

Synchronization Overlap Trade-Off for a Model of Spatial Distribution of Species

João Bioco[1,2,4(✉)], Paula Prata[1,2,4], Fernando Cánovas[3], and Paulo Fazendeiro[1,2,4]

[1] Universidade da Beira Interior, UBI, Covilhã, Portugal
[2] C4 - Centro de Competências em Cloud Computing (C4-UBI), Universidade da Beira Interior, Covilhã, Portugal
[3] Universidad Católica San Antonio de Murcia, Facultad de Ciencias de la Salud, Murcia, Spain
[4] Instituto de Telecomunicações (IT) Covilhã, Covilhã, Portugal

Abstract. Despite of the widespread implementation of agent-based models in ecological modeling and another several areas, modelers have been concerned by the time consuming of these type of models.

This paper presents a strategy to parallelize an agent-based model of spatial distribution of biological species, operating in a multi-stage synchronous distributed memory mode, as a way to obtain gains in the performance while reducing the need for synchronization. A multiprocessing implementation divides the environment (a rectangular grid corresponding to the study area) into stage-subsets, according to the number of defined or available processes. In order to ensure that there is no information loss, each stage-subset is extended with an overlapping section from each one of its neighbouring stage-subsets. The effect of the size of this overlapping on the quality of the simulations is studied. These results seem to indicate that it is possible to establish an optimal trade-off between the level of redundancy and the synchronization frequency.

The reported paralellization method was tested in a standalone multicore machine but may be seamlessly scalable to a computation cluster.

Keywords: Parallel programming · Multiprocessing · Agent-based modelling and simulation · Synchronization-reducing algorithms

1 Introduction

Agent-based modelling (ABM) is an approach centered in agents (unique individuals) that interact with each other and their environment in order to accomplish a certain goal. In this modelling approach (called bottom-up modelling) the behaviour of the system emerges by the local interactions between agents in the environment where they exist [1]. This fact makes this modelling approach widely applied in several areas, such as in biology, engineering, ecology, epidemiology, etc. [2,20,23,26]. One of the areas that has largely adopted agent-based

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12950, pp. 296–310, 2021.
https://doi.org/10.1007/978-3-030-86960-1_21

modeling is ecology. Generally in this field, agents are described as individual organisms with behaviours and proprieties that usually change during their life cycle [18].

In the last years due to the climate changes observed in the nature, the interest in the study of species distribution has increased, in the sense of creating mechanisms for species preservation. There are some aspects that have to be taken into account when using ABM to model and simulate the distribution of species, such as: the environmental conditions that influence the life cycle of the species, the available resources (i.e., food, water), the dimension of the environment where species exist (the study area), etc. [4]. The area under study can be small or large, and the coarseness of the simulation stage can be quite heterogeneous from simulation to simulation. In the cases where the dimension of the environment is very large and the required level of detail for the simulation is high, the simulation can be quite time consuming and requires very large computation power. In this case it is necessary to implement strategies to narrow down the time requirements.

This paper presents a strategy to parallelize an agent-based model of spatial distribution of biological species, running in a multi-stage synchronous mode enabling the reduction of the processing time while assuring that there is no significant information loss. This study parameterizes the size of the overlapping section sent with each stage-subset and the number of iterations done between each global synchronization. The findings show that it is possible to attain a good trade-off between the size of the internal computation and the time of data transferring while maintaining the correctness of the algorithm.

The remaining of this paper is organized as follows: the Sect. 2 describes some studies that implemented parallelization in ABM; the Sect. 3 presents our model proposal; the experimental results are presented in the Sect. 4; in the Sect. 5 the discussion of the study is presented, followed by the concluding section.

2 Related Work

ABM in ecology was firstly applied in forest modelling, and the application to other areas of ecology began to increase in the 1990 s [12]. A particular early use of ABM was to model the recruitment of fish populations, in order to understand and assess human impacts in the mortality of fish recruitment [11]. However, two main categories of models have motivated the development of ABM in ecology based on purpose [14]. One category is mainly used to model a specific type of species, population or ecosystem with management purposes; another category aims to help ecologists to have a better understanding of the factors behind of several ecological phenomena [19]. A significant advantage of ABM in ecology concerns the study of populations by considering the individuals' behaviours and relationships. This is a more natural way of looking at population, making simulation results closer to reality [5].

The two main approaches to parallelize spatial ABMs are [22]: 1) to divide the computations at agent level where each processing element (PE) is responsible

for a set of agents; 2) to divide the computations at the spatial environment level where each PE is responsible for a set of grid cells. In both cases, to handle the interaction between agents and its movement requires communication and synchronization, which constitutes the limiting factor for obtaining a scalable parallel model.

Existent proposals for parallel implementations of spatial ABMs range from multithreaded implementations in shared memory architectures [13,24] to implementations in Graphic Processing Units (GPU)s [6,7,17]. However, most of the proposals are based on distributed memory programming models which can be scaled to thousands of cores. These works include frameworks as FLAME [8,25] and Repast HPC [9,10] that use MPI for inter process communications. The work presented in [27] implements an ABM in the Apache Spark framework trying to take advantage of its in-memory computation model. In all these works the main performance bottleneck remains the communication cost.

3 Methodology

The model description follows the ODD (Overview, Design concepts, Detail) protocol [15,16]. Its main components are described in [4]. The purpose of the model is to explore different hypothesis on the space-time distribution of species. The model comprises an environment disposed in a regular rectangular grid. Each grid cell stores its suitability value (a value between 0 and 1 that sums up the environmental conditions of the cell), and the number of specimens that are present in the cell. This amount increases or decreases during the life cycle according to the suitability of the cell and the give-and-take of specimens with the neighbouring cells. Figure 1 shows in more detail the characterization of the environment.

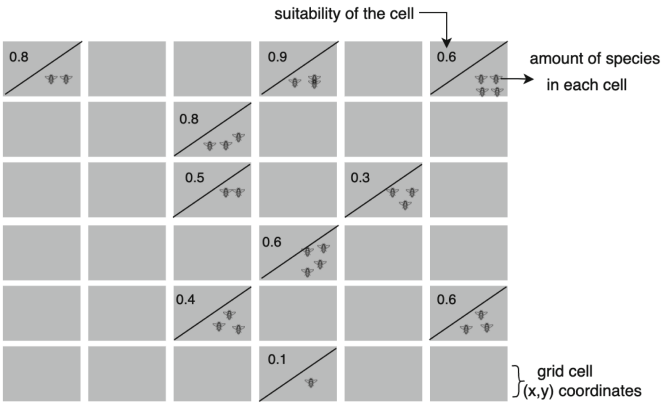


Fig. 1. Characterization of the environment.

3.1 Process Overview and Schedule

Figure 2 presents the main steps of the simulation process. It starts by initializing the environment stage (hereafter referred as patches), after setting the parameters of the model. The environment is initialized by instantiating each cell (hereafter also referred as patch) with its suitability value obtained by the combination of a set of environmental variables, and an initial quantity of species. In each iteration, to each cell is applied a birth and death rate affecting the quantity of species. These rates are conditioned by the suitability value of the cell, i.e., species are most likely to survive and reproduce in more suitable locations, on the other hand, species are more likely to disappear in less suitable locations. The spread of the species occurs through the neighbourhood of the cells. The reported model implements the Moore neighbourhood [21]. Therefore, at each iteration, each cell transfers an amount of material (quantity of species) to its neighbours, according to a spread rate. Algorithm 1 describes the updating mechanism that simulates the spatial dynamics of the spreading of a natural organism guided towards spatial self-organization (species life cycle).

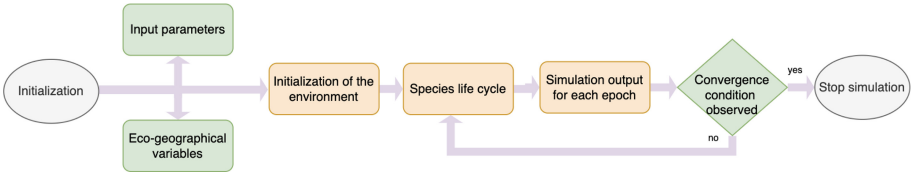


Fig. 2. General steps of the simulation process

Algorithm 1. Life cycle algorithm. The reproduce method contains the exchange policy of the cell with its neighbours depending on its birth, death and spread rates.

```
procedure Distribution_update(patches, previous_patches, steps)
    for k in steps do
        for i, row in patches do
            for j, patch in row do
                neighbours = patch.find_neighbours(previous_patches, i, j)
                patch.reproduce(previous_patches[i][j], birth_rate,
                                death_rate, spread_rate, neighbours)
            end for
        end for
    end for
    return patches
end procedure
```

The sequence of patches (different states) obtained during the iterative operation of the Algorithm 1 is called the evolution. In the sense defined in [3]

evolution is the result of the simulation task, representing the process under simulation. If the process converges to a stable global state, then the algorithm evolution has a termination. If that is not the case, then the evolution is infinite, exhibiting oscillatory or chaotic behaviour. Algorithm 2 represents the evolution snapshot for a fixed amount, t, of epochs.

Algorithm 2. Evolution of the simulation task.

```
Initialize patches, previous_patches, set epochs, output_interval and let steps ← 1
for t in epochs do
    patches = Distribution_update(patches, previous_patches, steps)
    if mod(t, output_interval) == 0 then
        Δ = sum(abs(previous_patches − patches))
        Copy patches into previous_patches
    end if
end for
```

Alternatively the evolution can be terminated once a convergence condition is observed. In that case the evolution stops, i.e. for all that matters it has converged, whenever the Cell-by-Cell difference, Δ, drops below a threshold τ given by

$$\tau = \epsilon \times N_c, \tag{1}$$

where N_c is the number of cells in the environment and ϵ stands for the amount of admissible error between any given pair of corresponding cells (or minimum distinguishability level). This threshold value can also be used as a reference value in order to analyse if two different evolutions exhibit, or not, a similar convergent behaviour.

3.2 Parallelization Strategy

The adopted parallelization strategy consists of assigning to each PE a set of grid cells. The species distribution map was divided into stage-subsets (hereafter referred as strips, since in our implementation any subset encompasses all the columns of the main stage), each one with a dimension given by the number of rows in the map divided by the number of available processes.

While pursuing high efficiency it is possible to break the equality between the initial sequential evolution and that of a decomposed parallel algorithm. Therefore in order to guarantee the equality of the sequential evolution to its parallelized version a set of correctness conditions, see [3], must be assured during the interaction between processes. Simply stated, "the problem is to organize the parallel operation in such a way that each domain interacts with the adjacent ones by exchanging data that are needed to be used in one of them for computing the next-states in the other". Moreover any given cell must be updated only once per iteration. Algorithm 3 presents the synchronous parallelization used in the reported experiments.

Algorithm 3. Synchronous parallel algorithm.

```
Initialize patches, previous_patches
Set number_processes, overlap, steps, epochs, output_interval
for t in epochs do
    strips = Build_strips(patches, number_processes, overlap)
    previous_strips = Build_strips(previous_patches, number_processes, overlap)
    for each strip_s do
        Run process P_s(Distribution_update(strip_s, previous_strip_s, steps))
    end for
    for each process P_s do
        Receive and fuse the P_s results into patches
    end for
    if mod(t, output_interval) == 0 then
        Δ = sum(abs(previous_patches − patches))
    end if
    Copy patches into previous_patches
end for
```

It goes without saying that the equality between sequential and parallel versions is only assured if all the cells whose values are necessary for updating any given cell of the strips are available. In this case, using a Moore neighbourhood, this means that in order to perform a parallel iteration each strip must be extended by adding a border region (overlap) of at least one row. When dealing with parallel processes one must also be aware of the interprocess communication costs, which may be reduced by preventing frequent synchronizations between processes. This can be achieved if, instead of joining the strips after each iteration, the different stages are allowed to evolve independently in their own parallel processes for a given number of evolutive steps. Once again, the equality between versions is only achieved if the overlap between strips (or partial stages) is composed of a number of rows equal or bigger than the number of steps.

To deal with the border region, the strips are extended with a set of adjacent rows coming from each neighbouring strip. In what follows those common regions are referred as overlap, as shown in Fig. 3 where it is emphasized the domain decomposition.

In the body of the simulation cycle, the data is divided into stage-subsets. Each chunk of data (including a strip of the current distribution map, $patches_k$, and its overlapping sections) is sent to a process running in parallel with the ones responsible for the other data chunks. Each process will perform over its data the number of iterations given by a step parameter. At the end of the parallel phase, just the arising strips are sent back to the main process, producing the next map instance ($patches_{k+step}$). This means that, at this phase, the overlapping sections are discarded since their main purpose is to act as a buffer, updating the frontier cells in the starting iterations of each process. The core strips of the distribution map are fused together and the main process continues as described.

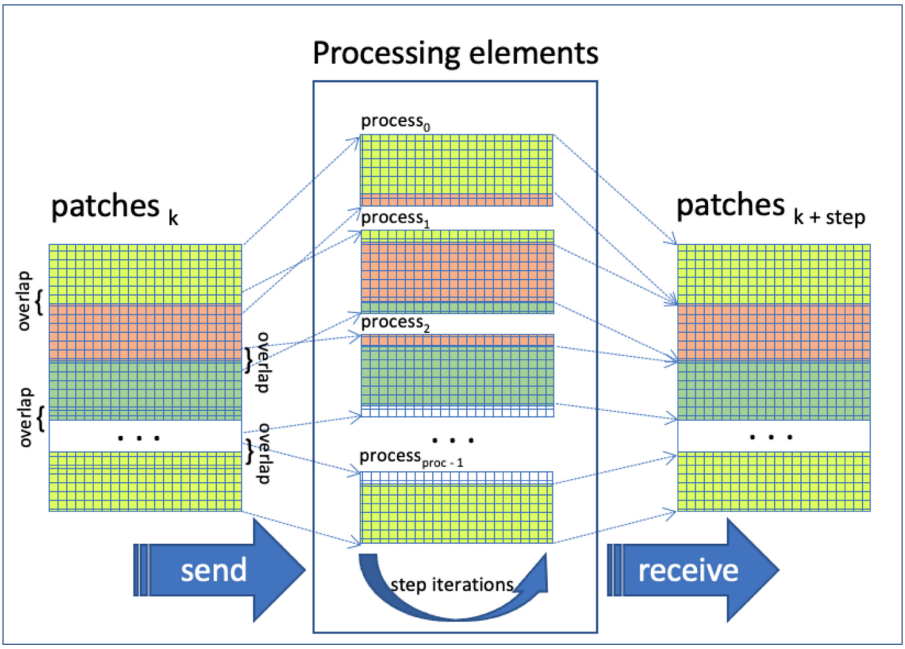


Fig. 3. Parallelization of the spatial environment. Decomposition of the study area into a set of overlapping strips for parallel processing.

4 Experimental Results

In the reported experiments were performed simulations using both sequential and parallel implementations of the model. Initially, the simulation with the sequential implementation was performed, followed by the simulation with the parallel implementation. Execution times were computed in a machine with the following hardware and software configurations: a) Operating System: Linux Ubuntu Desktop version: 18.04.5 LTS 64 bits, b) RAM: 64 GB, c) Processor: Intel® Core^{TM}i9-9900X CPU @ 3.50 GHz × 20, d) Python version: 3.7.2.

For these experiments the spatial distribution of the african honeybee Apis mellifera in the Iberian Peninsula was simulated. The set of variables of interest for the environment was composed by four eco-geographical variables (maximum temperature of the warmest month, rainfall seasonality, average annual temperature and minimum temperature of the coldest month) which together are the input values determining the suitability of each cell.

In order to facilitate the comparison between the two implementations (sequential and parallel) the quantity of species in each grid cell was initialized by using the same seed for each simulation.

4.1 Parameterization

Table 1 shows all the parameters of the model and the values used for both sequential and parallel implementations. For the parallel implementation, 12 parameter combinations were chosen: step of 10 varying the frontier with the values (4, 6, 8, 10); step of 50 varying the frontier with the values (20, 30, 40, 50); and step of 100 varying the frontier with values (40, 60, 80, 100). Therefore, for the parallel implementation 12 different simulations were performed. For the sequential implementation the results at each timestamp were saved; and for the parallel implementation the results were saved according to the chosen step. The number of processes was fixed to 12 in order to analyse the behaviour of the algorithms when the processes were subjected to a varying workload directly related with the dimension of the data used in the experiment.

Table 1. Models' parameters.

Parameters	Value
Initial population	200 000
Number of epochs	200
Cells capacity	1000
Output generation interval	According to the step
Birth rate	0.9
Death rate	0.2
Spread rate	0.6
Type of neighbourhood	Moore
Environment dimension	1210×1940

4.2 Quasi-equality Behaviour

Aiming at the determination of the rate of degradation, resulting from the reduction of the border region, the differences between the initial sequential evolution and the parallel evolution with different process steps (period where each strip evolves independently of the remaining ones, see Fig. 3) were analysed. For each fixed number of *steps* a set of experiments were conducted for different overlap levels: 100%, 80%, 60% and 40% of the performed steps.

For a fixed number of sequential epochs (200 in the reported experiments) the number of steps has a direct influence on the number of synchronizations (with its inherent communication costs) whereas the level of overlap (determining the overall number of cells treated by each process) has a major influence on the processes' workload.

As previously noted the equality between versions is only achieved if the overlap between strips (or partial stages) is composed of a number of rows equal or bigger than the number of steps.

Table 2 shows the sum of the differences (Cell-by-Cell comparison) between each parallel combination (different steps and overlaps) and the sequential implementation in the same set of iterations.

Table 2. Cell-by-Cell differences, Δ, between the results of the sequential implementation and the results obtained from the parallel implementation in the iterations: 50, 100, 150 and 200.

Steps	Overlap	50	100	150	200
10	10	0.00067	0.00067	0.00067	0.00065
10	8	0.14313	0.11767	0.10852	0.10361
10	6	5.20148	3.82719	3.41619	3.24228
10	4	58.60432	42.44981	38.00387	36.54078
50	50	0.0	≈0	≈0	≈0
50	40	≈0	≈0	≈0	≈0
50	30	≈0	≈0	≈0	≈0
50	20	0.00372	0.00170	0.00099	0.000706
100	100	–	0.0	–	0.0
100	80	–	≈0	–	≈0
100	60	–	≈0	–	≈0
100	40	–	≈0	–	≈0

For $\tau < 1E^{-5}$, see (1), the error values were denoted as "≈0".

As expected, Table 2 shows that the differences between the sequential and parallel evolutions increase with the reduction of the level of overlap between stage-subsets. Those differences are more apparent at the earliest stages of the simulation due to the convergence of the simulation to a stable state. Interestingly enough, it should be noted a somewhat counter-intuitive observation: in this model a large number of parallel steps tends to reduce the overall inequality between the parallel and the sequential evolutions. This means that, even when there is a significant relative gap on the number of rows necessary to guarantee an equal evolution, the local convergence of the algorithm is able to circumvent this lack of data as long as enough processing steps are allowed. In the following section some performance indicators for these and other selected experiments are reported.

4.3 Performance Comparison

As a way to analyse the performance of the proposed parallel strategy the improvements on the speed of execution for several parallel implementations with different parameter combinations were measured. In the reported experiments the speedup

of each parallel configuration P against the corresponding sequential version S is given by

$$S_P = T_S / T_P, \tag{2}$$

where T_S is the execution time of the sequential evolution and T_P is the execution time of the assessed parallel configuration.

The impact of varying the step and the overlap parameters for a fixed number of processes and several environment dimensions was studied. The speedups of a chosen configuration when the number of processes varies from 2 to 20 were also calculated. Figure 4 shows the speedups obtained when running the simulation for each parameter combination described in Sect. 4.1, considering three environment maps with dimensions: (1210 × 3880), (1210 × 1940) and (1210 × 970), that is, the initial map (1210 × 1940) was extended and shrunken along its second dimension. The simulations were ran by the 12 processes over 200 epochs. That number of iterations has shown to be enough to reach convergence.

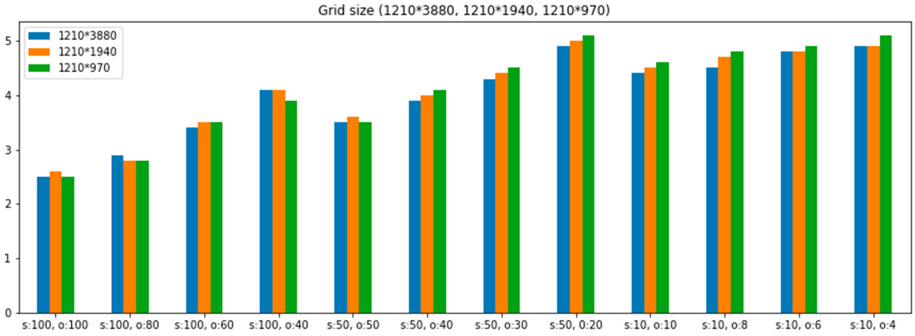


Fig. 4. Speedups obtained with maps of different number of columns, varying the number of steps s (10, 50 and 100) and for each step, the overlapping (o) size varies from 100% to 40% of the number of steps.

In Fig. 4 it is possible to observe that the speedups are approximately the same for the three maps. The speedups for the small map have values only insignificantly higher for the smallest step values. That is, increasing the number of columns has no impact in the speedup. When observing the figures with the same step value, as can be expected, the speedup increases when the overlap reduces. Less data redundancy implies less communication and less processing. Regarding the overlapping value, as seen in the previous section, it must be high enough to avoid information loss but the higher it is the worse will be the performance. In the studied cases the best speedups are obtained for the pair $steps = 50$ and $overlap = 20$ (i.e. 40% of the step value).

Then the variation of both dimensions of the map was studied, considering this pair of values. Figure 5 shows the speedups obtained when running simulations along 200 epochs in 12 processes for three maps with dimensions (605 ×

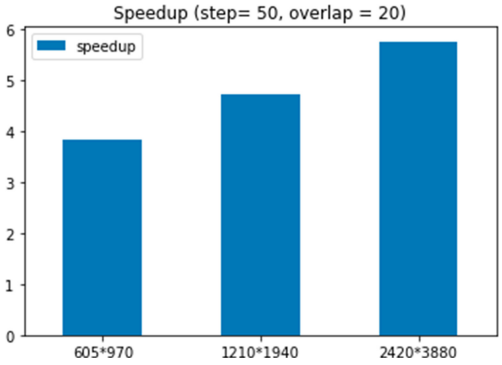


Fig. 5. Speedups obtained when increased data in both dimensions using step = 50, overlap = 20, 200 iterations and 12 processes.

970), (1210 × 1940) and (2420 × 3880). The overall parameters stay the same as before and the values of step and overlap were fixed on 50 and 20 respectively.

As can be seen the speedup increases when the dimension of the map increases. When the first dimension of the map is increased and the same number of processes is kept, the number of rows in each data strip increases. Keeping the same overlap value means less data redundancy when the size of each strip grows.

Finally, Fig. 6 shows the speedups obtained when the number of processes, p, varies from 2 to 20. The initial map with dimension (1210 × 1940) was used for a step 50 and overlap 20. The values of the remaining parameters were kept.

As can be seen from Fig. 6, the speedups increased steadily with the number of the available parallel processes to a value near of 5 times faster than the sequential implementation.

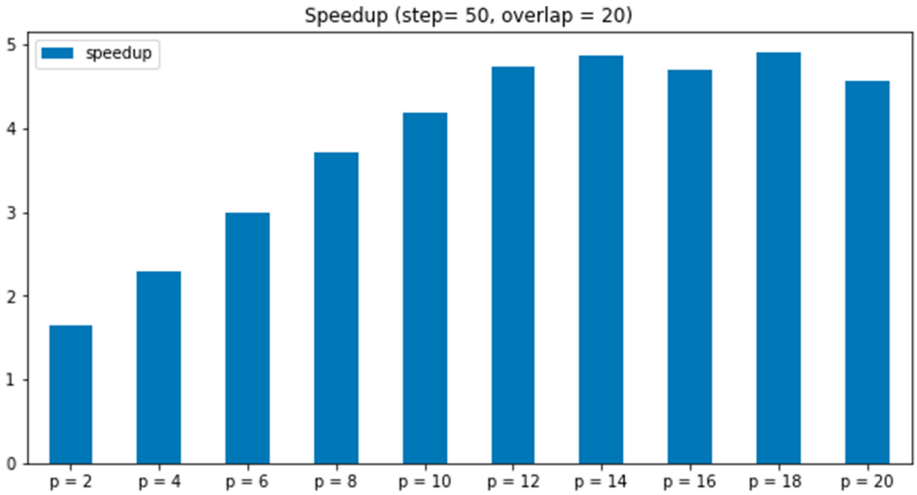
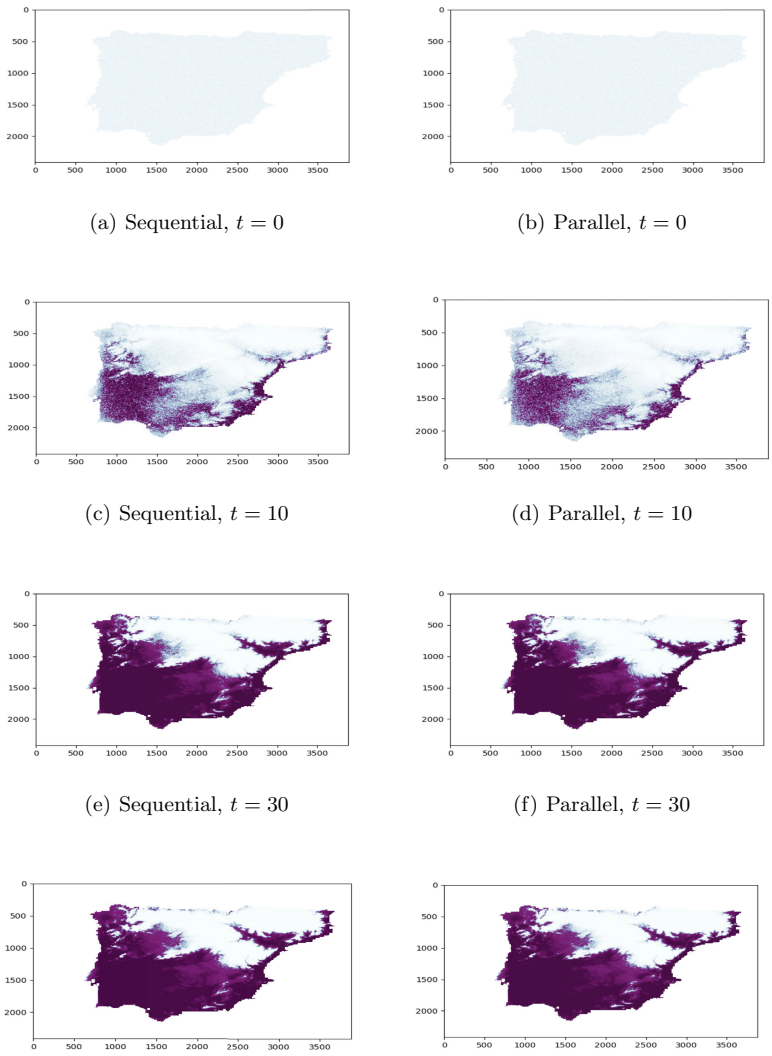


Fig. 6. Speedups obtained for the map dimension (1210 × 1940) when the number of processes, p, varies from 2 to 20, using step = 50, overlap = 20 and 200 iterations.

5 Discussion

The described parallel decomposition is able to preserve the absolute equality with the sequential evolution, provided that a widely enough border region is shared between contiguous stage-subsets. If the dimension of the border is less than the number of parallel inner-process steps, potentially the equality is broken.

Figure 7 shows the evolution of the sequential implementation side-by-side with the most different parallel evolution (step = 10; overlap = 4).



(a) Sequential, $t = 0$ (b) Parallel, $t = 0$

(c) Sequential, $t = 10$ (d) Parallel, $t = 10$

(e) Sequential, $t = 30$ (f) Parallel, $t = 30$

(g) Sequential, $t = 40$ (h) Parallel, $t = 40$

Fig. 7. Sequential evolution (on the left) and a non equal parallel evolution (on the right) with 10 inner-process steps and only 4 overlapping rows.

It worthwhile to notice that in Fig. 7 are depicted some snapshots of the evolution only at the initial steps, period where the major differences are accounted for. As a side note, notice that at plain sight it is very difficult to spot any differences even at this earliest stages. However the simulation runs until the convergence criteria (viz. the differences between consecutive states drop below $\tau = 1E^{-5} \times N_c$, see (1), for a number of epochs) is satisfied.

On the other hand, for the same relative level of overlap, given the local convergence characteristics of this particular model, any loss of equality is less noticeable when more inner-process steps are allowed. Thus if one is ready to relax the equality constraint for the parallel version, clearly it is possible to obtain a model presenting indistinguishable results from the sequential version while achieving at least a good level of speedup.

According to the performance results, the parallel implementation had a gain, with speedups of approximately 5. Different parameters' combinations of the parallel implementation constrain both the accuracy and the performance of the model.

6 Conclusion

This paper proposes a parallelization strategy of an agent-based model of spatial distribution of species aiming at a good trade-off between the synchronization requirements and the amount of data redundancy necessary to achieve the equality between the parallel and the sequential evolution.

The adopted strategy unveils the effects that the number of parallel evolutive steps and the size of overlap between stage-subsets have in the algorithm's performance. The relation between these two parameters is explored, in order to find which are the best parameter combinations that ensure increased speedups.

According to the empirical results there is a scale opportunity to model larger problems with almost negligible errors. It was found that due to the local convergence of the model (resulting from the use of the Moore neighbourhood in the decomposed stages) it is possible to attain almost indistinguishable results from the ones of the sequential version. This is verified even when the synchronizations are scarce and the overlap is kept at a parsimonious level, hence accomplishing significant performance gains.

Thus, a clear line of future developments is related with the scale-up of the implementation to a distributed setup in order to tackle higher cardinality problems.

However, for a sounder statistical evaluation and deeper inspection of the proposal, aiming at the generalization to another kinds of models (with different, non-regular types of iterations between stage-subsets), the performance of the algorithm must be assessed over averaged metrics for multiple runs in a set of carefully chosen descriptive parameters.

Acknowledgements. This work was supported by operation Centro-01-0145-FEDER-000019 - C4 - Centro de Competências em Cloud Computing, cofinanced by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio à Investigação Científica e Tecnológica - Programas Integrados de IC&DT. This work was also funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020.

References

1. Abar, S., Theodoropoulos, G.K., Lemarinier, P., O'Hare, G.M.: Agent based modelling and simulation tools: a review of the state-of-art software. Comput. Sci. Rev. **24**, 13–33 (2017)
2. An, L., Grimm, V., Turner II, B.L.: Meeting grand challenges in agent-based models. J. Artif. Soc. Soc. Simul. **23**(1), (2020)
3. Bandman, O.: Coarse-grained parallelization of cellular-automata simulation algorithms. In: Malyshkin, V. (ed.) PaCT 2007. LNCS, vol. 4671, pp. 370–384. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73940-1_38
4. Bioco, J., Fazendeiro, P., Cánovas, F., Prata, P.: Parameterization of an agent-based model of spatial distribution of species. In: Miraz, M.H., Excell, P.S., Ware, A., Soomro, S., Ali, M. (eds.) iCETiC 2020. LNICST, vol. 332, pp. 251–260. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60036-5_18
5. Breckling, B., Müller, F., Reuter, H., Hölker, F., Fränzle, O.: Emergent properties in individual-based ecological models-introducing case studies in an ecosystem research context. Ecol. Model. **186**(4), 376–388 (2005)
6. Chimeh, M.K., Heywood, P., Pennisi, M., Pappalardo, F., Richmond, P.: Parallel pair-wise interaction for multi-agent immune systems modelling. In: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1367–1373. IEEE (2018)
7. Chimeh, M.K., Heywood, P., Pennisi, M., Pappalardo, F., Richmond, P.: Parallelisation strategies for agent based simulation of immune systems. BMC Bioinf. **20**(6), 1–14 (2019)
8. Coakley, S., Gheorghe, M., Holcombe, M., Chin, S., Worth, D., Greenough, C.: Exploitation of high performance computing in the flame agent-based simulation framework. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, pp. 538–545. IEEE (2012)
9. Collier, N., North, M.: Parallel agent-based simulation with repast for high performance computing. Simulation **89**(10), 1215–1235 (2013)
10. Collier, N., Ozik, J., Macal, C.M.: Large-scale agent-based modeling with repast HPC: a case study in parallelizing an agent-based model. In: Euro-Par 2015. LNCS, vol. 9523, pp. 454–465. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27308-2_37
11. DeAngelis, D.L., Grimm, V.: Individual-based models in ecology after four decades. F1000Prime Rep **6**(39), 6 (2014)
12. DeAngelis, D.L., Gross, L.J., et al.: Individual-Based Models and Approaches in Ecology. Chapman & Hall, London (1992)
13. Fachada, N., Lopes, V.V., Martins, R.C., Rosa, A.C.: Parallelization strategies for spatial agent-based models. Int. J. Parallel Prog. **45**(3), 449–481 (2017)

14. Grimm, V.: Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future? Ecol. Model. **115**(2), 129–148 (1999)
15. Grimm, V., et al.: A standard protocol for describing individual-based and agent-based models. Ecol. Model. **198**(1–2), 115–126 (2006)
16. Grimm, V., et al.: The odd protocol for describing agent-based and other simulation models: a second update to improve clarity, replication, and structural realism. J. Artif. Soc. Soc. Simul. **23**(2) (2020)
17. Heywood, P., et al.: Data-parallel agent-based microscopic road network simulation using graphics processing units. Simul. Model. Pract. Theor. **83**, 188–200 (2018)
18. Huston, M., DeAngelis, D., Post, W.: New computer models unify ecological theory: computer simulations show that many ecological patterns can be explained by interactions among individual organisms. BioScience **38**(10), 682–691 (1988)
19. Lomnicki, A.: Population ecology of individuals. Monogr. Popul. Biol. **25**, 1–216 (1987)
20. Macal, C.M.: Everything you need to know about agent-based modelling and simulation. J. Simul. **10**(2), 144–156 (2016)
21. Moore, E.F.: Machine models of self-reproduction. In: Proceedings of symposia in applied mathematics. vol. 14, pp. 17–33. American Mathematical Society New York (1962)
22. Parry, H.R., Bithell, M.: Large Scale Agent-Based Modelling: A Review and Guidelines for Model Scaling. In: Heppenstall, A., Crooks, A., See, L., Batty, M. (eds.) Agent-Based Models of Geographical Systems. pp. 271-308. Springer, Dordrecht (2012) https://doi.org/10.1007/978-90-481-8927-4_14
23. Suárez-Muñoz, M., Bonet-García, F., Hódar, J.A., Herrero, J., Tanase, M., Torres-Muros, L.: Instar: an agent-based model that integrates existing knowledge to simulate the population dynamics of a forest pest. Ecol. Model. **411**, 108764 (2019)
24. Voss, A., et al.: Scalable social simulation: investigating population-scale phenomena using commodity computing. In: 2010 IEEE Sixth International Conference on e-Science, pp. 1–8. IEEE (2010)
25. Williams, R.A.: User experiences using flame: a case study modelling conflict in large enterprise system implementations. Simul. Model. Pract. Theor. **106**, 102196 (2021)
26. Wong, W.W.L., Feng, Z.Z., Thein, H.H.: A parallel sliding region algorithm to make agent-based modeling possible for a large-scale simulation: modeling hepatitis c epidemics in canada. IEEE J. Biomed. Health Inf. **20**(6), 1538–1544 (2016). https://doi.org/10.1109/JBHI.2015.2471804. Nov
27. Zhang, Q., Vatsavai, R.R., Shashidharan, A., Berkel, D.V.: Agent based urban growth modeling framework on apache spark. In: Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, pp. 50–59 (2016)