A Scalable Parallel Algorithm for Balanced Sampling (Student Abstract)

Alexander Lee,[*1] **Stefan Walzer-Goldfeld,**[*1] **Shukry Zablah,**[2] **Matteo Riondato**[1]

[1] Box 2232, Dept. of Computer Science, Amherst College, Amherst, MA 01002, USA
[2] Pallet Labs Inc.
{awlee22, swalzergoldfeld23, mriondato}@amherst.edu, shukry@pallet.xyz

Abstract

We present a novel parallel algorithm for drawing balanced samples from large populations. When auxiliary variables about the population units are known, balanced sampling improves the quality of the estimations obtained from the sample. Available algorithms, e.g., the cube method, are inherently sequential, and do not scale to large populations. Our parallel algorithm is based on a variant of the cube method for stratified populations. It has the same sample quality as sequential algorithms, and almost ideal parallel speedup.

Introduction

Many approximation algorithms and heuristics for important and diverse data analytics tasks use *random sampling*: a small subset (or sample) of the data (or population), chosen at random, is analyzed quickly at the price of introducing some error in the estimation of the quantities of interest. When available, *additional information about the population units* can be used to obtain a *balanced sample* (see Preliminaries), i.e., a *more representative sample*, thus improving the quality of the estimation obtained from it. Balance sampling is used in the New French Census and the French Master Sample (Dessertaine 2006) as well as to improve the consumer price index in Italy (Biggeri and Falorsi 2006).

Deville and Tillé (2004) introduced the cube method, an algorithmic framework for drawing balanced samples with desired inclusion probabilities, later improved by Chauvet and Tillé (2006) (related work is discussed in the online supplement[1]). The cube method is *inherently sequential*, thus it becomes impractical on the humongous populations, i.e., datasets, that are common in modern data analytics.

Contributions We introduce a *parallel algorithm* for drawing balanced samples from large populations. At the heart of our approach is a variant of the cube method for stratified balanced sampling (Chauvet 2009): we use "fake" strata to split the work among the different processors. Our algorithm has the same balancing quality as the sequential algorithm, and it exhibits an almost ideal speedup as the number of available processors increases.

[*]These authors contributed equally.

[1]https://github.com/acdmammoths/parallelcubesampling.

Preliminaries

Let $\mathcal{U} \doteq \{u_1, \ldots, u_N\}$ be a *population* of N *units*. Each unit u_k is associated to a vector $\mathbf{a}_k \doteq \langle a_{k1}, \ldots, a_{kn}\rangle^\intercal$ of n *known auxiliary variables*, and to a *variable of interest* y_k, whose value is *unknown* unless we explicitly "query" u_k, which is assumed to be costly. We let $\mathbf{T} \doteq \sum_{k=1}^{N} \mathbf{a}_k$ be the vector of *population totals of the* n *auxiliary variables*. Since querying the whole population would be expensive, we want to *estimate* a value $\theta \doteq \sum_{k=1}^{N} f(y_k)$ for a given function f from a *random balanced sample* of the population, i.e., a subset of the population drawn according to a specific probability distribution over a specific collection of population subsets, as follows.

Any subset A of $\mathcal{U}$ can be represented as the binary vector $\mathbf{s}_A \doteq \langle s_1, \ldots, s_N\rangle^\intercal$ where, for $k = 1, \ldots, N$, $s_k = 1$ if $u_k \in A$, and $s_k = 0$ otherwise. Let $\boldsymbol{\pi} \doteq \langle \pi_1, \ldots, \pi_N\rangle^\intercal$ be a given vector of *inclusion probabilities*. Given $A \subseteq \mathcal{U}$, the estimate for θ on A is the *Horvitz-Thompson estimate* $\sum_{k=1}^{N} f(y_k)s_k/\pi_k$. Consider the $n \times N$ matrix $\mathbf{W}$ whose (i, j) entry is a_{ji}/π_j. The *Horvitz-Thompson estimate* for $\mathbf{T}$ on $A \subseteq \mathcal{U}$ is the vector $\widetilde{\mathbf{T}}_A \doteq \mathbf{W}\mathbf{s}_A$. A *balanced sampling design* Φ is a probability distribution over the *support* $\mathcal{S} \doteq \{\mathbf{s} \in \{0, 1\}^N \ : \ \mathbf{W}\mathbf{s} = \mathbf{T}\}$ such that the probability that unit u_k belongs to the sample is π_k. I.e., if we let $\mathbf{s} = \langle s_1, \ldots, s_N\rangle^\intercal$ be a *random* binary vector sampled according to Φ, it holds that $\Pr(s_k = 1) = \pi_k$. Φ is *balanced* because in every sample (i.e., those whose vectors are in $\mathcal{S}$), the population totals estimate matches its exact value.

Algorithm

As discussed, a balanced sample is a vector $\mathbf{s} \in \{0, 1\}^N$ that satisfies the *balancing equations*

$$\mathbf{W}\mathbf{s} = \mathbf{T} \ . \tag{1}$$

Geometrically, $\{0, 1\}^N$ is the set of vertices of the N-dimensional hypercube $C \doteq [0, 1]^N$. The solution space of the balancing equations (1), is the hyperplane $Q \doteq \boldsymbol{\pi} + \ker(\mathbf{W})$. Thus, to obtain a balanced sample from $\mathcal{S}$ by drawing from Φ, one must select a vertex of C that belongs to $K \doteq C \cap Q$, while respecting the inclusion probabilities $\boldsymbol{\pi}$. The *cube method* (Deville and Tillé 2004) for balanced sampling is designed around this geometrical intuition.

The cube method has two phases: flight and landing. The *flight phase* performs a random walk on the constraint space K, starting from the vector of inclusion probabilities $\boldsymbol{\pi}$, with appropriate transition probabilities, until it reaches a vertex $\mathbf{s}$ of K. $\mathbf{s}$ may or not be a vertex of C, i.e., some of its entries may be *nonintegral*. When this issue, known as the *rounding problem*, is encountered, the *landing phase* then moves from $\mathbf{s}$ to a $\mathbf{s}' \in \{0,1\}^N$ *close* to K, i.e., $\mathbf{s}'$ is *approximately balanced* ($\widetilde{\mathbf{T}}_{\mathbf{s}'} \approx \mathbf{T}$), and returns it.

Chauvet (2009) provides an extension of the cube method for *stratified populations*, i.e., when $\mathcal{U}$ is partitioned in *strata* $U_1, \ldots, U_H$. First, a flight phase is performed *separately on each stratum*, then the resulting per-stratum sample vectors are *concatenated* into a single one, and a second "global" flight phase is performed, starting from the concatenated vector, followed by a landing phase to solve the rounding problem, if present.

Our parallel algorithm is based on the stratified cube method. Let p be the number of available processors. For ease of presentation, and w.l.o.g., assume that N is divisible by p. We create $H = p$ *fake* strata of size N/p, by *arbitrarily* assigning units to the strata. The flight phases, one per stratum, can then be performed *all in parallel*. The algorithm then continues as in the stratified cube method.

Experimental Evaluation

The first goal of our evaluation is to assess the *balancing quality* of the samples produced by our algorithm, i.e., how well they satisfy the balancing equations (1). The second goal is to evaluate how our algorithm scales as the population size and the number of processors grow.

We implemented our parallel algorithm and the fast cube method (Chauvet and Tillé 2006) in Python.[2]

We used artificial datasets with $n = 10$ and $N \in \{2500, 5000, 10^4, 10^5, 10^6, 10^7\}$. The values of auxiliary variables were generated by randomly selecting values from a uniform distribution over the interval $[0, 1)$. The inclusion probabilities $\boldsymbol{\pi}$ were also drawn uniformly at random from $[0, 1)$. We ran the experiments on a x86–64 virtual machine with 32 vCPUs, 128 GB of RAM, and Amazon 2 Linux.

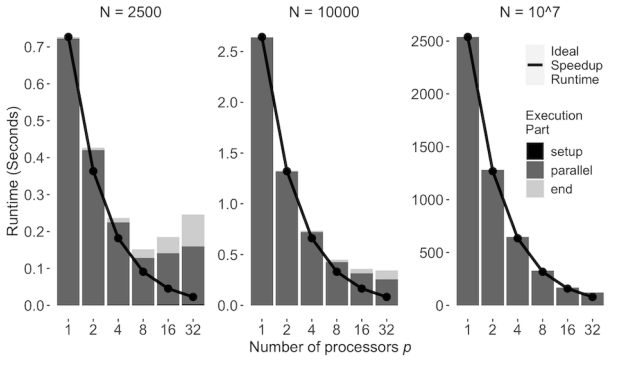

Figure 1: Parallel algorithm runtimes for different N and p.

[2]Implementations of all algorithms are available on our repository: https://github.com/acdmammoths/parallelcubesampling.

We measured the absolute relative deviation $100|\mathbf{W}\mathbf{s} - \mathbf{T}|/\mathbf{T}$ (where the division is component-wise) of both our implementations (parallel and sequential) and compared the obtained values with the implementations of the sequential cube method (Chauvet and Tillé 2006) in the `cubesampling` and `BalancedSampling` R packages. The balancing qualities of all four algorithms were extremely similar, confirming the correctness of our parallel approach. They all tend to slightly underestimate the population totals and the magnitude of the deviations is very small (less than 0.5%) (see Fig. 2 in the supplement.)

Fig. 1 shows the results of our runtime evaluation (results for other values of N are in the supplement). We split the runtime into three components: setup (creation of the fake strata and other data preparation), parallel (first flight phase), and end (concatenation, second flight phase, and eventual landing phase). At large population sizes common in modern data analytics ($N \geq 10^5$), our parallel algorithm shows close to an ideal speedup. When the population is smaller, the overhead at higher levels of parallelism becomes too costly, resulting in an *increase* of the runtime as the number p of processors grows. Additionally, the runtime for the "end" part is a larger portion of the total runtime, because the high number of strata implies a more serious rounding problem (more entries of the concatenated vector are nonintegral), thus the second flight phase and the landing phase take more time. The results of this experiment confirm the ability of our algorithm to make balanced sampling scalable for large populations.

Conclusion

We introduced a novel parallel algorithm for balanced sampling, to make this powerful statistical technique scale to large populations. Our approach offers a near-perfect parallel speedup, and produces samples of the same quality as sequential algorithms. In the future, we will develop novel applications of balanced sampling to data analytics.

Acknowledgements

This work is funded, in part, by NSF award IIS-2006765.

References

Biggeri, L.; and Falorsi, P. 2006. A probability sample strategy for improving the quality of the consumer price index survey using the information of the business register. In *Joint ECE/ILO meeting of the of Group of Experts on Consumer Price Indices*.

Chauvet, G. 2009. Stratified balanced sampling. *Survey Methodology*, 35(1): 115–119.

Chauvet, G.; and Tillé, Y. 2006. A fast algorithm for balanced sampling. *Computational Statistics*, 21(1): 53–62.

Dessertaine, A. 2006. Sondages et séries temporelles: une application pour la prévision de la consommation électrique. *Actes des journées françaises de Statistique*.

Deville, J.-C.; and Tillé, Y. 2004. Efficient balanced sampling: the cube method. *Biometrika*, 91(4): 893–912.