Modular Embedding of Problems onto Quantum Annealers

Ellis Wilson, Frank Mueller, Scott Pakin

North Carolina State University, Los Alamos National Lab

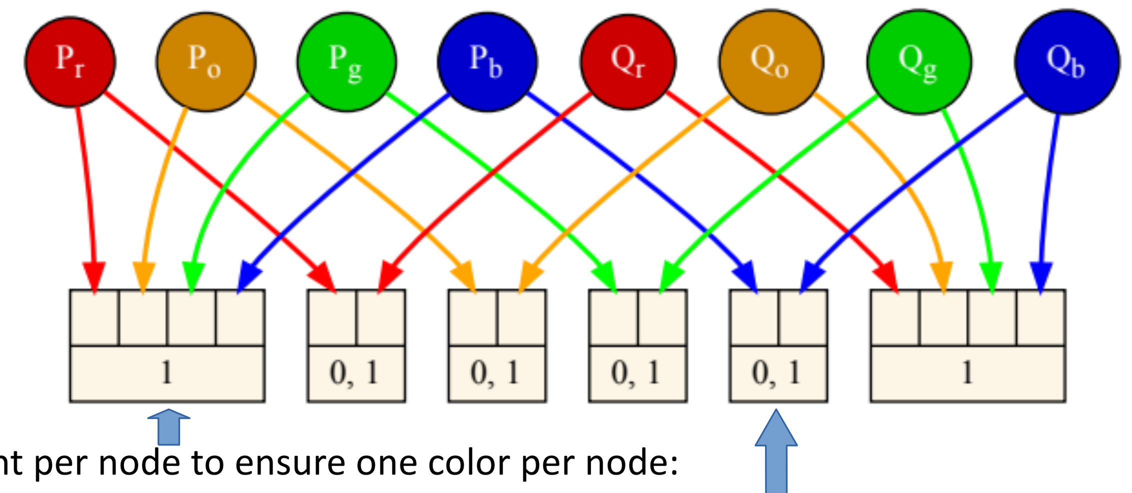
Motivation

- Quantum annealers benefit from high qubit connectivity
  - "Chains" of physical qubits used to represent single virtual qubits
- Embedding problems onto current topology is NP-Hard
- Previous work[1] shows improvement for specific problem by adding extra structural constraints
  - Separates the problem into cells of 2 qubits
  - Cells can only be embedded onto certain physical qubits on the Chimera structure
  - Similar to the idea behind DWave's *Locally Structured Embeddings*

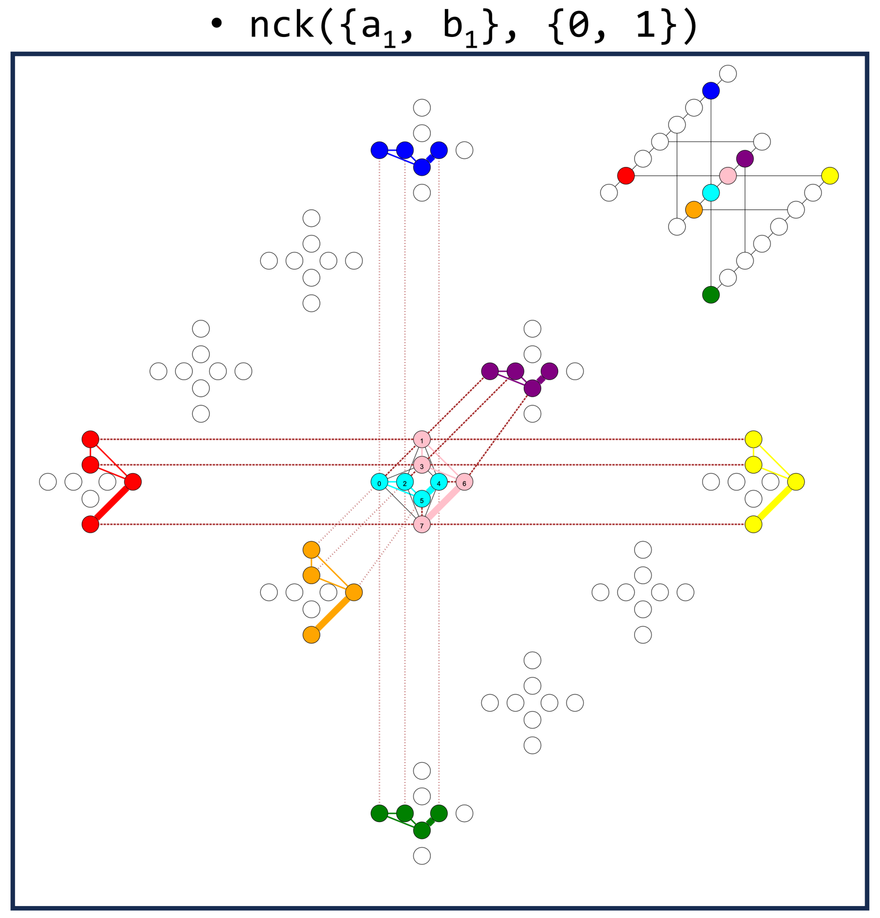
NchooseK

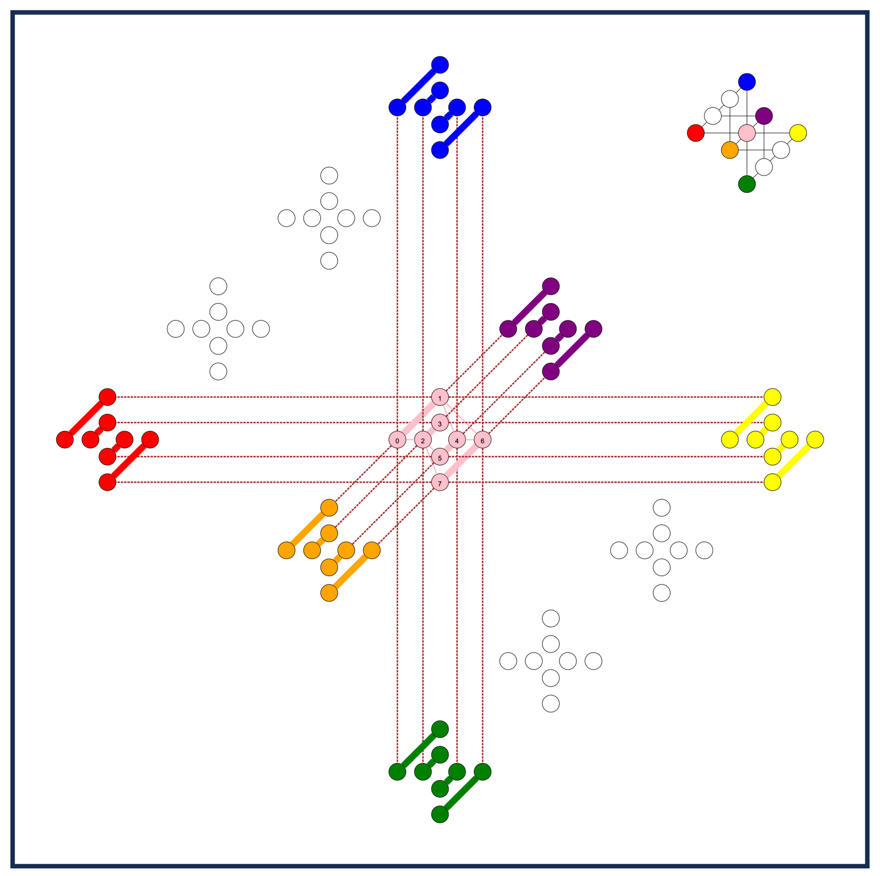
NchooseK is a domain specific, constraint-based language built for automatically setting up problems for both gate-based machines and quantum annealers[2].

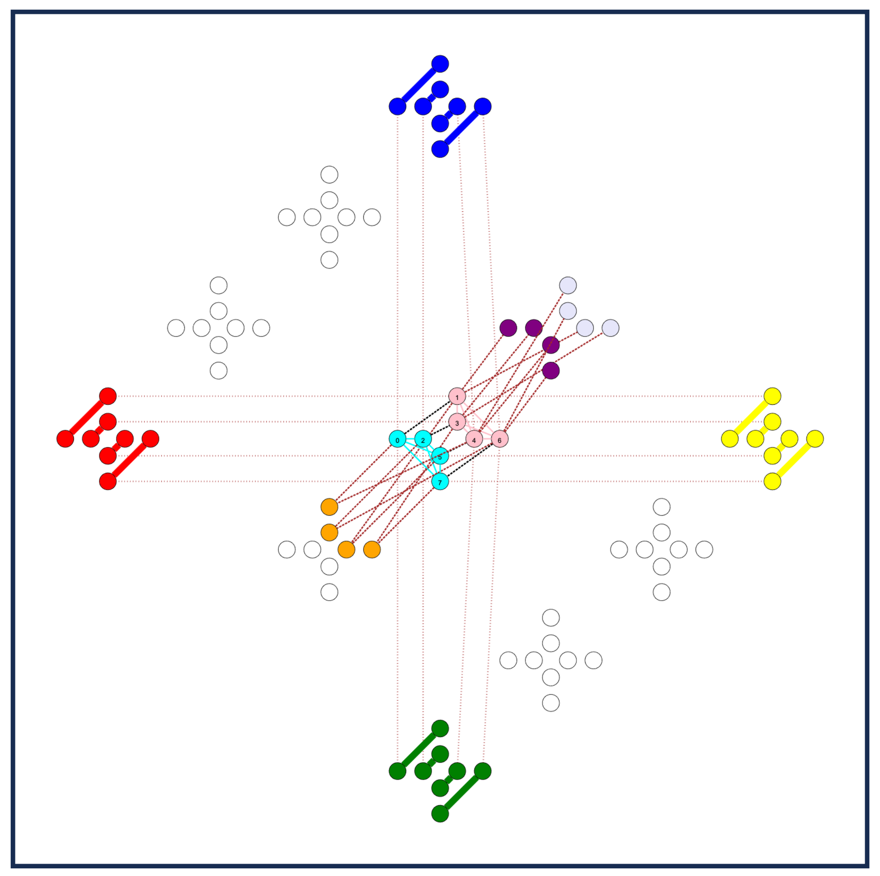
- Good candidates for modular embedding, thanks to constraint-based nature
- NchooseK uses constraints which say "Of N variables, K must be true"
- Constraints take the form `nck([N_1, …, N_n], {K_1, …, K_m})`
- Many NP problems have been solved with Nchoosek
  - One-hot encoding problems particularly suited to this type of embedding
    - Several qubits represent one variable
    - Qubit measured as |1> indicates "hot" value
  - Map coloring problem good example here
  - Map coloring uses 2 kinds of constraints, shown below:
    - Circles represent variables (regions P and Q)
    - Boxes represent constraints, number shows K for that constraint



- One constraint per node to ensure one color per node:
  - `nck({a_1, a_2, …, a_n}, {1})`
- n constraints per edge ensuring two nodes of the same color not connected:
  - `nck({a_1, b_1}, {0, 1})`







Connectivity of central cells on section of DWave Pegasus architecture
Abstract representation in upper right corner
Thick colored line indicates chain representing single virtual qubit

NchooseK Usage

```
import nchoosek
env = nchoosek.Environment()
P = ['Pr', 'Po', 'Pg', 'Pb']
Q = ['Qr', 'Qo', 'Qg', 'Qb']
for i in P + Q:
      env.register_port(i)
# One color per node
env.nck(P, {1})
env.nck(Q, {1})
# No shared color between regions
for idx, p in enumerate(P):
      env.nck([p, Q[idx]], {0, 1})
env.solve()
```

Methodology

We decided to start our investigation with the Map Color and Clique Cover problems due to their one-hot encoding lending to distinct cells with predictable connections between them.

- First tried DWave's current Pegasus architecture.
- Tried to find good encodings for maps of 3 and 4 colors
- These cells need the following properties:
- **Each cell must be a clique**
  - Each qubit within the cell connected to every other qubit within the cell
- **Corresponding qubits in connected cells must be connected to each other**
  - Ex: P_r connected to Q_r in NchooseK example
- Cell for 3 colors use 4 qubits (bottom left)
  - Thicker line indicates a chain, representing one virtual qubit
  - Cells have a degree of 4
- Two cells found for 4 colors
  - One version uses 8 qubits per cell (bottom center)
    - Cells have a degree of 6
  - Other version uses 4 qubits per cell (bottom right, unused)
    - Cells have degree of 2 or 3
    - Additional constraints on placement depending on connectivity
    - Cells need to be combined to connect with cells not on the same diagonal
- Minorminer was used to map on our abstract maps and on the full Pegasus map

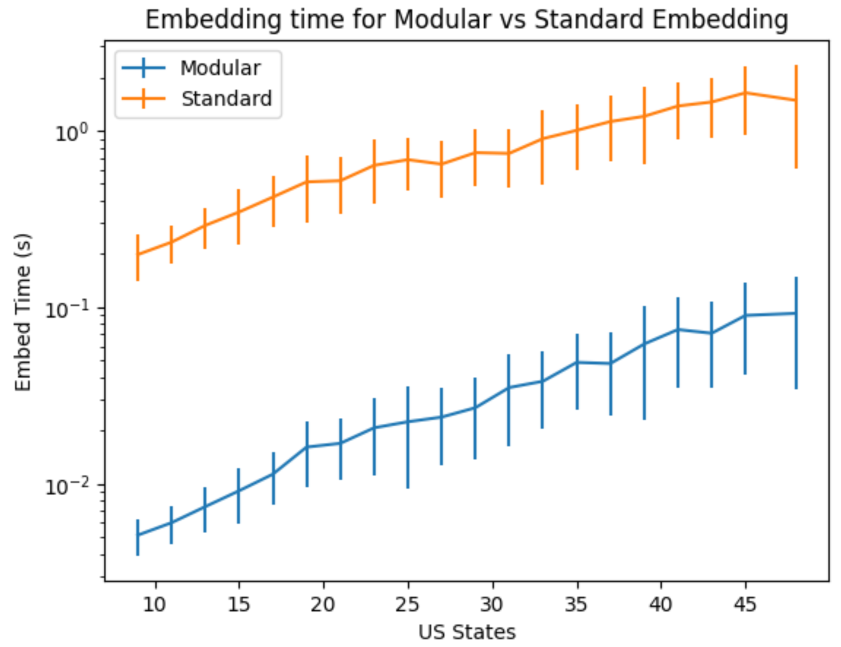
Results

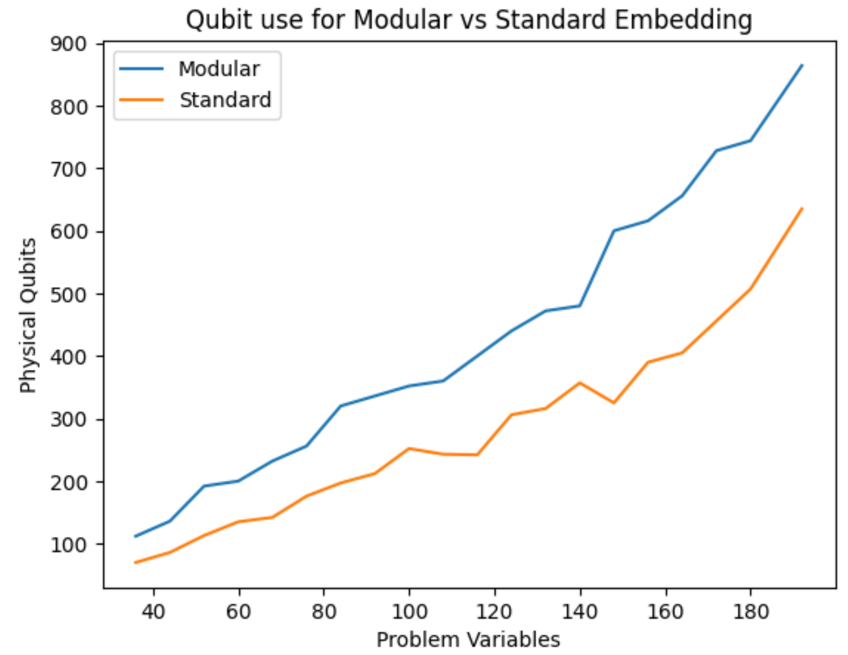
- Map color problem based on US continental map, starting with Tennessee
- Compared metrics of the mappings: Average/Max chain length, total number of qubits, embedding time
  - Modular (blue) performed worse than standard (red), except in embed time
- Ran some problems on physical machines, compared correctness
  - Modular mappings once again underperformed when compared to standard mapping

Number of States	Virtual Qubits	% Correct	Physical Qubits	Av. Chain Length	Max Length	Embed Time (s)
9	36	14	112	3.111	4	0.0051
11	44	6	136	3.091	6	0.0060
15	60	1	200	3.333	8	0.0091
19	76	0	256	3.368	6	0.0161
25	100		352	3.52	6	0.0224
35	140		480	3.429	8	0.0486
48	192		864	4.5	12	0.0922

Number of States	Virtual Qubits	% Correct	# Qubits	Av. Chain Length	Max Length	Embed Time (s)
9	36	25	70	1.944	3	0.1984
11	44	22	86	1.955	2	0.2333
15	60	5	122	2.033	4	0.3123
19	76	4	176	2.358	4	0.5123
25	100	1	252	2.520	5	0.6845
35	140	0	357	2.412	5	1.0033
48	192	0	635	3.307	9	1.4851

TABLE: Selected results for 4 color maps of US.
Top: Modular embedding. Bottom: Standard embedding.
3 color, clique cover, and DWave Chimera architecture performed similarly.





Conclusions

- Finding good modular maps is non-trivial
  - Modular embedding is faster but worse than full map embedding in this case
  - Modular embedding is better in other situations [1]
  - Likely better performance with cell size 2 or different necessary connections between cells
  - Unable to take advantage of many connections on Pegasus

References/Acknowledgements

https://github.com/lanl/NchooseK

[1] Joseph Fustero, Scott Palmtag, and Frank Mueller "Quantum Annealing Stencils with Applications to Fuel Loading of a Nuclear Reactor" in *IEEE International Conference on Quantum Computing and Engineering (QCE)*, Oct 2021

[2] Ellis Wilson, Frank Mueller, and Scott Pakin "Combining hard and soft constraints in quantum constraint-satisfaction systems" In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '22).* IEEE Press, 2022, Article 13, 1–14.

Research presented in this paper was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project number 20210397ER. Los Alamos National Laboratory is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy (contract no. 89233218CNA000001). This work was also supported in part by LANL subcontract 725530 and by NSF awards DMR-1747426, PHY-1818914, OAC-1917383, MPS-2120757, and CISE-2217020.