Title: Multilevel Quantum Annealing for Graph Partitioning

Author(s): Mniszewski, Susan M.
Ushijima-Mwesigwa, Hayato Montezuma
Negre, Christian Francisco Andres
Safro, Ilya

Intended for: Quantum Computing Workshop, Argonne National Laboratory,
2018-07-25/2018-07-27 (Lemont, Illinois, United States)

Issued: 2018-07-31
Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.
Multilevel Quantum Annealing For Graph Partitioning

Hayato Ushijima-Mwesigwa  Christian F. A Negre, Susan M. Mniszewski, Ilya Safro

Quantum Computing Workshop
Argonne National Laboratory

July 27, 2018
Motivation

**Challenge:**
Near-term Quantum Computing (QC) hardware size is small
- D-Wave 2000Q $\approx$ 2048 qubits
- IBM $\approx$ 50 qubits
- Google $\approx$ 72 qubits
- Intel $\approx$ 50 qubits

**Question:**
How can we efficiently use near-term QC devices for solving large-scale problems?

**Approach:**
Hybrid classical-quantum algorithms within the multilevel framework
Multilevel Methods:
- Technique useful for problems with multiple scales of behavior
- Major phases:
  - Coarsening Phase
  - Initial Solution
  - Uncoarsening Phase
    - Interpolation
    - Refinement
Applications of Multilevel Methods
Multilevel Methods For Optimization

- Line search multigrid for convex optimization
  (Goldfarb, Wen)

- PDE-constrained optimization
  (Borzi, Nash, Toint, ...)

- Multilevel trust-region methods
  (Gratton, Mouffe, Sartenaer, Toint, ...)

- Non-convex non-linear optimization for VLSI placement
  (Chan, Cong, Sze, ...)

- Linear programming - multilevel iterative methods
  (Gelman, Mandel, ...)

- Derivative-free multilevel optimization
  (Mendonca, Peckman, Toint, ...)
Coarsening
Create a hierarchy of restriction operators and corresponding coarse problems $P_0, \ldots, P_k$

Uncoarsening
Gradually approximate solutions $S_{k-1}, \ldots, S_0$ by (1) interpolation from previous level, and (2) further refinement

Exact solution
Multilevel Methods For Combinatorial Optimization

- Interpolation
- Relaxation
- Refinement

Exact (or best possible) solution

Coarsening

Coarsest graph

Uncoarsening

Graph Partitioning

July 27, 2018 7 / 31
Examples: VLSI Placement, Partitioning, Minimum Linear Arrangement, Minimum Bandwidth, Clustering, TSP, Community Detection, Segmentation, Visualization, ...

Quality: Usually exhibit superior results to other methods on practical test suites. Why? Because it is easy to combine the multiscale frameworks with other methods.

Time: Usually exhibit linear time complexity with no hidden coefficients.

Technical advantage: Admits parallelization. Suitable for various HPC configurations.
Question: Is the multilevel approach suitable for my problem, $P$?

**Refinement Requirements:**

- Refinement algorithm - Does a refinement algorithm exist?
- Can refinement algorithm handle additional restrictions caused by coarsening phase?
  - e.g., coarser graphs are weighted in GP
- For some problems, only known heuristics are based on construction rather than refinement
  - Not clear if multilevel can be applied
Coarsening Requirements:

- Solution in any of the coarsened spaces should induce a solution on the original space
  - current solution could be extended through all levels to a solution of the original problem
  - coarse solution should have the same cost with respect to objective function
  - goal is to find set of coarse variables that in future would interpolate their solution to fine variables
Graph Partitioning

Graph Partition Problem:
- Given $G = (V, E)$
- $V \sim$ nodes, $E \sim$ edges
- **Goal**: Partition $V$ into $k$ approximately equal parts minimizing the number of cut edges between parts

Applications:
- Graph-based QMD simulations
- VLSI design
- Load balancing - minimize communication between processors
- Sparse matrix-vector multiplication - Partition rows to minimize communication
- Social networks, cyber networks, ...
Partitioning large graphs is often an important subproblem for complexity reduction/parallelization

**Research in Graph partitioning**

- NP-hard: uses heuristics and approximation algorithms
- Very active area of research spanning over 50 years
- Most successful practical methods use multilevel paradigm
- Popular mutlilevel tools:
  - **CHACO** by Hendrickson and Leland, since 1993
  - **METIS** by Karypis and Kumar, since 1995
  - **SCOTCH** by Pellegrini, since 1996
  - **JOSTLE** by Walshaw, since 1995
  - **KAHIP** by Schulz, since 2013
Solving Optimization Problems on D-Wave 2X

- Formulate as unconstrained quadratic integer problem
  \[ \min_{q_1, \ldots, q_n} \left( \sum_{i=1}^{n} a_i q_i + \sum_{1 \leq i < j \leq n} a_{ij} q_i q_j \right) \]
  - Ising formulation if \( q_i \in \{-1, 1\} \)
  - QUBO formulation if \( q_i \in \{0, 1\} \)

- Map problem onto D-Wave hardware
  - Embed graph defined by \( a_{ij} \) into D-Wave hardware (Chimera) graph

Challenges:
- Sparse connectivity of chimera graph
- Limited precision
- Max size arbitrary QUBO \( \approx 45 \) variables
QUBO formulations for Graph Partitioning

Constrained formulation for 2 parts:

\[
\begin{align*}
\text{minimize} & \quad x^T L x \\
\text{subject to} & \quad \sum x_i = n/2 \\
& \quad x_i \in \{0, 1\}, \; i = 1, \ldots, n
\end{align*}
\]

Unconstrained (QUBO) formulation for 2 parts:

\[
\begin{align*}
\text{minimize} & \quad x^T L x + \alpha (\sum_i x_i - n/2)^2 \\
& \quad x_i \in \{0, 1\}, \; i = 1, \ldots, n
\end{align*}
\]

\(\alpha \sim \text{penalty constant (balanced parts)}\)
QUBO formulations for $k$-Graph Partitioning

Constrained formulation for $k$ parts:

\[
\text{minimize} \quad \sum_{j=1}^{k} x_j^T L x_j \\
\text{subject to} \quad \sum_{i} x_{i,j} = n/k, \quad j = 1, \ldots, k \\
\sum_{j} x_{i,j} = 1, \quad i = 1, \ldots, n \\
x_{i,j} \in \{0, 1\}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, k
\]

Unconstrained (QUBO) formulation for $k$ parts:

\[
\text{minimize} \quad \sum_{j=1}^{k} x_j^T L x_j + \sum_{j=1}^{k} \alpha_j (\sum_{i=1}^{n} x_{i,j} - \frac{n}{k})^2 \\
+ \sum_{i=1}^{n} \gamma_i (\sum_{j=1}^{k} x_{i,j} - 1)^2 \\
x_{i,j} \in \{0, 1\}
\]

- $\alpha_j, \gamma_i \sim$ penalty constants
Current work:

1. Coarsening Phase
   - Max edge weight matching
   - Algebraic Multigrid
   - Future work: coarsening with quantum device

2. Initial Partition
   - Exact solver
   - D-Wave

3. Uncoarsening/Refinement:
   - Kernighan-Lin and it’s variations
   - D-Wave refinement
Multilevel Graph Partitioning with D-Wave

D-Wave is used for
- Initial Partitioning
- Refinement

Multilevel Quantum Annealing for GP
Question: How good is D-Wave for initial partitioning?

Approach: We study the following,

1. Quality of partitioning unweighted graphs
2. Quality of partitioning weighted graphs with uniform volume
1. **Quality of partitioning unweighted graphs:**

   - **Graph data:**
     - Walshaw benchmark archive (http://chriswalshaw.co.uk/partition/)
     - Molecule electronic structure graphs from QMD simulations
     - Random graph models

   - **Tools:**
     - SAPI, D-Wave API
     - qbsolv: hybrid method with D-Wave and tabu search

   - **Experiment:**
     - D-Wave Vs KaHIP, (solution quality)
     - D-Wave Vs METIS, (solution quality)
Initial Partitioning: $k$-graph partitioning

- Dense random graphs
- Using sapi for embedding and solving
- Limited to $\approx 45$ node graph
- 15-node graph into 4 parts and 20-node graph into 3 parts used 900+ qubits
- Results comparable for SAPI, METIS and qbsolv
- Results using SAPI are typically equal to qbsolv

<table>
<thead>
<tr>
<th>$n$</th>
<th>$k$</th>
<th>SAPI</th>
<th>METIS</th>
<th>qbsolv</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>33</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>45</td>
<td>47</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>62</td>
<td>62</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>70</td>
<td>73</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>83</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>122</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>156</td>
<td>164</td>
<td>156</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>182</td>
<td>183</td>
<td>182</td>
</tr>
</tbody>
</table>
Initial Partitioning: \( k \)- Graph Partitioning

- Dense random graphs
- Using qbsolv for large graphs
- Produces \( kn \times kn \) QUBO
- Typically equal or better than METIS

<table>
<thead>
<tr>
<th>( n )</th>
<th>( k )</th>
<th>METIS</th>
<th>qbsolv</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>2</td>
<td>13691</td>
<td>13600</td>
</tr>
<tr>
<td>4</td>
<td>20885</td>
<td>20687</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>24384</td>
<td>24459</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>26224</td>
<td>26176</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>2</td>
<td>55333</td>
<td>54999</td>
</tr>
<tr>
<td>4</td>
<td>83175</td>
<td>83055</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>98073</td>
<td>97695</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>105061</td>
<td>105057</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>221826</td>
<td>221420</td>
</tr>
<tr>
<td>4</td>
<td>334631</td>
<td>334301</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>392018</td>
<td>392258</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>421327</td>
<td>420970</td>
<td></td>
</tr>
</tbody>
</table>
Initial Partitioning with D-Wave

Quality of partitioning weighted graphs:

- **Graph data:**
  - Random graph models
    - 420 nodes
    - Vary edge probability $p$
    - Edge weight $\sim \text{uniform}(1, 100)$

- **Tools:**
  - `qbsolv`

- **Experiment:**
  - D-Wave Vs KaffpaE, (solution quality)
    - Partition into $k = 2, 3, 4, 5, 6, 7$
    - KaffpaE run 20 times for each $k$
    - Save KaffpaE best, mean and worst cut value
    - Compare quality
Initial Partitioning: Weighted Graphs

**Experiment:**

- D-Wave Vs KaffpaE, (solution quality)
- Partition into $k = 2, 3, 4, 5, 6, 7$
- KaffpaE run 20 times for each $k$
- Save KaffpaE best, mean and worst cut value
- Compare quality

- Smaller than 1 means qbsolv was better

**Conclusion:** Positive results for initial partitioning
Question:
How to refine (improve) a given partition with D-Wave?

Kernighan-Lin algorithm review:
- An **iterative**, 2-way, balanced partitioning heuristic
- Each iteration:
  - Vertex pairs with the largest decrease or the smallest increase in cut size are exchanged
  - These vertices are then locked
  - locked vertices do not participate in any further exchanges
  - Process continues until all the vertices are locked
Refinement: D-Wave

KL Refinement Summary:
- At each pass, two nodes are swapped and gain function updated
- Developed for 2-way partitioning

D-Wave Refinement:
- Use D-Wave to swap set of free nodes $V' \subset V$ at once!
- Can handle $k$-way partitioning

Choice of free nodes:
- Current implementation: random choice of boundary nodes
**Question:** How powerful can quantum annealing be for refinement?

**Experiment:**
- Assume $h$ is size of quantum annealing hardware
- Start at random solution
- Choose $h$ nodes at random
- Optimize $h$ nodes at each iteration (system call)
- One iteration = One system call
- $h \approx 45$ for D-Wave 2X
Question: How powerful can quantum annealing be for refinement?

- If $h = 1024$, need less than 5 system calls.

Special Thanks: Fujitsu Digital Annealer for $h = 1024$!
Experiments: Final Partitioning Results

- Graph data
  - Walshaw benchmark graphs with less than 20k nodes

- Experiment
  - One V-cycle D-Wave Vs One V-cycle KaHip
  - Compare with best known solution
Results: Walshaw Graphs

- Graphs between 2000 – 17000 nodes
- Achieved best known value for 3 graphs with less than 80 system calls
- Results comparable with known solvers
Summary

- Multilevel framework ideal for near-term quantum computing hardware
- D-Wave gives high quality initial partitions
- Archived best known results with for 3 graphs with less 50 systems calls on average
Future Work:

- Coarsening for GP with quantum annealing
- Improved choice of free nodes in refinement algorithm
- Quantum enhanced coarsening for other combinatorial optimization problems