# MindFlayer: Efficient Parallel SGD with Random Worker Delays

Artavazd Maranjyan, **Omar Shaikh Omar**, Peter Richtárik

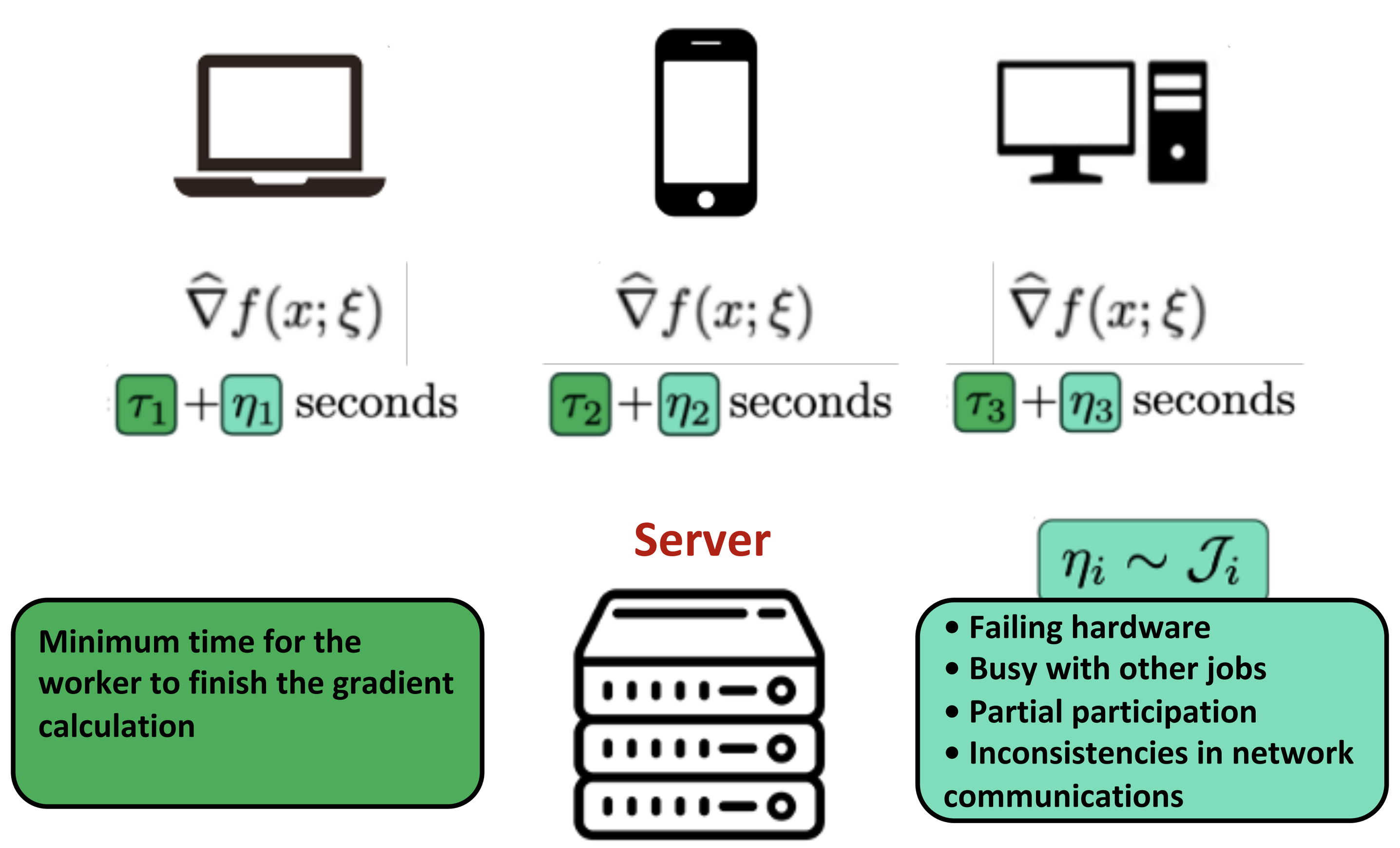
King Abdullah University of Science and Technology

## Problem Setup: Distributed Training

We address the nonconvex optimization problem:

$$\min_{x\in\mathbb{R}^d}\left\{f(x) := \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(x;\xi)\right]\right\},$$

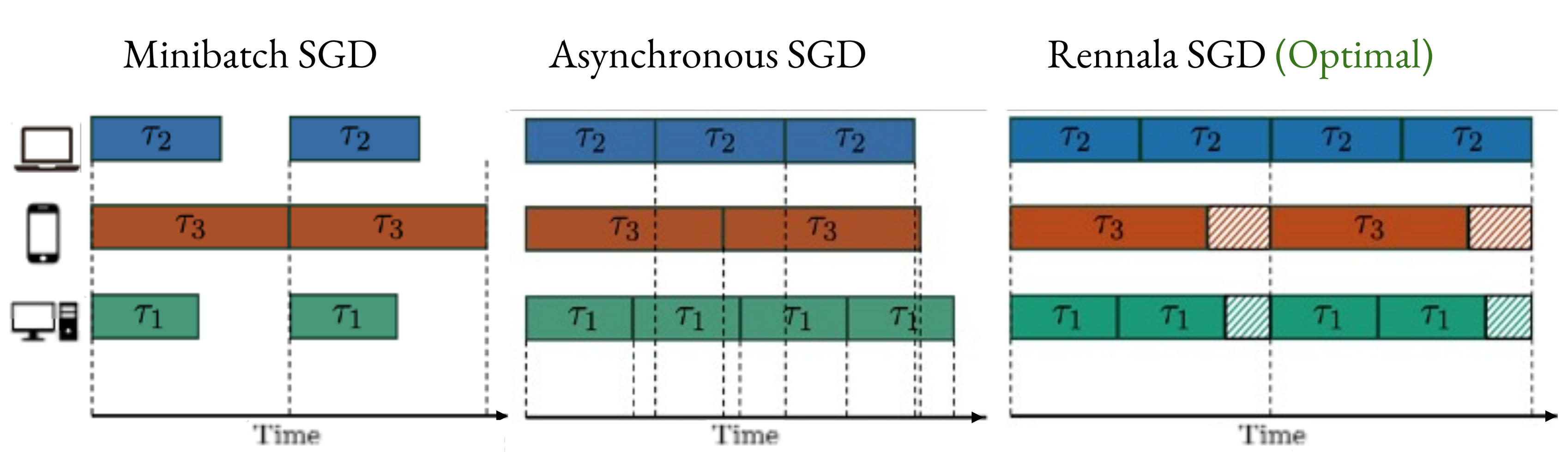
We assume we have access to n parallel workers that compute stochastic gradients independently




## From Deterministic Delays to Random Chaos

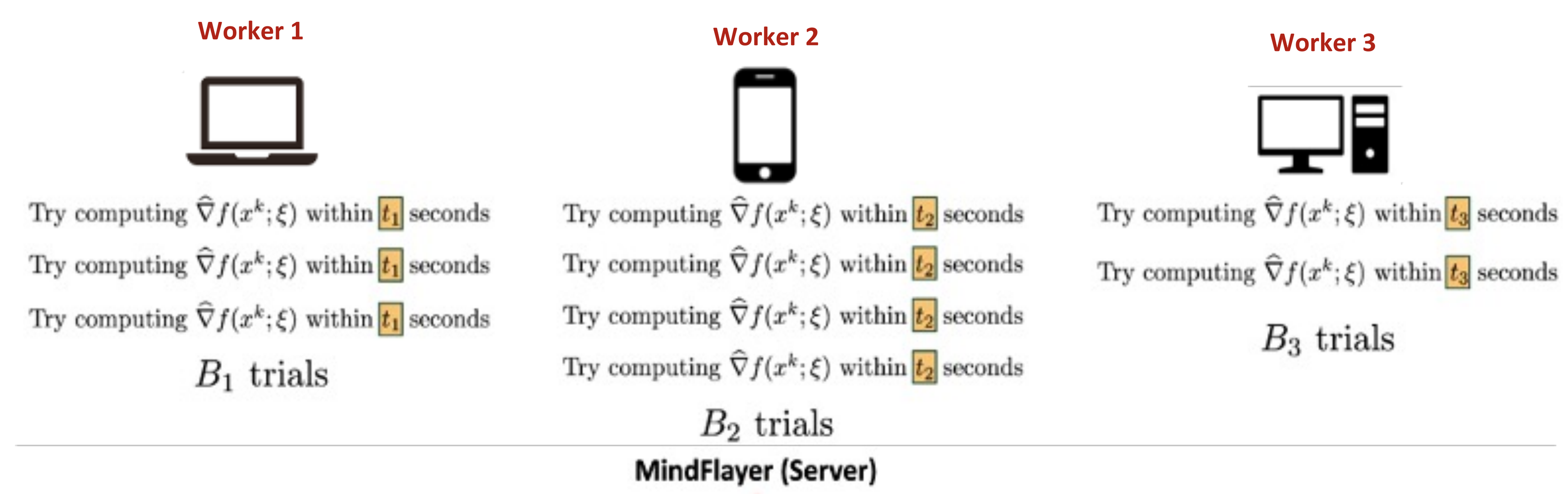
- **Minibatch SGD:** A classical method that processes gradients in parallel but is limited by the slowest worker.
- **Asynchronous SGD (ASGD):** Updates models as soon as any worker sends a gradient but suffers from delays caused by outdated gradients.
- **Rennala SGD [1]:**
  - Collects multiple gradients at the same model point.
  - Utilizes faster workers while ignoring slower ones.
  - Proven optimal for fixed compute times, no $\eta_i \sim \mathcal{J}_i$

**Challenge:** Rennala fails in realistic scenarios with random delays, necessitating a new approach for robust optimization.




Visualizing Client Utilization Across Minibatch SGD, ASGD, and Rennala SGD

## MindFlayer SGD




$$x^{k+1} = x^k - \gamma \frac{1}{B}\sum_{i=1}^n \sum_{j=1}^{B_i} I\left(\eta_i^j \le t_i\right) \nabla f\left(x^k;\xi_i^j\right)$$

$$B = \sum_{i=1}^n p_i B_i \text{ and } p_i = F_i(t_i) = P(\eta_i \le t_i)$$

**Algorithm 1** MindFlayer SGD

1: **Input:** starting point $x^0 \in \mathbb{R}^d$, stepsize $\gamma > 0$, allotted times $t_1, \dots, t_n \ge 0$, number of trials per client $B_1, \dots, B_n \ge 0$
2: **for** $k = 1, 2, \dots, K$ **do**
3: Put $g^k = 0$
4: Send $x^k$ to all clients
5: Run Method 2 in all clients $i = 1, 2, \dots, n$
6: **while** there is a client that has trials to perform **do**
7: Wait for the fastest client
8: Receive gradient $g$
9: $g^k = g^k + g$
10: **end while**
11: $g^k = \frac{g^k}{B}$, $\diamond$ $B = \sum_{i=1}^n p_i B_i$ and $p_i = F_i(t_i) = P(\eta_i \le t_i)$.
12: $x^{k+1} = x^k - \gamma g^k$
13: **end for**

**Algorithm 2** Client $i$-s $k$-th step

1: Receive $x^k$ from the server
2: **for** $j = 1, 2, \dots, B_i$ **do**
3: Sample $\eta_i^j \sim \mathcal{J}_i$
4: **if** $\eta_i^j \le t_i$ **then**
5: $g = \nabla f(x^k;\xi_i^j)$, $\xi_i^j \sim \mathcal{D}$
6: Send $g$ to the server
7: **end if**
8: **end for**

## Convergence and Time Complexity

**Assumptions.**

- Function $f$ is differentiable, and $L$-Lipschitz continuous: $\|\nabla f(x) - \nabla f(y)\| \le L\|x - y\|, \quad \forall x, y \in \mathbb{R}^d$
- There exists $f^{\inf} \in \mathbb{R}$ such that $f(x) \ge f^{\inf}$, for all $x \in \mathbb{R}^d$
- For all $x \in \mathbb{R}^d$, stochastic gradients $\nabla f(x;\xi)$ are unbiased and variance-bounded:

$$\mathbb{E}_\xi[\nabla f(x;\xi)] = \nabla f(x), \quad \mathbb{E}_\xi[\|\nabla f(x;\xi) - \nabla f(x)\|^2] \le \sigma^2$$

**Theorem.** With the above assumptions. Let $B = \sum_{i=1}^n p_i B_i$ and $\gamma = \frac{1}{2L}\min\left\{1, \frac{\epsilon B}{\sigma^2}\right\}$. Then, after

$$K \ge \max\left\{1, \frac{\sigma^2}{\epsilon B}, \frac{8L(f(x^0) - f^{\inf})}{\epsilon}\right\}$$

iterations, the method guarantees that $\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\|\nabla f(x^k)\|^2\right] \le \epsilon$

**Time Complexity.**

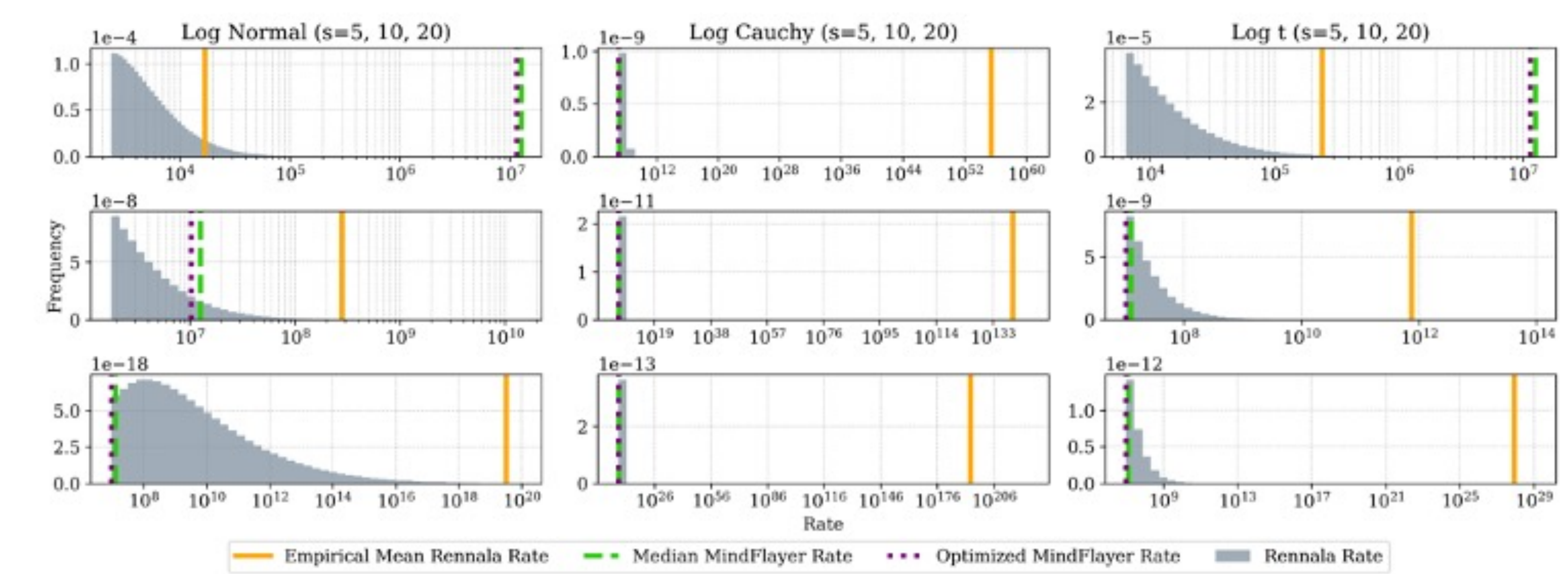
$$p_j = F_j(t_j) = P(\eta_j \le t_j) \qquad \Delta = f(x^0) - f^{\inf}$$

$$\min_{m\in[n]}\left\{\left(\frac{1}{m}\sum_{j=1}^m \frac{p_j}{\tau_j + t_j}\right)^{-1}\left(\frac{S}{m} + \frac{1}{m}\sum_{j=1}^m p_j\right)\frac{\Delta L}{\varepsilon}\right\}$$

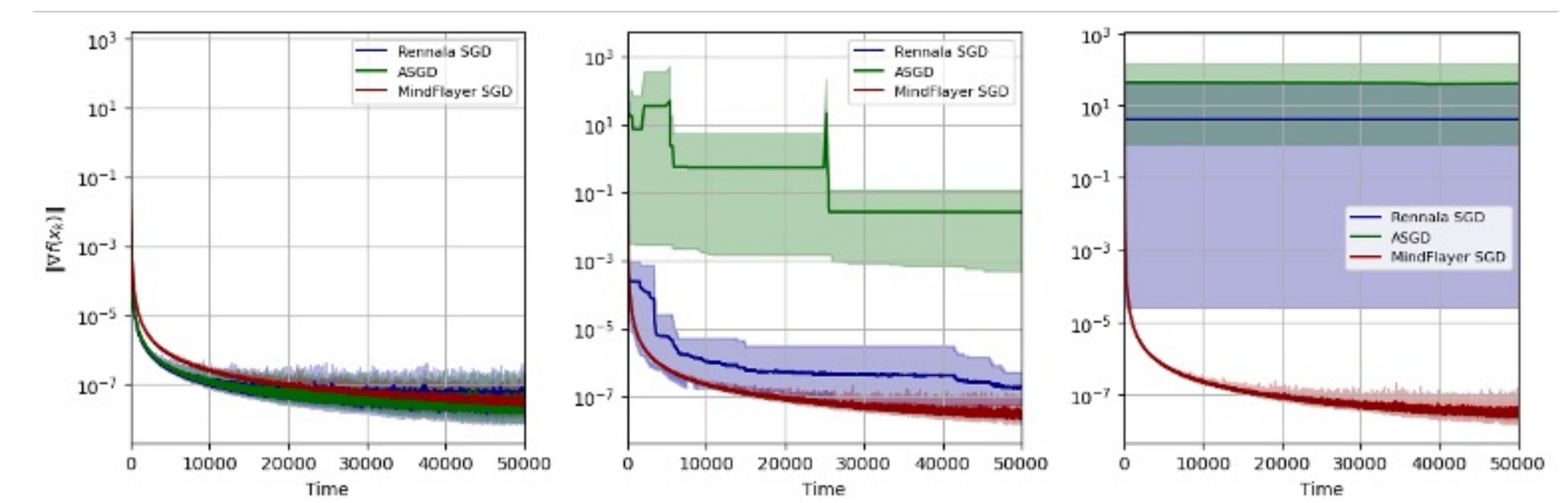
$$S = \max\left\{\frac{\sigma^2}{\varepsilon}, 1\right\}$$

## MindFlayer vs. Rennala: Theory and Experiments

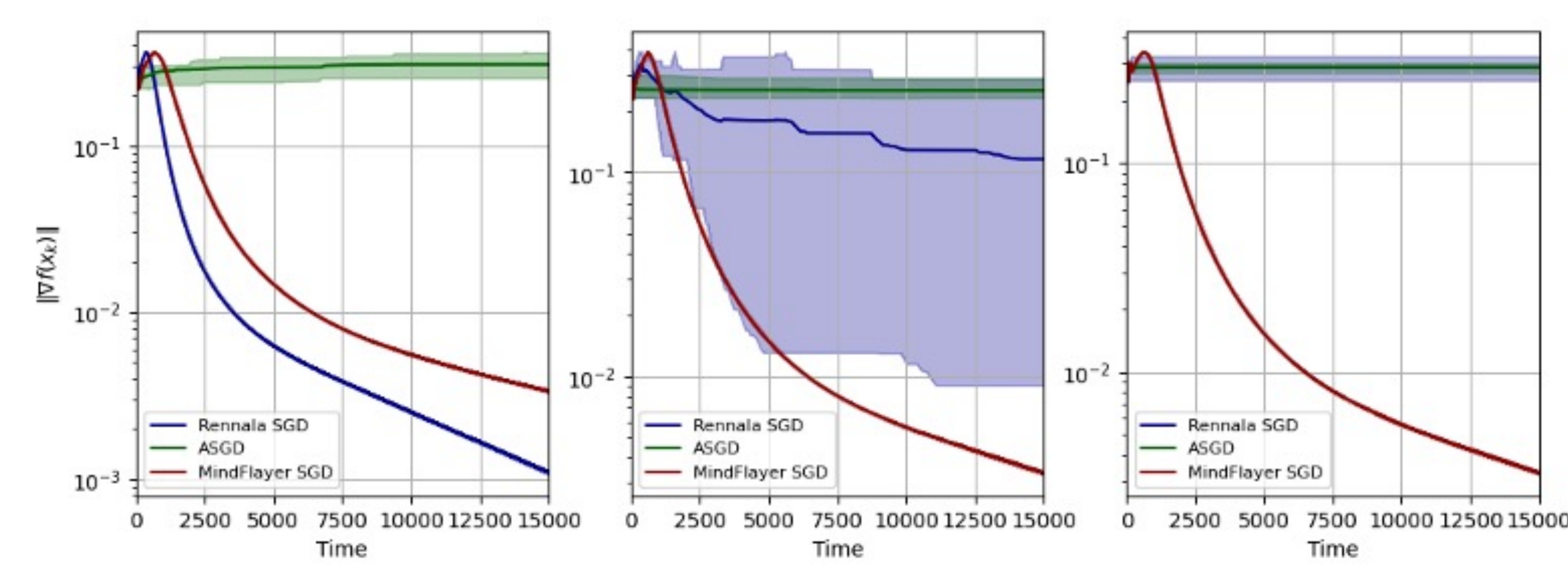
In this setting, Rennala's time complexity is a random variable depending on the random variable $T_B$ representing the time to collect a batch of size B




Time complexity comparison across various distributions. MindFlayer SGD consistently outperforms Rennala SGD under heavy-tailed distributions, with $t_i$ selected as the median or optimized via L-BFGS-B [2].




Performance of MindFlayer SGD, Rennala SGD, and ASGD on a quadratic problem (n=5 clients, $B_i$ set from Theorem). MindFlayer SGD demonstrates robustness under increasing variance (s=1,10,100), while Rennala SGD and ASGD degrade significantly.




Performance of MindFlayer SGD, Rennala SGD, and ASGD on a two-layer neural network with Log-Cauchy-distributed delays (s=1, 10, 100). MindFlayer SGD maintains consistent convergence across scales, while Rennala SGD and ASGD struggle under larger scale values.

### References

[1] Alexander Tyurin and Peter Richtárik. Optimal time complexities of parallel stochastic optimization methods under a fixed computation model. Advances in Neural Information Processing Systems, 36, 2024.

[2] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software (TOMS), 23(4):550–560, 1997.