Appl Intell (2016) 45:904–922
DOI 10.1007/s10489-016-0797-8

MASE-BDI: agent-based simulator for environmental land change with efficient and parallel auto-tuning

Cássio G. C. Coelho[1] · Carolina G. Abreu[1] · Rafael M. Ramos[1] · Aldo H. D. Mendes[1] · George Teodoro[1] · Célia G. Ralha[1]

Published online: 4 June 2016

Abstract This paper presents an agent-based simulator for environmental land change that includes efficient and parallel auto-tuning. This simulator extends the **M**ulti-**A**gent **S**ystem for **E**nvironmental simulation (MASE) by introducing rationality to agents using a mentalistic approach—the Belief-Desire-Intention (BDI) model—and is thus named MASE-BDI. Because the manual tuning of simulation parameters is an error-prone, labour and computing intensive task, an auto-tuning approach with efficient multi-objective optimization algorithms is also introduced. Further, parallelization techniques are employed to speed up the auto-tuning process by deploying it in parallel systems. The MASE-BDI is compared to the MASE using the Brazilian Cerrado biome case. The MASE-BDI reduces the simulation execution times by at least 82× and slightly improves the simulation quality. The auto-tuning algorithms, by evaluating less than 0.00115 % of a search space with 6 million parameter combinations, are able to quickly tune the simulation model, regardless of the objective used. Moreover, the experimental results show that executing the tuning in parallel leads to speedups of approximately 11× compared to sequential execution in a hardware setting with 16-CPU cores.

✉ Célia G. Ralha
ghedini@unb.br

[1] Computer Science Department, Institute of Exact Sciences, University of Brasília, P.O. Box 4466, Zip Code 70.904-970 Brasília, DF, Brazil

Keywords Belief-desire-intention · Environmental simulation tool · Land-use and cover change · Multi-agent system

1 Introduction

The establishment of modelling and simulation frameworks that perform land use and land cover change (or land change)—which allow humans to include environmental information in their decision-making processes—is becoming increasingly important. Land change is a term associated with the human modification of earth's terrestrial surface [1], that is one of the most intense causes of earth's recent alterations [2]. The intensification of land modification processes results in ecosystem changes at global scales, bringing negative consequences to the environment including biodiversity loss, climate change, deforestation, water, soil and air pollution, among other factors that impact human populations today [3].

Due to the large number of factors involved in land changes, any environmental modelling software is necessarily complex, involving several parameters related to many individual interactions. These software frameworks apply different techniques and technologies that include remote sensing, geospatial and image analysis along with an interdisciplinary assortment of natural and social scientific methods, as discussed in Section 3.1. Nevertheless, to the best of our knowledge, there is no agent-based environmental simulation tool to represent these complex individual interactions using rational agents, where each individual agent's decisions can vary based on its beliefs, desires and intentions. This representation affects agents' actions, thus

creating emerging behaviours through group interactions. Another novel aspect of this paper is the use of auto-tuning techniques in parallel and distributed machines to improve the MASE-BDI results.

Thus, in this paper, we present an extension to the **M**ulti-**A**gent **S**ystem for **E**nvironmental simulation (MASE),[1] which employs the **B**elief-**D**esire-**I**ntention (BDI) model to add rationality to agents through a mentalistic approach and is thus named MASE-BDI. According to [4], agent-based simulation platforms can represent the complexity of individual interactions through the general rule for ecological applications, a feature that is important for successfully supporting decision-making processes. Towards this goal, improvements in the agent's reasoning process amplify the agents' autonomy and capabilities for reasoning more independently about their actions.

The MASE-BDI also addresses the problem of automatically tuning the simulation model parameters, which is a complex, labour-intensive, and error-prone task. The MASE-BDI performs this task by employing efficient optimization algorithms to tune the simulation model parameters with respect to a user-defined single- or multi-objective function of interest. Because the tuning process is compute intensive (it requires executing the simulation several times), we have also proposed and implemented the ability to leverage parallel systems to speed up the tuning process. All our propositions are evaluated using the real-world Brazilian Cerrado biome land change case study.

The rest of the paper is organized as follows: Section 2 presents a short background overview of the BDI model and auto-tuning algorithms. Section 3 discusses related approaches, including environmental simulation tools and auto-tuning solutions. The MASE-BDI simulator, with its implemented architecture, a description of the agents and their reasoning, and the MASE-BDI auto-tuning module with parallel auto-tuning are presented in Section 4, and the Brazilian Cerrado case study used to demonstrate and evaluate the MASE-BDI simulator capabilities and the automatic tuning of the simulation settings is introduced in Section 5. The experiments and results, including MASE-BDI vs. MASE sequential executions and the MASE-BDI auto-tuning with distributed and parallel performance comparisons, are covered in Section 6. Finally, Section 7 discusses conclusions and future work.

2 Background

Artificial Intelligence (AI) concepts, methods, and techniques have been successfully applied to numerous domains. This paper focuses on the use of AI to support human decision-making processes in an environmental simulation context, where individuals' behaviour must be considered. To do this, we applied a multi-agent system (MAS) to integrate the individual interactions implemented in distributed and high-performance computational environments. Because the environmental simulation frameworks implement complex models with several parameters, we also applied tuning algorithms to adjust the simulation parameters with respect to user-defined goals. Thus, in this section, we provide a short background on the BDI model and on auto-tuning algorithms.

2.1 BDI model

The basis for implementing agents with rationality and mentalistic notions is to describe the behaviour of individuals. One of the most influential theories with respect to agent rationality is the BDI model conceived by [5] as a theory of human practical reasoning [6]. Practical reasoning is reasoning towards action and includes first deciding *what* state of affairs one wants to achieve, also known as deliberation, followed by deciding *how* to achieve this state of affairs, called means-end reasoning [7].

The main aspects of a BDI model are summarized in [8]. Beliefs represent the information an agent has about both the world it inhabits and its own internal state. A belief provides a domain-dependent abstraction of entities and affects the way an agent perceives and thinks about the world. Desires represent the motivational attitudes of agents, capturing an agent's wishes and driving the course of its actions. Nevertheless, an agent may have a conflicting set of desires. A goal deliberation process must select a subset of consistent desires that represents the states to be achieved or maintained and, therefore, the reasons why actions are executed. The concept of achieving a goal allows the modelling of agents that are not purely reactive but that exhibit proactive behaviour. For instance, a desire can be triggered by the occurrence of a set of events.

Plans are the means by which agents achieve their goals and react to occurring events. When an agent has decided on pursuing a goal with a certain plan, it commits itself (momentarily) to the accomplishment of that goal and, hence, has established an intention towards a sequence of plan actions. A well-established reasoning mechanism is required to determine the set of plans that are necessary to satisfy a certain goal and to foresee the consequences if a plan is not successful. Plans can be either abstract or quite concrete when composed of basic actions.

2.2 Auto-tuning algorithms

The process of tuning parameters is iterative and typically consists of running the simulator with a specific set of

[1] Visit the MASE website: http://mase.cic.unb.br

parameter values, checking for the quality of the simulation results, modifying the parameter values, and repeating this process until an acceptable result level is attained. The tuning process is time-consuming because the search space formed by the combination of parameter values involved in a simulation model may lead to millions of possible parameter combinations (approximately 6 million in our model). To alleviate this problem, auto-tuning algorithms can automate the tuning process and minimize the number of parameter sets that must be tested.

In this work, auto-tuning algorithms are implemented using Active Harmony (AH) [9], which was primarily designed and employed for tuning parameters of high-performance applications and kernels with the goal of maximizing performance. Therefore, we make a novel use of AH for searching parameter space to improve our application's quality analysis results. The AH framework includes two efficient optimization algorithms and both are integrated into the MASE-BDI simulator: *Nelder-Mead simplex (NM)* [10] and *Parallel Rank Order (PRO)* [11]. The NM and PRO algorithms try to minimize (or maximize by negation) an unknown function by probing and exploring the parameter search space. One motivating characteristic for using AH and its tuning algorithms is that they account for variability in the execution output for the same input parameter. This is essential for the MASE-BDI because multi-agent based simulators have intrinsic uncertainty levels, which leads to different results when executing a simulation multiple times with the same input. The characteristic of uncertainty is fundamental in models of rational agents, where the decision of each individual can differ based on their beliefs, the factors that affect their actions, and the creation of emerging behaviours through interactions of individuals in a group.

The NM algorithm uses a simplex or polytope of $k + 1$ vertices in k-dimensional search space. This simplex is updated in each iteration of the method by removing the vertex with the worst value (v_r) and replacing it with a new vertex that has a lower function value. This operation involves computing the centroid c of the remaining simplex vertices to replace v_r with a point on line $v_r + \alpha(c - v_r)$. Typical α values are 2, 3 and 0.5. The values of α define whether the transformation on the simplex is a reflection ($\alpha = 2$), an expansion ($\alpha = 3$), or a contraction ($\alpha = 0.5$). The NM method usually performs a reflection first, which may, depending on the results, be followed by an expansion or contraction. The original method has been modified in AH to deal with non-continuous search spaces.

The PRO algorithm uses a set of K points from a simplex ($K \geq N + 1$) for a N-dimensional space. Each iteration of the algorithm calculates up to $K - 1$ new vertices, which are computed by a reflection, expansion, and shrinking of the simplex around its vertex with the optimal value. Multiple vertices generated during each iteration may be evaluated in parallel. The reflection step succeeds if at least one of the evaluated vertices lead to an improvement of the optimization results. If no point succeeds during reflection, the simplex shrinks around the best vertex. The expansion check follows a successful reflection and is executed to accept or reject the simplex. The simplex is expanded when accepted, and then the search continues with a new iteration. The algorithm stops when it converges to a point in the search space or after a predetermined number of iterations have been executed. Figure 1 exemplifies the simplex transformations in a 4-point simplex with a 2D space. The PRO algorithm has better convergence guarantees than the NM method because the latter may converge to a degenerate simplex [11].

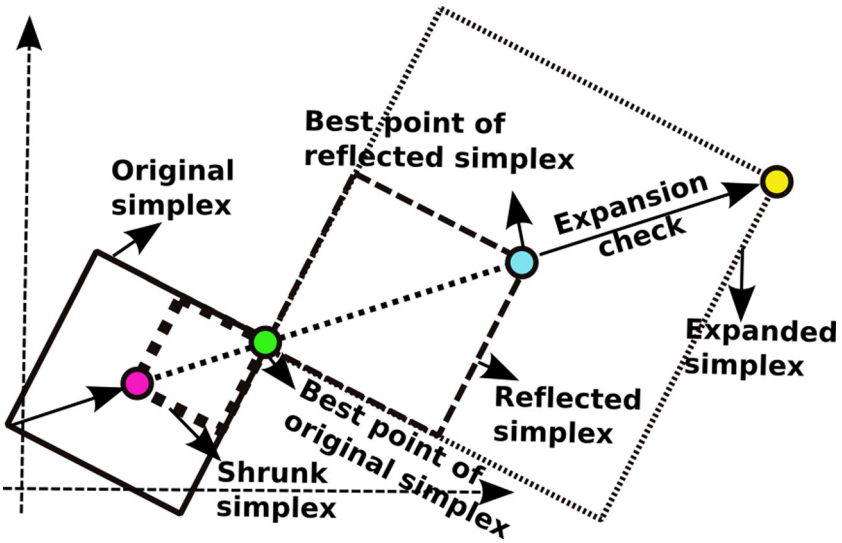

Fig. 1 Changes in a 4-point simplex (2D space)

3 Related work

Although the use of MAS in land change simulators is a valuable approach, it has not been well explored in the literature, especially considering agents' rationality. This is presented in more detail in Section 3.1, where we summarize and compare the features of well-known frameworks. In addition, auto-tuning techniques have been successfully applied to optimization problems, but not to environmental simulators as discussed in Section 3.2. Thus, in this section, we present related approaches to emphasize the main pros and cons of implementing efficient and parallel auto-tuning with the MASE-BDI simulator.

3.1 Environmental simulation tools

The development of environmental simulation tools, especially those that address land change problems, has been approached from different perspectives. The tools differ in methods, modelling techniques and model complexity. We briefly compare three very well-known frameworks by considering their different methods and modelling approaches (Table 1):

- IDRISI is a framework for GIS analysis, image processing, and modelling that was developed by Clark Labs at Clark University [12]. IDRISI is a PC grid-based system that offers a modelling tool for researchers and scientists engaged in monitoring and modelling Earth systems and includes tools for surface and statistical analysis, decision support, land change and prediction, and image time series analysis. The IDRISI Macro-Modeler provides a general-use graphical modelling environment for executing multi-step models and simulations. Various methods can be used in the models, for example, CA-MARKOV, which uses Markov chain matrices to determine the quantity of change along with suitability maps and cellular automata to spatially allocate change. Another possibility is the Land Change Modeler (LCM), a suite of tools in which land change analysis can be combined with other environmental issues to provide ways to assess and forecast changes in land cover to understand their implications for habitat and species biodiversity.
- ArcGIS is a commercial GIS (Geographic Information System) developed by Esri that enables one to visualize, examine, analyse and interpret data to reveal relationships, patterns, and trends. ArcGIS is one of the most popular frameworks for image processing, GIS analysis and modelling [13]. ArcGIS's ModelBuilder technology allows users to model, store, automate, publish, and run complex operations and workflows. The ModelBuilder can be used as an application that allows users to perform exploratory project work, or as an application for building generic tools that can be reused and shared. Using ModelBuilder, users can create models that chain tools together, using the output of one tool as the input to another tool.
- DinamicaEGO (**E**nvironment for **G**eoprocessing **O**bjects) is a freeware platform for environmental modelling developed at Federal University of Minas Gerais [14]. DinamicaEGO uses transition probability maps

Table 1 Overview of the general characteristics of each framework

Name	Model type	Modules	What it explains	Strengths	Weakness
IDRISI	Markov chain models	Markov_CA + LCM	Spatial patterns	LCM	Proprietary software
ArcGIS	Discrete finite state model	Several subroutines for different tasks	Spatial patterns	Image processing + GIS analysis	Proprietary software
Dinamica EGO	Spatial dynamic model	Several subroutines for different tasks	Environmental dynamics	User friendly + graphical user interface	Limited human decision-making
MASE	Spatial simulation	MAS to land change	Land change dynamics	Agents autonomy	Graphical user interface
MASE-BDI	Spatial simulation	MAS to land change + auto-tuning	Land Change dynamics	Agents autonomy + rationality	Graphical user interface

based on the weight of evidence and genetic algorithm methods. The transition probability maps simulate the spatial patterns of changes in the landscape using both Markov chain matrices to determine the quantity of change and a cellular automata approach to reproduce spatial patterns. DinamicaEGO is a generic modelling tool that enables the design and implementation of simulation models for environmental management.

– The MASE is a multi-agent based modelling and simulation tool for land change dynamics. A complete theoretical and methodological description is available in [15]. The MASE-BDI extends The MASE agents with rationality through the implementation of the BDI model. The MASE-BDI also implements an efficient and parallel auto-tuning module. Both versions of the framework allow multiple types of agents with different behaviours to represent the interactions and relations between agents and the physical environment. The MAS approach qualifies the flexible interaction of agents to integrate levels of the complexity of human decisions in typically spatially explicit models in the context of land change. Furthermore, the use of agents is well suited to solving complex tasks, which are processed into levels with actions performed in parallel or sequentially.

A comparison of the MASE and MASE-BDI to these three well-known frameworks is presented in Table 1. The comparison shows that the MASE is the only multi-agent based simulator. Nevertheless, there are other simulation frameworks that use agent-based approaches, such as Netlogo [16, 17], RePast [18] and CORMAS [19]. These tools implement simple agent approaches (purely reactive) that focus on understanding the coordination or relations of agents to the environment, but they do not involve agents' rationality. But the design of MAS includes not only properties for assigning reactivity, pro-activeness and social ability to agents, but also more complex aspects such as adaptability, learning and rationality through mentalistic notions (e.g., belief, desire, intention, obligation, choice) [7, 20].

The MASE and the MASE-BDI are also distinct from popular peer-reviewed land change models for the amount of pixels they correctly predict in land change using well-established statistical evaluation methods proposed by [21]. In [21] various land change applications were summarized and compared using two statistics: the null resolution and the Figure of Merit (FoM). Considering the FoM, the more accurate applications were the ones where the amount of observed net change in the reference maps is larger. The simulation model is considered good when its FoM is equal to or greater than 50—that value means that the amount of correctly predicted change is larger than the sum of the various types of errors. As stated by [21], twelve of the thirteen land change modelling applications in this paper's comparison contain more erroneous pixels than pixels of correctly predicted land change at fine resolutions of the raw data. The MASE and the MASE-BDI were able to surpass these statistics, presenting results that show high quality in the accuracy of their predictions. The complete explanation of the MASE simulation results using Pontius' statistical techniques of map comparison to land change models is presented in [15]. The MASE-BDI results are also discussed more detail in Section 6.

3.2 Auto-tuning systems and solutions

Tuning techniques have been successfully applied to optimization problems in a large spectrum of systems. Large-scale dynamic and open systems, with communication delays due to the interaction of many artefacts, automated and wireless devices, network access bandwidth, load control, and information management flows with distributed process control algorithms are natural foci for the application of auto-tuning. The success of auto-tuning in these problems and areas has motivated the development of a number of tuning techniques and systems that offer a diversity of tuning algorithms. Examples of auto-tuning systems and solutions include the Dakota toolkit [22], OKSI [23], AH [24], PATUS [25], and OpenTuner [26], among others.

In the MAS scenario of a naturally distributed application with a large exchange of information between agents, studying and applying different tuning strategies to achieve a totally decentralized fashion of stability formation holds many challenges [27–30]. In addition, real-world problems using distributed multi-agent based simulation models (allied to AI techniques) raise new aspects in the study of auto-tuning applications in diverse areas such as traffic control [31–33], automotive development process [34], environmental simulations—hydrological [35] and virtual air pollution monitoring stations in urban areas [36].

In [33] and [30], for instance, auto-tuning optimizations based on a genetic algorithm and a particle swarm optimization algorithm were employed to tune the parameter of a fuzzy logic controller model that drives the actions of a certain type of agent to improve the simulation quality. In our work, we apply auto-tuning to an agent-based system used in another problem: land change simulation. Additionally, we proposed tuning the system as a black-box, without exporting any information of the model being tuned to the auto-tuning algorithm. This strategy creates a more extensible solution that can be reused in other models and problems. Further, we perform auto-tuning with objective functions that may be defined by the user according to the goal of interest. We also support multi-objective optimizations that are critical in many contexts. Moreover, we

propose and implement an efficient execution strategy that makes use of parallel computing and distributed systems to accelerate the auto-tuning process, which is novel in the MAS area.

4 MASE-BDI simulator

In this section, the MASE-BDI simulator is described from two complementary directions: Section 4.1 describes the architecture, the agents, and the transformation agent (TA) reasoning process, and Section 4.2 presents the MASE-BDI auto-tuning process with distributed and parallel capabilities.

4.1 Architecture

The BDI model views an agent as a goal-directed entity that acts in a rational manner. The MASE is implemented in the **J**ava **A**gent **DE**velopment framework (JADE) [37], and the MASE-BDI is implemented in Jadex [8]. Jadex was chosen because it implements a BDI-infrastructure for JADE agents, keeping all the previous defined features. Figure 2 presents the MASE-BDI architecture, which is composed of three layers: the user interface layer, the utility layer, and the agent layer:

The user interface layer is subdivided into two modules: the Jadex Control Centre (JCC) and the Graphical Interface

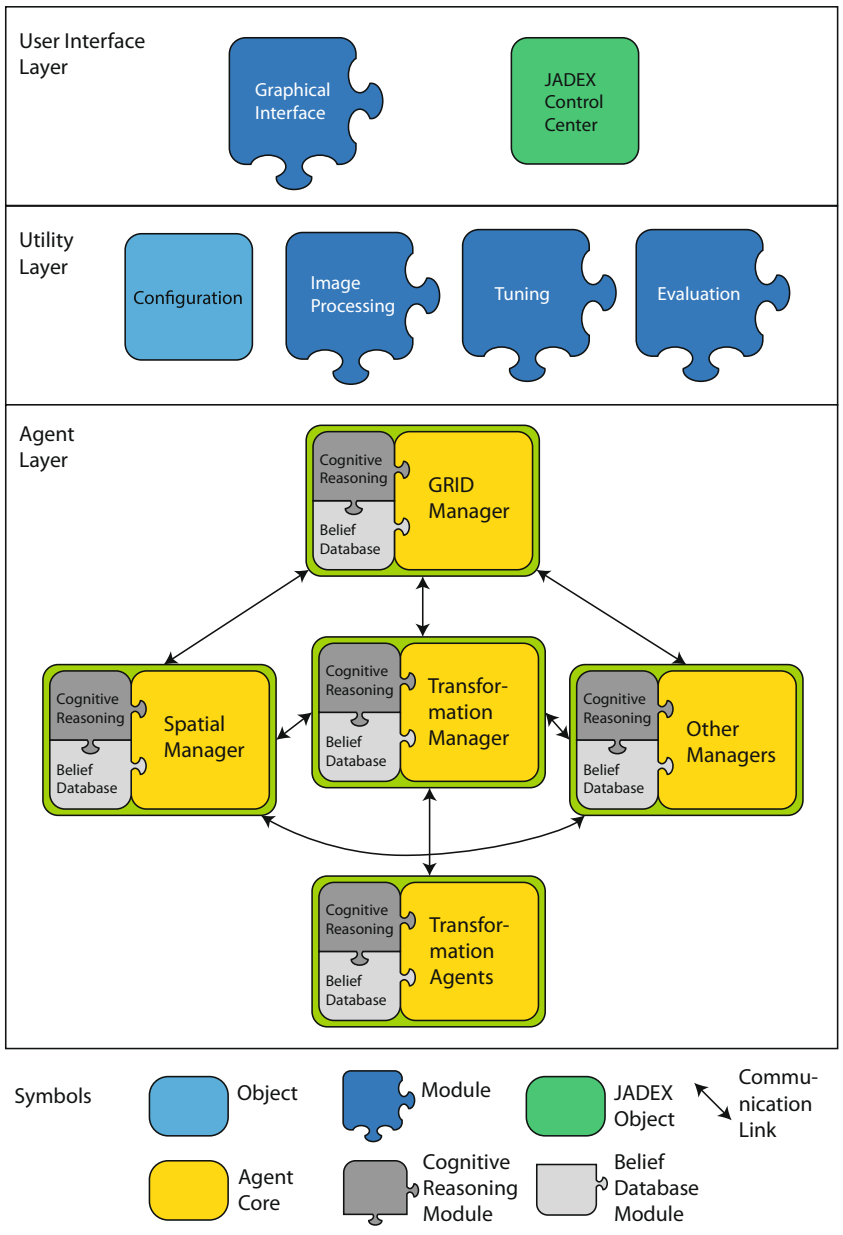

Fig. 2 The MASE-BDI architecture

(GI). The JCC is part of the Jadex framework; it provides tools for programming and debugging purposes. The GI is the module designed for users of the MASE-BDI system; it provides visualization and controls for the simulation model in relation to variable settings. By using the GI, it is possible to change the model attributes, including the number of agents and the number of simulation steps. It is also possible to pause or restart the simulation at any point during execution. Sometimes it is desirable to run the MASE-BDI in a console-only environment or users simply want to run the model simulation to obtain the results without changing its attributes through the GI module. When GI is not required, the model attributes can be defined directly in a configuration file (with the simulation settings), or passed as parameters when the application is invoked.

The utility layer implements several activities related to the simulation such as the configuration and evaluation of the simulation, the tuning of parameters and the preprocessing of input images and maps. User-defined settings need to be interpreted and processed before the simulation starts to be available for use by the MASE-BDI agents. Also, because the input can involve large data files, depending on the model attributes and size defined, the MASE-BDI already stores some preprocessed images in serialized files but deserializes them as needed. When the simulation execution reaches the predefined number of steps, the final simulation state is presented via a map. This map can then be compared to real maps of the environment to compute the correctly simulated scores. These tasks are accomplished by the evaluation and the image processing modules. After the user has entered the simulation settings, the configuration module checks whether the current settings match the ones in the serialized files. If so, this module deserializes them, otherwise, it evaluates the images provided by the user. After this process, the information is made available by the configuration module to the to-be-created agents in the simulation, using a common memory space. The image processing module is activated when the settings and images entered by the user do not match the stored files. In this case, the image processing deletes the old stored files, reads the new images and variables, treats them with the simulation rules, and makes the information available to the configuration module. It also serializes these new versions of processed images to be used in the future by the MASE-BDI system. The image processing module is called again at the end of simulation to convert the final simulated space into images and to provide information to the evaluation module. The evaluation module compares the final state of the simulation with the real representation of the environment to measure the similarities and discrepancies between them for correct scoring. This layer also hosts the tuning module, which is a configurable module that seeks to optimize configurations for the simulation with respect to some objective. The tuning module is described in detail in Section 4.2.

Agent layer is the bottom layer of the architecture, where agents are implemented. Agents are represented hierarchically based on their tasks in the simulation and their domains of influence. The agent at the top of the hierarchy is the GRID Manager (GGRID). This agent is created in the Jadex platform as soon as the configuration module completes. The GGRID's main purpose is to set up and coordinate the simulation. When the GGRID starts executing, it performs two tasks: (i) instantiate the Transformation Manager (TM), the Spatial Managers (SM), and the Transformation Agents (TAs); and (ii) setup the TM goals. When these tasks complete successfully, they return a success flag to the GGRID. After instantiating the TM, the SM, and the TAs, the GGRID runs the simulation. During this process, it is responsible for (i) receiving and processing requests (e.g., pausing or restarting the simulation) and (ii) counting the number of steps of the simulation. The step counting is tied to the work of the TAs. TAs are asynchronous agents, but are configured to do a certain number of transformations per step. The GGRID starts the steps, the TAs do their jobs, report their successes, and wait. When all of the TAs have reported success, the GGRID ends the step, increases the step counter and verifies whether the user-preconfigured number of steps has been reached. If so, a kill signal is sent to all managers and agents, and the simulation evaluations are carried out; otherwise, the next simulation step is started.

4.1.1 Agents' description

The TM is a management object responsible for setting up the TAs and solving their conflicts. The TM instantiates the TAs as its first goal, passing them an initial position. These positions are previously retrieved from the SM. The SM decides what the best positions are based on the policy (land-use rules) defined for the simulation. During simulation execution, the TM is responsible for conflict resolution, which occurs when two or more neighboring TAs are disputing over the land use. When such a conflict occurs, the TM decides which agent gets the desired location and reallocates the others. Thus, the SM manages the simulation spatial resources, providing the TM with new positions for the TAs.

The TAs move and explore the environment by rules predefined in the simulation modeler. When they are instantiated by the TM, they wait for the GGRID to start the simulation. TAs have two goals: explore (land-use) and move (backup objective if the first fails). When a TA tries to achieve its explore goal, it first checks whether the

area where it is currently located has sufficient exploration potential. If so, it uses the area's potential; otherwise, the explore objective fails. In this case, the TA moves to an adjacent location and explores that. When two or more TAs want to move to the same location (grid cell), they are in conflict and report to the TM, who solves the conflict by moving both TAs to nonclashing areas. Then, they restart their cycle to try to explore or move. The TA reasoning process is further explained in the following section.

4.1.2 Agents' reasoning

The agents' rationality in the MASE-BDI simulator is based on the mentalistic BDI model. The agents' internal reasoning process can be explained by describing how the practical reasoning interpreter works. Figure 3, based on the conceptual foundations of Jadex [38], illustrates the main elements of the practical reasoning module, which is composed of the means-end reasoning phase (handle internal events) and the goal deliberation phase (handle deliberation situations). The internal events are generated by agents' beliefs (condition events), goals (goal events) and plans (plan events). The goal deliberation module selects the new goals (restricted by agents' capabilities) based on agents' beliefs. The beliefs module imposes goal conditions to the goals module, i.e., a goal can automatically fail if a specific belief is false or does not exist. According to external events (incoming messages), the means-end reasoning module selects the agents' plans through the plans module. The plans module can read and write facts dynamically to the agents' belief modules. It can also dispatch sub-goals to the agent's goals module and send outgoing messages to other agents.

This reasoning engine is identical within each class of agents throughout the simulation execution. When the GGRID starts the simulation or any specific simulation step, it sends an incoming message to start TA execution. The TA execution cycle is presented in Fig. 4. The TA message receiver generates an event from this message (the internal events of Fig. 3). In this case, the receiving agent's first objective is triggered: explore the current area. The dispatcher selects the best plan to be executed at the moment from a repertoire of possible plans based on the current objective. This selection considers the nature of the event and the current beliefs and, finally, returns the best plan to be executed (made in the meta-level reasoning module). After the returned plan is chosen, its actions are stacked in a 'ready list' to be executed by the agent's scheduler.

If the objective is to explore the area, the agent will try to explore the land's full potential. If the agent explores it successfully, it will report success and end its actions for that step, generating both an internal and an external event. The internal event will update its own belief, and the

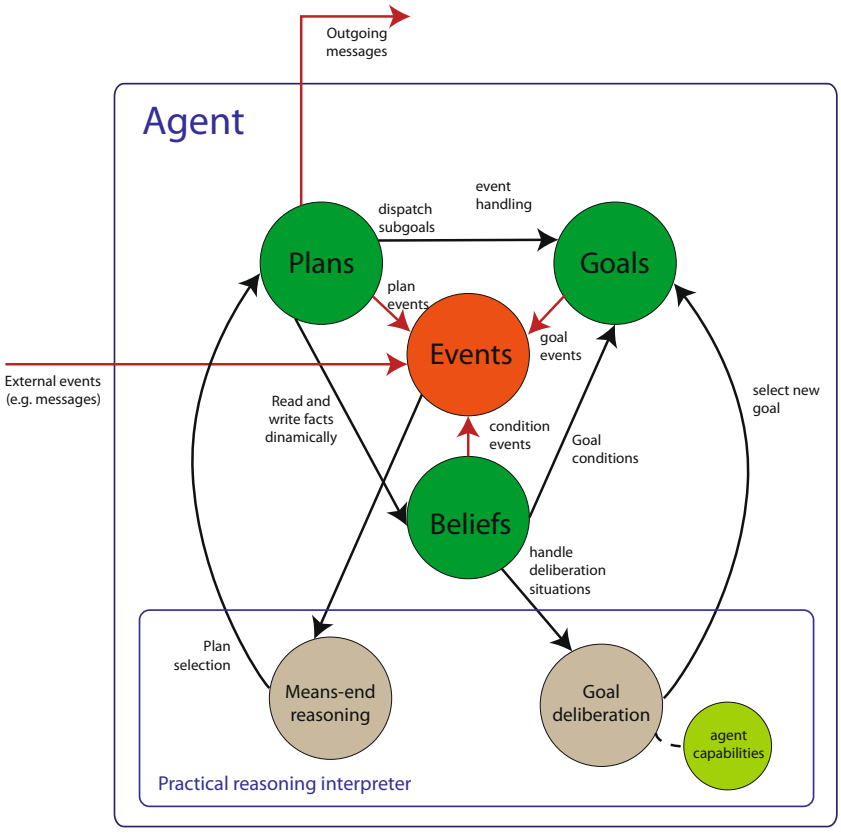

Fig. 3 Agents' internal abstract architecture

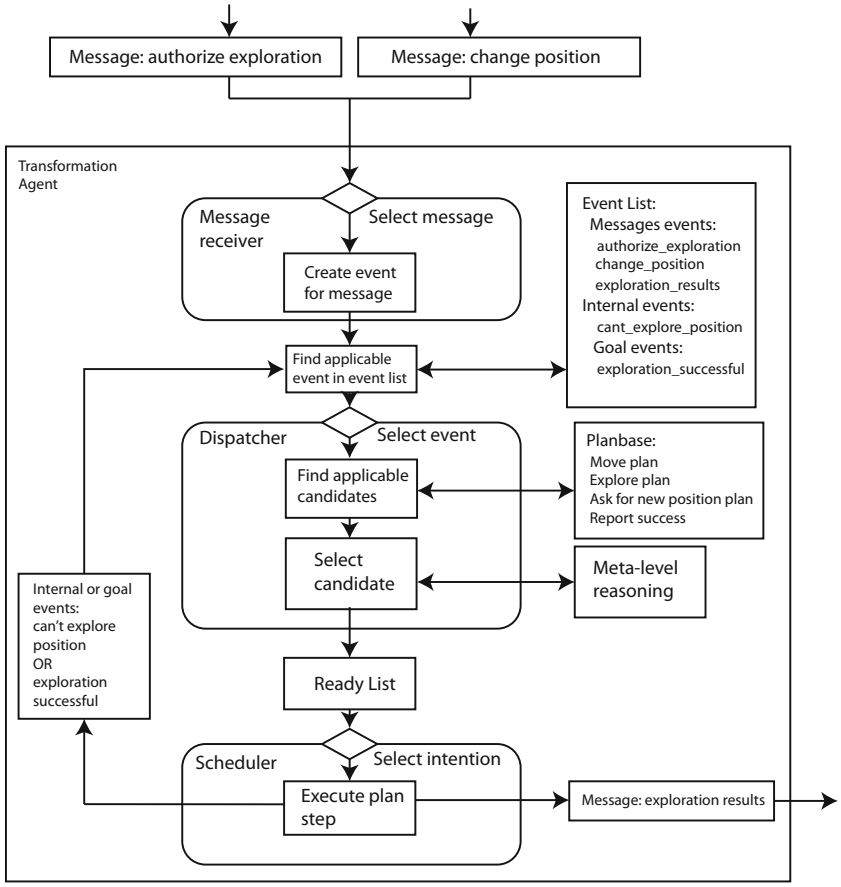

Fig. 4 Transformation agent execution cycle

external event will notify the manager (TM) about the exploration that was executed. However, if the area's potential is depleted but the agent still has actions in its scheduler, the explore goal will fail. In this case, only one internal event will be generated, updating the current objective to 'move' instead of to 'explore'. The agent's dispatcher will list the possible plans for handling that objective, and the meta-level reasoning module is once again activated. The actions for the chosen plan are stacked, and each is tried. If there is no nearby agent considering moving to the same location as the one desired by the agent, the move is successful, and the agent can change its location and finish its remaining actions for the explore goal. If another agent is considering a move to the possible locations, the agents are in conflict, and they report to the TM.

When in conflict, the TAs generate an internal event to update their own belief database. The conflict state is perceived by the TM, who asks for new locations from the SM. After the SM informs the TM of new areas, the TM chooses one of the conflicting agents to take the desired location and re-allocates the others. This is performed by sending a message to the conflicting TAs to change position, informing each which new position it should move to. This message is again transformed into an external event by the agent's message receiver, and its plans to handle the event are considered by the dispatcher. Because the message already indicates which position the TA should take, the dispatcher simply chooses the plan that moves the agent to the determined area. The TAs then move and continue with the rest of their actions.

4.2 MASE-BDI auto-tuning

The auto-tuning process is illustrated in Fig. 5. The figure shows the interactions of the three MASE-BDI architectural layers in the auto-tuning execution. In this process, users interact with the Graphical Interface to provide typical simulation rules and inform the simulation of the ranges of the parameter values to be tuned (1). In the next step (2), the auto-tuning module receives the information about the parameters to be varied and their range values as input. One of the tuning algorithms (presented in Section 2.2) from the auto-tuning module is used to select a value for each simulation parameter being tuned. This information is forwarded to the Configuration component, which outputs a configuration file with all the data needed to execute the simulation (3). The MASE-BDI reads the configuration information (4), and the agent layer is invoked to run the simulation. The output of the simulation is then evaluated for accuracy or quality by the evaluation module (5), using the FoM

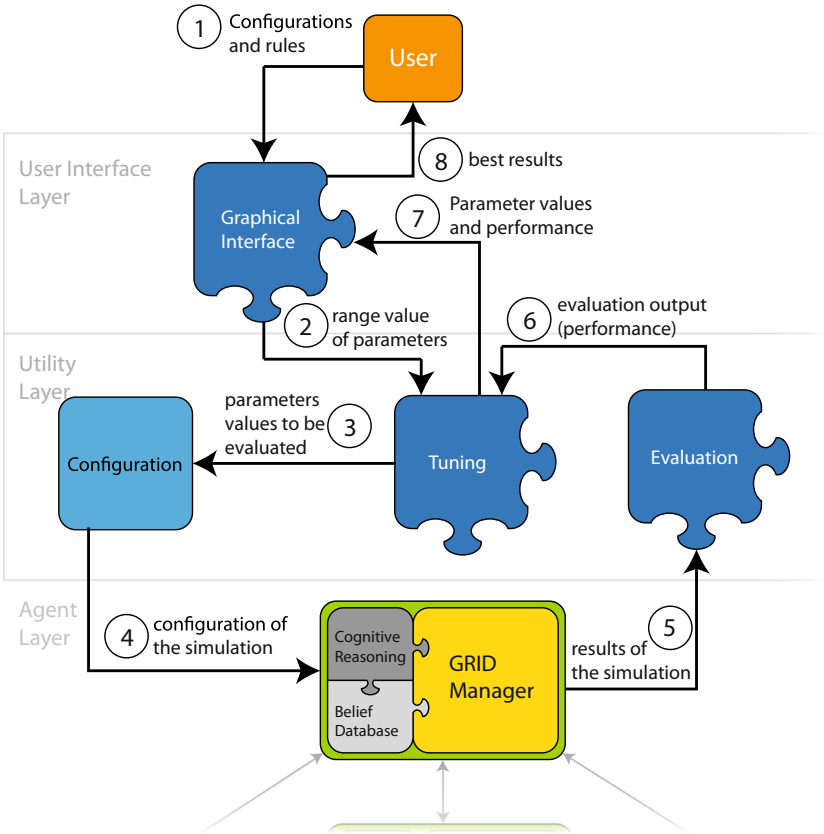

Fig. 5 The interactions between the simulation layers when the auto-tuning module is activated

simulation quality metric (as presented in Section 3.1). The output simulation metric is fed back to the auto-tuning system (6), and a new set of parameters (or multiple sets, for parallel execution) may be generated for execution with the MASE-BDI.

This cyclic process continues until the tuning algorithm converges to a final solution or the maximum number of simulations (preset by the user) have been executed. After the tuning process completes, the set of parameter values with the best performance is provided to the user along with the simulation results (7 and 8). In our system, tuning may be performed for single or multiple objectives, which could have combined goals, for example, both simulation quality and execution speed. This is possible because the user may define arbitrary functions that combine different metrics into a value to be minimized or maximized by the auto-tuning algorithms. In this work, we evaluate single tuning with the goal of maximizing the simulation quality and multi-objective tuning using a goal that combines typically conflicting metrics—maximizing the simulation quality while minimizing the execution time.

4.2.1 Parallel auto-tuning

This section presents the approach used to employ distributed and parallel computing environments to accelerate the MASE-BDI tuning process. The distributed tuning in our systems is attained with the parallel and independent execution of the MASE-BDI simulations with different input parameter values, which consists of evaluating multiple points of the search space in parallel. This approach is possible with the PRO algorithm, which allows for a configurable number of points or parameter sets to be retrieved into a single iteration of the tuning process.

Figure 6 presents the internal auto-tuning architecture for parallel and distributed computing environments. The auto-tuning module has an Auto-Tuning Manager (ATM) and Worker processes. The ATM is a multi-threaded process responsible for running the tuning algorithm and assigning simulations for execution in the distributed and parallel system with the Workers. The communication between ATM and Workers is carried out using the Message Passing Interface (MPI), which is a standard tool for efficient communication and process management in high-performance distributed machines [39].

After a Worker process has been assigned a simulation task (or a specific parameter set to be evaluated), the configuration of the model to be computed is generated, the simulation executes, and the quality of the simulation output is evaluated using the MASE-BDI components previously described in the tuning process overview (Fig. 5). The evaluation of the simulations is also sent to the ATM process.

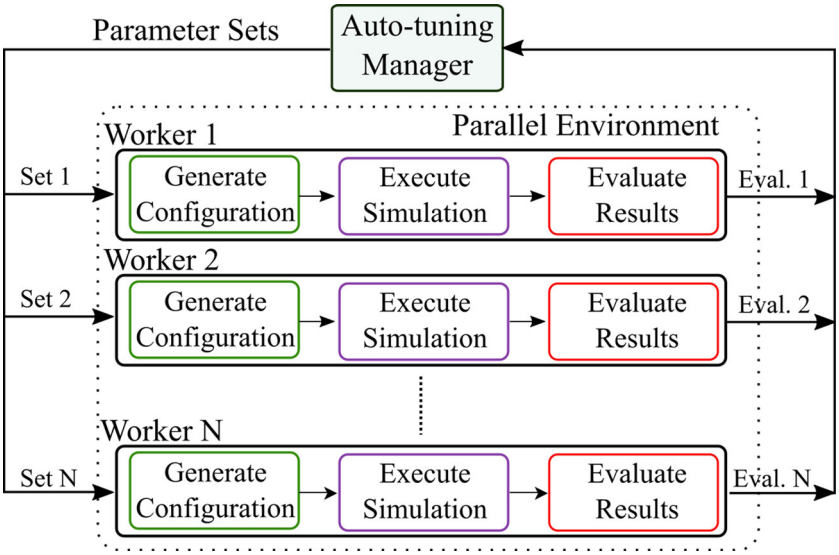

Fig. 6 Internal auto-tuning architecture for parallel and distributed environments

As in the sequential execution scheme, the ATM will run the selected tuning algorithm using the evaluation results and—if the tuning algorithm does not converge or the maximum number of iterations has not been reached—the tuning will proceed into another round.

5 Cerrado case study

Figure 7 presents the Federal District of Brazil, which is located in the Central-West region (15°47′42″S 47°45′28″W) with a total area of 5,779.999 km^2. Located in the Central Plateau of the Brazilian Highlands, the Federal District is in the Cerrado biome. The Cerrado (Spanish/archaic Portuguese for hilly) is a vast tropical savanna ecoregion of Brazil. The Cerrado's climate is typical of moister savanna regions of the world, with a semi-humid tropical climate. Thus, the Federal District climate has only two seasons: a wet season (summer) and a dry season (winter). During the dry season, the humidity can reach critical levels.

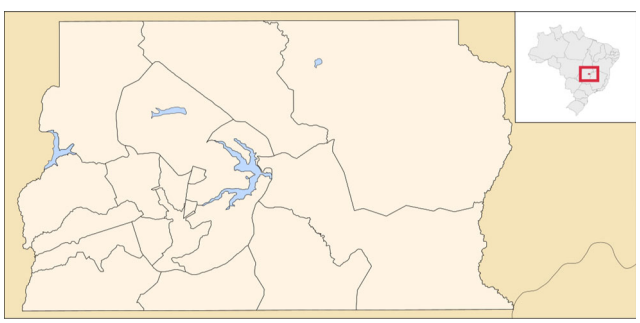

Fig. 7 Location of the Federal District of Brazil

A vast amount of research has demonstrated that the Cerrado is one of the richest of all tropical savanna regions and has high levels of endemism [40]. Characterized by enormous ranges of plant and animal biodiversity, the World Wide Fund for Nature named it the most biologically rich savanna in the world, with approximately 10,000 plant species and 10 endemic bird species. There are nearly 200 species of mammals in the Cerrado, although only 14 are endemic.

Even though it is a land of high ecological value, the Cerrado is currently one of the most threatened biomes in South America due to the rapid expansion of agriculture [41]. Data from the Brazilian Institute of Environment and Renewable Resources (IBAMA) show a cumulative loss of 47.8 % of the Cerrado natural vegetation cover in the last three decades. Experts point out that conservation efforts fall far short of the real needs of the biome. Only 2.2 % of the territory occupied by the Cerrado is legally protected [42].

These changes come mainly from the replacement of native vegetation for agricultural activities because the Cerrado is considered one of the last agricultural production frontiers in the world [43], responsible for almost 70 % of Brazil's beef cattle production. Surpassing even the large-scale grazing operations is soybean production, in which soy farmers apply chemical lime ($CaCO_3$) that changes the pH of soils, resolves aluminium toxicity, and mobilizes plant nutrients that were previously tightly bound to clay particles [44]. The expansion of these activities is driven by a series of interconnected socio-economic factors, often encouraged by government policies.

To understand this complex land change dynamic, Brazil's Federal District, specifically, the Cerrado biome was chosen as a case study for the MASE and used for

comparison in this work. The physical state of the cells in the simulation grid correspond to the real input using land cover maps provided by IBAMA at an initial time (2002–t0) and a final time (2008–t6). The input maps are LANDSAT ETM satellite images from the Remote Sensing Centre (CSR) of IBAMA, classified by the PROBIO software (IBAMA/Environment Ministry of Brazil-MMA). These maps are public domain data being used by the government for deforestation control and land use planning. The maps contain two basic categories: anthropic land use and native vegetation; the water courses were not explored.

5.1 Simulation settings

The experiments were executed with the MASE and the MASE-BDI to verify how adding rationality to the agents would affect the accuracy and execution times of the resulting simulations. The initial model setting considers the environmental changes between the years of 2002 and 2008, making up a total of 365 simulations steps (one week in real time is converted to one simulation step).

The total area of study was divided into cells in which every set of four cells represents one hectare, and each hectare is occupied by a different agent. The physical state of the cells corresponds to the set of real spatial data that are preprocessed using the image processing module. Considering that the exploitation of land use depends on aspects such as social, economic, geographic and political factors, information from the maps is taken into account to create land use simulations, which are called proximal variables. Thus, the simulations include six proximal variables: (i) water courses (rivers); (ii) water bodies (lakes); (iii) buildings; (iv) highways; (v) streets; and (vi) protected areas. The political aspects are also considered as a compelling force in the simulation by translating the PDOT (Federal District Spatial Plane) onto an influence matrix for the TAs. In both simulations, the proximal variables are used to create a probabilistic model that reflects the likelihood of exploration.

The natural environment (not anthropized) has an exploratory potential of 1,500 per area, while the anthropized area has only 500. The TAs were defined as farmers or ranchers, who could be individual or group producers. An individual TA can consume only 500 of the exploratory potential of its current area per step while grouped producers can explore 1,500 per step. When the potential of the area reaches zero, the agent moves. After a TA moves and begins exploring an area that was originally natural, that area changes state to anthropized (500 of exploratory potential), independently of the amount of potential actually consumed.

The proximal matrix is a combination of the environmental attributes and represents the likelihood of a TA to move to a specific area. In the Cerrado case, area proximity to water bodies, water courses, and highways attracts TAs (farmers or ranchers in this case), but area proximity to buildings, streets and protected areas repels them. The total attractive forces minus the total repulsive attributes composes the first part of the proximal matrix. The PDOT influence is subsequently applied in the matrix using the following rules: matrix elements that are in the prohibited polygon of PDOT are set to zero; elements that are in the incentive polygon have their value boosted by ten percent; and the rest of the elements are left intact, retaining their original sum.

6 Experiments and results

In this section, the MASE-BDI functionalities are evaluated and compared to the MASE using the Brazilian Cerrado biome case. Section 6.1 describes the experimental results with sequential executions and Section 6.2 details the parallel and distributed auto-tuning experimental results.

6.1 MASE-BDI vs. MASE

The MASE-BDI vs. the MASE experiments were executed using a machine with an Intel® Core i5 and 4GB RAM memory, running the Windows 7 operating system and with the Java 7 64-bit platform installed. The experiments were conducted by varying the number of TAs and registering the simulated environment in the last step, as well the elapsed time, from the start of the simulation to its end. Starting with 20 TAs, 10 ranchers and 10 farmers, the simulation is executed until the 365 step mark is reached, and then, the simulated environment is registered. After the simulation completes, it restarts, increasing the number of TAs by 10 TAs of each type and cycling the executions until there is no available space for the number of exploring agents and the steps have not reached the 365 mark. For the experiments using MASE, the saturation state was reached with 150 agents of each type, while when using MASE-BDI it was reached with 70 agents of each type. It is important to highlight that in auto-tuning experiments, these configurations are modified automatically by the tuning algorithms used (Section 6.2).

When the saturation state was reached, the simulated final states were compared to the real representation of the Federal District Cerrado in 2008 using the methodology described in [21]. The comparison was performed by acquiring each area in the simulated state and correlating it to the equivalent area in the real observed location. An area anthropized in both simulated and observed spaces is considered a correct change, but when it remains in the same state in both simulated and observed spaces, it is

Table 2 MASE experimental results

Agent quantity (each type of TA)	FoM	Simulation time (hh:mm:ss)
10	13.14499263	0:14:23
20	23.38390255	0:17:25
30	30.96450862	0:26:20
40	33.16913627	0:35:11
50	40.20544277	0:59:19
60	41.76096015	1:24:24
70	43.83395872	1:49:05
80	48.33650662	2:34:13
90	52.93650662	3:00:45
100	50.76171565	3:24:19
110	50.37117590	4:03:04
120	50.84269911	5:12:51
130	52.40097387	6:23:41
140	51.95890604	6:46:13
150	51.42224730	7:45:12

considered a correct permanence. However, when a simulated area changed but the observed area is preserved, or conversely, when a simulated area is preserved but the observed area was anthropized, it is considered an incorrect change or an incorrect persistence, respectively, and the error is calculated. Some areas of the total space are omitted from comparison because they should not influence the result, such as pixels in a middle of a river or a street. At the end of the comparison, the number of correct changes is divided by the sum of the correct changes, incorrect changes and incorrect persistence. The resulting quotient is called the FoM, which measures how closely the simulated model approached the observed space (as stated in Section 3.1).

The final FoM and the simulation time of the MASE and the MASE-BDI experiments are presented in Tables 2 and 3, respectively. Note that the MASE simulations reached the best FoM (52.93) with 90 TAs (time = 3 h 45 s) as listed in Table 2. The MASE-BDI simulations reached the best FoM (54.04) with 30 TAs (time = 43 s) as presented in Table 3.

Table 3 MASE-BDI experimental results

Agent Quantity (each type of TA)	FoM	Simulation time (hh:mm:ss)
10	14.87036418	0:00:14
20	29.79210311	0:00:27
30	54.04066592	0:00:43
40	51.98603923	0:01:04
50	52.27247897	0:01:50
60	51.75358084	0:02:22
70	50.87789301	0:04:21

As shown, the rationality added to the MASE-BDI agents did not affect either the correct scoring or the execution time of simulations because the MASE-BDI maintained the quality of results with only 1/3 the number of agents (30 TAs; FoM = 54 for the MASE-BDI compared to 90 TAs; FoM = 52 for the MASE) and did so in less time (43 s vs. 3 h 45 s). The simulations presented considered TA agent type variations (ranchers and farmers), but percentage variations in group agents were also used to represent condominiums using different land exploitation rates.

To provide a visual assessment of the nature of the MASE-BDI simulation results, we used a validation technique that overlays three maps as proposed by [21], namely, the reference map from the initial time (2002), the reference map of the ending time (2008), and the prediction map for the ending time (2008). This three-map comparison visually distinguishes the pixels that are correct due to persistence versus the pixels that are correct due to change. Figure 8 presents the best MASE-BDI experimental result with 30 TAs and an FoM of 54.04 (Table 3). The black pixels show the proximal variables in different layers for rivers, lakes, buildings, highways, streets, railways and PA that were not considered in the land transformation simulation. The dark grey pixels show where the land change model predicted change correctly, while the medium grey pixels show the error locations where change was observed at locations where the model predicted persistence (anthropic persistence). The light grey pixels show the error locations where persistence was observed at locations where the model predicted change (native vegetation persistence). Finally, the white pixels show locations where the land change model correctly predicted persistence (anthropic loss). When there are more light grey pixels than medium grey pixels, the land change model predicted more change than was observed.

The presented results show how the space occupation is well-represented by the BDI metaphor. Even using fewer agents than the MASE, the MASE-BDI simulation results scored by FoM reached the 50-point threshold in the MASE-BDI. Thus, agents with rationality are more efficient because they are capable of reasoning about their next action on their own. The MASE also had the handicap of TAs that were too dependent on the TM. In the MASE, when a TA had explored its area completely, it always needed to ask the TM for a new space, which overburdens the TM as this scheme requires it to be able to handle several requests for every step of the simulation. The TA scheduling used in the MASE was modified in the MASE-BDI architecture. In the MASE, TAs are scheduled in small groups and in a sequential manner. In other words, when one group finishes executing, another group starts executing.

In the MASE-BDI, independent of quantity, all TA are executed simultaneously when the step opens because TAs in the MASE-BDI are aware of their possibilities, local

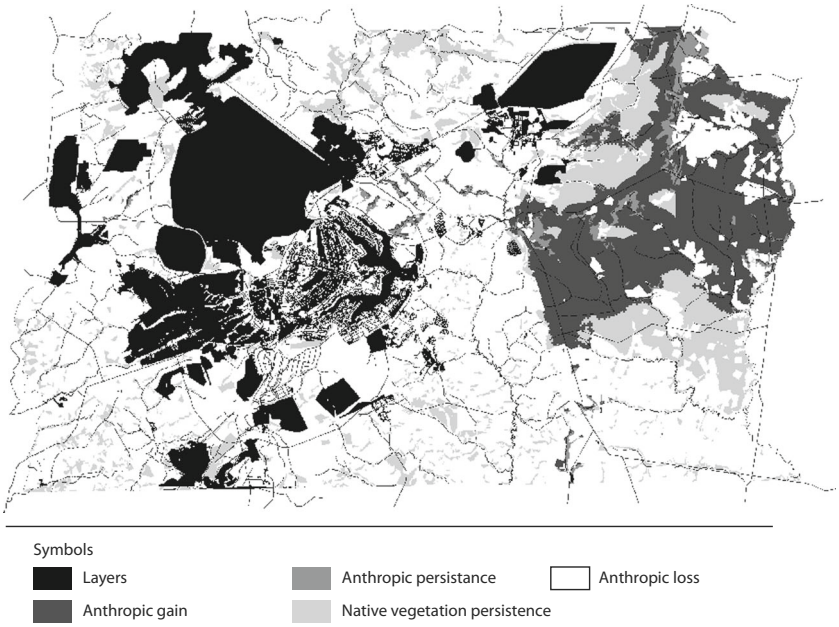

Fig. 8 Validation map for the Federal District of Brazil simulation by overlaying the three maps

environment, and other agents. In the MASE-BDI, the TM authorizes all agents to work as soon as the step opens. Subsequently, the TM has to handle only requests for relocation by conflicting agents (agents disputing the same space). The changes implemented in the MASE-BDI explain the massive improvement in simulation execution time as compared to the MASE execution times. For comparison, the execution times of the MASE (with 90 TAs) and the MASE-BDI (with 30 TAs), for the configurations in which they attained the highest FoM metric values show that the MASE-BDI is 535× faster than the MASE. This performance improvement is even more impressive because it is achieved simultaneously with an improvement in the FoM metric (which increased by approximately 1.1).

6.2 MASE-BDI auto-tuning

The parameters to be tuned in the simulation model in our system are presented in Table 4 along with their descriptions and range values. The search space size with these parameters leads to more than 6 million possible parameter configurations.

The MASE-BDI auto-tuning experimental evaluation was also carried out using the Cerrado case (detailed in Section 5). The auto-tuning algorithms (Section 2.2) were configured to adjust the parameters presented in Table 4 to maximize different objective functions. The details of the actual objective functions used are presented along with the experimental results. These experiments were executed using a machine with a dual socket Intel® Xeon® E5-2640 v3 processor clocked at 2.6 GHz (for a total of 16 physical CPU cores with 2-way hyper-threading) with 64 GB of RAM memory and running the Linux operating system.

To evaluate the proposed auto-tuning techniques in the MASE-BDI, we organized the experimental evaluation as follows: (i) in the next three sections (Sections 6.2.1 to 6.2.3), we perform a comparative analysis of the auto-tuning algorithms in single- and multi-objective configurations and analyse their stability with respect to the initialization conditions; and (ii) we examine the performance improvements attained using the parallel auto-tuning in Section 6.2.4.

6.2.1 Auto-tuning for quality

This set of experiments intends to evaluate the ability of the auto-tuning algorithms to find a set of parameters that improve the quality of the simulation output using the FoM metric (see Section 5.1). In this experiment, FoM is computed using the MASE-BDI evaluation module (Fig. 5) for each simulation output (parameter value used) and the result is fed back directly to the tuning algorithm. We have

Table 4 MASE-BDI simulation model parameters

Parameter	Range value		
	min	max	step
Agent quantity	10	100	1
Percentage of group agents	10	100	1
Agent potential of individual exploration	200	1500	50
Group potential of exploration	200	1500	50

evaluated the performance of both the NM and PRO tuning algorithms (Section 2.2) and compared their results to those attained by the model using parameters that were set by a specialist in a manual tuning.

As a result, the NM and PRO auto-tuning algorithms were able to reach FoM values of 55.377 and 54.828, respectively. This level of quality is better than the best value attained with the manual tuning (FoM = 54.04 with 30 TAs, 43 s) presented in Table 3 (Section 6.1). This is an important result because the auto-tuning reached better quality results as compared to the manual tuning while minimizing the user interaction required for this labour-intensive and error-prone task. Further, for the same experiments, the NM and PRO converged in 69 and 62 iterations of the tuning process (or simulation executions), respectively, with the PRO attaining higher FoM values with a smaller number of iterations. We want to emphasize that the cost of retrieving a new set of parameter values from the auto-tuning algorithms is no higher than 10 ms, which is much smaller than the cost of executing the actual simulation runs (on the order of tens of seconds). As such, minimizing the number of iterations while achieving good output quality results is the main goal of the auto-tuning in this evaluation.

6.2.2 Multi-objective auto-tuning

In this section, we analyse the auto-tuning algorithms in scenarios with multi-objective tuning tasks. These objective functions combined simulation output quality and execution time. These functions have the form $G = w_1 \times q/100 + w_2 \times t'$, where q and t' refer to the quality (FoM) and the execution time, respectively. The weighting factors w_1 and w_2 can be used to prioritize the components of the objective functions according to the user's goal. The FoM quality metric assumes values between 0 and 100, but this is not true of execution time. Therefore, we scaled the components to the same scale (0 to 1). The q component is simply divided by 100. We normalized the execution time t to compute t', which is also expected to be between 0 and 1. The normalized execution time is computed as $t' = (t_{slowest} - t)/(t_{slowest} - t_{fastest})$, where $t_{slowest}$ and $t_{fastest}$ are, respectively, the slowest and fastest execution times of the simulation with the given parameter range values. In this normalization, the value of t' tends towards either 0 or 1 because t is close to either the slowest or fastest possible execution time.

To estimate the slowest and fastest execution times of our simulation model with the used parameter value ranges, we executed 15,000 simulations with parameter values randomly selected from the range values defined by the user (Table 4) and extracted the slowest and fastest execution times from those simulations to be used in our normalization.

The experimental results of auto-tuning for the multi-objective function G as the value of the weight w_1 is varied while maintaining a fixed value of $w_2 = 1$ are listed in Table 5. In this set of experiments, both the NM and PRO algorithms were evaluated by selecting a single parameter set for evaluation per iteration of the tuning process. As presented, the auto-tuning algorithms were able to quickly tune our system model under different scenarios. The maximum number of iterations (and model simulation executions) until convergence were 101 (0.00167 % of the search space) and 126 (0.00208 % of the search space) for NM and PRO, respectively, out of a search space with more than 6 million possible parameter value combinations (Table 4). Although NM and PRO both attained good quality (q) in all tuning experiments, the latter was able to achieve higher values of q for all tuning tasks. Also, as the w_1 value increases, the optimization prioritizes the quality component of the equation; therefore, the resulting q value tends to increase. This compromise is most noticeable for the configuration where $w_1 = 1$ in which the best value was reached with substantially smaller execution times.

6.2.3 Impact of auto-tuning seed to performance

This section evaluates the impact of varying the auto-tuning seed parameter set to its performance. Our tuning algorithms allow for an initial seed point or parameter set to be first evaluated by the tuning algorithm and then selected in the interval of the input parameter range values. This point or parameter set is calculated as a shift from the beginning of the range value by a percentage of the range size of each parameter.

Figure 9 presents the best performance ($G/(w_1 + w_2)$) of the auto-tuning algorithms for $w_1 = 3$ and $w_2 = 1$ as the seed point used is varied by modifying the shift value used.

Table 5 Auto-tuning of the function G as the weight of the quality component (w1) of the objective function is increased. The q and t values refer to those measured with the best G presented.

w_1	Policy	Iterations	$\frac{G}{w_1 + w_2}$	q	t
1	NM	101	0.679	51.00	24
	PRO	101	0.672	53.62	27
2	NM	68	0.589	54.84	37
	PRO	126	0.591	55.01	37
3	NM	72	0.550	54.63	39
	PRO	92	0.608	55.11	31
4	NM	82	0.588	54.35	30
	PRO	66	0.583	55.39	35
5	NM	67	0.582	55.06	32
	PRO	84	0.574	55.23	36

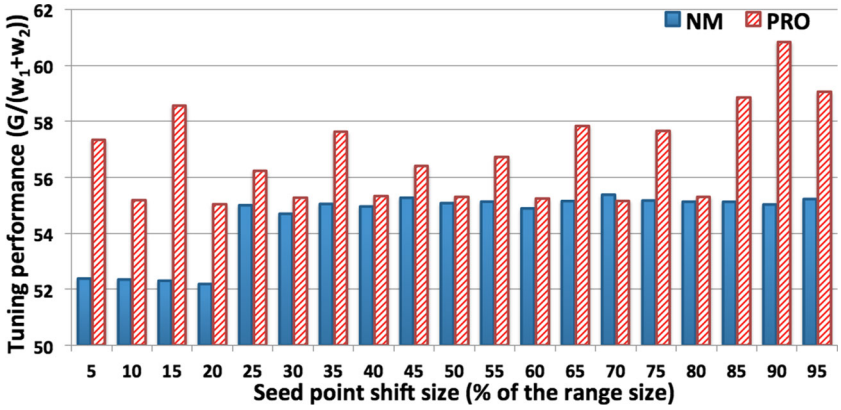

Fig. 9 Performance ($G/(w_1 + w_2)$) attained by NM and PRO as the initial point in the search space (parameter set) evaluated by the tuning is varied. Each parameter is initialized with a value that is a shift from the beginning of the parameter range, and the shift value is calculated as a percentage of the range size of each parameter

As presented, although the performance of the auto-tuning algorithms is affected by the seed parameter set chosen, they are still able to attain good performance for all shift values. Also, the difference between the best and worst G values attained by the tuning algorithms as the seed point is varied is no higher than 1.09×. Further, the PRO algorithm attained better performance (G value) than NM for more than 95 % of the seed point configurations tested, and it improved the result of G by approximately 1.1× as compared to NM. The default shift value used in previous experiments was 90 % because it leads to better performance for both auto-tuning algorithms.

6.2.4 Performance of parallel auto-tuning

This section evaluates the ability of our system to speed up the auto-tuning process by taking advantage of a parallel environment. This evaluation is carried out in two steps. We first investigate the upper limit gains of our parallelization scheme by executing a batch of simulations in parallel as the number of the MPI processes used is varied. The second set of experiments consists of an actual auto-tuning employing the PRO algorithm to perform the tuning task in the parallel infrastructure we proposed and implemented in this work.

The results for the first set of experiments are presented in Fig. 10. In this evaluation, a batch with 1,000 simulation tasks using the same parameters is created to analyse the speedups attained by executing them in parallel on the proposed architecture. As shown, we were able to achieve a speedup value of up to approximately 11× using 16 MPI processes and, as a consequence, 16-CPU cores are employed to perform the simulations in parallel. Using additional processes to take advantage of hyperthreading in our processor did not lead to any additional performance gains.

Although the performance improvement with the use of 16 MPI processes is high, the speedup achieved is below linear. The sub-linear gains could be caused either by (i) higher costs of the simulation runs as the machine is shared by multiple simulations (or MPI processes) or by (ii) a bottleneck in the parallelization scheme. Table 6 shows the average execution time of the simulation runs as the number of MPI processes used is varied. As presented, there is an increase of 1.47× in the simulation execution time as the number of MPI processes grows from 1 to 16. This increase in execution times is a consequence of sharing the machine resources. More specifically, competition for the caching and memory subsystems impact the simulations because they are memory-intensive. The execution time with 16 MPI processes is 1.47× higher than when 1 MPI process is used, which limits the efficiency of the parallelization with 16 MPI processes to up to 0.68 (1/1.47). Because the measured efficiency in our execution for 16 MPI processes was 0.67, the increasing execution times explain most of our speedup limitations that, as a consequence, cannot be attributed to any bottleneck in our parallelization strategy.

Finally, we executed a parallel version of the auto-tuning tasks using PRO on the same tuning tasks evaluated in Section 6.2.3, which maximizes G for $w_1 = 3$ and $w_2 = 1$. In this experiment, we retrieved multiple parameter sets from PRO in each iteration of the tuning process for parallel

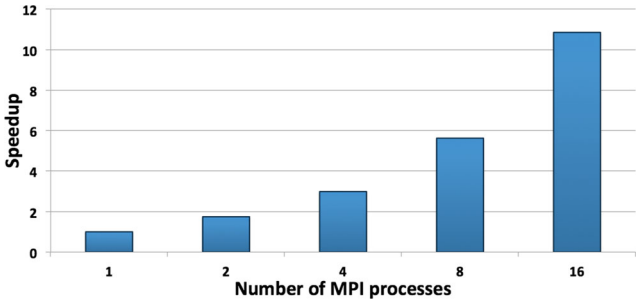

Fig. 10 Improvement in execution time attained for execution of a batch of 1,000 simulations as the number of MPI processes and CPU cores used was varied

Table 6 Average execution time (in seconds) of simulation runs according to the number of MPI processes used

	Number of MPI processes				
	1	2	4	8	16
AVG time (s)	46.3	53.0	63.3	66.7	68.1

evaluation in each of the MPI processes used. We scaled our execution until the 16 CPU computing cores available in our system were all being used and were able to reduce the tuning time compared to the sequential case by up to 10.2× while achieving the same level of performance (G value). This new parallel auto-tuning feature of our system allows us to quickly introduce and tune new models and simulation objective functions.

7 Conclusions

This article presented the MASE-BDI agent-based simulator for environmental land changes coupled with an efficient and parallel auto-tuning method. Compared to popular peer-reviewed land change tools such as IDRISI, ArcGIS, and DinamicaEGO, the MASE-BDI and the MASE are the only multi-agent based tools. In addition, compared to well-known agent-based simulation frameworks including Netlogo, RePast and CORMAS, the MASE-BDI is the only one that implements rational agents through mentalistic notions using the BDI model. The MASE-BDI and the MASE simulators were validated using the real world Brazilian Cerrado Case study—currently one of the most threatened biomes of South America. The results show that the MASE and the MASE-BDI are distinct from popular land change frameworks in the number of pixels for which they correctly predicted land change according to the statistical methods proposed by [21]. Moreover, the MASE-BDI was able to attain slightly better simulation results than the MASE with sequential execution using the FoM metric (1.03×) as well as a significant improvement on the execution time (535× faster).

We also developed a novel auto-tuning module in the MASE-BDI to automate the work of finding appropriate simulation parameters that maximize or minimize a metric of interest set by users (i.e., simulation quality and/or execution time). We evaluated different auto-tuning algorithms to implement a module for efficient parallel execution of the tuning process. The experimental results demonstrated that the evaluated auto-tuning strategies were able to quickly converge while guiding the simulator to provide good results. For example, out of a search space with more than 6 million possible parameter configurations, the auto-tuning evaluated less than 0.00115 % of the possible parameter sets while maximizing the simulation score. Further, the parallel auto-tuning version, executing in a hardware setting with 16-CPU computing cores, achieved a speedup approximately 11× faster compared to sequential execution, while maintaining the level of performance (G value).

We expect that the MASE-BDI simulator may be helpful to evaluate other real environmental scenarios to support human decision-making processes because the implemented rational agents can represent the complexity of individual interactions that are desirable for ecological applications. Also, the implemented distributed and parallel auto-tuning features may allow the quick introduction and tuning of new models and simulation objectives, which are also important qualities for environmental simulators. Finally, the auto-tuning approach was designed to be flexible, which will be useful for future work related to sensitivity analysis. The MASE-BDI simulation model sensitivity analysis is directed to evaluate the degree of uncertainty level embodied in the multi-agent approach, which is a novel research topic in the context of environmental land change frameworks.

Acknowledgments This work was supported in part by the Brazilian National Council for Scientific and Technological Development (CNPq) and the Coordination for the Improvement of Higher Education Personnel (CAPES).

References

1. Ellis E (2013) Land-use and land-cover change. Retrieved from http://www.eoearth.org/view/article/154143
2. BL Turner II, WC Clark, RW Kates, JF Richards, JT Mathews, WB Meyer (eds) (1993) The Earth as transformed by human action: global and regional changes in the biosphere over the past 300 years. Cambridge University Press. ISBN 052144 6309
3. Foley JA, DeFries R, Asner GP, Barford C, Bonan G, Carpenter SR, Chapin FS, Coe MT, Daily GC, Gibbs HK, Helkowski JH, Holloway T, Howard EA, Kucharik CJ, Monfreda C, Patz JA, Prentice CI, Ramankutty N, Snyder PK (2005) Global consequences of land use. Science 309(5734):570–574. doi:10.1126/science.1111772. ISSN 1095-9203
4. Railsback SF, Lytinen SL, Jackson SK (2006) Agent-based simulation platforms: review and development recommendations. Simulation 82(9):609–623. doi:10.1177/0037549706073695. ISSN 0037-5497
5. Bratman ME (1987) Intention, plans, and practical reason. Harvard University Press, Cambridge. ISBN 1575861925. doi:10.2307/2185304
6. Rao AS, Georgeff MP (1995) Bdi agents: from theory to practice. In: Proceedings of the 1st international conference on multi-agent systems (ICMAS-95), pp 312–319

7. Wooldridge M (2009) An introduction to multiagent systems, 2nd edn. Wiley Publishing. ISBN 978-0-470-51946-2
8. Braubach L, Lamersdorf W, Pokahr A (2003) Jadex: implementing a BDI-Infrastructure for JADE agents. EXP 3(3):76–85
9. Ţăpuş C, Chung I-H, Hollingsworth JK (2002) Active harmony: towards automated performance tuning. In: Proceedings of the 2002 ACM/IEEE conference on supercomputing, SC '02. IEEE Computer Society Press, Los Alamitos, pp 1–11
10. Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313. doi:10.1093/comjnl/7.4.308
11. Tabatabaee V, Tiwari A, Hollingsworth JK (2005) Parallel parameter tuning for applications with performance variability. In: Proceedings of the 2005 ACM/IEEE conference on supercomputing, SC '05. IEEE Computer Society, Washington, DC, p 57. doi:10.1109/SC.2005.52. ISBN 1-59593-061-2
12. Eastman JR (2015) IDRISI 32.2—guide to GIS and image processing. Clark Labs, Clark University, Worcester. Retrieved from https://clarklabs.org/
13. Esri (2015) ArcGIS. Retrieved from http://www.esri.com/software/arcgis
14. Soares-Filho BS, Rodrigues HO, Souza Costa WL (2009) Modeling envitonmental dynamics with dinamica ego. Belo Horizonte, Minas Gerais, Brazil. Dinamica EGO guidebook, ISBN 978-85-910119-0-2
15. Ralha CG, Abreu CG, Coelho CGC, Zaghetto A, Macchiavello B, Machado RB (2013) A multi-agent model system for land-use change simulation. Environ Model Softw 42:30–46. doi:10.1016/j.envsoft.2012.12.003
16. Tisue S, Wilensky U (2004) Netlogo: a simple environment for modeling complexity. In: International conference on complex systems, pp 16–21
17. Wilensky U (1999) NetLogo. Retrieved from http://ccl.northwestern.edu/netlogo/
18. North MJ, Collier NT, Ozik J, Tatara ER, Macal CM, Bragen M, Sydelko P (2013) Complex adaptive systems modeling with Repast Simphony. Complex Adapt Syst Model 1:3. doi:10.1186/2194-3206-1-3. ISSN 2194-3206
19. C Le Page, F Bousquet, I Bakam, A Bah, C Baron (2000) Cormas: a multiagent simulation toolkit to model natural and social dynamics at multiple scales. In: Workshop "The ecology of scales"—Wageningen (The Netherlands)
20. Weiss G (ed) (2013) Multiagent systems, 2nd edn. MIT Press, Cambridge. ISBN 978-0-262-01889-0
21. Pontius RG, Boersma W, Castella J-C, Clarke K, de Nijs T, Dietzel C, Duan Z, Fotsing E, Goldstein N, Kok K, Koomen E, Lippitt CD, McConnell W, Sood AM, Pijanowski B, Pithadia S, Sweeney S, Trung TN, Veldkamp AT, Verburg PH (2008) Comparing the input, output, and validation maps for several models of land change. Ann Reg Sci 42:11–37
22. Eldred MS, Giunta AA, Van Bloemen Waanders BG, Wojtkiewicz SF Jr, Hart WE, Alleva MP (2002) DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis - version 3.0 users manual
23. Vuduc R, Demmel JW, Yelick KA (2005) OSKI: a library of automatically tuned sparse matrix kernels. J Phys Conf Ser 16(1):521
24. Tiwari A, Hollingsworth JK (2011) Online adaptive code generation and tuning. In: 2011 IEEE international parallel distributed processing symposium (IPDPS), pp 879–892. doi:10.1109/IPDPS.2011.86
25. Christen M, Schenk O, Burkhart H (2011) PATUS: a code generation and autotuning framework for parallel iterative stencil computations on modern microarchitectures. In: Proceedings of the 2011 IEEE international parallel & distributed processing symposium, IPDPS '11. IEEE Computer Society, Washington, DC, pp 676–687. doi:10.1109/IPDPS.2011.70. ISBN 978-0-7695-4385-7
26. Ansel J, Kamil S, Veeramachaneni K, Ragan-Kelley J, Bosboom J, O'Reilly U-M, Amarasinghe S (2014) OpenTuner: an extensible framework for program autotuning. In: Proceedings of the 23rd international conference on parallel architectures and compilation, PACT '14. ACM, New York, pp 303–316. doi:10.1145/2628071.2628092. ISBN 978-1-4503-2809-8
27. Munz U, Papachristodoulou A, Allgower F (2008) Delay-dependent rendezvous and flocking of large scale multi-agent systems with communication delays. In: 47th IEEE conference on decision and control, 2008. CDC 2008, pp 2038–2043. doi:10.1109/CDC.2008.4739023
28. Lendek Z, Babuska R, De Schutter B (2009) Stability of cascaded fuzzy systems and observers. IEEE Trans Fuzzy Syst 17(3):641–653. doi:10.1109/TFUZZ.2008.924353. ISSN 1063-6706
29. Daneshfar F, Bevrani H (2010) Load-frequency control: a ga-based multi-agent reinforcement learning. IET Gener Transm Distrib 4(1):13–26. doi:10.1049/iet-gtd.2009.0168. ISSN 1751-8687
30. Jarraya Y, Bouaziz S, Alimi AM, Abraham A (2014) Multi-agent evolutionary design of beta fuzzy systems. In: 2014 IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 1234–1241. doi:10.1109/FUZZ-IEEE.2014.6891722
31. Yang S, Gechter F, Koukam A (2008) Application of reactive multi-agent system to vehicle collision avoidance. In: 20th IEEE international conference on tools with artificial intelligence, 2008. ICTAI '08, vol 1, pp 197–204. doi:10.1109/ICTAI.2008.134
32. Balaji PG, German X, Srinivasan D (2010) Urban traffic signal control using reinforcement learning agents. IET Intell Transp Syst 4(3):177–188. doi:10.1049/iet-its.2009.0096. ISSN 1751-956X
33. Abdelhameed MM, Abdelaziz M, Hammad S, Shehata OM (2014) A hybrid fuzzy-genetic controller for a multi-agent intersection control system. In: 2014 international conference on engineering and technology (ICET), pp 1–6. doi:10.1109/ICEngTechnol.2014.7016755
34. Klein CE, Bittencourt M, Coelho LS (2015) Wavenet using artificial bee colony applied to modeling of truck engine powertrain components. Eng Appl Artif Intell 41:41–55. doi:10.1016/j.engappai.2015.01.009. ISSN 0952-1976
35. Oliveira GG, Pedrollo OC, Castro NMR (2015) Simplifying artificial neural network models of river basin behaviour by an automated procedure for input variable selection. Eng Appl Artif Intell 40:47–61. doi:10.1016/j.engappai.2015.01.001. ISSN 0952-1976
36. Shahraiyni HT, Sodoudi S, Kerschbaumer A, Cubasch U (2015) A new structure identification scheme for {ANFIS} and its application for the simulation of virtual air pollution monitoring stations in urban areas. Eng Appl Artif Intell 41:175–182. doi:10.1016/j.engappai.2015.02.010. ISSN 0952-1976
37. Bellifemine FL, Caire G, Greenwood D (2007) Developing multi-agent systems with JADE. Wiley, New York. ISBN 0470057475
38. Braubach L, Pokahr A (2011) Jadex active components framework—BDI agents for disaster rescue coordination. In: Essaaidi M, Ganzha M, Paprzycki M (eds) Software agents, agent systems and their applications, chap. 3, vol 32. IOS Press, pp 57–84
39. Message P Forum (1994) MPI: a message-passing interface standard. Technical report, Knoxville
40. Jepson W (2005) A disappearing biome? Reconsidering land-cover change in the Brazilian savanna. Geogr J 171(2):99–111. doi:10.1111/j.1475-4959.2005.00153.x. ISSN 1475-4959

41. Marquis RJ, Oliveira PS (2002) The Cerrados of Brazil: ecology and natural history of a Neotropical Savana, 1st edn. Columbia University Press, New York
42. Klink CA, Machado RB (2005) A conservação do Cerrado brasileiro. Megadiversidade 1(1):147–155. doi:10.1590/S0100-69912009000400001
43. Sano EE, Rosa R, Silva Brito JL, Ferreira LG (2008) Mapeamento semidetalhado do uso da terra do Bioma Cerrado/Semidetailed land use mapping in the Cerrado. Pesqui Agropecu Bras 43(1):153–156
44. Killeen TJ (2007) A perfect storm in the amazon wilderness: development and conservation in the context of the initiative for the Integration of the Regional Infrastructure of South America (IIRSA), vol 7. Advances in Applied Biodiversity Science (AABS). ISBN 1-934151-07-6