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

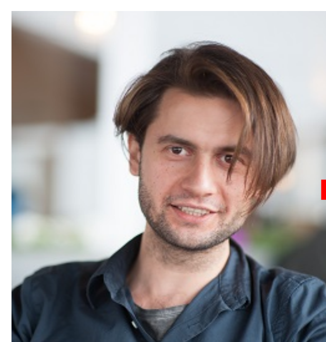
“Three Schools of Thought” in Algorithms & Complexity

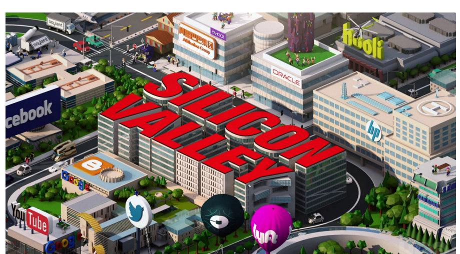
- **Bay area** (Stanford & Berkeley)
  - **No time for philosophy,** driven by applications and societal needs
  - **“Let’s start a company and change the society!”**: machine learning/AI, fairness, social networks, privacy

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 $\boldsymbol{V_1}, \boldsymbol{V_2}, \ldots, \boldsymbol{V_k}$:
  - 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 $\approx \mathbf{2 \times 10^9}$, **#edges** $\approx \mathbf{10^{12}}$

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
  - ≈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 (≥ 0)
  - Each vertex i has $\boldsymbol{d}$ weights: $w_{i,1}, w_{i,2}, \ldots, w_{i,\boldsymbol{d}}$
  - Let $w_j(S) = \sum_{i \in S} w_{ij}$ for each $j \in [\boldsymbol{d}]$
  - Want $w_j(V_t) = \frac{(1 \pm \epsilon) w_j(\mathrm{V})}{k}$ for each part V_t
- Balanced graph partitioning: $\boldsymbol{d} = 1, \forall i: w_{i1} = 1$
- Balance of the sum of degrees in each part:

$$w_{i2} = \deg(i)$$

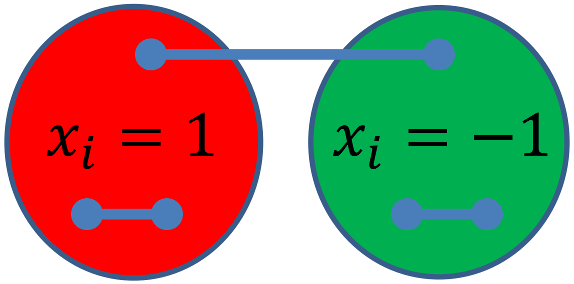
- **Note**: can be impossible as weights are unrelated

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 **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:
  - $i \in \boldsymbol{V_1}: x_i = 1$
  - $i \in \boldsymbol{V_2}: x_i = -1$



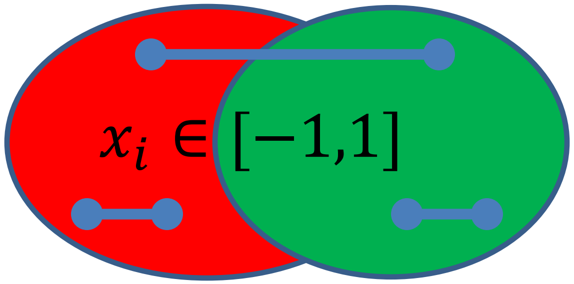
Maximize: $$\sum_{(i_1,i_2)\in E} \frac{1}{2}\left(x_{i_1}x_{i_2} + 1\right)$$

Subject to: $$\left|\sum_{i=1}^{n} w_{ij} x_i\right| \le \epsilon \sum_{i=1}^{n} w_{ij} \quad \forall j \in [\boldsymbol{d}]$$

$$x_i \in \{-1,1\} \quad \forall i \in V$$

Non-convex relaxation

- $x_i \to$ continuous variables



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

Subject to: $\left|\sum_{i=1}^{n} w_{ij}x_i\right| \le \epsilon \sum_{i=1}^{n} w_{ij} \quad \forall j \in [\boldsymbol{d}]$

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

Randomized Projected Gradient Descent

- Objective: $f(\boldsymbol{x}) = \boldsymbol{x}^T A\, \boldsymbol{x}$ (up to constants)
  - $\nabla f(\boldsymbol{x}) = A\boldsymbol{x}, \;\; \nabla^2 f(\boldsymbol{x}) = A$
- Projected Gradient Descent
  - Set $\boldsymbol{x}_0 = \mathbf{0}$
  - For $i = 1 \ldots t$:
    - Gradient step: $\boldsymbol{y}_i = \boldsymbol{x}_i + \boldsymbol{\gamma} \cdot \nabla f(\boldsymbol{x}_i) = \boldsymbol{x}_i(I + \boldsymbol{\gamma} A)$
    - Project on the feasible space: $\boldsymbol{x}_{i+1} = Proj(\boldsymbol{y}_i)$
- Note that $\mathbf{x}_0 = \mathbf{0}$ is a saddle point
  - Add random noise: $\boldsymbol{x}_i' = \boldsymbol{x}_i + N_{\boldsymbol{d}}(0,1)$

Projection Step

- Proj($\boldsymbol{y_i}$) is $\boldsymbol{x}$ = closest* point to $\boldsymbol{y_i}$ satisfying:

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

$$x_i \in [-1,1] \qquad \forall i \in V$$

* closest in ℓ_2 (Euclidean distance)

- Projection is a computationally expensive step
  - For $\boldsymbol{d} = 1$ can be done in $O(n)$ time [Maculan, et al. '03]
  - For $\boldsymbol{d} = 2$ we give an $O(n \log^2 n)$ time algorithm
  - **Open:** Give $\tilde{O}(n)$ time algorithm for any fixed $\boldsymbol{d}$

Badger Rampage:
Bal**A**nce**D** **GR**aph Partitioining via **RA**ndo**M**ized **P**rojected **G**radient **DE**scent

- Set $\boldsymbol{x}_0 = \boldsymbol{0}$
- For $i = 1 \dots t$:
  - Gradient step: $\boldsymbol{y}_i = (\boldsymbol{x}_i + N_{\boldsymbol{d}}(0,1)) \cdot (I + \boldsymbol{\gamma} A)$
  - Project on the feasible space: $\boldsymbol{x}_{i+1} = Proj(\boldsymbol{y}_i)$

- * If fractional values remain, use them as rounding probabilities
- **Open:** What can we say about convergence?
  - Randomized PGD converges to a local minimum **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: $\boldsymbol{B}_\infty \cap \left(\bigcap_{j=1}^{d} \boldsymbol{S}_\epsilon^j\right)$, where:
  - ℓ_∞-ball $\boldsymbol{B}_\infty = \{x \in R^n \mid x_i \in [-1;\ 1]\}$
  - Slice $\boldsymbol{S}_\epsilon^j = \{x \in R^n \mid \left|\sum_{i=1}^n w_{ij}\, x_i\right| \leq \epsilon \sum_{i=1}^n w_{ij}\}$
- Approaches:
  - Solve exactly using KKT conditions
  - Alternating projections:
    $P_{\boldsymbol{B}_\infty}(P_{\boldsymbol{S}_\epsilon^1}(P_{\boldsymbol{S}_\epsilon^2}(\ldots P_{\boldsymbol{S}_\epsilon^d}(P_{\boldsymbol{B}_\infty}(\ldots(y)\ldots)$
    - Finds a point in the feasible space, not necessarily closest
  - Dykstra's projection algorithm
    - Converges to the projection

Projection problem

Minimize: $f(\boldsymbol{x}) = ||\boldsymbol{x} - \boldsymbol{y}||_2^2$

Subject to:

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

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

$$\sum_{i=1}^{n} w_{ij} x_i \geq -c \qquad \forall j \in [d]$$

After simplifying KKT conditions…

- KKT is equivalent to finding $\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{\boldsymbol{d}}$ such that $\boldsymbol{x}$ satisfies the constraints, where
  - $x_i = [y_i - \sum_j \boldsymbol{\lambda}_j w_{ij}]$, where [] is rounding to [-1,1]
  - I.e. shift y by a lin. combination, then project on B_∞
- $\boldsymbol{x}$ is the projection if it satisfies constraints:
  - $\boldsymbol{\lambda}_j < 0 \Rightarrow \sum_i w_{ij} x_i = c$
  - $\boldsymbol{\lambda}_j = 0 \Rightarrow \sum_i w_{ij} x_i \in [-c, c]$
  - $\boldsymbol{\lambda}_j > 0 \Rightarrow \sum_i w_{ij} x_i = -c$

Finding $\lambda_1, \ldots, \lambda_{\boldsymbol{d}}$

- For each j there are 3 cases:
  - $\boldsymbol{\lambda}_j < 0 \Rightarrow \sum_i w_{ij} x_i = c$
  - $\boldsymbol{\lambda}_j = 0 \Rightarrow \sum_i w_{ij} x_i \in [-c, c]$
  - $\boldsymbol{\lambda}_j > 0 \Rightarrow \sum_i w_{ij} x_i = -c$
- Try 3^d combinations. Select the best point
  - For each unknown $\boldsymbol{\lambda}_j$ we have equality constraints
  - Projection on $\boldsymbol{B}_\infty \cap \left(\cap_{i=1}^{\boldsymbol{d}} \boldsymbol{A_i}\right)$, where $\boldsymbol{A_i}$ are hyperplanes
- Can find $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{\boldsymbol{d}}$ using nested binary search
  - $O(n \log n)$ for $\boldsymbol{d} = 1$ and $O(n \log^2 n)$ for $\boldsymbol{d} = 2$
  - **Conjecture:** $\tilde{O}(n)$ for any fixed $\boldsymbol{d}$

Balanced Graph Partitioning

- Implementation in Apache Giraph

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

Graph	Badger Rampage	SocialHash	Spinner
FB-2.5B	**5.11%**	8.75%	13.30%
FB-55B	**4.99%**	11.75%	12.79%
FB-80B	**5.21%**	12.04%	8.64%
FB-400B	6.88%	**5.82%**	6.31%
FB-800B	5.52%	**5.25%**	6.83%

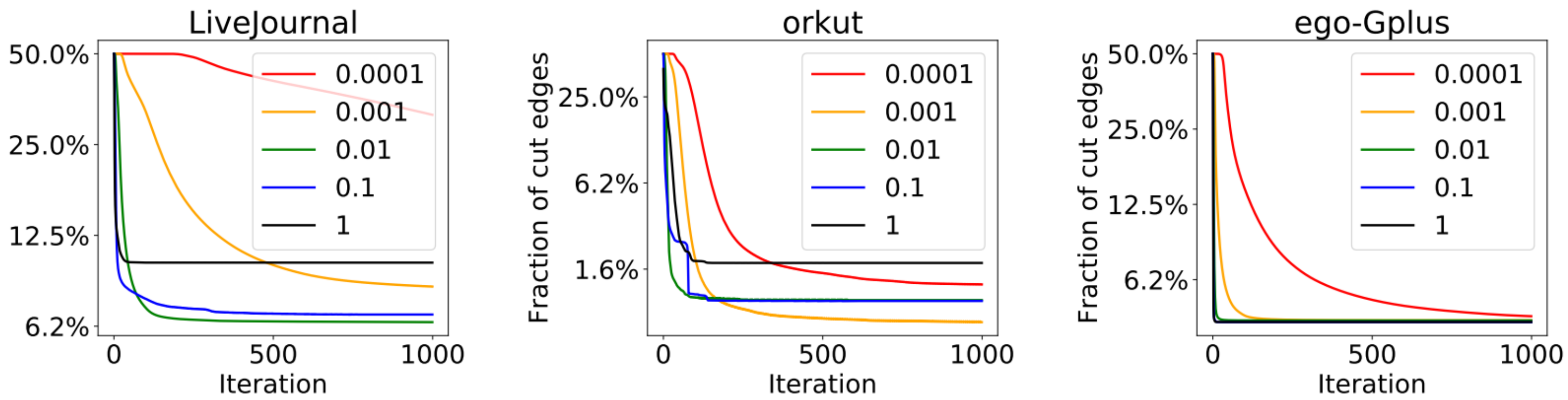
2D Balanced Graph Partitioning

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

Graph	Badger Rampage– exact projection	Badger Rampage – alternating projection	Spinner
LiveJournal	**6.74%**	6.74%	9.53%
Orkut	5.14%	**4.9%**	5.68%
ego-Gplus	**12%**	12.2%	44.5%

Step size selection (γ)

- Cut size per iteration as a function of γ



Future work

- $\tilde{O}(n)$ algorithm for fixed $\boldsymbol{d}$?
- 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