Motivation

Why Approximate?
- Big data: GB, TB, PB, EB, and still increasing.
- Slow algorithm: from $O(n^2)$ to NP-Complete everywhere.
- Approximate or not approximate?
  - Big data beats clever algorithm.
  - Approximated answers are usually sufficient for analytics.
  - Approximated good vs. exact bad. Which will you choose?
  - Answer delayed is answer denied.

Why Auto?
- Developing "big data" solution is hard.
  - That's why we are having this conference.
- Designing approximation algorithms is hard.
- Designing distributed approx. algorithms is hard + hard.

Our Target
- Minimum level of user involvement.
- Approximation "for dummies".
- Ultimate Goal: "One button" approximation.

Our approach

Sample the PROGRAM, not the GRAPH!
- Why graph sampling is NOT working:
  - The vertices and edges are not independent at all.
  - The "connections" are exactly what we care most.

How we do it?
- Background: iterative, vertex-centric graph analytics.
- We ignore some of the program instructions. That's it!
- And carefully calibrate the answer to keep it unbiased.
  (See below.)

Why it works?
- Can we always do it?
  - Mostly. Even for complex queries, their vertex-centric instructions are usually simple and regular.
- Is the error small?
  - Mostly. Under mild assumptions, we show the error will converge (to a small value).
  - We analyze it by analogizing it to the social network information diffusion and denote it as error drifting. Check paper for details.

Toolbox for Program Sampling

For example, if the user sums all incoming messages, we sample each five of them.
- This is not graph sampling!
- We find such opportunities using program analyzing.
- And compensate the final sum to keep it unbiased.

Other alternatives:
- Memorization: remember past answers.
- Task skipping: take a nap this time. Work next call.
- Interpolation: simple function replace complex one.
- System function replacement: $1 + x$ for $exp(x)$.
- And still extending!

Does it work?

We conducted experiments on a wide spectrum of algorithms and datasets.
They demonstrates our approach works quite well on both convex and non-convex (!), continuous and discrete (!) cases.

x-axis is relative computation time compared to the original (un-approx.) algorithm, y-axis is relative error (in log scale), dots for datasets. Dot size for data size.

Opportunities and Challenges

By manipulating parameters it generates a wide spread trade-off between accuracy and time. We are investigating how to make full use of it. (Cost model? Discussions are welcomed.)

x-axis is relative computation time compared to the original (un-approx.) algorithm, y-axis is relative error (in log scale), dots for combinations of parameters.

Works for both synchronize and asynchronous computing. We will try distributed environment later.