Badger Rampage: Multi-Dimensional Balanced Partitioning of Facebook-scale Graphs

Grigory Yaroslavtsev (Indiana University, Bloomington)
http://grigory.us/blog

Dmitry Avdyukhin (Indiana University), Sergey Pupyrev (Facebook)

2nd Workshop on Local Algorithms, MIT, June 14, 2018
“Three Schools of Thought” in Algorithms & Complexity

- **Boston (MIT & Harvard)**
  - **Youthful & innovative attacks on problems**, driven by PhD students with new ideas (“grad student descent”)
  - **“Relentless optimism ;)****: faster algorithms, e.g. sublinear time, gradient descent, unconditional results
“Three Schools of Thought” in Algorithms & Complexity

- **New York & Chicago** (Princeton, NYU, U Chicago)
  - Abstract and skeptical theory building, driven by fundamental questions and big agendas
  - “Life is hard...”: polynomial-time, hardness of approximation, conditional hardness, beyond-worst case analysis
“Three Schools of Thought” in Algorithms & Complexity

• **Bay area** (Stanford & Berkeley)
  
  – *No time for philosophy*, driven by applications and societal needs
  
  – “Let’s start a company and change the society!”: machine learning/AI, fairness, social networks, privacy
This talk

• “Boston school”
  – Fast, optimistic and specific: sublinear time, streaming, distributed, gradient descent

• “Bay area school”
  – Driven by applications, does it work in practice and scale to large data?
Balanced Graph Partitioning

• Partition $G(V, E)$ into $k$ parts $V_1, V_2, \ldots, V_k$:
  – Each part contains $(1 \pm \epsilon) \frac{|V|}{k}$ vertices
  – # of edges inside the parts is maximized

• Goal: make it work for the real Facebook graph
  – Load balancing
  – Community detection
  – Selecting representative subsets for training
  – ...
Facebook Graph

# vertices \( \approx 2 \times 10^9 \), #edges \( \approx 10^{12} \)
Hard in Theory, Important in Practice

• Minimizing the cut
  – No constant-factor approximation for $\epsilon = 0, k \geq 3$ unless $P = NP$ [Andreev, Racke’06]
  – Best approximation: polylog [Feige, Krauthgamer’02]

• Max $n/2$-UNCUT
  – $\approx 0.64$ via SDP [Halperin, Zwick, IPCO’01]

• If approximate balance is allowed, what is the hardness of this problem?
Hard in Theory, Important in Practice

• Previous generation tools:
  – METIS [Karypis, Kumar, ‘95]

• Google:
  – Linear embedding: [Aydin, Bateni, Mirrokni, WSDM’16]

• Facebook:
  – Label propagation: [Ugander, Backstrom, WSDM’13]
  – SocialHash partitioner: [Kabiljo, Karrer, Pundir, Pupyrev, Shalita, Akhremtsev, Presta, VLDB’17]
  – Spinner [Martella, Logothetis, Loukas, Siganos, ICDE’17]

• Some other papers:
  – FENNEL [Tsourakakis, Gkantsidis, Radunovic, Vojnovic, WSDM’14]
Multidimensional Balanced Graph Partitioning

• Balance according to multiple weights ($\geq 0$)
  – Each vertex $i$ has $d$ weights: $w_{i,1}, w_{i,2}, \ldots, w_{i,d}$
  – Let $w_j(S) = \sum_{i \in S} w_{ij}$ for each $j \in [d]$
  – Want $w_j(V_t) = \frac{(1 \pm \epsilon)w_j(V)}{k}$ for each part $V_t$

• Balanced graph partitioning: $d = 1, \forall i: w_{i1} = 1$

• Balance of the sum of degrees in each part:
  \[ w_{i2} = \text{deg}(i) \]

• **Note:** can be impossible as weights are unrelated
Existing approaches are combinatorial

• Local search, branch and bound, “linear embedding”, etc ...

• Difficult to extend to the multi-dimensional case
  – Don’t scale very well
  – Don’t produce good results

• Our approach is \textbf{gradient descent based}:
  – Easy to implement
  – Scales well on Facebook-scale graphs
  – Handles multiple balance constraints naturally
Quadratic Integer Program

• Variable $x_i$ for each vertex:
  • $i \in V_1: x_i = 1$
  • $i \in V_2: x_i = -1$

Maximize: $\sum_{(i_1, i_2) \in E} \frac{1}{2} (x_{i_1} x_{i_2} + 1)$

Subject to: $|\sum_{i=1}^{n} w_{ij} x_i| \leq \epsilon \sum_{i=1}^{n} w_{ij} \quad \forall j \in [d]$  
$x_i \in \{-1,1\} \quad \forall i \in V$
Non-convex relaxation

- $x_i \rightarrow$ continuous variables

Maximize:
\[
\sum_{(i_1, i_2) \in E} \frac{1}{2} (x_{i_1} x_{i_2} + 1)
\]

Subject to:
\[
|\sum_{i=1}^{n} w_{ij} x_i| \leq \epsilon \sum_{i=1}^{n} w_{ij} \quad \forall j \in [d]
\]
\[
x_i \in [-1,1] \quad \forall i \in V
\]
Randomized Projected Gradient Descent

- Objective: $f(x) = x^T A x$ (up to constants)
  - $\nabla f(x) = Ax, \; \nabla^2 f(x) = A$

- Projected Gradient Descent
  - Set $x_0 = 0$
  - For $i = 1 \ldots t$:
    - Gradient step: $y_i = x_i + \gamma \cdot \nabla f(x_i) = x_i(I + \gamma A)$
    - Project on the feasible space: $x_{i+1} = Proj(y_i)$

- Note that $x_0 = 0$ is a saddle point
  - Add random noise: $x'_i = x_i + N_d(0,1)$
Projection Step

• Proj($y_i$) is $x = \text{closest}^*$ point to $y_i$ satisfying:

\[
\left| \sum_{i=1}^{n} w_{ij} x_i \right| \leq \varepsilon \sum_{i=1}^{n} w_{ij} \quad \forall j \in [d]
\]

\[
x_i \in [-1,1] \quad \forall i \in V
\]

* closest in $\ell_2$ (Euclidean distance)

• Projection is a computationally expensive step
  • For $d = 1$ can be done in $O(n)$ time [Maculan, et al. ‘03]
  • For $d = 2$ we give an $O(n \log^2 n)$ time algorithm
  • Open: Give $\tilde{O}(n)$ time algorithm for any fixed $d$
Badger Rampage: \textbf{BalanceD GRaph Partitioning via RAnDoMized Projected Gradient DEscent}

• Set $x_0 = 0$
• For $i = 1 \ldots t$:
  • Gradient step: $y_i = (x_i + N_d(0,1)) \cdot (I + \gamma A)$
  • Project on the feasible space: $x_{i+1} = \text{Proj}(y_i)$

* If fractional values remain, use them as rounding probabilities.

**Open:** What can we say about convergence?
  – Randomized PGD converges to a local minimum \textbf{if all constraints are equalities} [Ge, Huang, Jin, Yuan, COLT’15]
  – With inequalities even computing Frank-Wolfe conditional gradient is NP-hard
Projection Problem

- Feasible region: $B_\infty \cap \left( \bigcap_{j=1}^{d} S_j^\epsilon \right)$, where:
  - $\ell_\infty$-ball $B_\infty = \{ x \in R^n \mid x_i \in [-1; 1] \}$
  - Slice $S_j^\epsilon = \{ x \in R^n \mid |\sum_{i=1}^{n} w_{ij} x_i| \leq \epsilon \sum_{i=1}^{n} w_{ij} \}$

- Approaches:
  - Solve exactly using KKT conditions
  - Alternating projections:
    $P_{B_\infty} (P_{S_1^\epsilon} (P_{S_2^\epsilon} ( \ldots P_{S_d^\epsilon} (P_{B_\infty} ( \ldots (y) \ldots ) ) \ldots ))$
    - Finds a point in the feasible space, not necessarily closest
  - Dykstra’s projection algorithm
    - Converges to the projection
Projection problem

Minimize: \( f(x) = \|x - y\|_2^2 \)

Subject to:

\[
x_i^2 \leq 1 \quad \forall i \in [n]
\]

\[
\sum_{i=1}^{n} w_{ij} x_i \leq c \quad \forall j \in [d]
\]

\[
\sum_{i=1}^{n} w_{ij} x_i \geq -c \quad \forall j \in [d]
\]
After simplifying KKT conditions...

- KKT is equivalent to finding $\lambda_1, \ldots, \lambda_d$ such that $\mathbf{x}$ satisfies the constraints, where:
  - $x_i = [y_i - \sum_j \lambda_j w_{ij}]$, where $[\cdot]$ is rounding to $[-1,1]$.
  - I.e. shift $y$ by a lin. combination, then project on $B_\infty$.

- $\mathbf{x}$ is the projection if it satisfies constraints:
  - $\lambda_j < 0 \Rightarrow \sum_i w_{ij} x_i = c$
  - $\lambda_j = 0 \Rightarrow \sum_i w_{ij} x_i \in [-c, c]$
  - $\lambda_j > 0 \Rightarrow \sum_i w_{ij} x_i = -c$
Finding $\lambda_1, \ldots, \lambda_d$

- For each $j$ there are 3 cases:
  - $\lambda_j < 0 \Rightarrow \sum_i w_{ij} x_i = c$
  - $\lambda_j = 0 \Rightarrow \sum_i w_{ij} x_i \in [-c, c]$
  - $\lambda_j > 0 \Rightarrow \sum_i w_{ij} x_i = -c$

- Try $3^d$ combinations. Select the best point
  - For each unknown $\lambda_j$ we have equality constraints
  - Projection on $B_\infty \cap (\cap_{i=1}^d A_i)$, where $A_i$ are hyperplanes

- Can find $\lambda_1, \ldots, \lambda_d$ using nested binary search
  - $O(n \log n)$ for $d = 1$ and $O(n \log^2 n)$ for $d = 2$
  - Conjecture: $\tilde{O}(n)$ for any fixed $d$
Balanced Graph Partitioning

• Implementation in Apache Giraph

• Percentage of cut edges on subsets of the Facebook graph (allowed vertex imbalance – 3%).

<table>
<thead>
<tr>
<th>Graph</th>
<th>Badger Rampage</th>
<th>SocialHash</th>
<th>Spinner</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB-2.5B</td>
<td>5.11%</td>
<td>8.75%</td>
<td>13.30%</td>
</tr>
<tr>
<td>FB-55B</td>
<td>4.99%</td>
<td>11.75%</td>
<td>12.79%</td>
</tr>
<tr>
<td>FB-80B</td>
<td>5.21%</td>
<td>12.04%</td>
<td>8.64%</td>
</tr>
<tr>
<td>FB-400B</td>
<td>6.88%</td>
<td>5.82%</td>
<td>6.31%</td>
</tr>
<tr>
<td>FB-800B</td>
<td>5.52%</td>
<td>5.25%</td>
<td>6.83%</td>
</tr>
</tbody>
</table>
2D Balanced Graph Partitioning

- Percentage of cut edges on public graphs (allowed imbalance on vertices and degrees – 1%).

<table>
<thead>
<tr>
<th>Graph</th>
<th>Badger Rampage–exact projection</th>
<th>Badger Rampage – alternating projection</th>
<th>Spinner</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiveJournal</td>
<td>6.74%</td>
<td>6.74%</td>
<td>9.53%</td>
</tr>
<tr>
<td>Orkut</td>
<td>5.14%</td>
<td>4.9%</td>
<td>5.68%</td>
</tr>
<tr>
<td>ego-Gplus</td>
<td>12%</td>
<td>12.2%</td>
<td>44.5%</td>
</tr>
</tbody>
</table>
Step size selection ($\gamma$)

- Cut size per iteration as a function of $\gamma$
Future work

• $\tilde{O}(n)$ algorithm for fixed $d$?
• Guarantees on convergence of Badger Rampage?
• Practical algorithm for more than 2 parts
  – Currently use recursive partitioning
  – Can modify the approach to support $k$ parts, but time and memory increase by factor $k$