OpenMP parallelization of agent-based models

Filippo Castiglione and Massimo Bernaschi
*Istituto Applicazioni del Calcolo (IAC) “M. Picone”, CNR,
Roma, Italy

Federico Massaioli
*CASPUR, Interuniversity Consortium for SuperComputing,
Roma, Italy
Many complex systems are composed by many heterogeneous elements which interact with each other.

A possible approach for studying such systems is Agent-Based Simulation.

An Agent is an entity which has:

- an internal data representation (memory or state)
- means for modifying its internal data representation (perception)
- means for modifying its environment (behaviour)
The complex *perception/behaviour* of the entities corresponds to precise *state-changes* upon interaction.

Every single entity can be thought of as a **Stochastic Finite State Machine (SFSM)** which processes information and changes its state according to the result of the interactions with other entities, or with the environment.

There is a very limited number of floating point operations in an Agent-Based simulation. Most of the data structures are a combination of **integers** and **pointers**.
Application I: Immunology

The whole automaton corresponds to a single lymphnode

- the space is defined as an hexagonal 2D lattice
- the dynamics is probabilistic
- the evolution of each site depends just on the site itself (*internal dynamics*), but:
  - entities move from site to site (*diffusion process*)

Entities on the lattice are the major classes of cells of the *Lymphoid* lineage (TH, CTL, B, PLB) and some of the *Myeloid* lineage (MA, DC)
Application II: Financial Markets

Three kinds of agents, trading with different strategies for a set of N assets (stocks, commodities, etc.)

*Fundamentalists* consider a reference (or “fundamental”) value to determine the “right” price of an asset

*Noisy* traders do not follow any reference value and do not look at charts. Their behaviour is mostly random. There is a very high number of noisy traders but most of them have a very limited capital at their disposal

*Technical traders* also called *chartists* represent those agents who take into account information about the evolution of the price (in our case the moving average over certain time horizons)

An agent is not a broker: broker = collection of agents
Trading Strategies

Fundamentalists
buy if $p_t \leq f_t$
otherwise sell

Noisy
buy and sell randomly (probability 1/2)

Chartists
sell if $MA_t(h) > p^+$
buy if $MA_t(h) < p^-$
otherwise stay inactive
A generic agent data structure

typedef struct tagAgentblock {
  int aclass; /* agents type identifier */
  int atid; /* type of moving average */
  int strategy[MAX_NUM_ASSETS]; /* buy-sell decision */
  int nstocks[MAX_NUM_ASSETS]; /* num. of possessed stocks */
  int ordertime[MAX_NUM_ASSETS]; /* time an order may be outstanding 
                             >0 means limit order 
                             <0 means stop order */
  double orderprice[MAX_NUM_ASSETS]; /* orderprice: 
                                    ordertime>0 means limit price, 
                                    ordertime<0 means stop price, 
                                    no meaning if ordertime==0 */
  double money; /* liquidity: available money 
                  not blocked in any order */
  double iwealth; /* initial wealth */
  double invested[MAX_NUM_ASSETS]; /* money invested in stocks */
  double activity; /* probability to trade */
  void (*policy[NSTRATEGIES])(); /* strategy function */
  struct tagAgentblock *next;
} AGENTBLOCK;
The Book of Orders!!

The “interaction” on each asset is regulated by a book-of-orders defined by lists of buy-orders and sell-orders

<table>
<thead>
<tr>
<th>BUY ORDERS</th>
<th>SELL ORDERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>time</strong></td>
<td><strong>price</strong></td>
</tr>
<tr>
<td>21005</td>
<td>11122</td>
</tr>
<tr>
<td>25008</td>
<td>11121</td>
</tr>
<tr>
<td>24506</td>
<td>11118</td>
</tr>
<tr>
<td>19002</td>
<td>11108</td>
</tr>
<tr>
<td>20148</td>
<td>11106</td>
</tr>
</tbody>
</table>

Orders are stored according to type (buy or sell), price (descending and ascending) and time, operating on singly linked lists

A transaction occurs whenever the cheapest price among the sell list matches the most expensive offer in the other list

Matching orders are immediately satisfied (filed) whereas the rest wait for the arrival of a matching order for a time defined in the order itself.
The simulation

Repeat forever
   Update fundamentalists’ models [stochastic]
   While no price changes
      Pick another agent [order is not important, fairness is!]
      Have it take a decision
      If not inactive
         Decide what to trade, and how much
         Decide the order type [market or limit order?]
         Check if it has enough resources to buy/sell
         Post the order in the BO [possible contentions!!]
         If order can be fulfilled
            Update involved agents [possible contentions!!]
            Possibly update price [possible contentions!!]
         Save info, compute moving averages, …
   …and orders could expire so check now and then [to contend or to serialize?]
Relevant issues

Distributing agents among threads
- Probable contentions for Book of Orders
- Dynamic lists of orders
- Agent competition has to be fair
- Price updates while agents are trading
- Periodic removal of expired orders: possible races

Distributing Book of Orders among threads
- Probable contentions for agents’ money boxes
- False sharing on agents’ assets
- Load balancing (stocks can become inactive for a while)

Usual (and boring) stuff
- RNGs
- mallocs
Questions?

Comments?

Suggestions?

Ideas?
Application II: Financial Markets

Asset price

\[ L^2 = 400, \ N = 18000 \]

Moving averages