Parallelizable Recursive Problems in Economics and Finance

Econon 899 Computational Economics, Fall 2018
University of Wisconsin-Madison

September 14, 2018

Introduction

- Recursive problems are one of the most, if not the most, widely used modelling technique in dynamic economics.
- Due to the fixed point nature of this type of problems, they can be solved in a computer with simple iterative algorithms. (e.g. value function iterations).
- However, large scale recursive models (i.e. many state variables) are subject to the *curse of dimensionality*.
- That is, models with multiple state variables may take a LONG time to solve.
- Fortunately, because of the recursive nature of this type of model, we may speed up solving them by *parallelization*.

A Simple Example

- Consider this simple recursive problem of a household

$$V(a) = \max_{c,a'} \; u(y + a(1+r) - a') + \beta V(a')$$

s.to

$$a' \geq 0$$

- a is savings, y is income, r is the exogenous interest rate
- To make things really simple, we assume away uncertainty of income
- The only state variable then is a

Solution Concept

- Within each iteration of the value function, we have to solve the maximization problem on the right hand side of the above Bellman equation
- Given a value function V, we solve the maximization problem by seeking
  1. A policy function for saving $a'^{*} = g(a)$
- Note that both functions are functions of the state variable and only the state variable.
- This means that we can solve for them at each value of the state a independently.
- This nice feature makes the whole problem highly parallelizable.

Putting the Problem in a Computer

- Since a is continuous state variable, the first thing we do is to create discrete grid for it
- Suppose that we choose to create a evenly space grid of a of size N
- That is $\{a_1, a_2, a_3, ..., a_N\}$.
- We then solve the maximization problem at each point of the grid.
- The following pseudo-code illustrates a basic algorithm.

Pseudo-code

- Let i denote the index of the grid points.
- For each i we solve the maximization problem as follows.

$$g(a_i) = arg \max_{a'} \; u(y + a_i(1 + r) - a') + \beta V(a')$$

s.to

$$a' \geq 0$$

- Note that to solve the above problem at each i, the only thing we need to condition on is a_i.
- Since we we have N points in the grid of the state, we need to solve the above, independently at each i, N times.

Sequential vs. Parallel Computing

- We may easily write a loop in the computer to solve the maximization problem at each i one after one (i.e. sequentially).
- However, when N is huge this way might be slow.
- Fortunately, with multi-core CPUs, we may *parallelize* the problem.
- That is, we divide our state grid into smaller segments and let each core of our CPU work on one segment independently.
- The following pseudo-code illustrates a parallel algorithm.

Pseudo-code—Parallel

- Suppose that $N = 10000$; and suppose further that there are 4 cores in our CPU.
- Let's first divide the original state grid into four smaller sets, each of which has 2500 points.
- Simultaneously and separately, we tell the 4 cores to do the following
  1. let core 1 solve the problem for set 1 (i.e. $i = 1$ through 2500)
  2. let core 2 solve the problem for set 2 (i.e. $i = 2501$ through 5000)
  3. let core 3 solve the problem for set 3 (i.e.$i = 5001$ through 7500)
  4. let core 4 solve the problem for set 4 (i.e. $i = 7501$ through 10000)
- This way our 4 cores work simultaneously and each of them works on a smaller segment
- Computational time is significantly reduced

Generalization

- In general, the maximization step in a recursive problem consumes most of the computational time.
- Parallel computing exploits the feature that the maximization problem can be solved at each state independently.
- This greatly reduces the computational time of the most timing consuming part of the whole problem.
- Note that recursive problems with multiple states can be parallelized too, using the same idea.
- In those cases we just need to collapse all state variables into one big multidimensional grid. Then solve the maximization problem at each point of that grid.

Fortran Compilers

- There are two popular Fortran/C++ compilers
  1. Open source (free): gfortran. Download it here ▸ Link
  2. Commercial (free for students): Intel Fortran (ifort)
- Command to compile via gfortran: gfortran myps2.f90 -o solution.exe
- Command to compile via gfortran w/ OpenMP: gfortran myps2.f90 -o solution.exe -fopenmp
- Command to compile via Intel Fortran: ifort myps2.f90 -o solution.exe
- Command to compile via Intel Fortran w/ OpenMP: ifort myps2.f90 -o solution.exe /Qopenmp