DAI project

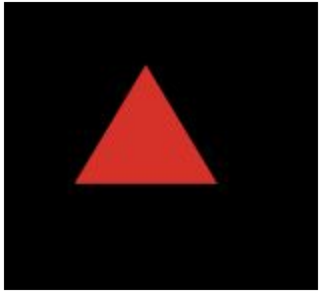
Swarm and cooperative behaviors for problem solving in MAS

Matteo Mosconi

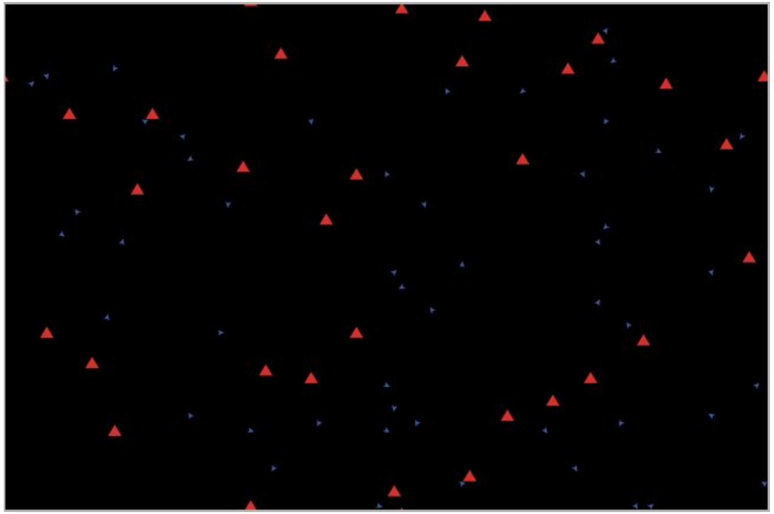
What is it?

It is a model about the interaction of multiple generic entities (the solvers) trying to solve generical problems which are already, and can appear, in the environment.
It tries to show the importance of swarm intelligence (in particular the behavior of slime mold in aggregation) and cooperative behaviors (a sort of contract net-like protocol) that combined together can greatly improved the solution of possibly inherently distributed problems.

a solver

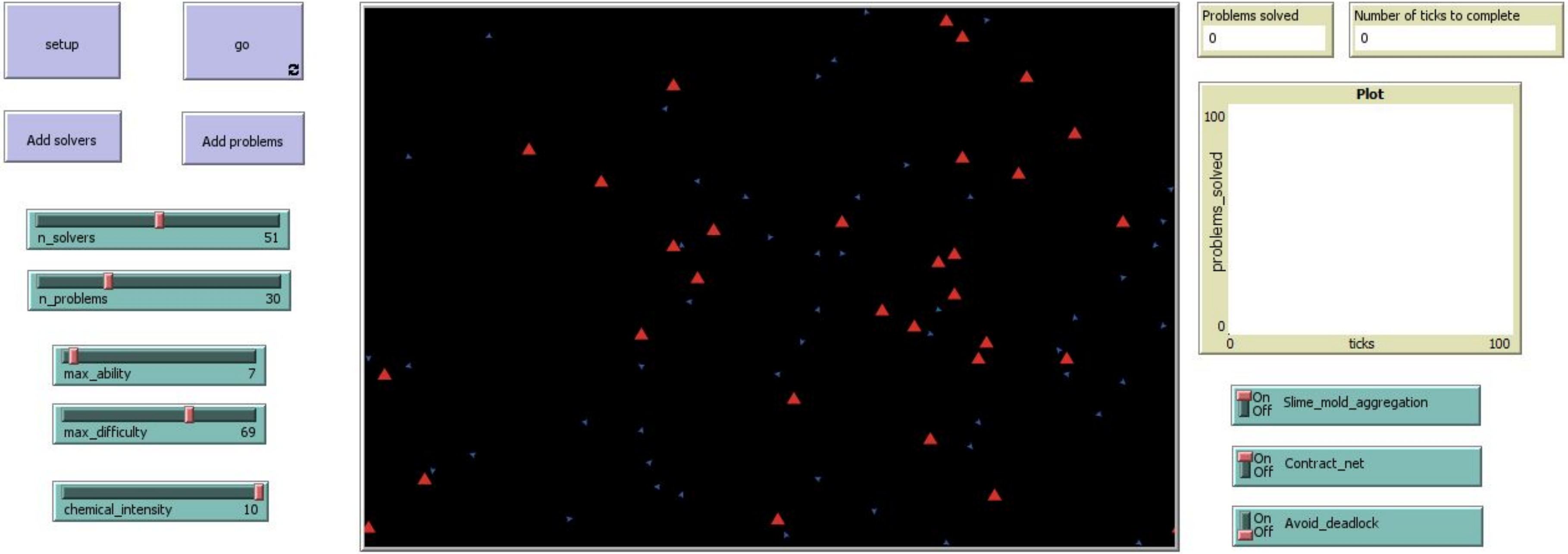


a problem



the environment

The interface



How the model works

This is a "modular" model, in the sense that I've provided 3 switches to able/disable the key features of the model, I will describe the general functioning, with all the features "On".

Once created, the agents simply start moving randomically in the environment, but at the same time they also sense the environment around them, searching for chemical; If the chemical on a particular patch is the biggest around (and greater than a certain threshold) they will turn toward that patch until they will end up eventually (together with other agents) on a problem. At the beginning there is no chemical on the patches, in fact, unlike what happens with the slime mold, agents do not spread continuously chemical, but only when they arrive on a problem unsolved. So at the start the first agent that arrives on a problem, starts spreading the chemical that will help other agents to arrive there and to group with it, we can see this initial spreading of the first agent as a call for proposal in the contract net.

The problems have different difficulties, and the solvers different abilities to solve them; Once on the same problem, this will be solved only when there is enough ability (the sum of agents abilities there, they can be seen as the bidders of a contract net protocol).

In some condition there is the need of introducing a maximum time agents can stay on a problem without solving it to avoid deadlock, I've wanted also to introduce a switch that "allows deadlocks" to underline that if this condition happens the system could be under dimensioned (i.e. the number of problems or their difficulties is too large for the number of solvers or their abilities).

How to use it

Other than the setup and go buttons, it is also possible to add solvers and problems on the fly, it can be useful for example to see how a new group of agents can impact on the world (e.g. in a deadlock situation).

There are two sliders to control the number of solvers and problems, together with two for maximum ability and maximum difficulty, the actual number of these properties for each agent is actually pick at random from 1 to the maximum specified by the user with the slider. There is at the end a slider to control the intensity of the chemical that each agent will drop (and the environment will spread) once arrived on a problem.

As mentioned before there are 3 switches granting the possibility for various combinations for the simulation.

There are monitor and plot to visualized the number of solved problems in real time; Finally a monitor to see in how many ticks (that can be considered time for NetLogo) all the problems on the environment have been solved.

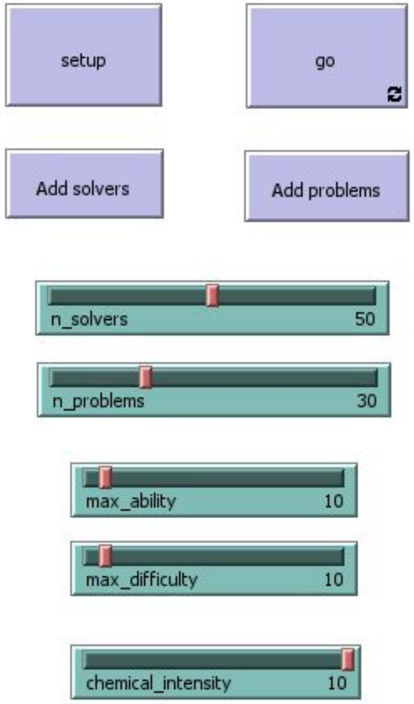
For more details see the NetLogo Info and Code tabs.

Some experiments…

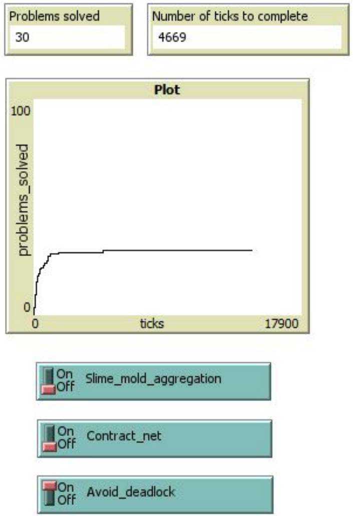
I’ve started by setting out the parameters for this first series of experiments.

It is important to notice that I’ve set the maximum ability for both problems and solvers to the same value, in this way it is possible to evaluate if a cooperative behavior is still improving the results, even in the presence of agents that can solve problems by themselves.

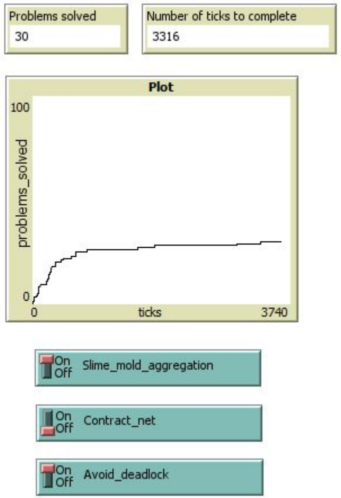
I want to add that for having consistent results would be necessary to do hundreds or even thousands of runs (for statistical reliability). Alternatively it would be possible to fix the parameters of each agent and proceed with those fixed values throughout the experiment. This is not the purpose of this simulation, so I will only state qualitative results.



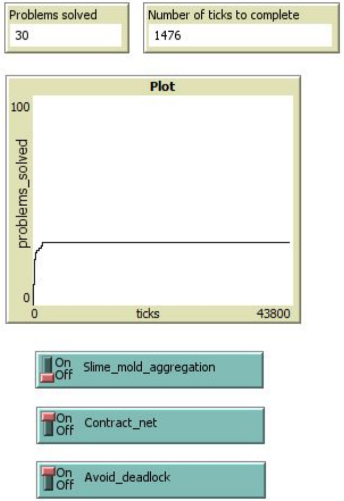
First I show the results without Slime_mold_aggregation and Contract_net (for now Avoid_deadlock is turned on, its usefulness will become clear on a later experiment). (They have an average completion time of around 5000 ticks based on the average between 10 runs and trimming out the extremes, to note that as said before for a statistical validity much more runs would be necessary).



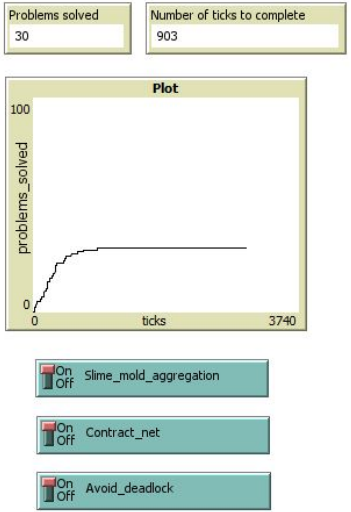
With only Slime_mold_aggregation turned on the improvement is not so relevant, because even if solvers start to aggregate on a problem they do not cooperate. (Around 4000 ticks for completion with the same average technique as before).



With only the Contract_net turned on the improvement start to be significantly relevant (this underline how a cooperative behavior can help even in presence of purely random movements in the environment, because I’ve not turned on the Slime_mold_aggregation so solvers are not attracted to problems at all). (Around 1500 ticks for completion with the same average technique as before).



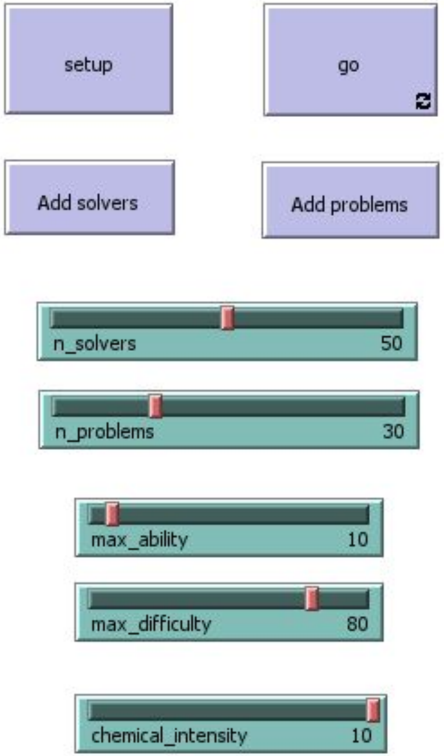
The definitive version of the experiment is with all the features turned on; As expected, in this case the solvers are faster in solving the problems. Still, from a qualitative perspective at least, the greater boost is given with the enabling of the cooperative behavior. (Under 1000 ticks for completion with the same average technique as before).



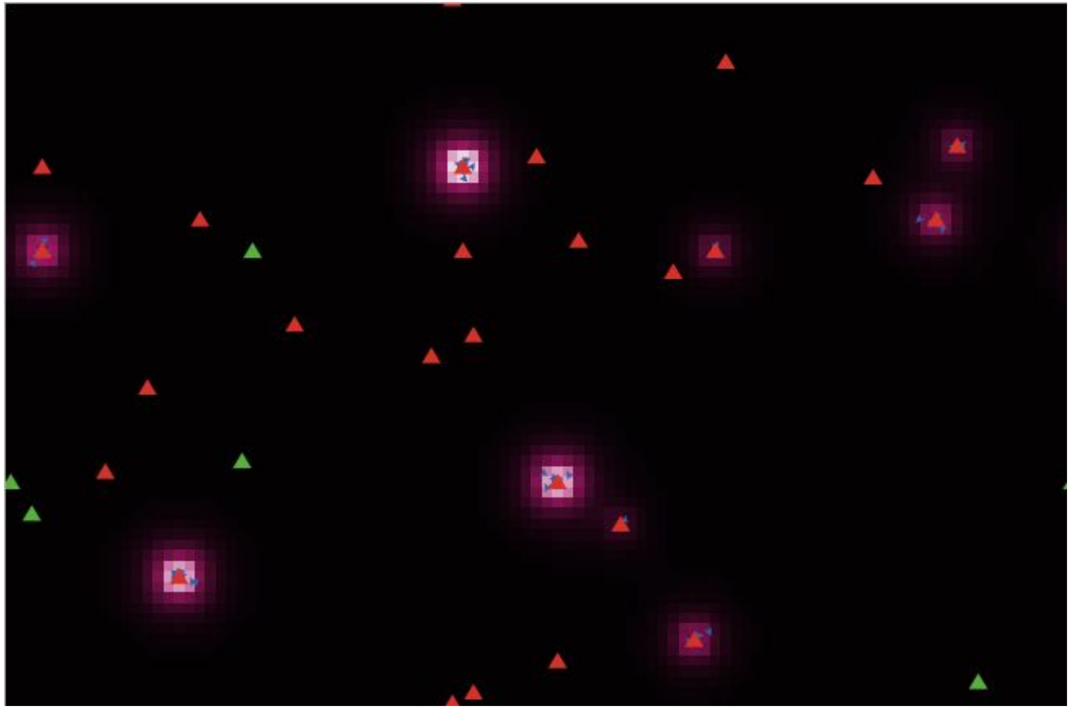
Another example…

I’ve tried to simulate the behavior of this model in presence of inherently distributed problems (i.e. problems that can not be solved by a single agent only by itself). One can achieve this simply setting the max_difficulty of the problems greater than the max_ability of the solvers as in the image on the right.

In particular I put the single max_difficulty much higher than the max_ability for showing a behavior that I think could be interesting. In many of this simulations in fact, deadlock appears. A situation in which some solvers are waiting on a problem for other solvers to show up because they cannot solve the problem by themselves. However this aid will never happen because the other solvers too are in the same situation. (An example is the image in the next slides).



Deadlock example



It is clear that the switch for avoiding or not deadlock is needed to underline this condition, that, as mentioned before, can mean that the system is under dimensioned.

This situation can be solved either by adding solvers that “patrol” the environment or by incrementing their abilities, due to the fact that in real world cases you cannot act on the problems (i.e. you cannot modify their difficulties or their number).

Conclusions

In general, the model is useful to show how swarm intelligence and cooperative behaviors combined together can greatly improved the solution of possibly inherently distributed problems in multi agents systems.

Even from a qualitative point of view, the solution shows improvement of one order of magnitude with reference to the cases without this behaviors enabled. Not to mention that they introduce also adaptivity, thanks to swarm intelligence (randomness and mass are both present on the model).

This model is open to new possible extensions. A possible one, in order to make it more complicated, would be introducing new classes of problems (for example differentiated by color) with the solvers that instead of having a single ability to solve, will have multiple different abilities to solve different classes of problems. Another possibility would be introduce an implementation of a more constrained environment, instead of this open and generic one.

References

Wilensky, U. (1997). NetLogo Slime model. http://ccl.northwestern.edu/netlogo/models/Slime. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Thanks for the attention!