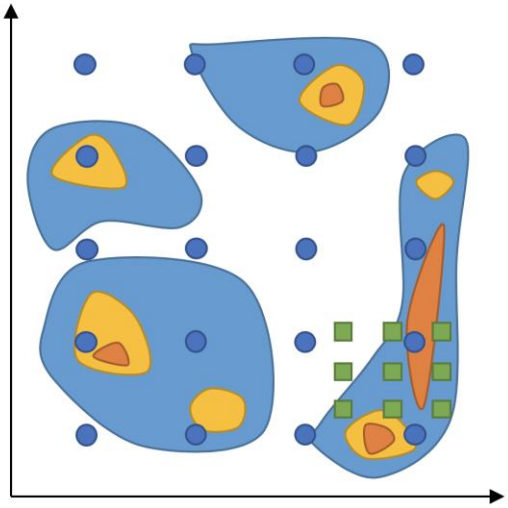
Combinatorial design can be generally described as a function of characteristics, x_n, $f(x_1, \cdots, x_n)$ that define a design space, such as a parametric Computer Aided Design (CAD) model, a function of constraints, y_n, $f(y_1, \cdots, y_n)$, such as fits, limits and tolerances, and more often than not, an objective function that determines how well a solution performs. The task is to then identify solutions in the design space that satisfy the constraints, and in the presence of an objective function, select the optimal solution(s) for the problem (Biskjaer et al., 2014). However, the challenge lies in the increasingly open design spaces we wish to explore (e.g., Additive Manufacturing) and increasing number of constraints being imposed on our problems. This is coupled with dimensional representation (e.g., continuous, discrete, ordinal, range, and dependency), resulting in complex hyper-parameter design spaces that engineers need to navigate to discover a solution.

Over the years, methods such as constraint-based modelling, hyper-parameter optimisation, genetic algorithms, linear programming, and generative approaches have been developed to support engineers in traversing design spaces (Lin and Chen, 2002; Sapossnek et al., 1991; Zhang and Xie, 2014). These methods tap into the power of classical computing to explore design spaces to a greater extent than that of an engineer manually generating and evaluating solutions. In general, these methods can be categorised as either design space mapping or directed search (Figure 1).

(a) Optimum solutions to our design problem.



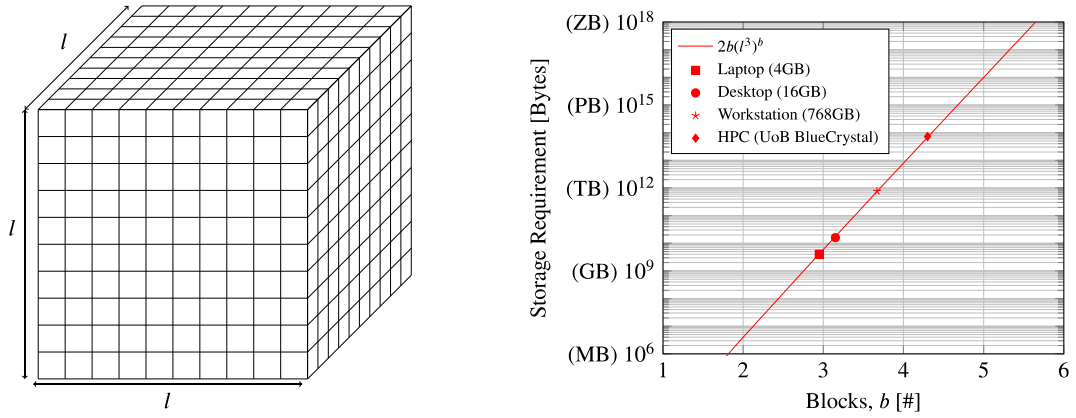
(b) Mapping. The design space is computed for a range of input parameters (e.g., grid) with each cross-over representing a solution. These computed solutions are stored for later retrieval. Challenges arise in the computation and storage of enough solutions that can suitably represent the design space.



(c) Directed search. Circles represent an initial grid-search that is later refined, based on the results, to a local search represented by the green squares. This will occur multiple times until a desired solution is reached.

Figure 1. Combinatorial design modelling approaches.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press



(a) Cube whose length, width and depth is integer l.

(b) Power-law scaling storage requirement.

Figure 2. Modelling a design space cube.

Design space mapping methods populate the design space with solutions that satisfy the constraints (Figure 1b). This can be achieved using techniques such as grid/random sampling and/or contour traversal, to produce a map of the design space. The map can then be queried, and solutions selected to meet the designer's needs. Curve/surface fitting algorithms (e.g., linear and polynomial interpolation) are often employed to inference between known results. The challenge for design space mapping is in generating enough solutions in the permitted product development timeframe to fully characterise the design space. This is often countered by increasing computational resource (power, storage).

Directed search couples the objective function and design space (Figure 1c). Methods start by populating the design space with solutions, typically through grid, random or Bayes sampling. The results then inform subsequent iterations where algorithms, such as gradient descent/ascent and genetic, are employed. This leads to a more focused navigation of the design space. Challenges exist in identifying local optima, traversal of disjoint design spaces, and convergence to an optimal solution within the permitted product development timeframe. This is also countered by increasing computational resource.

Thus, no matter the approach, increasing computational resource is often the answer to a higher fidelity understanding of the design space. This becomes a barrier if the design problem being solved does not scale well with computational resource and has been observed by Gopsill and Hicks (2020) in their analysis of design spaces which require power-law scaling computational resource to model them fully.

To illustrate, let's consider a design space that can be represented by a cube whose length, width and depth is integer l. The cube is then discretised so that b blocks of unit size can be positioned within it giving a design space, D, where $D \subseteq \mathbb{Z}$ representing the candidate block positions. Let's also allow the blocks to be positioned anywhere in D with overlaps and discontinuous structures. The number of design solutions, S, can be written as a $f(l, b)$:

$$S = (l^3)^b \tag{1}$$

Taking $l = 10$ gives a cube volume of 1×10^3. Realising that 8 bits in a byte results in $2^8 = 256$ binary sequences, the number of bytes to index the position of a single block can be determined by:

$$\log_{256} 1 \times 10^3 = 1.25 < 2 \tag{2}$$

Thus, the storage requirement for S becomes $2b(l^3)^b$ (Figure 2b). What we observe is a design problem whose search/storage requirement exhibits power-law scaling and quickly becomes an intractable problem for classical computing.

Thus, it is rare to compute and store all solutions in classical computing but to store only the solutions that are valid for a set of constraints. If these constraints were to be updated, then a new traversal of the design space has to be made. It is also the reason that the topology of a design space is often described rather than individual solutions. However, this can be problematic for discontinuous design and solution spaces. Even if one adds constraints, such as the constraint of an individual constructing in the space, the power-law scaling remains with the constraints translating the relationship down the y-axis. For example, 10 Minecraft bricks constrained by being the same type, single starting position, forms a continuous structure and features no-overlap produces 102.9×10^9 design solutions (Gopsill, 2018).

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press

Thus, we arrive at the problem of our design spaces requiring power-law computational resource to model them fully, and classical computing not scaling well to these types of problem. A problem that this paper poses could be solved through the inherent properties of quantum computing.

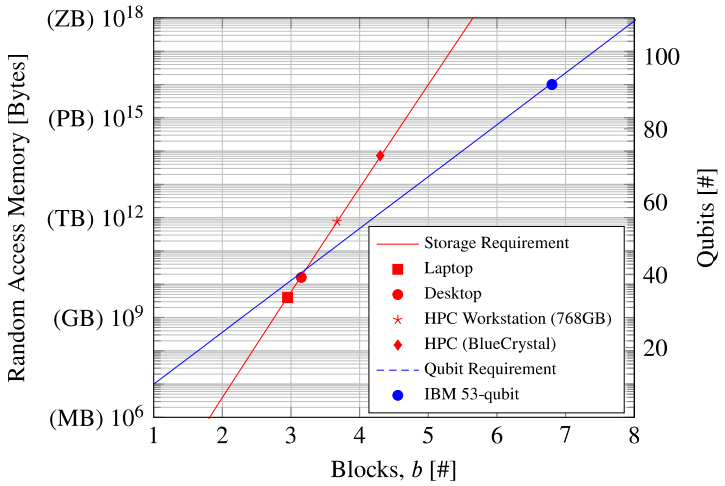


Figure 3. Polynomial scaling of quantum computing in evaluating $S(b)$ for our $l = 10$ design space cube.

The paper's contribution is therefore in design science and the examination of quantum computing's ability to model combinatorial design spaces through both theoretical and empirical study. The theoretical discusses the nature of qubits and their ability to represent multiple states in a single instance, and how this feature could provide the ability to study design spaces (Section 2). The empirical applies quantum computing to a 2D tiling problem where we propose a quantum circuit that can represent all valid solutions to the design problem (Section 3). This is extended to consider the placement of constraints to the problem and the world's first set of quantum circuits to represent design constraints along with an investigation in combining constraints (Section 4 & Section 5). In doing so, we demonstrate the potential of quantum computing to complement and in some cases, supersede classical computing in representing design spaces. We also demonstrate quantum design problem structuring is a non-trivial process with Section 6 eliciting six research questions for the design community. Solving these would open avenues for a new suite of design tools to support us in developing the products of tomorrow. The paper then concludes by highlighting the key findings (Section 7).

2 QUANTUM COMPUTING

Quantum computing is receiving increasing attention with considerable work on the underlying information theory along with the rapid pace of development of real-world quantum computers that provide a means to test and validate quantum circuits and algorithms. Here, we define a circuit as the ability to represent and perform a computation in a quantum computer, while an algorithm is a quantum circuit that has demonstrable performance improvements over classical counterparts. The maturity of the National Quantum Technologies Programme in the UK demonstrates the field is starting to deliver theory into practice (Anon, 2020). Alexandru et al. (2020) and Linden et al. (2020) are examples of taking quantum algorithms from their pure mathematical constructs and contextualising them to real-world problems, such as Heat Transfer. While researchers are beginning to explore the application of Quantum Computing for engineering modelling problems, a search on the Design Society's website, and Design Science, Research in Engineering Design, Design Studies and Mechanical Design journals reveal no papers that explore the potential implications it has on the design process and the act of designing.

Quantum computing has the potential to overcome the power-law scaling challenges exhibited in classical computed design spaces through its ability to represent multiple states via superposition. Qubits, the quantum equivalent to classical bits, can represent both 0 & 1 compared to 0 or 1 in classical computing. This allows n qubits to represent 2^n states in a single instance. This has the potential to drastically reduce resource requirements as a design space can be represented as an ensemble of quantum states.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press

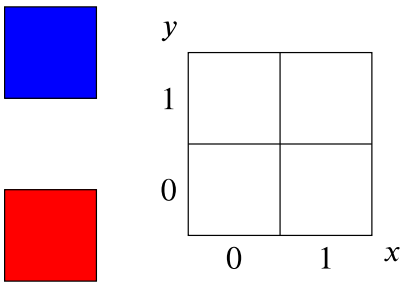
Returning to our problem of representing $S(b)$ for our design space cube, the number of qubits, Q, (resource requirement) can be written as:

$$Q = \log_2(l^3)^b \tag{1}$$

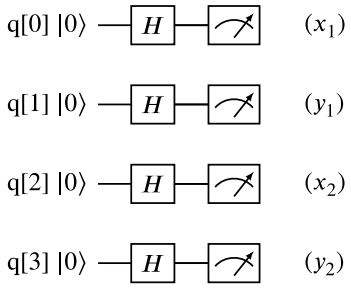
Figure 3 overlays the result from Figure 2b where we observe polynomial scaling for a quantum computer as opposed to power-law scaling in classical computing (n.b., second y-axis). In fact, today's IBM 53 qubit computer is theoretically capable of superseding classical High-Performance Computing in representing our design space cube. While the theoretical and increasingly practical capability exists, challenges remain in orchestrating the ensemble of states in order to both represent a design space, provide solutions with respect to a set of constraints and retrieve solutions through measurement.

3 REPRESENTING DESIGN SPACES WITH QUBITS

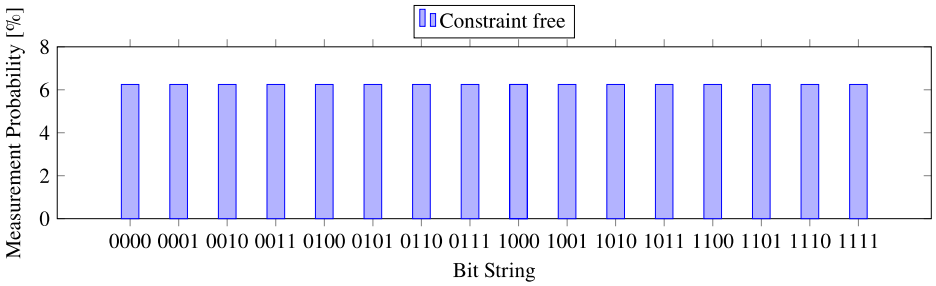
To examine quantum computing's capability, let's take a 2D tiling design problem where two tiles are to be placed on a 2×2 grid. Each tile can be placed in one of four locations (Figure 4a). Each tile can be represented by two qubits, one for each dimension (x, y). To place them in superposition, we use a Hadamard gate, H, which gives equal probability of the qubit being 0 or 1 when measured. Combined, they form all the solutions to placing a single tile in the design space, $S = 2^2 = 4$. The quantum circuit is shown in Figure 4b, which features the four qubits, two for each tile, all set into superposition using Hadamard gates and being measured, , to return a classical binary state.
With two tiles, the number of solutions becomes $2^2 \times 2^2 = 2^4 = 16$ and the quantum computer represents each of these solutions with equal probability as shown by the results from the classical simulation of the quantum circuit[1], Figure 4c. When measured, the quantum computer returns one of the solutions to our tiling problem. To represent all these solutions in a classical computer would require $16 \times 4 = 64$ bits. This demonstrates Quantum Computing's ability to represent the design space as an ensemble of quantum states. The next step is in how we can manipulate the quantum circuit to report only solutions for our combinatorial design problem.



(a) A 2 tile 2×2 *tiling problem.*



(b) 2×2 *design space QC model*

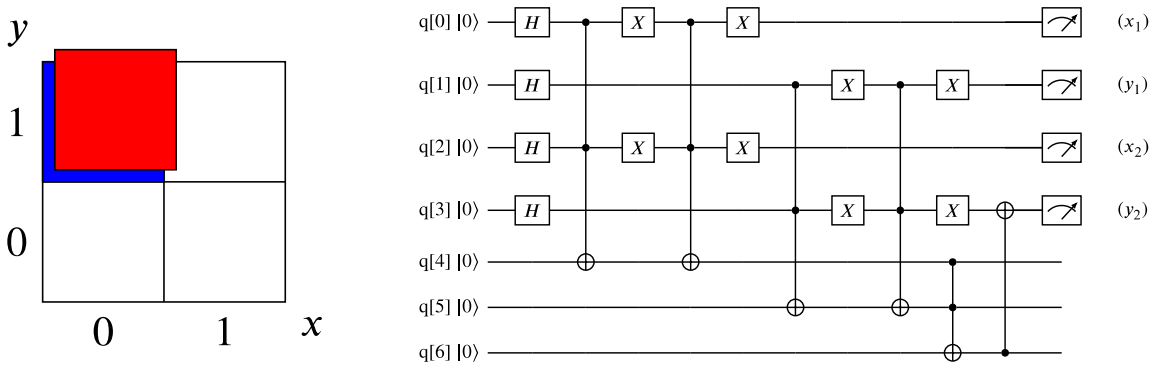


(c) Measurement probabilities for the different configurations of tiles on the 2×2 *grid.*

Figure 4. Tiling problem represented in a quantum computer

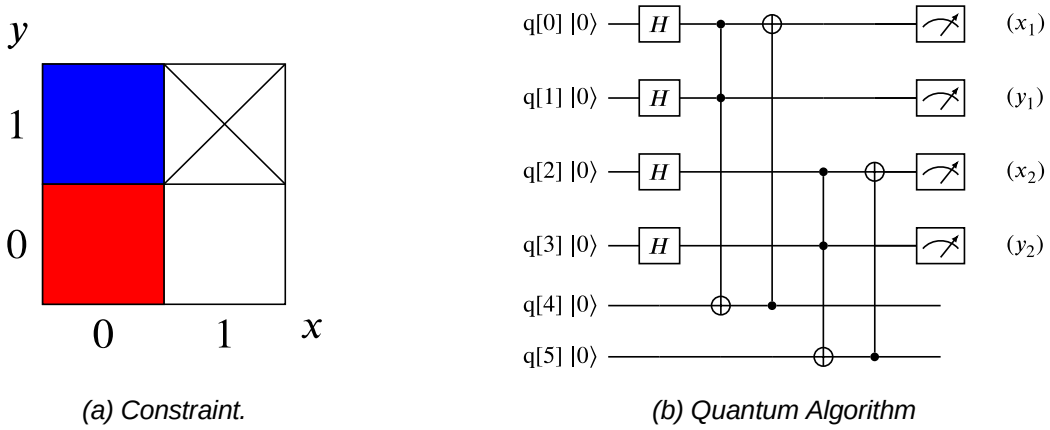
[1] Computed using QisKit. The code for each of these problems can be found at data.bris.ac.uk.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press



(a) Constraint. *(b) Quantum algorithm.*

Figure 5. No-overlap constraint.



(a) Constraint. *(b) Quantum Algorithm*

Figure 6. No (1,1) constraint.

4 QUANTUM CONSTRAINTS

In Section 3, we demonstrated how we can take a tiling design problem and represent all the solutions using a quantum circuit. However, design is fraught with constraints and it is the role of the engineer to navigate around these constraints to develop a solution to the problem. In this section, we present two constraints one might apply to our tiling problem, how they can be represented in a quantum circuit and how they could be combined to form a multi-constraint model. The development of the quantum circuit solutions followed an iterative trial and error design process by researchers with knowledge of quantum computing circuits.

4.1 The no-overlap constraint algorithm

Design problems typically involve geometric constraints. In our tiling problem, we may want to say the tiles cannot be placed in the same location (Figure 5a). Thus, we need to set a constraint in our model to remove these from S. Quantum Information theory states that no information can be lost, which results in the sum of the measurement probabilities across all combinations the quantum computer always being 1. For example, lets say we want to remove 1111 from the set of results in Figure 4c. The 6% measurement probability would have to be re-distributed across the remaining combinations in some manner. Therefore, we need to manipulate the qubits to only report correct solutions for our problem and this is achieved by re-distributing the measurement probability across the solutions that meet our constraint. The proposed circuit for this constraint is shown in Figure 5b.

The circuit starts with the four qubits, q[0-3], to provide the bit string that will represent the locations of the two tiles. q[0-1] represents the location of the first tile and q[2-3] represents the location of the second tile. We start by setting q[0-3] in superposition through Hadamard gates, as in Section 2, which defines our design space.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press

We accompany this with three additional qubits, q[4-6], that we use to detect and control the states of the four q[0-3]. Following the Hadamard gates, there is a Toffoli gate comparing q[0] and q[2] followed by two Pauli X, $\sqrt{X}$, gates, another Toffoli and two more Pauli X gates. A Toffoli (also known as a CCNOT gate) is a two-bit input, one-bit output gate which inverts the state of the output bit if and only if, the two input bits are 1. In this case, we're checking whether q[0] and q[2] are 1 and if so, q[4] becomes 1. A Pauli X gate inverts the state of a qubit. We're using Pauli X gates to flip the state of q[0] and q[2] so we can check for instances of 00 and 11 and a second set of Pauli X to ensure we return q[0] and q[2] to their original state. The combination of gates forms a bit comparator that returns q[4] as 1 when q[0] and q[2] are equivalent (Menon and Chattopadhyay, 2020; Ramos et al., 2006).
The series of gates are repeated for q[1] and q[3] giving us a 1 on q[5], if they are equivalent. The last Toffoli gate checks both sets of qubits for equivalency and if so, q[6] becomes |1>. The final element is the CNOT gate that applies a Pauli X gate to q[3] (could be any one of q[0-3]) if q[5] is 1. Applying the Pauli X gate will flip the state of q[3] so that it is out of sequence with q[1] thereby re-distributing the overlapping cases to non-overlapping case and thus, meet our design constraint. The final elements in Figure 5b represent the measurements being taken and stored in classical form. These are only taken for q[0-3] as these are the ones of interest in terms of representing a solution.

4.2 The no (1,1) constraint

Design spaces may also have positions that we would not like to place objects. In the case of our tiling problem, let's say that we would not like a tile in location (1,1) (Figure 6a). The proposed circuit for this constraint is shown in Figure 6b. Each qubit set has a Toffoli gate and applies a Pauli X gate to q[4,5], respectively. A CNOT gate is then placed after the Toffoli gate. This applies a Pauli X gate to the target q[0,3] if q[4,5] are 1, respectively, preventing either tile being present at (1,1).

4.3 Combined constraints

With the circuits formed, a further experiment was set-up to examine how they can be combined to provide a no overlap scenario where neither tile is in the (1,1) position. The combined model is presented in Figure 7. In this case, we chose to simply append the quantum circuits to investigate whether combining constraints can be achieved through a trivial addition process.

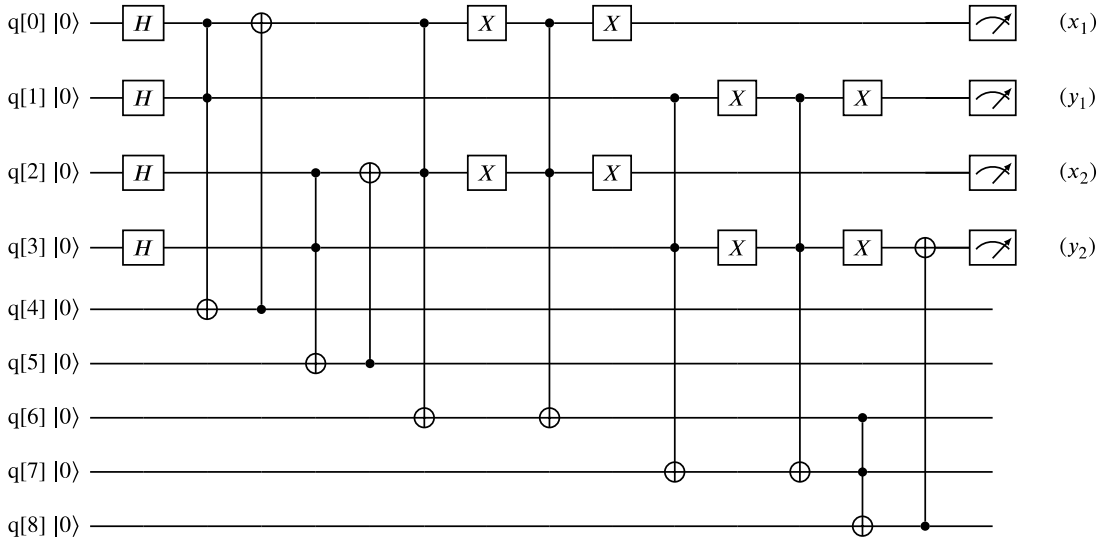


Figure 7. Combined quantum circuit.

5 RESULTS

Table 1 presents the results to the four design problems: constraint free, no overlap, no (1,1) placement and combined (no overlap and no (1,1) placement). In the constraint free tiling problem, we can see that S is represented by the circuit and solutions feature the same measurement probability. Moving to the no overlap case, we see the circuit is effective in eliminating the overlap solutions for our problem. Given the law of conservation of quantum information, it can be seen that the measurement probabilities have

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press

been re-distributed from the invalid solutions to the valid solutions. This is also observed in the no (1,1) placement constraint. In both cases, the re-distribution is not even across S with some featuring higher probabilities of being measured than others. This presents bias in S as a result of our quantum circuit representation of the combinatorial design problem.

When combined, we observe the interplay between the two constraints. The combined circuit is effective in representing S except for the presence of one invalid solution (1101). As a result, the combined circuit was revised as shown in Figure 8. The circuit features an additional Toffoli gate between q[2] and q[3] to check whether they are both 1 and if so, the accompanying CNOT gate alters the state of q[2]. The results of which are shown in the final column of Table 1 and demonstrates the additional quantum gate series counters the interplay between the two constraints. It reveals that the formation of design problems with sets of constraints will be non-trivial. The combination of constraints also shows bias to particular solutions in S.

Table 1. Measurement probabilities and solution correctness of the quantum circuits.

Bit String				Solution Measurement Probability [%]				
x_1	y_1	x_2	y_2	Constraint Free	No Overlap	No (1,1)	Combined	Combined v2
0	0	0	0	6.25	0.00	6.25	0.00	0.00
0	0	0	1	6.25	6.25	6.25	6.35	5.47
0	0	1	0	6.25	6.25	12.50	36.13	36.43
0	0	1	1	6.25	6.25	0.00	0.00	0.00
0	1	0	0	6.25	0.00	6.25	6.93	5.66
0	1	0	1	6.25	0.00	6.25	0.00	0.00
0	1	1	0	6.25	6.25	12.5	10.55	14.36
0	1	1	1	6.25	12.5	0.00	0.00	0.00
1	0	0	0	6.25	12.50	12.50	19.04	18.07
1	0	0	1	6.25	6.25	12.50	13.47	20.02
1	0	1	0	6.25	0.00	25.00	0.00	0.00
1	0	1	1	6.25	6.25	0.00	0.00	0.00
1	1	0	0	6.25	6.25	0.00	0.00	0.00
1	1	0	1	6.25	12.50	0.00	7.52	0.00
1	1	1	0	6.25	6.25	0.00	0.00	0.00
1	1	1	1	6.25	0.00	0.00	0.00	0.00

Valid Solution Set (S)
Invalid Solution Set

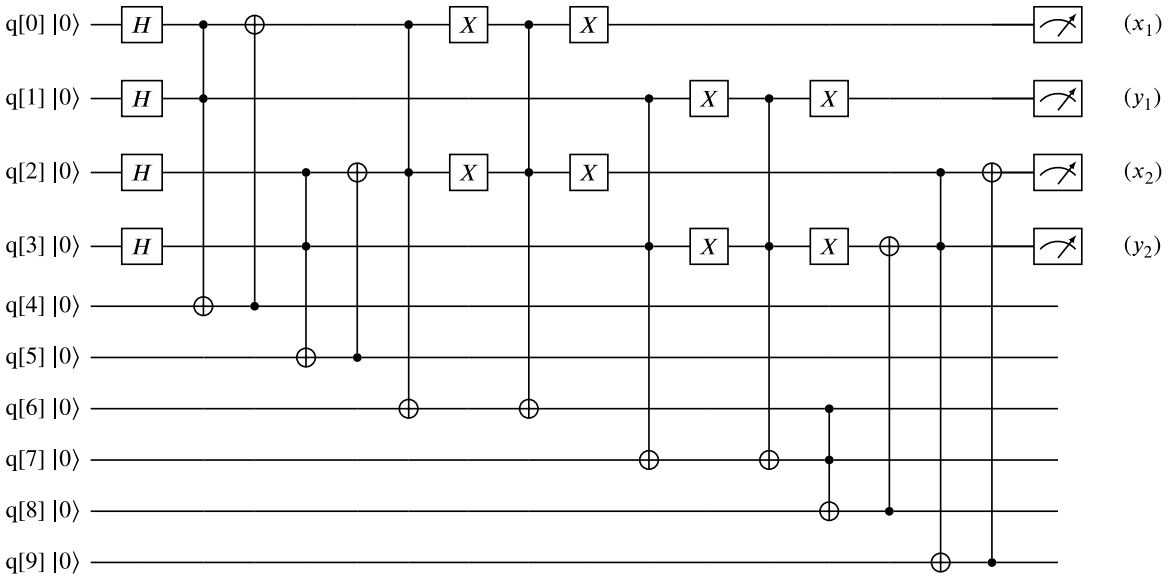


Figure 8. Combined constraints v.2.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press

6 DISCUSSION

The paper demonstrates quantum computing's potential in representing combinatorial design problems and the ability to realise design spaces in a single instance that would be infeasible through classical computation. While we have indicated quantum supremacy in representing design spaces, the test presented assumes that there is a need to compute all S for a design problem in order to select an optimal solution. An issue that is well-recognised in classical computing and has led to the advancement of algorithms that navigate design spaces. Therefore, a more suitable test might be one that reflects quantum computations' ability to find optimal solutions faster as well as optimal solutions that classical algorithms would not arrive at. This is the next step in the process where we are seeking to score the solutions against an objective function and to determine what, if any, quantum algorithms exist in returning optimal solutions from S, and whether they provide a computational advantage over classical solutions.

At present, repeated sampling of the quantum circuits is likely to be comparable to random sampling of a classical design space representations. However, opportunities exist in manipulating the measurement probabilities in order to promote preferred solutions. Subsequent evaluation of the measurement distribution would effectively provide the ranking of solutions. In understanding how we may manipulate the measurement probabilities and apply known quantum algorithms that outperform their classical counterparts, we will be able to evaluate how the methods complement one another, where the trade-offs are, and how we can determine where and when a quantum approach should be taken for an engineering design problem. We have also provided the first quantum circuits that represent and constrain design spaces. The successful will be deployed by the engineering design community to evaluate combinatorial design problems. An emergent property that needs to be managed is in the shifting probability distributions so as to not bias particular solutions (unless for the purposes of promoting them with respect to the objective function). Reflecting on Figure 8, randomly assigning the CNOT gate to q[0-3] between each run of the circuit could provide a means of averaging the distribution. Thus, it may be that successive runs of different quantum logic-gate configurations are required.

The challenge in retrieving all the solutions exists with repeated measurement likely to return results that we have already measured. Thus, incorporating methods of constraining the design space through the inclusion of known solutions could be explored. In addition, being able to retrieve the number of solutions that exist would be an interesting metric ahead of investigating the form of S as this may indicate the level of uncertainty in our design requirements and whether we need to tighten, relax, add or remove constraints. One such metric would be determining the number of solutions that satisfy our problem (D'Helon and Protopopescu, 2002). Continuing along this train of thought, it may be that describing the topology of the design space would be just as powerful and useful as returning specific solutions.

The algorithms also demonstrate the use of qubits as control bits to constrain the state of the design space. While this is crucial for circuit function, we ultimately lose computing capacity in representing the design space. Thus, effective use/re-use of these control qubits will become important in maximising the capacity of a quantum computer in modelling a design space.

In summary, the emergent research questions for the design community are:

1. How can one score solutions against an objective function?
2. How can measurement probabilities be manipulated to assist solutions identification?
3. How can we manage bias introduced by the introduction of our constraints?
4. How does the construction of quantum circuits scale with size?
5. How could Quantum Computing be used to describe the topology of a design space?
6. What guidelines, support and design processes for constructing combinatorial design problems for Quantum Computing are required?

7 CONCLUSION

Combinatorial design problems are widely observed in engineering and are crucial in the derivation of solutions to solve our design problems. Design problems that are becoming increasingly complex and vast with classical computing exhibiting power-law scaling behaviour in the computational resource they require to resolve them.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press

To counter this and be able to continue to explore complex design spaces, this paper has successfully demonstrated the potential of quantum computing to represent design spaces with polynomial scaling resource requirements. The paper has gone further by contributing four circuits to represent and constrain a 2 tile 2D tiling design problem.
Quantum supremacy has been discussed with it potentially being closer than we think. However, considerable research is still required in providing the underlying constraint circuits and algorithms that will help us construct our design problems for quantum computers to solve. While practical application of quantum computing is still years away, this paper has started us on the journey of creating a toolbox of circuits and algorithms that engineers will be able to deploy when it arrives.

ACKNOWLEDGEMENTS

The work has been undertaken as part of the Engineering and Physical Sciences Research Council (EPSRC) grants – EP/R032696/1 and EP/V05113X/1. We would like to thank the University of Bristol and Centre for Modelling and Simulation (CFMS) in supporting this research, the IBM Quantum Experience, and colleagues at the Quantum Information Institute.

REFERENCES

Alexandru, C.-M., Bridgett-Tomkinson, E., Linden, N., MacManus, J., Montanaro, A., and Morris, H., 2020. Quantum speedups of some general-purpose numerical optimisation algorithms. arXiv: 2004.06521 [quant-ph].
Anon, 2020. Uk national quantum technologies programme [Online]. Available from: http://uknqt.epsrc.ac.uk/ [Accessed August 27, 2020].
Biskjaer, M., Dalsgaard, P., and Halskov, K., 2014. A constraint-based understanding of design spaces. Proceedings of the conference on designing interactive systems: processes, practices, methods, and techniques, dis [Online]. Available from: https://doi.org/10.1145/2598510.2598533.
D'Helon, C. and Protopopescu, V., 2002. New summing algorithm using ensemble computing. Journal of physics a: mathematical and general, 35(42), p.L597.
Feng, C.-X. and Kusiak, A., 1995. Constraint-based design of parts. Computer-aided design [Online], 27(5), pp.343–352. Available from: https://doi.org/https://doi.org/10.1016/0010-4485(95)96798-Q.
Gopsill, J.A., Shindler, J., and Hicks, B.J., 2017. Using finite element analysis to influence the infill design of fused deposition modelled parts. Progress in additive manufacturing [Online]. Dataset: https://doi.org/10.15125/ BATH-00420. Available from: https://doi.org/10.1007/s40964-017-0034-y.
Gopsill, J., 2018. Examining the solution bias of construction kits. Proceedings of the international conference on design [Online]. Available from: https://doi.org/10.21278/idc.2018.0192.
Gopsill, J. and Hicks, B., 2020. The principles of construction, standard parts, interfaces and constraints, and their representation of the design space. Royal society open science. In Preparation.
Lin, L. and Chen, L.C., 2002. Constraints modelling in product design. Journal of engineering design [Online], 13(3), pp.205–214. eprint: https://doi.org/10.1080/09544820110108908. Available from: https://doi.org/10.1080/09544820110108908.
Linden, N., Montanaro, A., and Shao, C., 2020. Quantum vs. classical algorithms for solving the heat equation. arXiv: 2004.06516 [quant-ph].
Mathias, D., Boa, D., Hicks, B., Snider, C., Bennett, P., Taylor, C., et al., 2017. Design variation through richness of rules embedded in lego bricks. Ds 87-8 proceedings of the 21st international conference on engineering design (iced 17) vol 8: human behaviour in design, vancouver, canada, 21-25.08. 2017, pp.099–108.
Medland, A.J. and Mullineux, G., 1993. A constraint approach to feature-based design. International journal of computer integrated manufacturing [Online], 6(1-2), pp.34–38. eprint: https://doi.org/10.1080/09511929308944553. Available from: https://doi.org/10.1080/09511929308944553.
Menon, V. and Chattopadhyay, A., 2020. Quantum string comparison method. arXiv: 2005.08950 [quant-ph].
Ramos, R.V., Sousa, P.B. de, and Oliveira, D.S., 2006. Solving mathematical problems with quantum search algorithm. arXiv: quant-ph/0605003 [quant-ph].
Sapossnek, M. et al., 1991. Research on constraint-based design systems.
Zhang, L. and Xie, L., 2014. Modeling and computation in engineering iii. CRC Press.

https://doi.org/10.1017/pds.2021.512 Published online by Cambridge University Press