A domain-wall encoding of discrete variables*

HPC 2021

Nicholas Chancellor

July 29, 2021

*Based on results from arχiv:2102.12224 (with co-authors Jie Chen and Tobias Stollenwerk, accepted in IEEE Transactions on Quantum Engineering DOI: 10.1109/TQE.2021.3094280) and background from other sources

Aside: UK activities which may be relevant to this audience

QEVEC

- Test early applications for quantum in exascale computing
- Very recently funded (UKRI)
- Focused on using quantum within HPC for real applications (including material science for example)
- Contact Viv Kendon for more info: viv.kendon@durham.ac.uk

CCP-QC

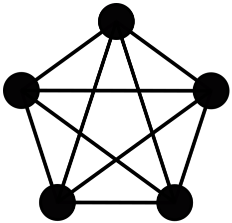
- Collaborative computational project on quantum computing
- Network focused on early academic research applications
- webpage ccp-qc.ac.uk

One-hot constraints

- Common constraints found in optimisation problems
- Enforce that **exactly one** mutually exclusive option is taken
- Examples: a vehicle can only take one route to get to a destination, a piece of equipment can only be doing one thing at a time, etc...

Easy to enforce using quadratic interactions, term proportional to $(\sum_i x_i - 1)^2$ equal to zero if exactly one variable is 1 and all others are 0, higher otherwise

...but requires interaction between all variables



A slightly different perspective: discrete variables

- Rather than thinking of individual binary variables under a constraint, treat like a single m value variable
- A mostly philosophical distinction, but crucial for understanding other encodings
- Define two index objects:

$$x_{i,\alpha} = \begin{cases} 1 & \text{variable } d_i \text{ takes value } \alpha \\ 0 & \text{otherwise} \end{cases}$$

- Discrete Quadratic models, (DQM), made from pairwise interactions of x terms:

$$H_{\mathrm{DQM}} = \sum_{i,j} \sum_{\alpha,\beta} D_{(i,j,\alpha,\beta)} x_{i,\alpha} x_{j,\beta}$$

one-hot and binary encoding as a DQM

$$H_{\text{one hot}} = H_{\text{DQM}} + \lambda \sum_i \left(\sum_{\alpha=0}^{m-1} x_{i,\alpha} - 1 \right)^2$$

- Each $x_{i,\alpha}$ is an individual binary variable, add one-hot constraints to force them to be single valued
- Easy to forget distinction in this case, but in general $x_{i,\alpha}$ does not need to map to a single binary variable
- Binary encoding, each binary variable $b \in \{0, 1\}$ is a digit in a binary number $x_{i,\alpha}$ is a string of $\log_2(m)$ b and $1 - b$ terms

Is there another way?

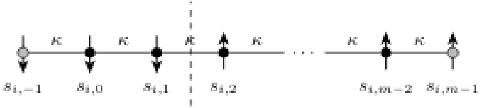
YES!

1. Constrain Ising $s_{i,\alpha} \in \{-1, +1\}$ variables with strong ferromagnetic coupling

$$H_{\text{chain}} = -\kappa \left(\sum_{\alpha=-1}^{m-2} s_{i,\alpha} s_{i,\alpha+1} \right)$$

2. Constrain $s_{i,-1} = -1$ and $s_{i,m-1} = 1$ so that the chain consisting of variables $1 \ldots m-2$ is frustrated and contains at least one domain wall
3. Define DQM terms

$$x_{i,\alpha} = \frac{1}{2} \left(s_{i,\alpha} - s_{i,\alpha-1} \right)$$



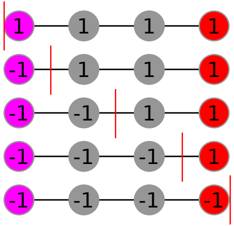
Discrete variables into binary, three ways*

Variable size$=m$

performance metric	binary	one-hot	domain wall
# binary variables	$\lceil \log_2(m) \rceil$	m	$m-1$
# couplers for encoding	0 if $m=2^n$ $n \in \mathbb{Z}$ complicated otherwise	$m\,(m-1)$	$m-2$
intra-variable connectivity	N/A or complicated	complete	linear
maximum order needed for two variable interactions	$2\,\lceil \log_2(m) \rceil$	2	2

Binary= assign bitstrings to configurations
One hot= constrain variables so exactly one can be 1
Domain wall= new method we discuss here

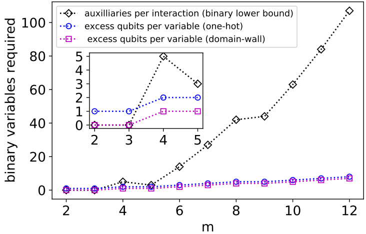
encoded value	qubit configuration
0	1111
1	-1111
2	-1-111
3	-1-1-11
4	-1-1-1-1



*For details see: Chancellor, Quantum Sci. Technol. 4 045004

Binary encoding

- A variable of size m can be encoded in $\lceil \log_2(m) \rceil$ qubits
- Arbitrary interactions require high order terms in Hamiltonian
- Only quadratic interactions $\rightarrow$ gadgets $\rightarrow$ auxilliary variables
- Fair counting needs to include auxilliary variables as well

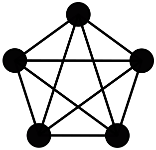


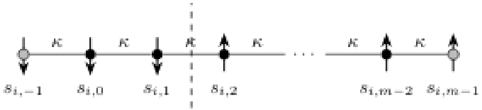
This is a losing proposition for general interactions*

*Extensive discussion of this point recently added to arχiv:2102.12224; binary may still be best for interactions with special structure, example, variable multiplication: Joseph et. al. Phys. Rev. A 103, 032433

How does this domain-wall encoding stack up against one-hot?

- One fewer qubit per DQM variable than one-hot
- $x_{i,\alpha}$ is linear in s terms, therefore $x_{i,\alpha}x_{j,\beta}$ is quadratic, maps to Ising model with only second order interactions
- Simple degree of freedom counting arguments $\rightarrow$ domain-wall encoding uses the smallest possible number of variables for all $x_{i,\alpha}x_{j,\beta}$ terms to be quadratic
- Less connectivity within variables, linear versus all-to-all



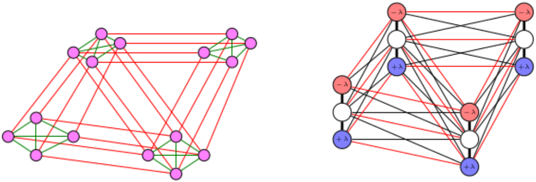


Test this on an example: colouring problems*

Simple test problem with structure: penalty between nodes if and only if they are the same colour

Use natural structure of problem to ‘spread out’ embedding

Four colouring example, ‘layered’ structure in Domain wall (right), no structure in one hot, (left)

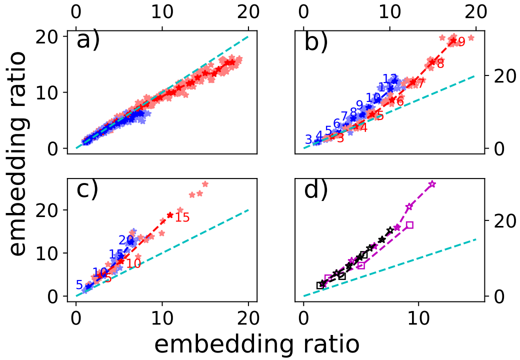


three-colouring → randomly generated edges with 50% probability

k-colouring → twice as many nodes as colours, random edges with 75% probability

*Example taken from Chancellor, Quantum Sci. Technol. 4 045004

Test this numerically with minor-miner*



Embedding ratio= number of qubits per logical variable
(Y-axis=one-hot, X-axis=domain-wall)
chimera in red, Pegasus in blue

(a) Max-three-colouring problems (domain-wall slightly worse)

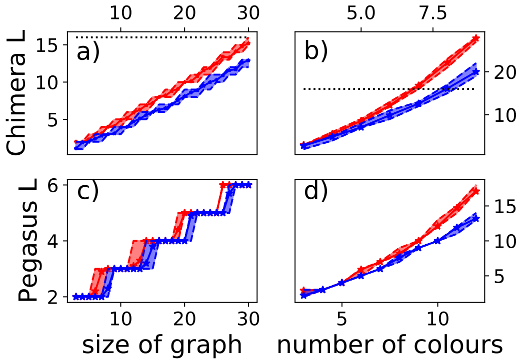
(b) Max-k-colouring problems (domain-wall much better)

(c) Artificial scheduling problem (domain-wall much better)

---

*Example taken from Chancellor, Quantum Sci. Technol. 4 045004

Bigger problems can be embedded on the same size device*



Size of device required to embed problems of different sizes
right column= three colouring, left= k-colouring

- Can embed larger problems for both graphs and problem types
- Fewer binary variables makes up for slightly worse embedding ratio in three-colouring case

*Example taken from Chancellor, Quantum Sci. Technol. 4 045004

What about dynamics?

- Makes embedding more efficient...
- But does it translate to performance gains on actual annealers? how does the encoding affect the dynamics?

Not easy to answer a priori, on one hand

- Changing the values of domain-wall variables is non-perturbative*
- This isn't true for one-hot

On the other hand

- Configurations now have a defined order and are traversed in a linear way, rather than one-hot where there is no "order"
- Not clear if this will be helpful or harmful

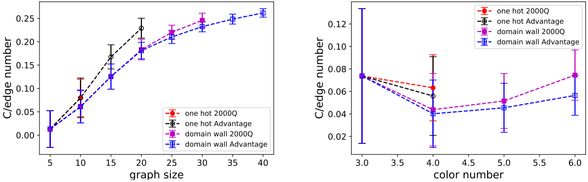
Need to test **experimentally**, our recent paper arχiv:2102.12224 does this

*all valid configurations can be reached without having to pass through invalid configurations

The results*

For both k and three colouring problems the domain-wall encoding performs better on both Advantage and 2000Q

three colouring (left), k-colouring (right)



C=number of places same colour touches

Even looks like domain-wall on 2000Q out-performs one-hot on Advantage!

Use hypothesis testing to verify that this is a statistically significant result, test 100 instances on each and see how much each processor/encoding combination wins for all 6 combinations

*arχiv:2102.12224

Hypothesis testing, three colour*

Green=statistically significant result (95% confidence)

	Adv. dw/oh		2000Q dw/oh		dw Adv./2000Q		oh Adv./2000Q		(dw, Adv.)/(oh, 2000Q)		(dw, 2000Q)/(oh, Adv.)	
5 node (b,w)	0	0	0	0	0	0	0	0	0	0	0	0
5 node p												
10 node (b,w)	42	0	37	0	2	0	19	21	39	0	40	0
10 node p	2.27×10^{-13}		7.28×10^{-12}		2.50×10^{-1}		6.82×10^{-1}		1.82×10^{-12}		9.09×10^{-13}	
15 node (b,w)	85	2	95	3	32	34	70	22	94	1	91	2
15 node p	2.47×10^{-23}		4.95×10^{-25}		6.44×10^{-1}		2.67×10^{-7}		2.42×10^{-27}		4.41×10^{-25}	
20 node (b,w)	99	0	100	0	43	41	94	3	100	0	93	2
20 node p	1.58×10^{-30}		7.89×10^{-31}		4.57×10^{-1}		9.60×10^{-25}		7.89×10^{-31}		1.15×10^{-25}	
25 node (b,w)	100	0		FAIL	66	20		FAIL		FAIL	98	2
25 node p	7.89×10^{-31}				3.33×10^{-7}						3.98×10^{-27}	
30 node (b,w)	100	0		FAIL	72	20		FAIL		FAIL	97	2
30 node p	7.89×10^{-31}				2.30×10^{-8}						7.81×10^{-27}	
35 node (b,w)	100	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
35 node p	7.89×10^{-31}											
40 node(b,w)	100	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
40 node p	7.89×10^{-31}											

- Domain wall 2000Q beats one hot Advantage (in a statistically significant way)
- Trend continue up to size where no longer possible to embed in 2000Q (FAIL)
- Otherwise results are expected → 2000Q worse than Advantage, one hot worse than domain wall

*arχiv:2102.12224

Hypothesis testing, k colour*

Green/red=statistically significant result (95% confidence)

	Adv. dw/oh		2000Q dw/oh		dw Adv./2000Q		oh Adv./2000Q		(dw, Adv.)/(oh, 2000Q)		(dw, 2000Q)/(oh, Adv.)	
3 color (b,w)	0	0	0	0	0	0	0	0	0	0	0	0
3 color p												
4 color (b,w)	34	1	37	2	11	3	26	16	44	1	33	7
4 color p	1.05×10^{-9}		1.42×10^{-9}		2.87×10^{-2}		8.21×10^{-2}		1.31×10^{-12}		2.11×10^{-5}	
5 color (b,w)	91	1	78	1	34	18	23	59	88	1	91	1
5 color p	1.88×10^{-26}		1.32×10^{-22}		1.82×10^{-2}		≈ 1		1.45×10^{-25}		1.88×10^{-26}	
6 color(b,w)	99	0		FAIL	59	15		FAIL		FAIL	99	0
6 color p	1.58×10^{-30}				1.28×10^{-7}						1.58×10^{-30}	
7 color(b,w)	92	0	FAIL	FAIL		FAIL		FAIL		FAIL	FAIL	
7 color p	2.02×10^{-28}											

- Domain wall 2000Q beats one hot Advantage (in a statistically significant way)
- Trend continue up to size where no longer possible to embed in 2000Q (FAIL)
- One case where 2000Q beats advantage for the same decoding (one hot)*

*This goes away when the decoding strategy for broken chains is changed so probably an artefact of majority vote decoding

*arχiv:2102.12224

Experimental results summary*

- Encoding makes a bigger difference to solution optimality even than choosing a more advanced processor
- Domain wall constraints seem much less "fragile"
- Encoding still helps with chain breaks, but advantage is smaller → QPU structure makes a bigger difference

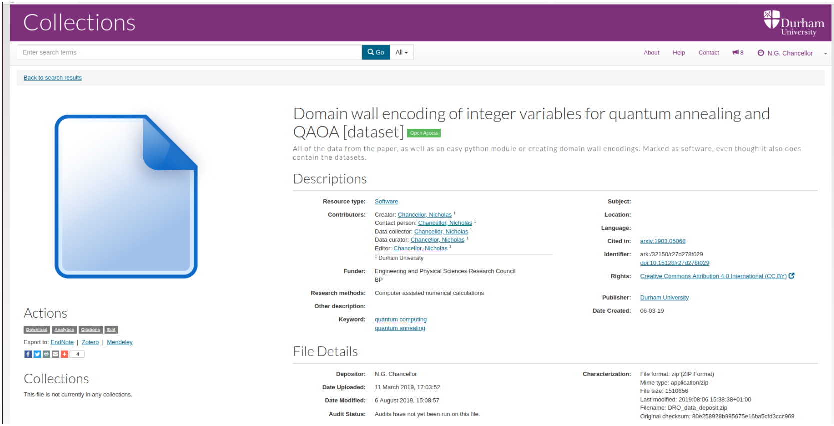
Experiments didn't find any metrics where one-hot does better

No observed downside to using domain-wall encoding, but some major advantages

*Additional results on supplemental slides if people are interested, can also be found in arχiv:2102.12224

Want to try it yourself?

Python code to create domain wall encodings available at https://collections.durham.ac.uk/: "Domain wall encoding of integer variables for quantum annealing and QAOA [dataset]"*



*https://doi.org/10.15128/r27d278t029

How to use the repository code

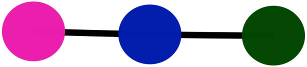
- Load the ez_domain_wall module
- The function make_domain_wall_encoding creates a domain wall encoding of a discrete problem
- ez_dw_examples jupyter notebook with some examples of how to use the code

Inputs to make_domain_wall_encoding

1. Variable sizes: a list of the sizes of all variables, $\rightarrow$ call the sum of these sizes R
2. Penalties: single body terms which penalize different values of individual variables, 1-D array (or list) of length R
3. Interactions: interactions between variables upper triangular R by R array

Outputs: J_core, h_core terms for enforcing domain wall constraints, J_prob, h_prob rest of Hamiltonian

A simple example: three colouring a line of three nodes



1. Three variables with three possible colours, therefore variable sizes $\rightarrow [3, 3, 3]$ ($R = 3 \times 3 = 9$)
2. No single term penalties, therefore penalties term is all zeros (length 9)
3. Interaction terms: 9×9 array of zeros with 6 terms which take value 1 (using Python style zero indexing)
   $(0, 3)$, $(1, 4)$, $(2, 5)$ $\rightarrow$ prevent the first and second node from being the same colour
   $(3, 6)$, $(4, 7)$, $(5, 8)$ $\rightarrow$ prevent the second and third node from being the same colour

The outputs for our simple example

h_core: $\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$

J_core: $\begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

h_prob: $\begin{bmatrix} 0.25 & -0.25 & 0.5 & -0.5 & 0.25 & -0.25 \end{bmatrix}$

J_prob: $\begin{bmatrix} 0.0 & 0.0 & 0.5 & -0.25 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.25 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & -0.25 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.25 & 0.5 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$

$$H = \sum_{i<j} J\_prob_{ij}\ s_i s_j + \sum_i h\_prob_i\ s_i +$$
$$\kappa \left(\sum_{i<j} J\_core_{ij}\ s_i s_j + \sum_i h\_core_i\ s_i \right)$$
$$s \in \{-1, 1\}$$

Possible application: quantum simulation

- Treat each variable in the DQM as a point in space
- Quadratic coupling can emulate many differential equation terms (may also be useful in solving other diff. eqs)
- Transverse fields can emulate local quantum fluctuations in quantum field theories (field fluctuates to nearby values)

Under development with Steven Abel and Michael Spannowsky in Durham IPPP*

Not the main focus of the talk, but worth highlighting

*see: S. Abel, N. Chancellor, M. Spannowsky Phys. Rev. D 103, 016008 (2021) and S. Abel, M. Spannowsky arχiv:2006.06003

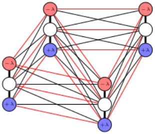
Further outlook (Chancellor, QST. 4 045004)

Drivers which preserve valid subspace out of two body terms*

- Constructed by combining $Z_{i-1}X_i$ and $-X_iZ_{i+1}$ terms $\rightarrow$ rotations only happen if domain wall is present
- May be useful in QAOA or other gate model algorithm (maybe even just with transverse field drivers)
- Annealers with muti-body drivers (longer term)

Layered structure for important problem types such as colouring and scheduling

- Suggests application specific (ASIC if superconducting) designs for future annealers



*analogous to what was done in Hadfield et. al. Algorithms 12.2 (2019): 34

Summary

Domain-wall encoding can lead to superior performance over one-hot for quantum annealing

- No downside seen yet
- Can get substantial gains, more than new hardware in the cases we tested

Should try it if you are using discrete variables

- Python code available; don't have to code "from scratch"
- **dw encoding to be included in DLR (German aerospace organization) software package when released**
- QAOA has not been tested, not clear how it would perform

Potential for using annealers as simulators

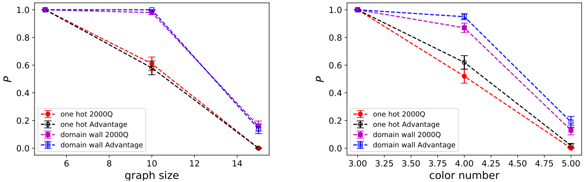
- Early results, still being explored (**ongoing work with Quantum Computing Inc. to do this, watch this space!**)
- More rigorous understanding may be helpful

Supplemental slides: More details on domain-wall performance

Summary of more results from arχiv:2102.12224

Same pattern holds for probability to find optimal*

three colouring (left), k-colouring (right)



Note that each run was only performed with 100 reads, better results could be attained with more reads

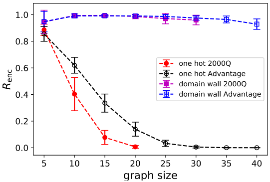
All QPU-encoding combinations found optimal solution at smallest size $\rightarrow$ explains no "winners" in hypothesis testing

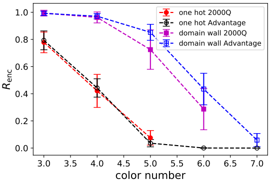
*arχiv:2102.12224

Digging deeper into performance: encoding failures*

What fraction of solutions have all one-hot/domain-wall constraints satisfied

three colouring (left), k-colouring (right)





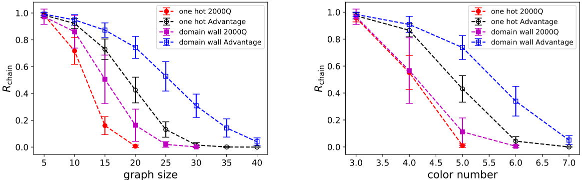
Domain-wall constraints are much less "fragile" especially with only three colours, makes a much bigger difference than processor structure

* arχiv:2102.12224

Digging deeper into performance: chain breaks*

What fraction of solutions have no unbroken minor embedding chains

three colouring (left), k-colouring (right)



Note: bars are standard deviation, standard error is 10x smaller

QPU structure seems to make a bigger difference here, but domain-wall encoding still leads to an improvement

*arχiv:2102.12224