Data Driven Resource Allocation for Distributed Learning

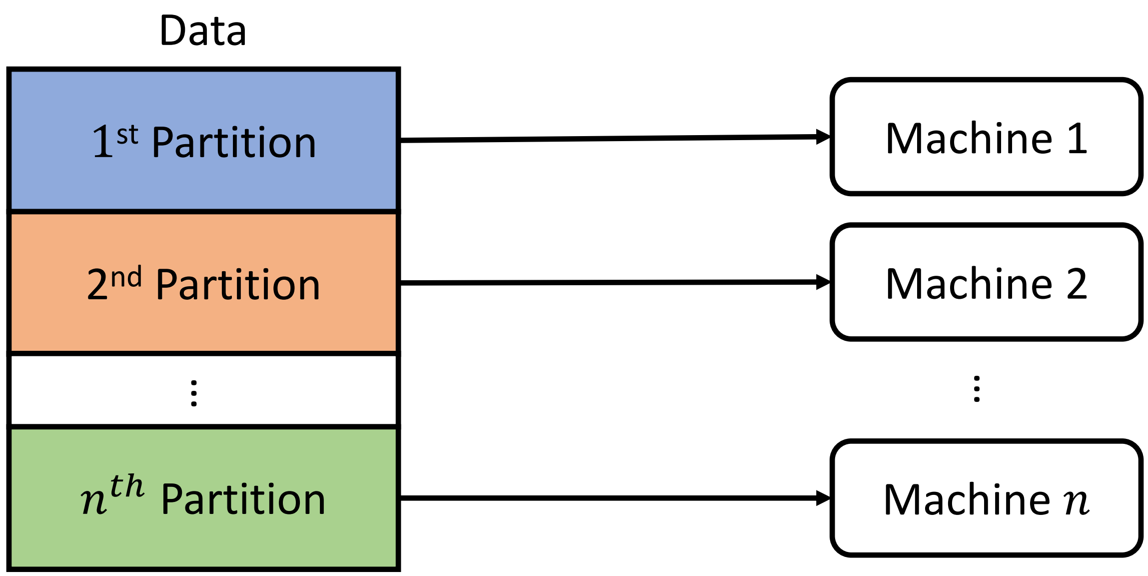
Travis Dick, Mu Li, Venkata Krishna Pillutla,
Colin White, Maria-Florina Balcan, Alex Smola

Outline

1. Problem
2. Clustering-based Data Partitioning
3. Summary of results
4. Efficiently Clustering Data
    1. LP-Rounding Approximation Algorithm
    2. Beyond worst-case analysis: k-means++
    3. Efficiency from subsampling
5. Experimental results
    1. Accuracy comparison
    2. Scaling experiments

The problem

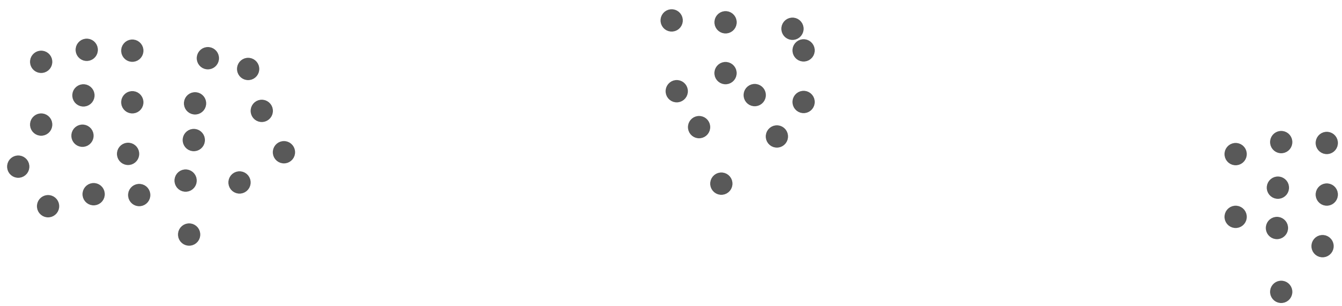
- Want to distribute data to multiple machines for efficient machine learning.
- ***How should we partition the data?***



- Common idea: randomly partition the data.
  - Clean both in theory and practice, but suboptimal.

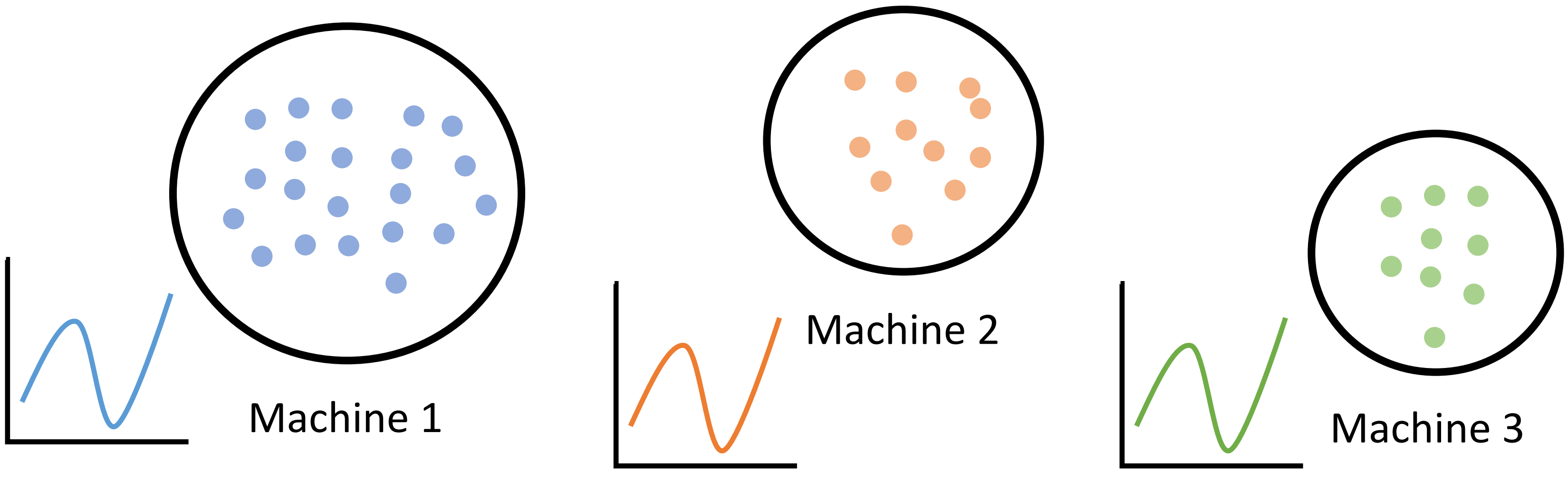
Our approach

- Cluster the data and send one cluster to each machine.
- Accurate models tend to be ***locally simple*** but ***globally complex.***



Our approach

- Cluster the data and send one cluster to each machine.
- Accurate models tend to be ***locally simple*** but ***globally complex.***



- Each machine learns a model for its ***local*** data.
- Additional clustering constraints:
  - ***Balance:*** Clusters should have roughly the same number of points.
  - ***Replication:*** Each point should be assigned to p clusters.

Summary of results

1. Efficient algorithms with provable worst-case guarantees for balanced clustering with replication.
2. For non-worst case data, we show that common clustering algorithms produce high-quality balanced clusterings.
3. We show how to efficiently partition a large dataset by clustering only a small sample.
4. We empirically show that our technique significantly outperforms baseline methods and strongly scales.

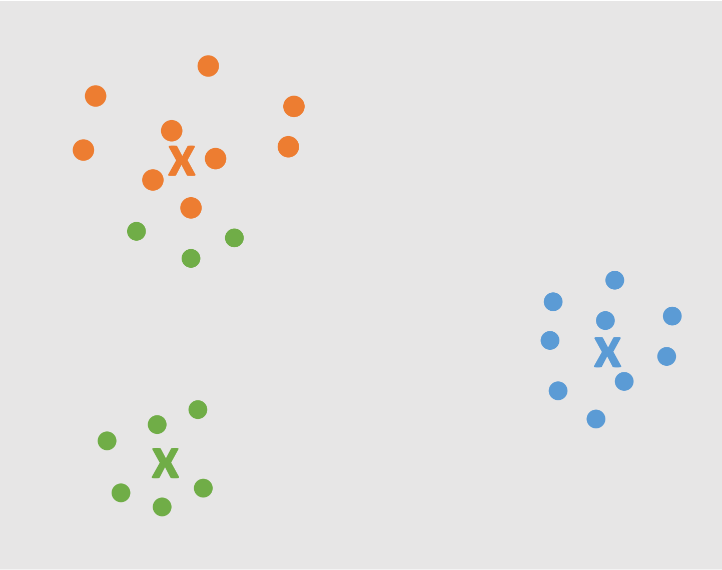
How can we efficiently compute balanced clusterings with replication?

Balanced k-means with replication

Given a dataset S…

- Choose k centers $c_1, \ldots, c_k \in S$,
- Assign each $x \in S$ to p centers: $f_1(x), \ldots, f_p(x) \in \{c_1, \ldots, c_k\}$

- k-means cost: $\sum_{x \in S} \sum_{i=1}^{p} d\left(x, f_i(x)\right)^2$.
- Balance constraints: Each center has between $\ell|S|$ and $L|S|$ points.



LP-Rounding Algorithm

- Can formulate problem as an ***integer program*** (NP Hard to solve exactly).
- The ***linear program*** relaxation can be solved efficiently…
  - but gives ”fractional” centers and “fractional” assignments.

1. Compute a coarse clustering of the data using a simple greedy procedure.
2. Combine centers within each coarse cluster to form “whole” centers.
3. Find optimal assignments by solving a min-cost flow problem.

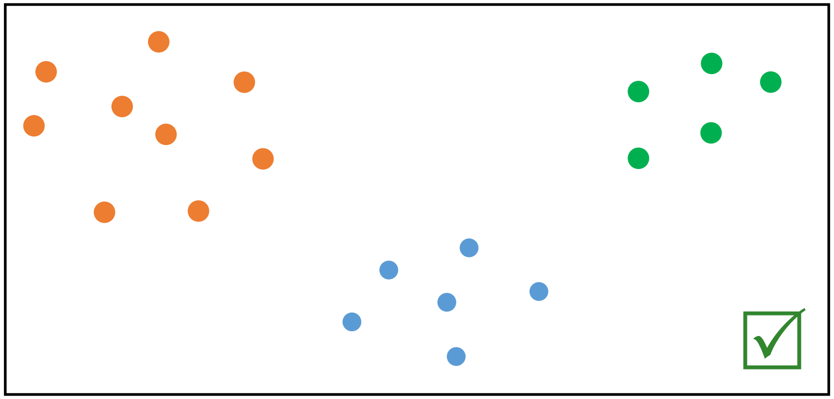
Theorem

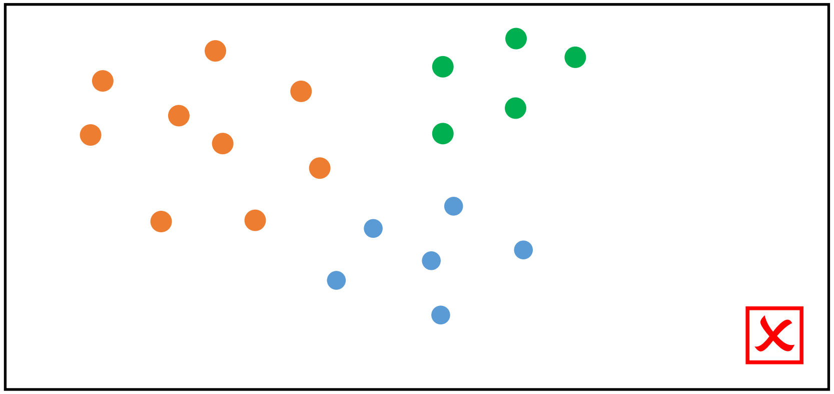
The LP-rounding algorithm returns a constant factor approximation for balanced k-means clustering with replication when $p \geq 2$ and violates the upper capacities by at most $\frac{p+2}{2}$.

* We have analogous results for k-median and k-center clustering as well.

Beyond worst-case: k-means++

- For non-worst-case data, common algorithms also work well!
- a clustering instance satisfies (α, ε)-***approximation stability*** if all clusterings C with $cost(C) \leq (1 + \alpha)OPT$ are ε-close to the optimal clustering. [1]





Theorem

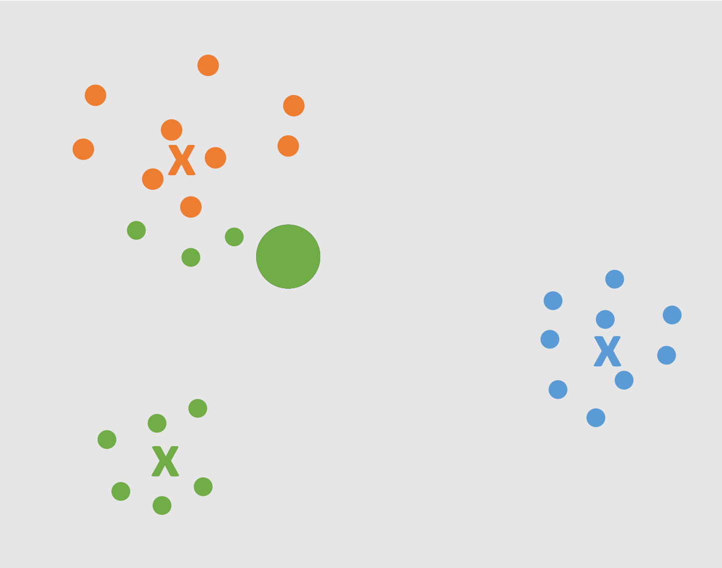
k-means++ seeding with greedy pruning [5] outputs a nearly optimal solution for balanced clustering instances satisfying (α, ε)-approximation stability

Efficiency from Subsampling

Goal: Cluster a small sample of data and use this to partition entire dataset with good guarantees.

Assign new point to the same clusters as its nearest neighbor.

- Automatically handles balance constraints.
- New point costs about the same as neighbor.



Theorem

If the cost on sample is $\leq r \cdot OPT$, then the cost of the extended clustering is $\leq 4r \cdot OPT + O(rpD^2\epsilon + pr\alpha + r\beta)$

- D is the diameter of the dataset
- α, β measure how representative the sample is, and go to zero as sample size grows.

How well does our method perform?

Experimental Setup

Goal: Measure the difference in accuracy from different partitioning methods.

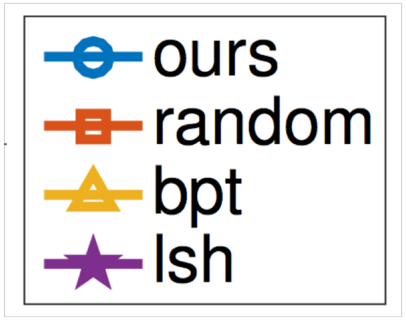
1. Partition the data using one of the partitioning methods.
2. Learn a model on each machine.
3. Report the test accuracy.

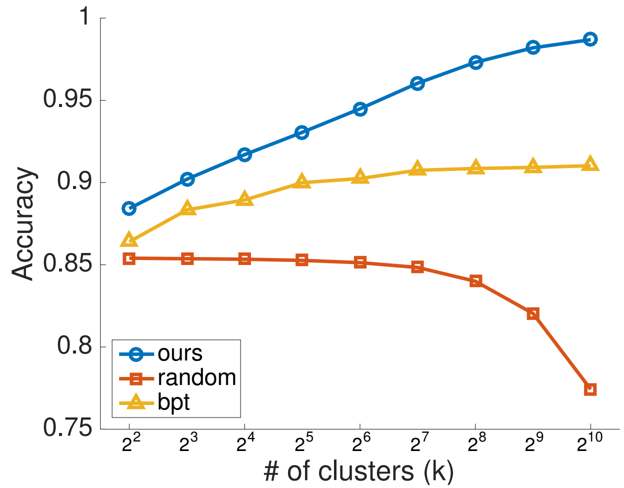
- Repeat for multiple values of k
  - Larger values of k are more parallelizable.

Baseline Methods

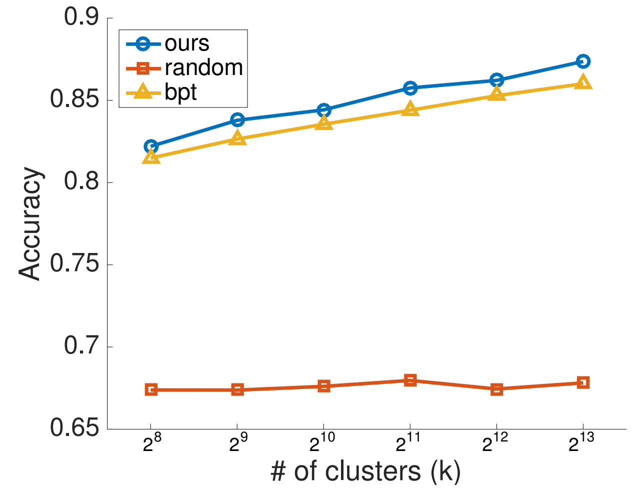
Method	Fast?	Balanced?	Locality?
Random			X
Balanced Partition Tree (kd-tree)			1/2
Locality Sensitive Hashing		X	
Our Method			

Experimental Evaluation

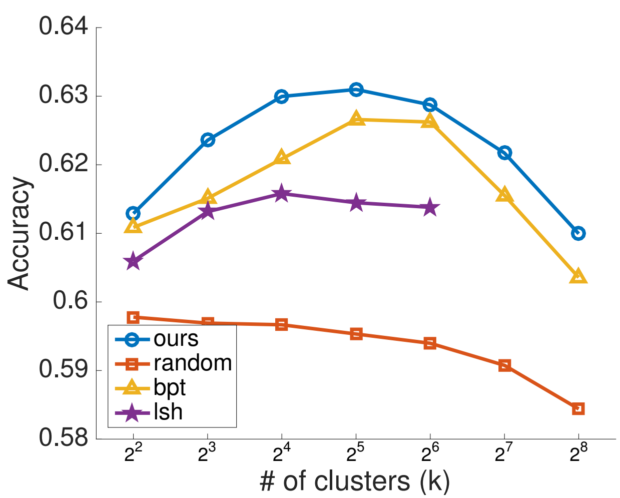




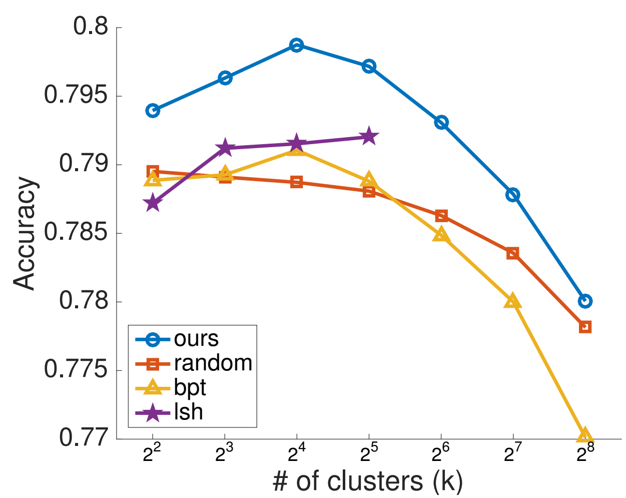
MNIST-8M



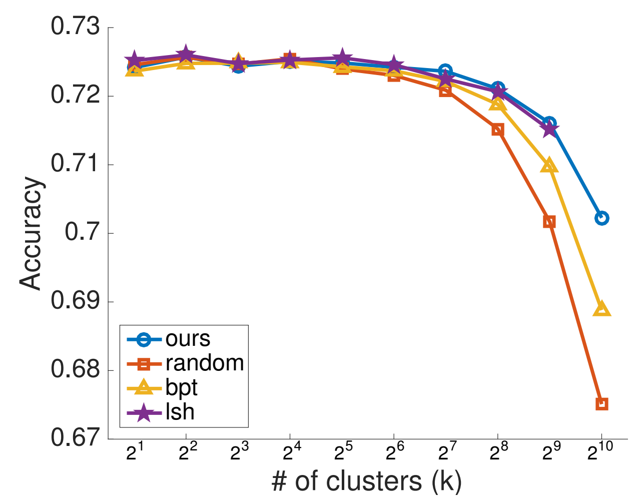
Synthetic



CIFAR10_IN3C



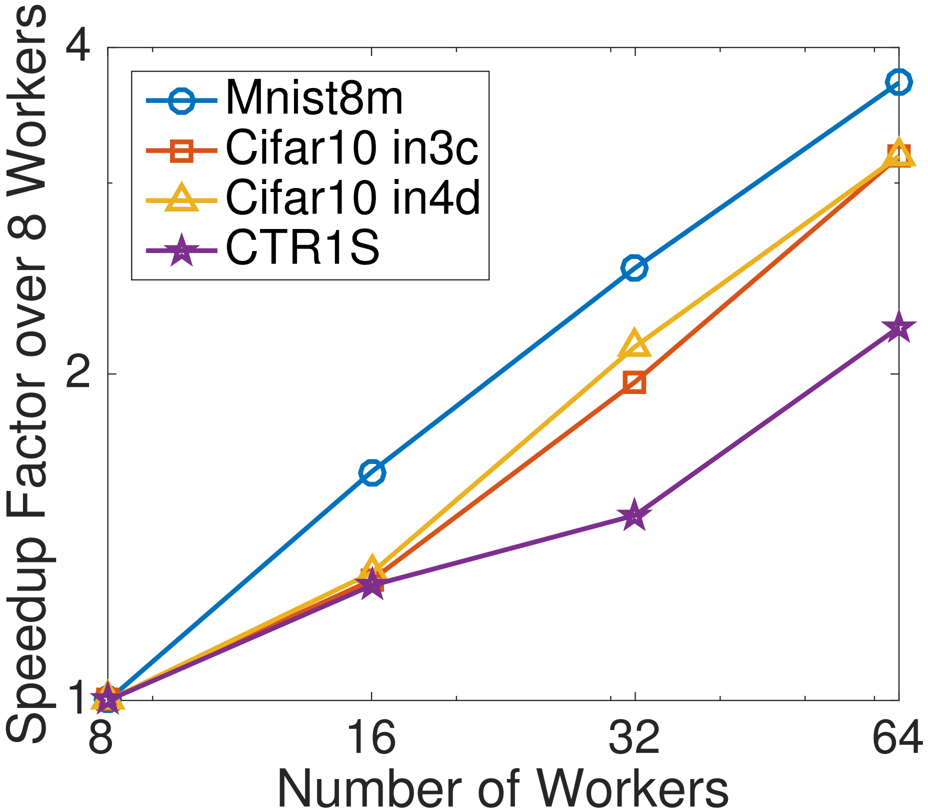
CIFAR10_IN4D



CTR

Strong Scaling

- For a fixed dataset we evaluate the running time of our method using 8, 16, 32, or 64 machines.
- We report the speedup over using 8 machines.
- For all datasets, doubling the number of workers reduces running time by a constant fraction (i.e., our method strongly scales).



Conclusion

- Propose using balanced clustering with replication for data partitioning in DML.
- LP-Rounding algorithm with worst-case guarantees.
- Beyond worst-case analysis for k-means++.
- Efficiently partition large datasets by clustering a sample.
- Empirical support for utility of clustering-based partitioning.
- Empirically demonstrated strong scaling.

Thanks!

Extra Slides (You’ve gone too far!)

Capacitated k-means with replication

Choose k centers and assign every point to p centers so that points are "close" to their centers and each cluster is roughly the same size.

As an Integer Program:

- Number the points 1 through n.
- Variable y_i = "1 if point i is a center, 0 otherwise."
- Variable $x_{i,j}$ = "1 if point j is assigned to point i."

Minimize $\sum_{i,j} x_{i,j}\, d(i,j)^2$
Subject to:

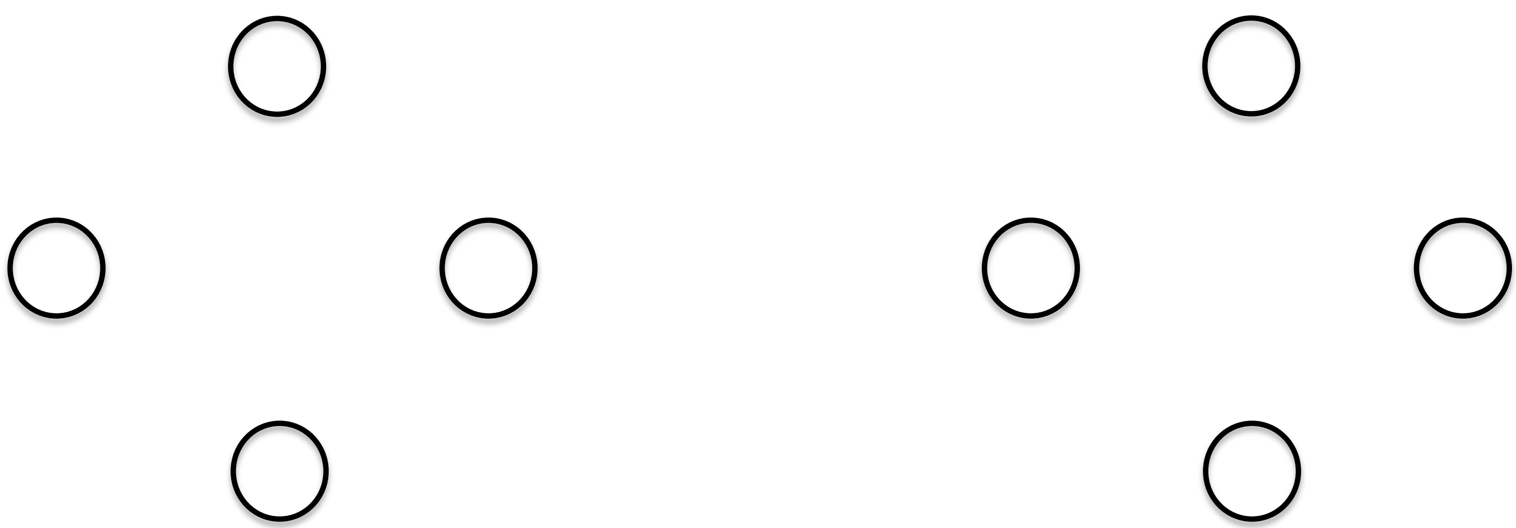
- $\sum_i x_{i,j} = p$ for all points j (p assignments)
- $\sum_i y_i = k$ (k centers)
- $\ell n y_i \leq \sum_j x_{i,j} \leq L n y_i$ for all points i (balancedness)
- ~~$x_{i,j}, y_i \in \{0,1\}$ for all points i, j.~~

Linear Program Relaxation: $x_{i,j}, y_i \in [0,1]$

xs are "fractional assignment", ys are "fractional centers"

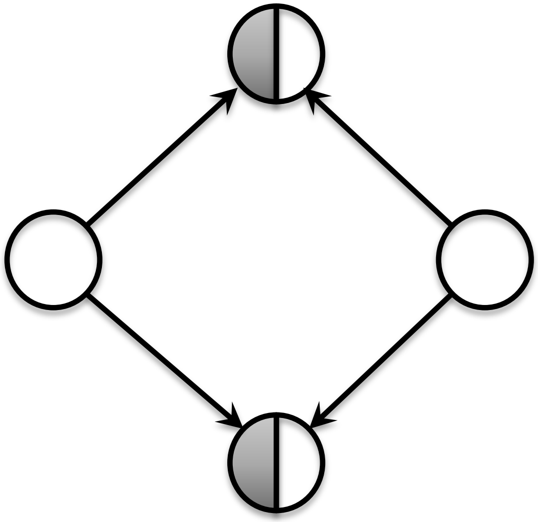
LP-Rounding algorithm

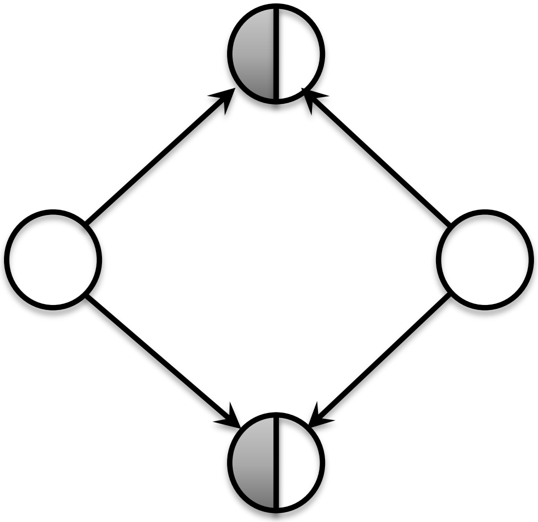
1. Solve the LP to get fractional ys and xs.
2. Compute a very coarse clustering of the points using a greedy procedure.
3. Within each coarse cluster, round the ys to 0 or 1.
4. Find the optimal integral xs by solving a min-cost flow.



LP-Rounding algorithm

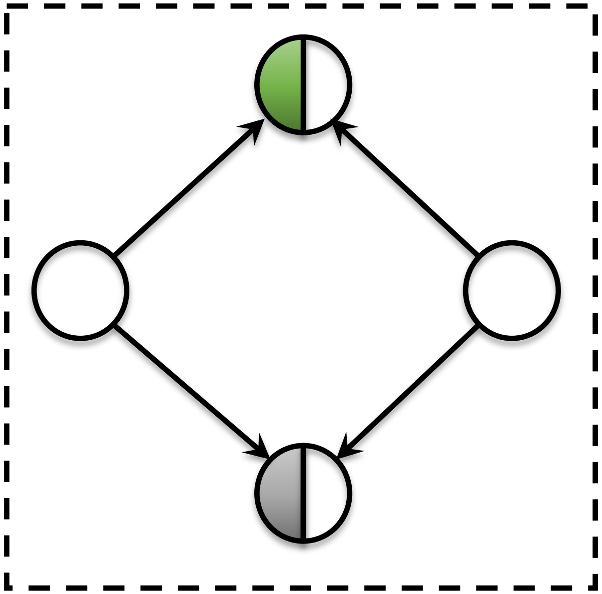
1. Solve the LP to get fractional ys and xs.
2. Compute a very coarse clustering of the points using a greedy procedure.
3. Within each coarse cluster, round the ys to 0 or 1.
4. Find the optimal integral xs by solving a min-cost flow.

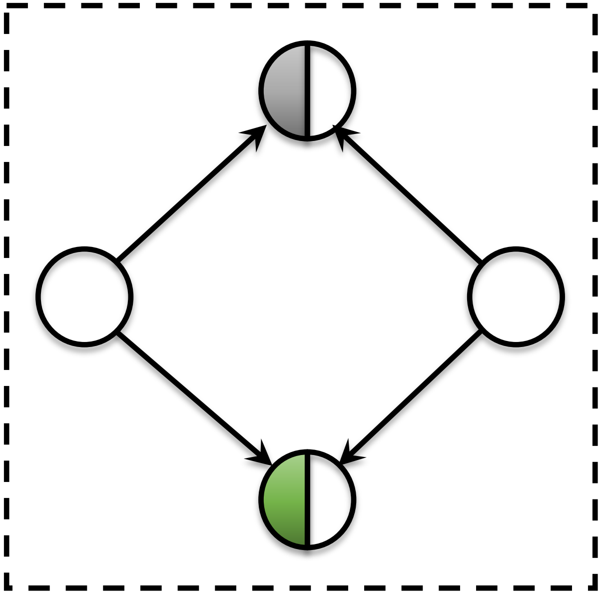




LP-Rounding algorithm

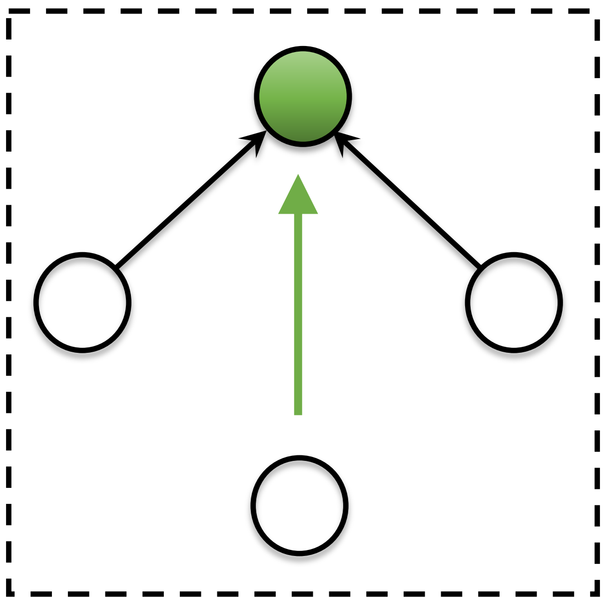
1. Solve the LP to get fractional ys and xs.
2. Compute a very coarse clustering of the points using a greedy procedure.
3. Within each coarse cluster, round the ys to 0 or 1.
4. Find the optimal integral xs by solving a min-cost flow.

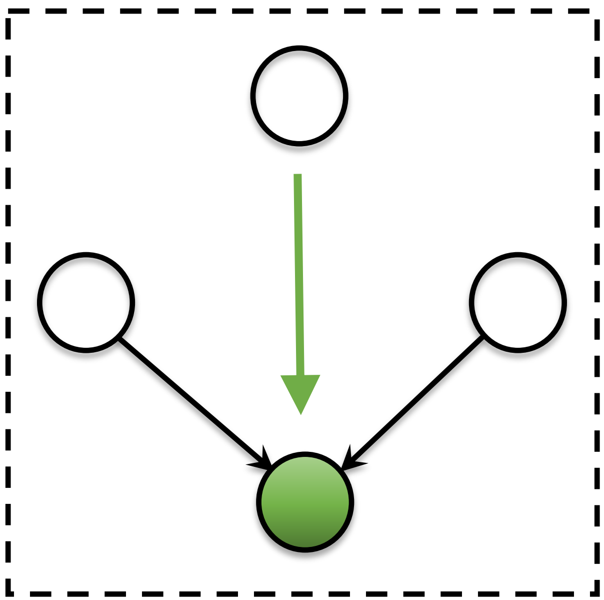




LP-Rounding algorithm

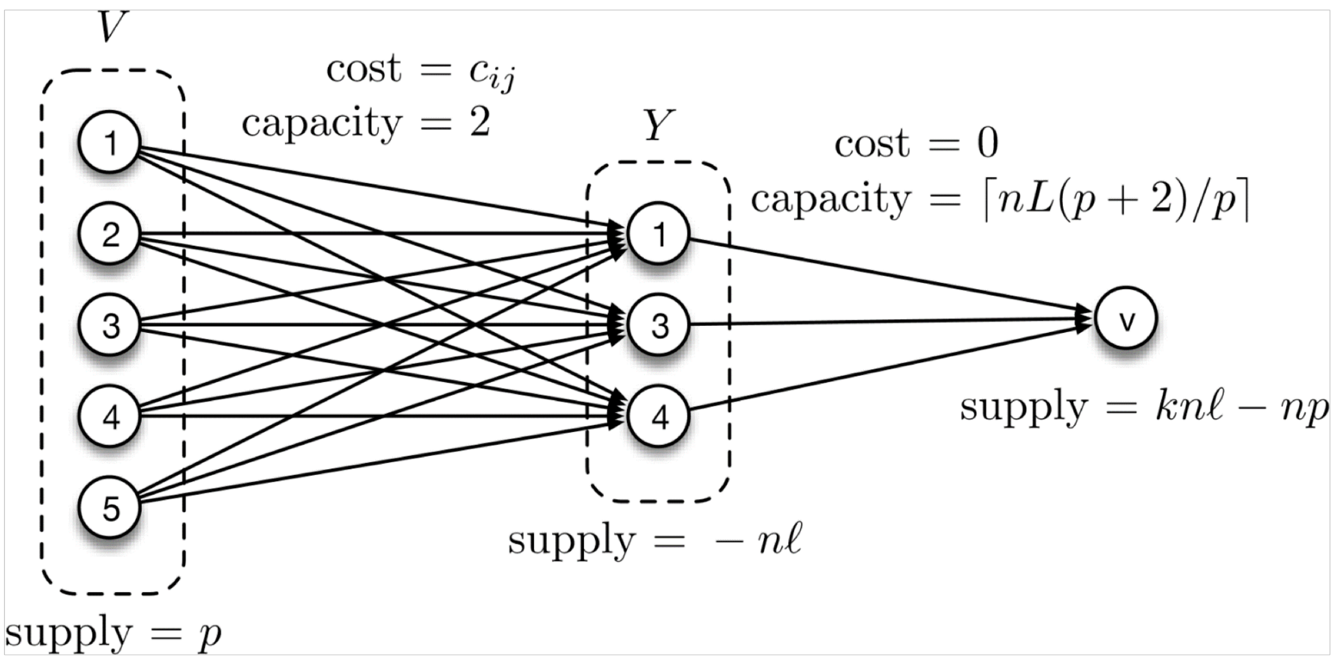
1. Solve the LP to get fractional ys and xs.
2. Compute a very coarse clustering of the points using a greedy procedure.
3. Within each coarse cluster, round the ys to 0 or 1.
4. Find the optimal integral xs by solving a min-cost flow.





LP-Rounding algorithm

1. Solve the LP to get fractional ys and xs.
2. Compute a very coarse clustering of the points using a greedy procedure.
3. Within each coarse cluster, round the ys to 0 or 1.
4. Find the optimal integral xs by solving a min-cost flow.



LP-Rounding Algorithm

1. Solve the LP to get fractional ys and xs.
2. Compute a very coarse clustering of the points using a greedy procedure.
3. Within each coarse cluster, round the ys to 0 or 1.
4. Find the optimal integral xs by solving a min-cost flow.

Theorem

The LP-rounding algorithm returns a constant factor approximation for capacitated k-means clustering with replication when $p > 1$ and violates the upper capacities by at most $\frac{p+2}{2}$.

* We have analogous results for k-means and k-center clustering as well.