Perils and Advantages of Dynamic Scheduling in financial markets ABMs

Alessandro N. Cappellini[12*] and Gianluigi Ferraris[1]

1 University of Torino
2 ISI Foundation
* cappellini@econ.unito.it

Keywords: ABM, scheduling, financial market, simulation

In the simulations time has to be advanced as well as events have to be handled, i.e. they have to be scheduled in order to let them happen. In the same way agents' actions also have to be scheduled, in order to allow agents, i.e. pieces of software that need to be executed into the computer, to operate. For the such of simplicity both agents' actions and other events will be generally addressed as events.
To run an ABM simulation a lot of programs have to be executed in the right order at the correct simulated time; when the simulation involves interaction among a large number of agents, thousand of independent instances of several programs have to be activated, so the scheduling task could use a sensible amount of the computing capability seriously slowing the simulation.
Usual simulation tools provide simple solution for scheduling the activity and let the events happen, like what is implemented in the Swarm (1996) simulation libraries, that consists in a fixed sequence of objects activations, to be repeated in loop until the simulation stops.
This approach is quite useful for a large number of simulations: the central scheduler tells all the agents when they have to operate and what they are requested to do, as well as it handle environmental events. The structure of the model is very clear and the dynamic of the simulation appear to be immediately understandable. The cons of such a strategy emerge when the agent are allowed to decide if and when they will operate: all the agents have to be activated, each time, to ask them to decide if they are going to perform some tasks or not. In this way a lot of machine instructions are needed each simulation step, dramatically increasing the simulation run time.
Generally speaking the problem of scheduling activities and events becomes more and more complex if the events order may change during the simulation: for each step the order for the events to happen has to be dynamically computed by employing appropriate routines; the computational effort could be very high.
The scheduling problem was well known yet in the seventies, Franta and Maly (1977) proposed a specialized data structure to handle discrete events simulations in a high efficient way, even their sequence may dynamically change.

The basic idea consists in representing such a sequence by means of a chained list where new events are continuously inserted, in the appropriate place according with their desired execution time. To fast the search into the list to insert new events Franta and Maly used a two level pointer structure, the higher one is used to store the initial and ending time of each group of events, i.e. pieces of the list, whereas the lower one consists in the corresponding interval in the whole events chained list. In this way the search algorithm operate in two steps:

1. first it looks for the appropriate part of the chained list where the event has to be inserted then
2. it browses the sub list to find the precise place for the events.

The second step is avoided when the events time is outside the interval assigned to the lists. This approach could fasten the scheduling.

In order to sketch an Agent Based Model classification according to the centrality and autonomy of the Agent (versus the Model) we can find out three main classes:

- "When" and "What to do" is decided a priori by the Model,
- "When" is decided a priori by the Model, but "What to do" is decided each time by the Agent,
- "When" and "What to do" is decided by the Agent itself.

In the first case, the Model is the clock, the scheduler, or better, it acts as a dictator, or better again, it is a stroke of a boat. It is charged to activate the agents each time telling them what they are asked to do. Following this approach agents become simple pawns used to build the actions or to simulate a predefined process.
In the second case the Model still acts as a stroke, as a clock, but the Agents become free to decide about their actions. So they are strictly asked each time to compute an action or a strategy, but they can act in different ways or not act at all.
The third case realizes the centrality and the higher independence of the Agents, by decentralizing several algorithms and functions, but creating as well some complexity, both in the simulation and in the interpretation of its results.

In this paper we would like to explore advantages and perils of the third proposed approach. Among the former we can enumerate:

- time span and time granularity are variable and arise endogenously during the Simulation,
- reduction of the simulation execution time,
- the Agent can show an enhanced sensibility and higher interaction with the environment (as well as with other agents).

Among the latter we can list:

- an increase in code complexity that brings both to some possible errors, and to a lower degree of comprehension,
- the possibility to create loops or to stop some agents,

– the causality and the events sequence are no more predefined, and they have to be discussed and investigated.

Another question we would to address is about what kind of model this solution is suitable for. We will use demo models to show these usages and suggest some couplings. This kind of solution, developed by a focused algorithm inspired to Franta-Maly works, appeared suitable for those models in which time is a crucial variable for agent behaviours.

Typical instances of those models are financial market simulations, where a large number of agents have to operate in a fully autonomous way. Some example of those techniques are provided through the SUM/SumWEB platform (Terna 2000, Cappellini and Ferraris 2008), able to face different market institutions as well as different scheduling mechanisms.