Large-scale agent-based models perspectives and requirements

Filippo Castiglione

Istituto Applicazioni del Calcolo (IAC) "M. Picone",

National Research Council (CNR), Roma, Italy

IMA "Hot Topics" Workshop:

Agent Based Modeling and Simulation

Minneapolis, USA

November 3-6, 2003

Overview

I. From spin models to ABM

II. Optimization requirements

- Immune System Simulation
- Stock Market Simulation

Large scale simulations using state-of-art HPC techniques and parallel processing have been "limited" to classic applications like fluid-dynamics, materials science or meteorology.

Floating point simulations:

(i) Discretize the model-eqs;
(ii) Do computer simulation;
(iii) Double check w. r. t. continuous eqs.

Efficient algorithms, optimized codes and compilers

ODE, PDE describe well the macroscopic properties

but

- Difficult to explain the origin (micro)
- Don't handle well discontinous systems
- Don't handle well heterogenity in the population

Complex systems (in biology or finance) are composed by many heterogeneous elements which interact each other

The rules governing the micro-behaviour of the entities are mostly unknown

What is available is the behaviour at macro level (empirical observations, clinical data, financial records)

→ Bottom-Up simulations (it's a computational paradigm)

Ising-like models

Ernst Ising (1925):

- model for magnetization (also for liquid-gas transition)
- spin up (+1) or down (-1)

John von Neumann (1966):

- self-reproducible automata

John Conway (Gardner, 1970):

- Game of LIFE Cellular Automata
- cells dead (0) or alive (1)
- model for excitable medium (Belousov-Zhabotinskii reaction, Greenberg-Hasting model)

Frish, Hasslacher, Pomeau (1986):

- model for fluid dynamics
- 2D Lattice Gas (FHP): 6 velocities

	Ising-like models	Agent Based Models
Discrete lattice of cells	One, two, three dimensions (also more)	Yes/No?
Internal state	Simple representation (0,1, …, N, N small)	Complex representation (many states from an enumerable set and even more)
Heterogenity	Usually not	Yes
Interactions	Usually with the neighbourhood (local interactions)	Long-range interaction
Synchronism	Yes (parallel update dynamics)	No. Asyncrony comes from eligible states in the state-transition rule.

Ising-models are statistical mechanics models, i.e.

- very simple interactions
- very large number of microscopic entities

Optimized codes for Ising-like models (e.g. multispin coding, Swendsen and Wang's cluster update, …)

There is no standard for ABM high-performance simulations

→

- Serialize computation/simulation
- Ad-hoc solutions

Lymphoid System

(organs of the Immune System)

Primary = development

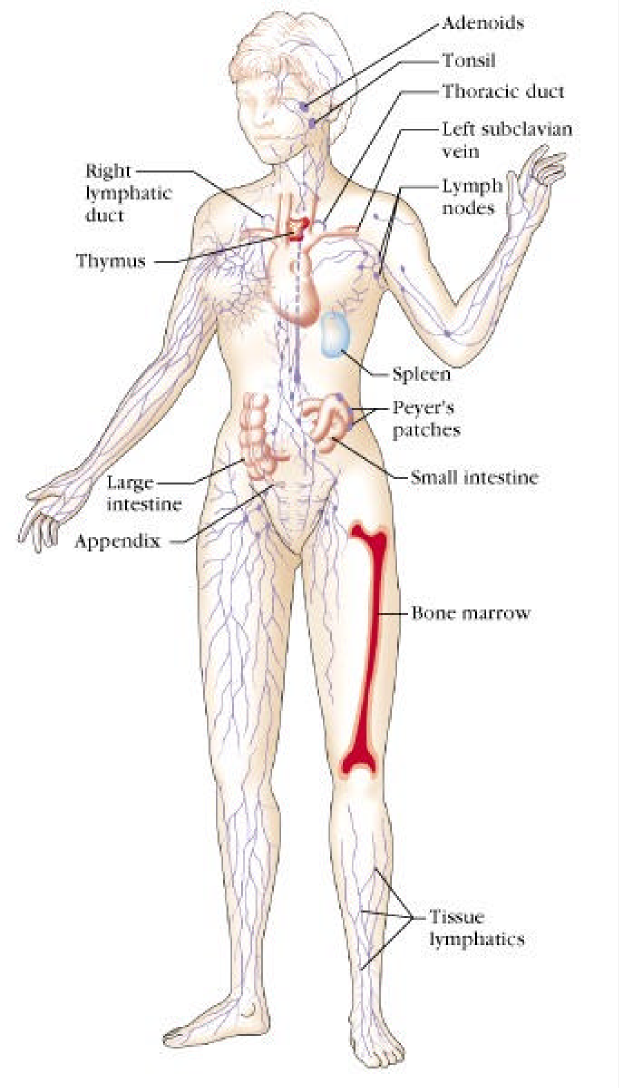
Bone Marrow
Thymus
Lymph Nodes
Spleen

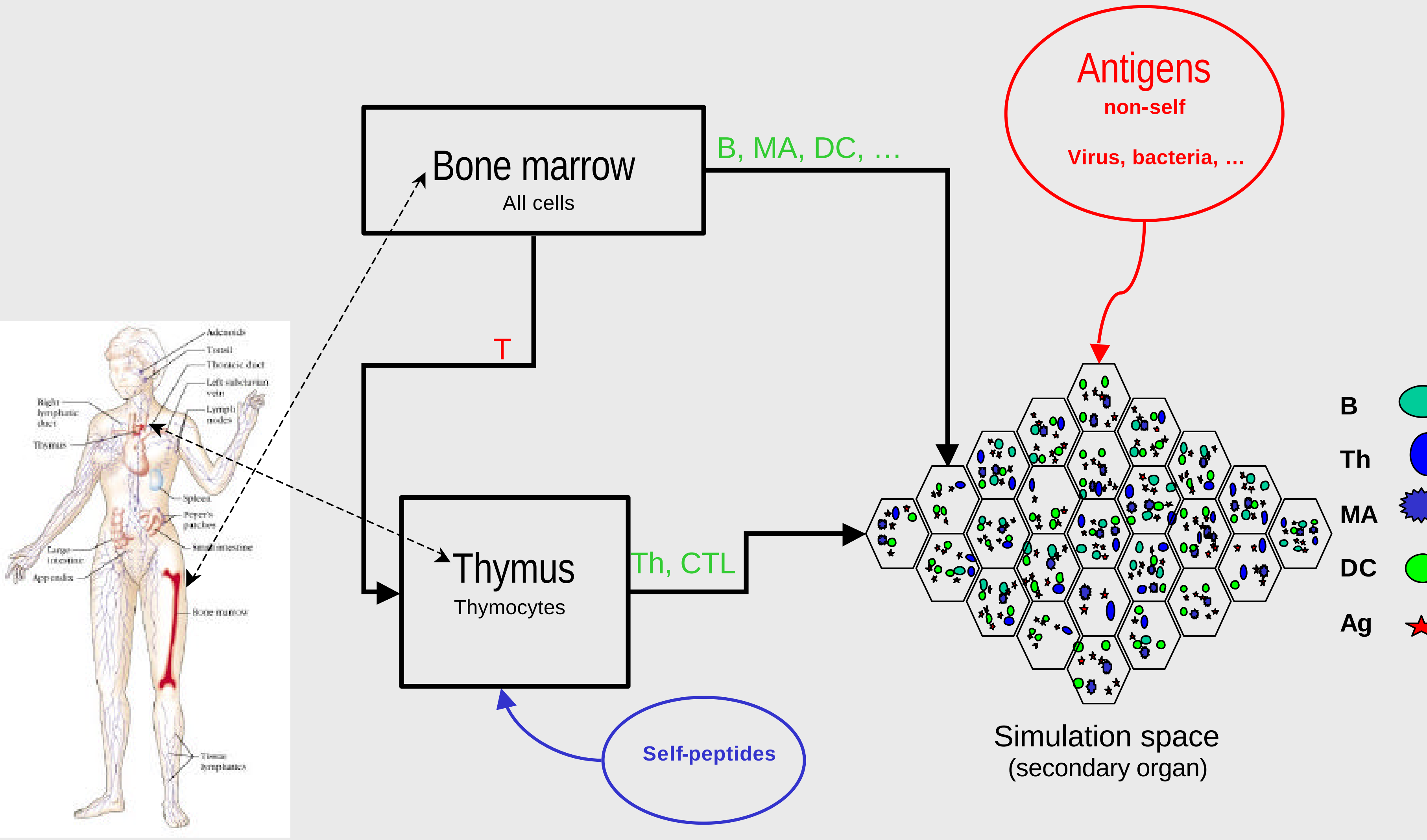
Secondary = Ag encounter

Adenoids
Tonsils
Peyer's Patch
Appendix

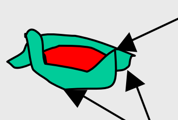
Tertiary = effectors

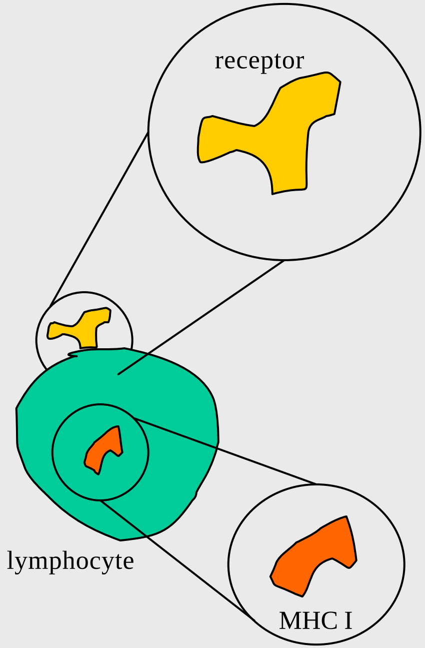
Intestine, skin, etc.




Antigens
non-self
Virus, bacteria, …
Bone marrow
All cells
B, MA, DC, …
T
Thymus
Thymocytes
Th, CTL
Self-peptides
B
Th
MA
DC
Ag
Simulation space
(secondary organ)
Adenoids
Tonsil
Thoracic duct
Left subclavian vein
Lymph nodes
Right lymphatic duct
Thymus
Spleen
Peyer's patches
Small intestine
Large intestine
Appendix
Bone marrow
Tissue lymphatics

- Molecules are represented by binary strings (BCR, Igs, TCR, antigens, immunocomplexes)





True potential repertoire

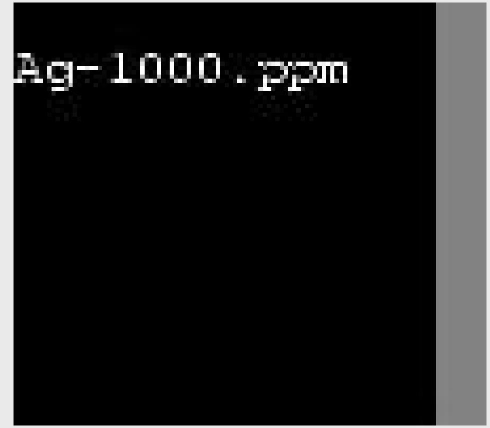
- BRC 10^{11} ($\sim 2^{36}$)
- TCR 10^{16} ($\sim 2^{53}$)

- $l = 24$ is the maximum bit-string length reached up to date

$$\longrightarrow 2^{24} \cong 1.6 \cdot 10^{7}$$

- Cellular agents are not striving for an overall goal
- Immune system goals:
  - ✓ recognition
  - ✓ response
  - ✓ memory
- No centralized control

Viral infection

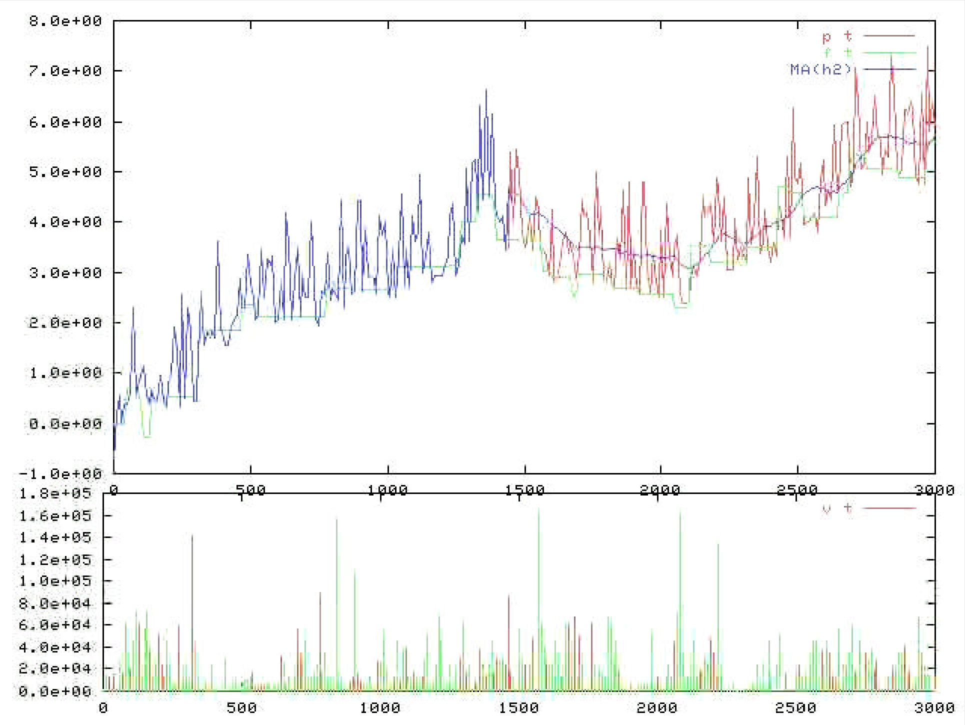


The model is useful for:

- What-if scenarious
- Virtual experiments, i.e. optimize protocols for
  - ✓ antiretroviral HIV threrapy (HART)
  - ✓ tumor vaccines

Financial Markets

Agents trading with different strategies for a set of N assets



Fundamentalists:

consider a "right" price of an asset

Noisy:

behave randomly

Technical traders:

look at charts

At each time step the agents decide whether to trade or to stay inactive

- Active agents follow different decision paths (depending on their trading strategy)
- may take different positions with respect to each stock in the market.

- Traders are displaced on a 2D-lattice (space represents the social / communication network)

- They diffuse to next-neighbors sites at each time step spreading their preference

- Traders on the same lattice belong to the same "group" of investors and get influenced by components of the same group

- Leadership: Few agents have a greater influence than others

Book of Orders (the "interaction")

- Traders choose randomly the kind of order for a random subset of assets
  - Market orders
  - Limit Orders
- Orders are matched in the book of orders (one for each asset)

BUY ORDERS		SELL ORDERS	
SHARES	PRICE	SHARES	PRICE
7,900	26.2400	1	27.0000
100	26.1200	100	27.0000
100	26.1200	100	27.0000
200	26.0100	2,000	27.0500
1,100	26.0100	1,000	27.0600
100	26.0100	100	27.0700
100	25.9700	100	28.9500
100	24.0000	200	29.9900

BUY ORDERS					SELL ORDERS			
time	trader	shares	price		price	shares	trader	time
21005	240	4	11122		11123	4	576	19802
25008	207	70	11121		11124	4	876	14706
24506	647	3	11118		11125	2	806	12150
19002	820	2	11108		11130	49	201	17203
20148	100	12	11106		11130	4	792	20101

Orders are stored according to type (buy or sell), price and time

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

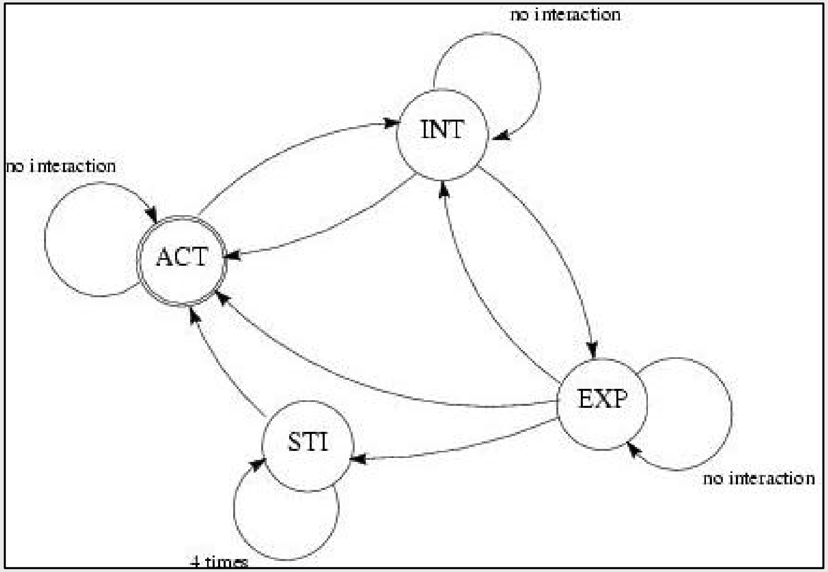
What these models have in common?

The architecture

The complex perception/behaviour of the entities corresponds to precise *state-changes* upon interaction.

Every agent can be represented 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 external fields.

There is a very limited number of floating point operations. Most of the data structures are a combination of integers and pointers (i.e., the *numerical stability* is guaranteed)



Agent's trading strategies

Fundamentalists

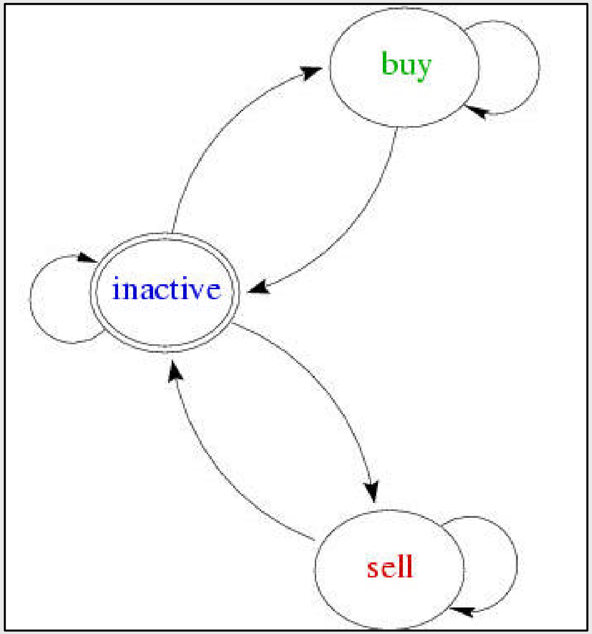
buy if $p_t = f_t$
otherwise sell

Noisy

buy randomly (probability 1/2)

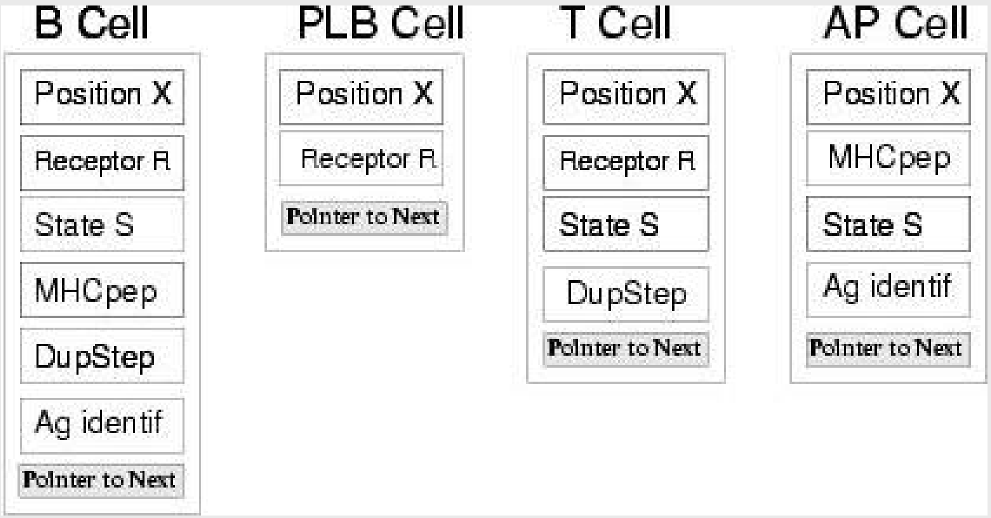
Chartists

sell if $MA_t(h) > p^+$
buy if $MA_t(h) < p^-$
otherwise do nothing



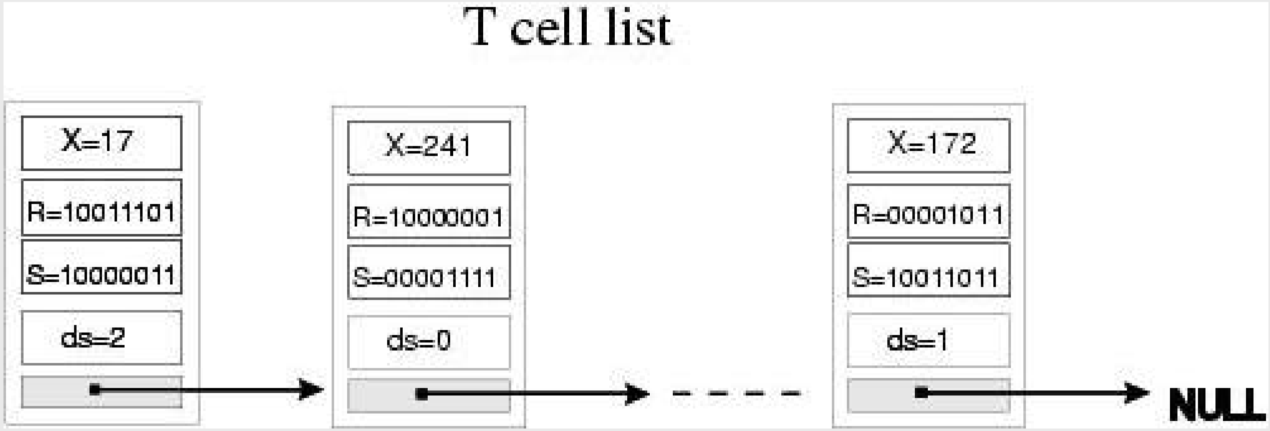
Dynamic memory allocation

agents (cells, molecules, traders on the market etc) are represented as a collection of information or *attributes*.



The information is organized in blocks of variables, *one block for each cell*.

```
/* T helper lymphocyte */
typedef struct tagTHblock {
        int x;                          /* lattice position */
        int CD4;                        /* T cell receptor, TCR */
        int MHCIpep;                    /* MHCI-peptide complex */
        int NVirus;                     /* viral load */
        int Age;                        /* age of the cell in units of 1/3 day */
        int tau;                        /* determines the death rate */
        int Nduplications;              /* number of times entering mitotic phase */
        unsigned short dupStep;         /* duplication phase */
        unsigned short Flags;           /* state-flags */
        AGCOMPBLOCK *AGIdI;             /* infecting HIV structure */
        struct tagTHblock *Next;        /* next block */
} THBLOCK;
```

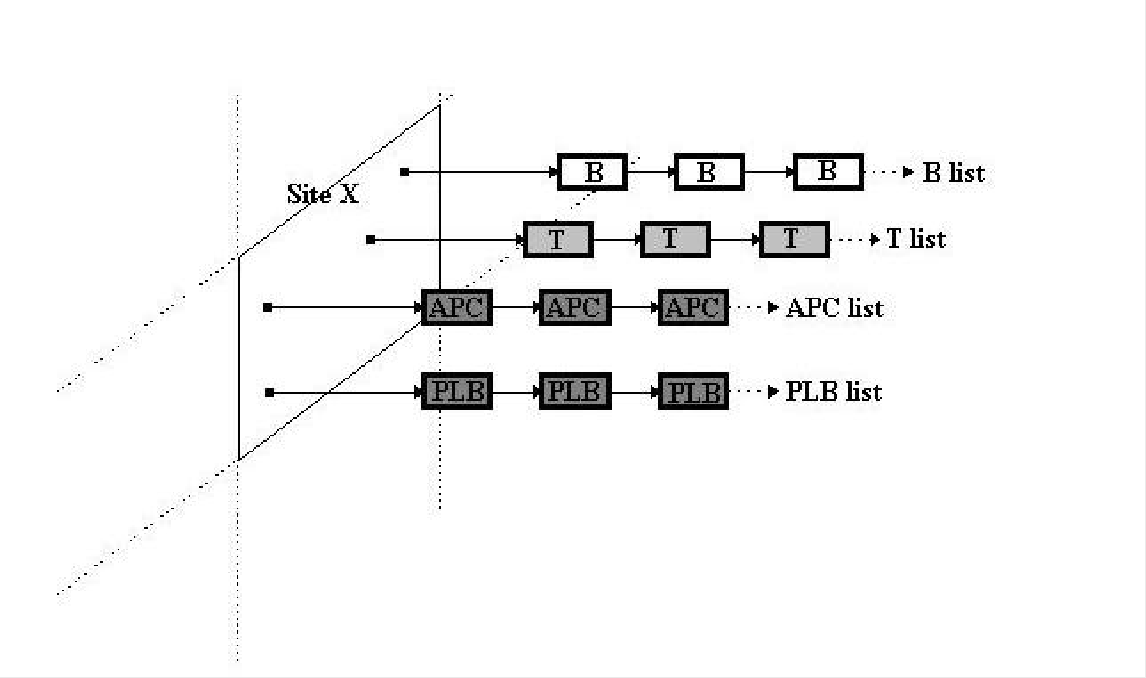


Trader's data structure

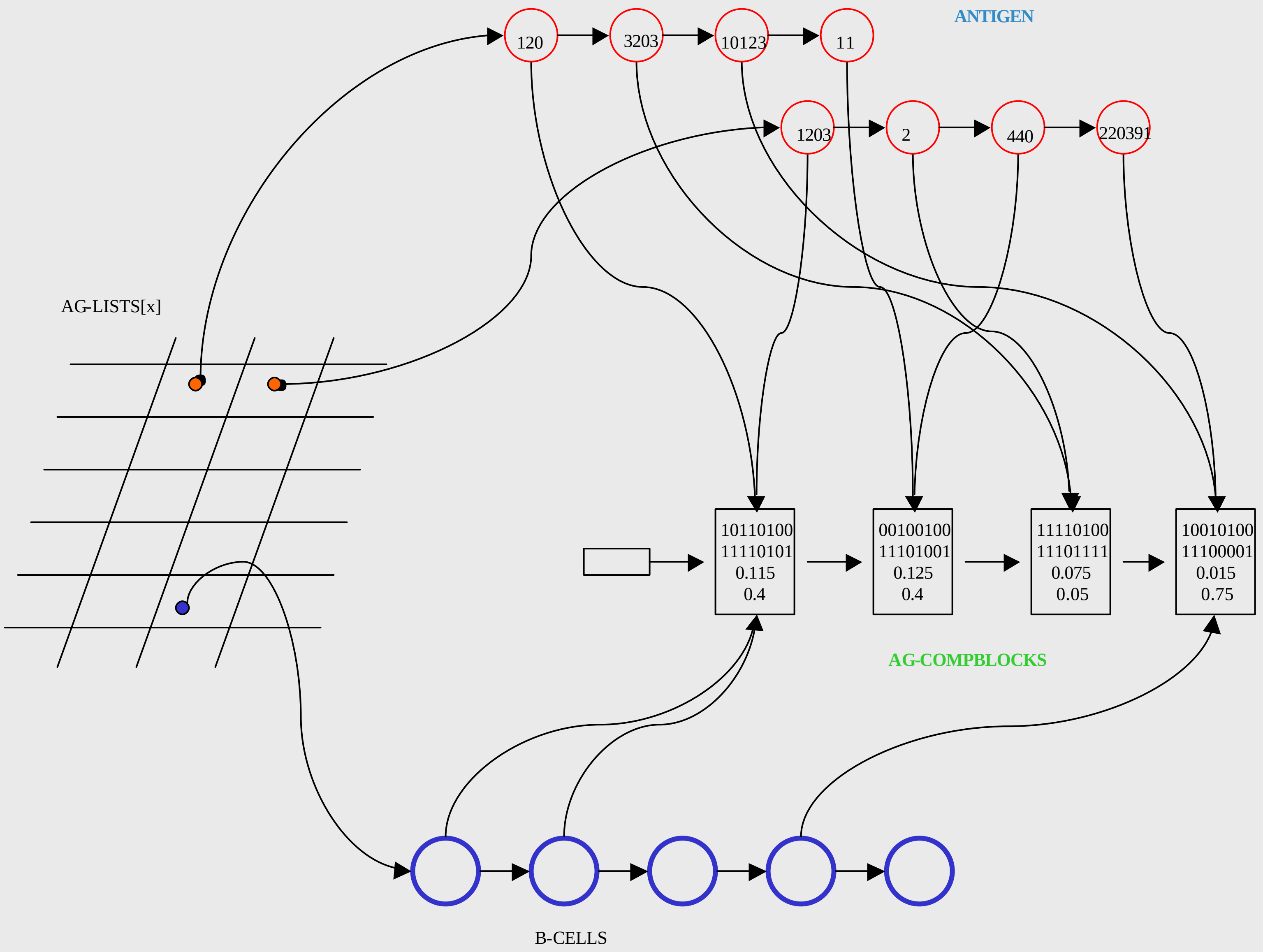
```
typedef struct tagAgentblock {
    int aclass;                         /* agents type identifier   */
    int atid;                           /* type of moving average   */
    int strategy[MAX_NUM_ASSETS];       /* buy-sell decision        */
    int nstocks[MAX_NUM_ASSETS];        /* num. of possessed stocks */
    int ordertime[MAX_NUM_ASSETS];      /* time an order may be outstanding
                                           ordertime=0 means market order
                                           >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 lists are dynamic: existing agents run out of money and new agents enter the market during the simulation

The blocks are linked in *forward* lists, one for each class of agent.
One list for each lattice site

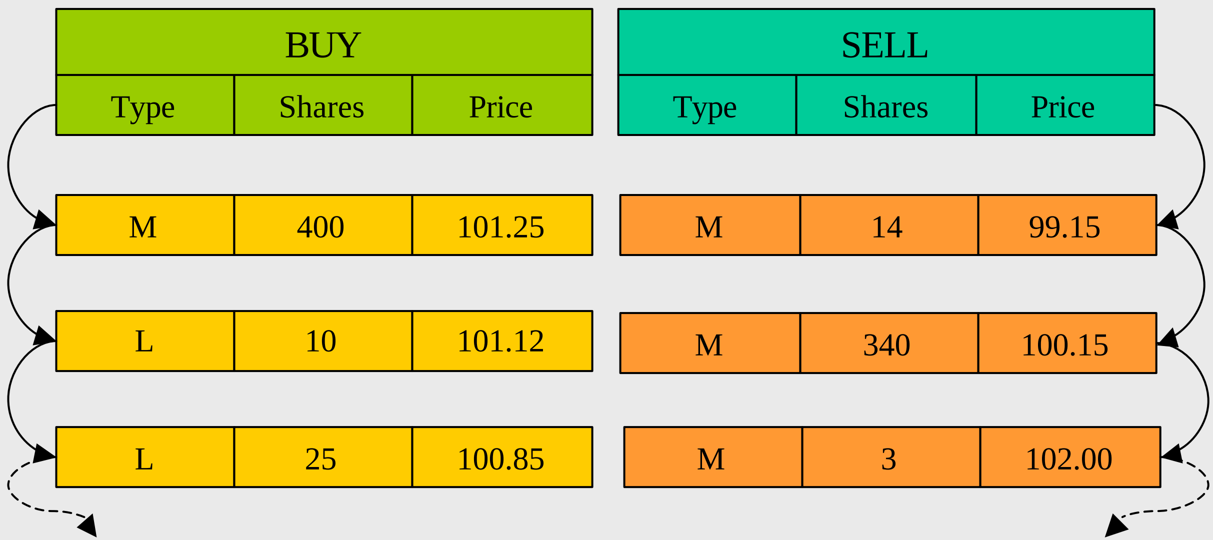


The lists are initialized at startup time and are managed dynamically at run time


ANTIGEN
120
3203
10123
11
1203
2
440
220391
AG-LISTS[x]
10110100
11110101
0.115
0.4
00100100
11101001
0.125
0.4
11110100
11101111
0.075
0.05
10010100
11100001
0.015
0.75
AG-COMPBLOCKS
B-CELLS

The buy-orders have to be sorted from the highest to the lowest order price whereas the sell-orders must be sorted in the reverse order; for each stock.

Since most of the orders have a limited life-time (in the real word, up to few days) it is necessary to check if an order is expired, which means a periodic scanning of the lists.



In a real market there are hundreds of different stocks and hundreds of millions of transactions every day. A *quasi*-realistic simulation requires a significant amount of computing time for the management of the book-of-orders.

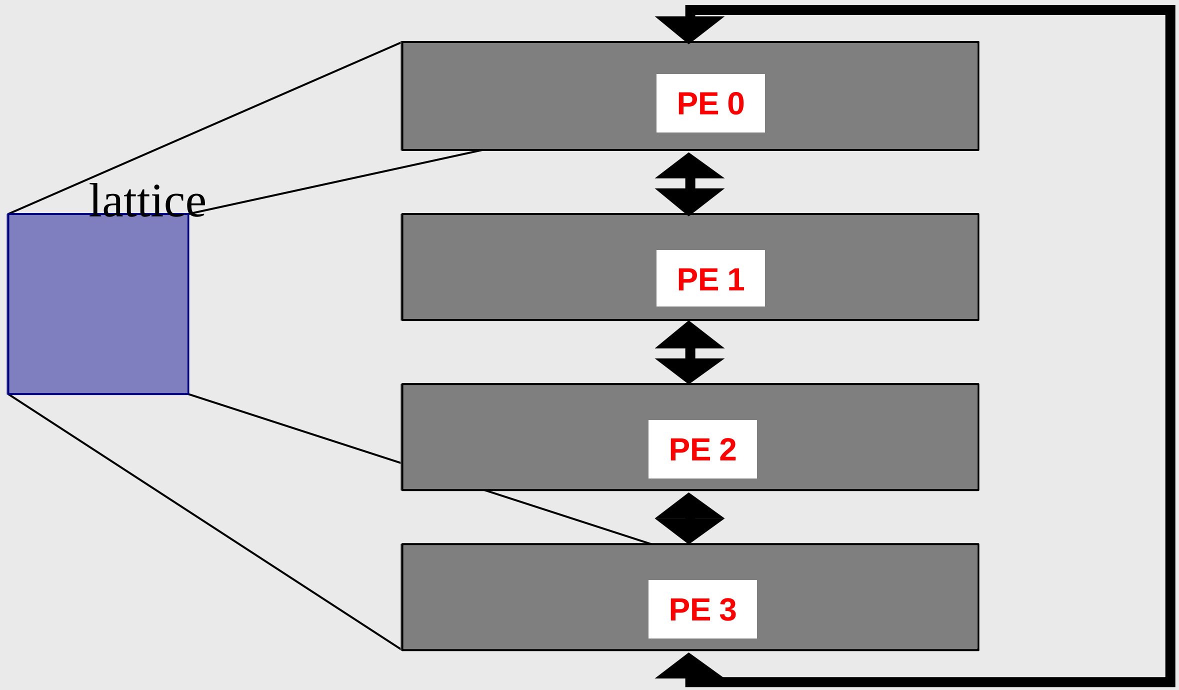
What do we need?

1. Optimized C/C++ code
2. Special libraries for special functions (e.g., list permutation, optimized insertion/deletion of nodes, list lookups)
3. Compiler directives for optimized list processing (cache-friendly)

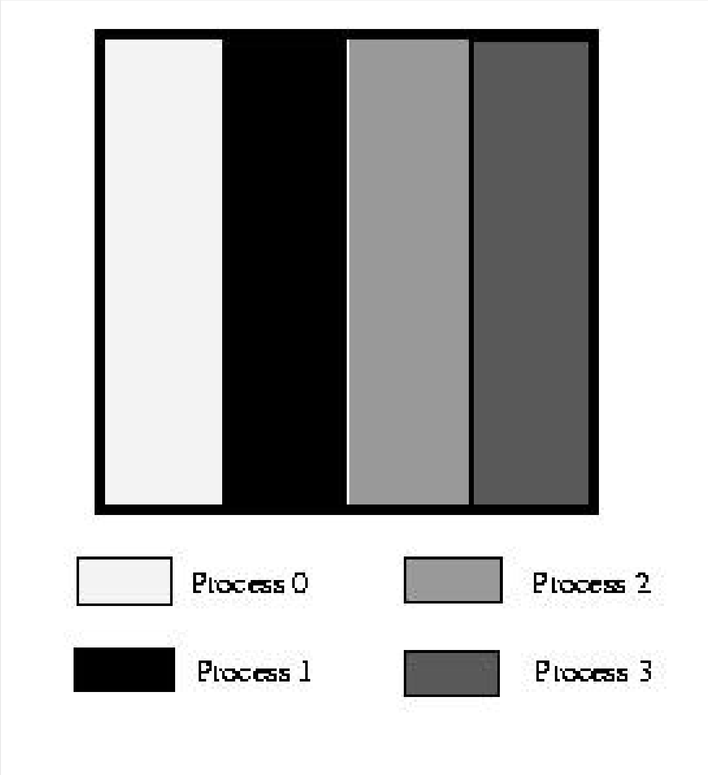
Parallelization of the Immune System simulator

The lymphonode is mapped on a two-dimension grid L_X x L_Y with periodic boundary conditions in both directions

Each task of a parallel run is in charge of a subset of the total number of agents



Each Processing Elements (PEs) works on a subset of the lattice sites. The lists that describe the entities are “local” to the PEs. This means that there is no single list split among the processors but as many independent lists as the number of PEs in use.



The problem is not “embarrassingly parallel” for two features of the simulation.

• The *diffusion* phase: If an entity leaves the “domain” of a PE to migrate to one of the “nearest neighbor PEs”, it is necessary:

- to delete the entity from the original list
- to *pack* all its attributes in a message
- to *send* the message to PE_d, that is the destination PE.

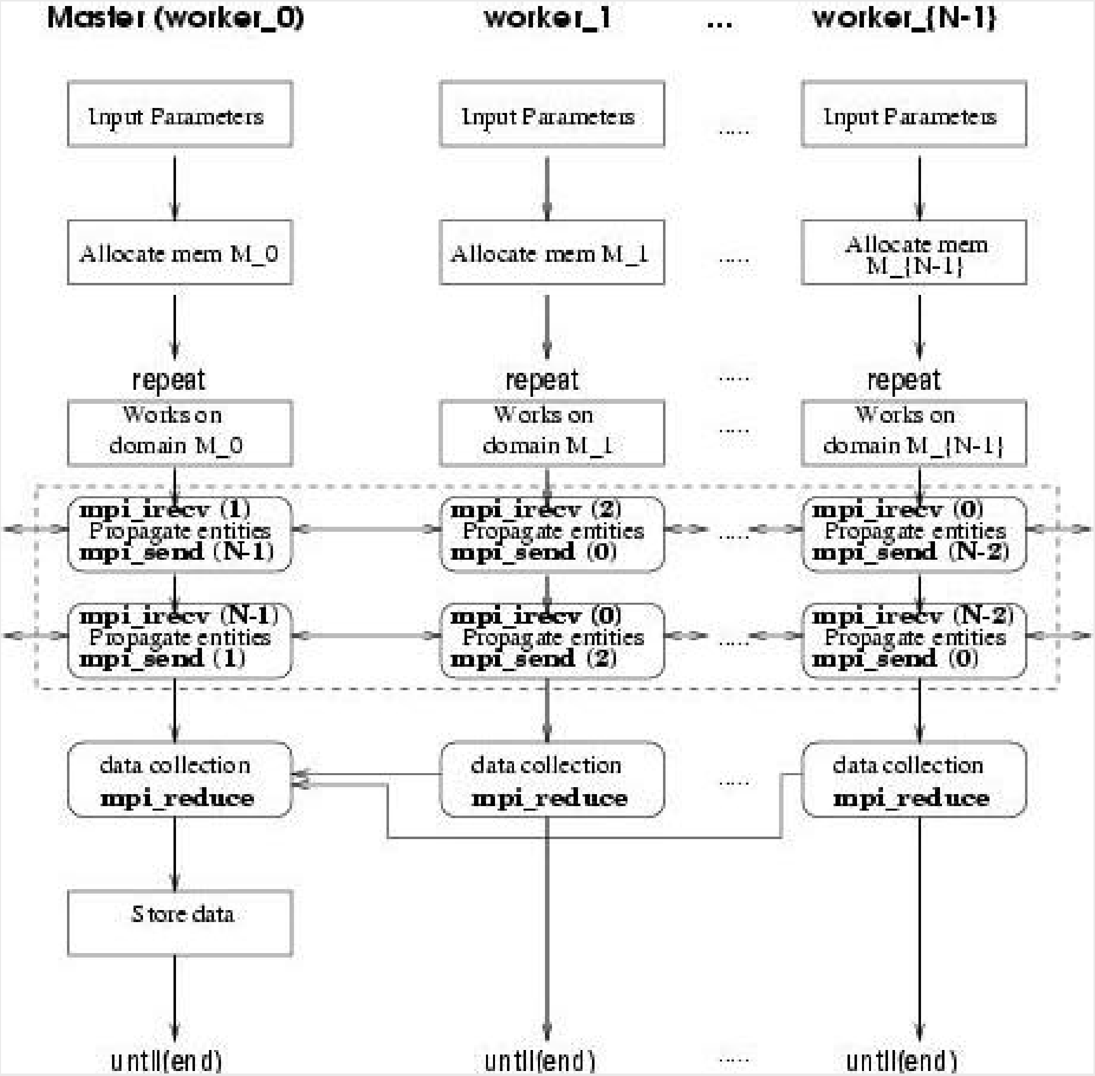
The destination PE

- *unpacks* the message
- inserts the attributes of the incoming entity in a new element that becomes the head of the corresponding list

• The output phase: a master has to collect global data

Communication scheme

Numbers represent the source/target for the **mpi_irecv/mpi_send**, of the point-to-point communication operations.



What do we need?

1. Parallel constructs to handle generic data structure for the agents for message-passing operations

2. Parallel construct for list-to-list interactions

3. User friendly parallel directives for mutual exclusive access to global data structures

Conclusions

High performant ABMs require:

Compilers: we need special constructs to define and manage agents

- – Smart data structures
- – Smart list processing

- Parallel computing
  - – Minimize Message Passing
  - – Use Shared Memory Machines

Thank you

Questions?

This work has been done in collaboration with M. Bernaschi, IAC-CNR