Parallel GSSA

Yongyang Cai, Hoover Institution

July 31, 2013

Outline

- Review of GSSA
- Analysis before parallelization
  - computational complexity
  - memory requirement
- Parallelization
  - in integration step
  - in fitting step
- Parallelization results

Model for Illustration

- N-country optimization problem

$$\max_{\left\{c_t^h, k_{t+1}^h\right\}_{t=0,\ldots,\infty}^{h=1,\ldots,N}} E_0 \sum_{h=1}^{N} \lambda^h \left[\sum_{t=0}^{\infty} \beta^t u^h\left(c_t^h\right)\right] \tag{1}$$

  - subject to the aggregate resource constraint,

$$\sum_{h=1}^{N} c_t^h + \sum_{h=1}^{N} k_{t+1}^h = \sum_{h=1}^{N} k_t^h (1-\delta) + \sum_{h=1}^{N} \theta_t^h A f^h\left(k_t^h\right), \tag{2}$$

  - the countries' productivity levels,

$$\ln \theta_{t+1}^h = \rho \ln \theta_t^h + \epsilon_t^h, \qquad h = 1, ..., N, \tag{3}$$

    where ϵ_t^1, ..., ϵ_t^N are correllated normal random variables

Euler Equations

- Euler Equations:

$$u_c^h\left(c_t^h\right) = E_t\left\{\beta u_c^h\left(c_{t+1}^h\right)\left[1-\delta+\theta_{t+1}^h A f_k^h\left(k_{t+1}^h\right)\right]\right\}, \quad h=1,...,N, \tag{4}$$

  - this implies

$$k_{t+1} = E\left\{F\left(\theta_{t+1}\right) \mid \left(k_t,\theta_t\right)\right\} \approx \widehat{K}\left(k_t,\theta_t;b\right), \tag{5}$$

  where

$$F\left(\theta_{t+1}\right) \equiv \beta \frac{u_c\left(c_{t+1}\right)}{u_c\left(c_t\right)}\left[1-\delta+\theta_{t+1} A f_k\left(k_{t+1}\right)\right] k_{t+1}$$

- Randomness: θ_{t+1} , and c_{t+1} when it is dependent on θ_{t+1} if we do not use the one-node MC method
- KKT conditions also implies

$$u_c^h\left(c_t^h\right)\lambda^h = \mu_t \tag{6}$$

for any $h=1,...,N$, where μ_t is the Lagrange multiplier of the aggregate resource constraint at time t.

GSSA Algorithm

- *Initialization:*
  - Given the initial state (k_0, θ_0). Simulate a path of θ_t for $t = 1, ..., T-1$, denoted by $\{\vartheta_t\}_{t=0,...,T-1}$
  - Choose a functional form $\widehat{K}\left(x_t; b^h\right)$, where $x_t = (k_t, \vartheta_t)$.
  - Choose an initial guess on the coefficients vectors $b^1, ..., b^N$.
- Step 1. Simulation step. Use $k_{t+1}^h = \widehat{K}\left(x_t; b^h\right)$ to generate $\{k_t\}_{t=0,...,T}$
- Step 2. Integration step.
  - Compute the intratemporal choice c_t and c_{t+1}
  - Compute $y_t = E\left\{F\left(\theta_{t+1}\right) \mid x_t\right\}$
- Step 3. Fitting step. Compute $\widehat{b}^h$ such that $\widehat{K}\left(x_t; b^h\right)$ approximates $\left(x_t, y_t^h\right)$, for each country h
  - update b^h: $b^h = (1-\xi)\, b^h + \xi \widehat{b}^h$

Approximation

- Approximation: Choose a functional form

$$k_{t+1}^h = \widehat{K}\left(x_t; b^h\right),$$

for each country h.

  - basis approximation:

$$k_{t+1}^h = \sum_{i=1}^{m} b_i^h \psi_i(x_t),$$

  - degree-n complete polynomial approximation:

$$k_{t+1}^h = \sum_{|\alpha| \leq n} b_\alpha^h x_t^\alpha,$$

- Minimization of approximation errors

$$\widehat{b}^h = \arg\min_{b^h} \sum_{t=1}^{T} \left\| y_t^h - \widehat{K}\left(x_t; b^h\right) \right\|.$$

Integration

- Quadrature with J nodes

$$y_t = E_t \left\{ F\left(\theta_{t+1}\right) \mid x_t \right\} \approx \sum_{j=1}^{J} \omega_j F\left(\theta_{t+1,j}\right)$$

- To compute c_t,

$$\sum_{h=1}^{N} c_t^h + \sum_{h=1}^{N} k_{t+1}^h = \sum_{h=1}^{N} k_t^h \left(1-\delta\right) + \sum_{h=1}^{N} \vartheta_t^h A f^h\left(k_t^h\right),$$

$$u_c^h\left(c_t^h\right) \lambda^h = \mu_t, \quad h = 1, ..., N$$

  - $N+1$ unknown variables, $N+1$ equations
  - In general, we may need an equation solver to find the solution of the system
  - For this specific example, it exists an explicit formula

Integration

- Similar for computing $c_{t+1,j}$ for $j = 1, ..., J$:

$$\sum_{h=1}^{N} c_{t+1,j}^{h} + \sum_{h=1}^{N} k_{t+2,j}^{h} = \sum_{h=1}^{N} k_{t+1}^{h} (1 - \delta) + \sum_{h=1}^{N} \theta_{t+1,j}^{h} A f^{h} \left(k_{t+1}^{h} \right),$$

$$u_{c}^{h} \left(c_{t+1,j}^{h} \right) \lambda^{h} = \mu_{t+1,j}, \quad h = 1, ..., N$$

where $k_{t+2,j}^{h} = \widehat{K} \left(k_{t+1}, \theta_{t+1,j}; b^{h} \right)$

RLS fitting

- Regularized least square fitting (RLS-Tikhonov):

$$\widehat{b}^h = \left(X'X + \eta I\right)^{-1} X' y^h$$

where

$$X = \begin{pmatrix} \psi_1(x_1) & \cdots & \psi_m(x_1) \\ \vdots & \ddots & \vdots \\ \psi_1(x_T) & \cdots & \psi_m(x_T) \end{pmatrix}, \quad y^h = \begin{pmatrix} y_1^h \\ \vdots \\ y_T^h \end{pmatrix}$$

Computational Complexity

- Before designing a parallel algorithm, do a complexity analysis at first
- No need to think about parallelization on fast steps
- Simulation step: Use $k_{t+1}^h = \widehat{K}\left(x_t; b^h\right)$ for $h = 1, ..., N$ to generate $\{k_t\}_{t=0,...,T}$.
  - Assume that we use degree-n complete polynomials.
  - Number of coefficients of degree-n complete polynomials for N countries: $m = \begin{pmatrix} 2N + n \\ n \end{pmatrix}$
  - computational cost: $O(TNm)$
    - When $N = 100$, $n = 2$, $T = 30,000$, it is 61 Gflops

Cost of Integration Step

Integration step:

- Compute the intratemporal choice c_t and $c_{t+1,j}$ for $j = 1, ..., J$
  - fast for the illustration model, computational cost: $O(N(J+1))$
  - may be time-consuming for a general model using an equation solver
  - $f(N)$: cost of computing c_t
- Compute $y_t = \sum_{j=1}^{J} \omega_j F\left(\theta_{t+1,j}\right)$
  - computational cost: $O(f(N)(J+1)MN)$
  - M: the cost for computing one $F^h\left(\theta_{t+1,j}\right)$ for given c_t, $c_{t+1,j}$ and $\theta_{t+1,j}$
  - For the illustration model using one Gaussian quadrature node for integration ($J = 1$): it is very fast.

Cost of Fitting Step

Fitting step: compute $\widehat{b}^h = (X'X + \eta I)^{-1} X'y^h$ for $h = 1, ..., N$

- Compute $A = X'X$, computational cost: $O\left(m^2 T\right)$: When $N = 100$, $n = 2$, $T = 30,000$, it is 8 Teraflops.
- Naive way:
  - compute $B = (A + \eta I)^{-1}$, computational cost: $O\left(m^3\right)$
  - compute $C = BX'$, computational cost: $O\left(m^2 T\right)$
  - compute Cy^h for $h = 1, ..., N$, computational cost: $O\left(mTN\right)$
- Less-computational-cost way:
  - Compute $z^h = X'y^h$ for $h = 1, ..., N$, computational cost: $O(mTN)$
  - Cholsky factorization: $A + \eta I = R'R$, computational cost: $O\left(m^3\right)$
    - but much faster than computation of $(A + \eta I)^{-1}$
    - e.g., in Matlab, 14 seconds to compute inverse of a positive definite matrix, but only 1 second for computing its cholesky factorization.
  - Solve $R'\widehat{z}^h = z^h$ for $h = 1, ..., N$, computational cost: $O\left(m^2 N\right)$
  - Solve $R\widehat{b}^h = \widehat{z}^h$ for $h = 1, ..., N$, computational cost: $O\left(m^2 N\right)$
- $N << m$ for nonlinear polynomial approximation, and $m < T$

Large Memory is Required

- To store X, it needs $8Tm$ bytes
- To store $A = X'X$, it needs $8m^2$ bytes
- If we use quadratic approximation,

N	T	m	Store X	Store A
50	10,000	5,151	0.4 GB	0.2 GB
80	20,000	13,041	2.1 GB	1.4 GB
100	30,000	20301	4.9 GB	3.3 GB
200	100,000	80601	64 GB	52 GB

- My laptop: 8 GB
- One beagle node: Shared memory 32 GB (users can use up to about 31 GB)

Parallelization in Integration Step

- For each $t = 1, ..., T$,
  - compute the intratemporal choice c_t and $c_{t+1,j}$ for $j = 1, ..., J$, and then $y_t = \sum_{j=1}^{J} \omega_j F(\theta_{t+1,j})$
- Parallelization across t
- Parallelization across computation of c_t and $c_{t+1,j}$ for $j = 1, ..., J$

Parallelization in Fitting Step

- Compute $A = X'X$ in parallel
  - One example of naive parallelization: compute $X'X_1$, ..., $X'X_m$ in parallel
- Compute Cholesky factorization of $A + \eta I = R'R$ in parallel
- Compute $X'y^h$ for $h = 1, ..., N$ in parallel

Parallelization in Simulation Step

- Use $k_{t+1}^h = \widehat{K}\left(x_t; b^h\right)$ for $h = 1, ..., N$ to generate $\{k_t\}_{t=0,...,T}$.
- Parallelization across h

More Parallelization

- The most time-consuming part of fitting step:
  - compute $A = X'X$
  - the Cholesky factorization $A + \eta I = R'R$
- Independent between
  - computing A and the Cholesky factorization in the fitting step
  - computing y^h for $h = 1, ..., N$ in the integration step
- Parallelization:
  - One cluster computes A and the Cholesky factorization in parallel, another computes y^h for $h = 1, ..., N$ in parallel
  - If y^h is very time-consuming to be computed even in parallel, then the naive way that computes $C = (A + \eta I)^{-1} X'$ explicitly may be good
  - If computation of A and the Cholesky factorization is more time-consuming than computation of y^h, then the another cluster keeps computing $X'y^h$ for $h = 1, ..., N$ in parallel

Parallelization Results

- Use OpenMP for parallelization over one beagle node (a 24-core computer)
- Solve the illustration model using one-node Gaussian quadrature method
- Use quadratic polynomial approximation, one GSSA iteration

N	T	one beagle core	one beagle node
50	10,000	99 seconds	12 seconds
80	20,000	21 minutes	2 minutes
100	30,000	73 minutes	6 minutes

Parallelization Results

- Use linear and quadratic polynomial approximation, iterates until convergence

N	T	n	one beagle core	one beagle node
50	10,000	1	77 seconds	20 seconds
		2	26 minutes	3 minutes
80	20,000	1	57 seconds	17 seconds
		2	-	1.5 hours
100	30,000	1	101 seconds	31 seconds
		2	-	5.8 hours

Under Development

- Combine MPI and OpenMP
- Solve a general model using an equation solver
- Parallelization to overcome the memory limit