Electrical Power and Energy Systems 42 (2012) 367–374

Contents lists available at SciVerse ScienceDirect

Electrical Power and Energy Systems

journal homepage: www.elsevier.com/locate/ijepes

Multi-agent systems applied to reliability assessment of power systems

Mauro A. da Rosa [a,*], Armando M. Leite da Silva [c], Vladimiro Miranda [a,b]

[a] Institute for Systems and Computer Engineering of Porto Technology and Science, INESC TEC, Portugal
[b] Faculty of Engineering of the University of Porto, FEUP, Portugal
[c] Department of Electrical Eng., Federal University of Itajubá, UNIFEI, Brazil

ARTICLE INFO

Article history:
Received 17 October 2008
Accepted 11 March 2012
Available online 17 May 2012

Keywords:
Multi-agent systems
Monte Carlo simulation
Distributed systems
Power system reliability
Reliability

ABSTRACT

This paper discusses the development of a Multi-Agent Systems (MAS) technology-based platform with potential applications in management and simulation processes in power systems. In order to explore some of the features of MAS, a new methodology is proposed to assess power systems reliability based on Monte Carlo simulation (MCS), exploiting the benefits of the distributed artificial intelligence area and, mainly, the use of the distributed capacity in two ways: building autonomous behaviors to the applications and mitigating computational effort. Through the use of this technology, it was possible to divide the MCS algorithm into distinct tasks and submit them to the agents' processing. Two different approaches to solve generating capacity reliability problems based on chronological MCS illustrate the potential of MAS in power systems reliability assessment.

1. Introduction

In artificial intelligence (AI) research, agent-based systems technology is a new paradigm for conceptualizing, designing, and implementing software systems. Intelligent agents are sophisticated computer programs that act autonomously on behalf of their users, across open and distributed environments, to solve a growing number of complex problems. Increasingly, however, applications have required multiple intelligent agents (IA) that can work together. A multi-agent system is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver.

Several technical works using MAS technology have been published in the literature [1–7], showing potential applications to solve problems related with electric power systems. Recently, two important contributions have organized the concepts, approaches, technical challenges, technologies, standards, and tools for building MAS applied to power engineering [1,2]. Many aspects were discussed in relation to the benefits of this technology besides providing guidance and information on the state-of-the-art of MAS research for the power industry. Some favoured issues in power agent applications have been indicated: modelling, simulation and distributed control. In particular, the modelling of electricity market as a complex adaptive system [3] and the optimization of decisions in retail markets [4] have been discussed. In the last few years, some other applications were carried out to solve power engineering problems [1,2] such as systems integration with database, systems and equipment monitoring, interpretation and analysis of outage data, system restoration [5], support to decision in micro-grids operation [6], and environment simulation [7]. These approaches have shown the great potential of MAS, exploring the concepts of communication and autonomous behaviors offered by this technology. In order to explore some other features of the MAS technology, an application to power systems reliability assessment is being proposed in this work. It uses the benefits from the distributed artificial intelligence area in two ways: building autonomous behaviors and reducing computational effort on power systems reliability evaluations.

Many works have explored the use of the distributed or parallel environment in power systems evaluations [8–10], some of them including reliability assessment [8]. Different from these works, which generally aim mitigating computational effort and make feasible some expensive simulations, this work intends to build a software environment where agents are capable of sensing it through the use of basic intelligence offered by the agent-based technology. This environment facilitates the agents to recognize others specific agents, within the platform, as well as to promote communication among them through the use of *performative* structures. In order to introduce computational intelligence in power system reliability assessment, MAS using Java technology is being proposed, where the agent base is supported by distributed computing technology [11,12].

This paper discusses the development of a MAS technology-based platform to assess generating capacity reliability indices,

* Corresponding author.
E-mail addresses: mauro_eng@yahoo.com.br, marosa@inescporto.pt (M.A. da Rosa).

0142-0615/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ijepes.2012.03.048

based on sequential or chronological MCS processes. Although the focus will be on the application of MAS to this particular reliability problem, other applications based on simulation can also be benefited in a similar manner. The MAS construction was designed under the concept of reliability assessment by the sequential MCS. JADE Library (Java Agent Development Framework) [13], standardized by FIPA (Foundation Intelligent Physical Agents: www.fipa.org), is used and, sometimes, only Java language is called for constructing agents. In the next sections, one will present an overview of the power agent platform, the application of MAS in power systems reliability assessment based on two different approaches, and the results of the reliability evaluations of two electric systems: the IEEE-RTS, with 32 unit generators, and a modified configuration of the Brazilian South/South-eastern System (BSS), with 242 unit generators. Discussions will be provided to demonstrate the potential of MAS technology.

2. Overview of the power agent platform

The proposed power agent platform can be described through layers, as shown in Fig. 1. In the first level, at the bottom (in red[1]), there is the class diagram that represents the electric power system. Representations such as generators, transmission lines, distribution network, and buses have a relationship among them, based on the Oriented Object Modeling (OOM). In the second level, at the right side (in green), there is the class diagram named *tools class* that represents the tools linked to each evaluation algorithm, such as reliability assessment by a sequential MCS. In the third level, at the top (in blue), there is the class diagram with the MAS technology, where agents are built and provided with some abilities to use the first and the second levels, previously described. Fig. 1 is also showing a *Generator Agent* using the *Monte Carlo Class* and the *Generator Class*, which after a perception on its specific environment, has taken an action. Obviously, this simple representation intends to show the philosophy of the simulation platform using one of the main characteristic of agents: autonomy.

In a simpler sense, an agent can be classified as software or hardware elements located in a specific environment, which is able to note changes and react to these changes [14]. Following the same idea, power engineers can compare this concept with an old and simple well-known agent: an over-current relay, which after its perception about an over-current will take an action, switching off the electric circuit. Characteristics such as *Reactivity*, *Pro-Activity* and *Social Ability* are described in several references on Artificial Intelligence (AI) and MAS [14], and here they are applied to this proposed simulation platform. Essentially, the MAS technology is the combination of these inherent characteristics and others that allows its clear distinction from other conventional tools in the power systems analysis. One of the most important characteristics of this technology appears as applying Java to build agents with its ability to distribute applications and run concurrency programming. This feature enables the building of power models using intelligent behaviors in a very distinctive way.

3. Reliability assessment using MAS

The reliability assessment of real large power system by sequential MCS is known as being a very expensive computational evaluation. Moreover, to analyze the sampled states, heavy computational tools such as optimal power flows are intensively used [8,15]. The solution time for analytical techniques is relatively short, but usually requires a simplification of the systems [16].

[1] For interpretation of color in Figs. 1, 4, 6–8, the reader is referred to the web version of this article.

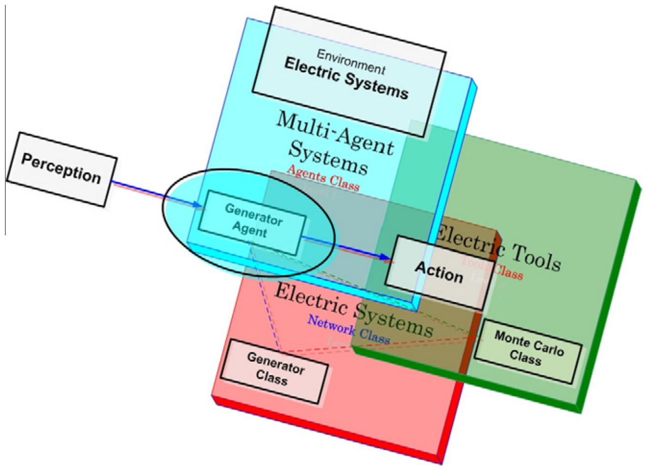

Fig. 1. Representation of the platform layers.

However, sequential MCS makes possible the use of a wide range of detailed models including non-Markovian representation for generation and transmission equipment, correlated chronological load models, etc. In order to reduce this expensive computational cost, different techniques have been proposed: distributed processing [8], artificial neural networks [15], variance reduction [17,18], pseudo-chronological simulations [19]. There are other related techniques to reduce CPU costs that can be found in the bibliography lists of the previous references.

3.1. Using IA to support AI in power system reliability assessment

The Monte Carlo technique is a statistical-based method which can be naturally divided in three inherent stages, in order to assess reliability indices: state selection, state evaluation and index calculation. Many works have explored AI topics as search techniques, knowledge representation, reasoning and learning systems as well as some mathematical approaches aiming to incorporate intelligence on these stages [15,20,21]. Generally, these approaches appear as competitors with non-sequential Monte Carlo simulation, in order to achieve a more efficient simulation process. Fig. 2 shows an overview of these representations. This paper intends to construct an agent-based application in order to support the natural transition from AI to IA in power system reliability assessment. The main idea is to construct a basic intelligent distributed environment that can be able to support sequential Monte Carlo representation using IA features.

Instead of applying AI techniques in order to improve a single stage of non-sequential MCS, this work intends to build an IA architecture that supports sequential MCS as well as AI techniques in more than one stage of the simulation process. Consequently, it will be possible to measure the gains of an agent-based distributed structure in reliability assessment before applying AI topics, using an intelligent agent architecture design. Different from a simple distributed approach, which uses signals to control computers inside a network [8], this work uses agent communication language (KQML: Knowledge Query and Manipulation Language) [13] in order to establish communication among agents. KQML is essentially a knowledge-level message language, which defines a number of performatives and makes explicit an agent's intentions [14]. Firstly, the aim is combining KQML with autonomous behaviors of each stage of the sequential Monte Carlo representation to define an IA architecture design. Secondly, it will prepare this IA architecture to support different AI techniques.

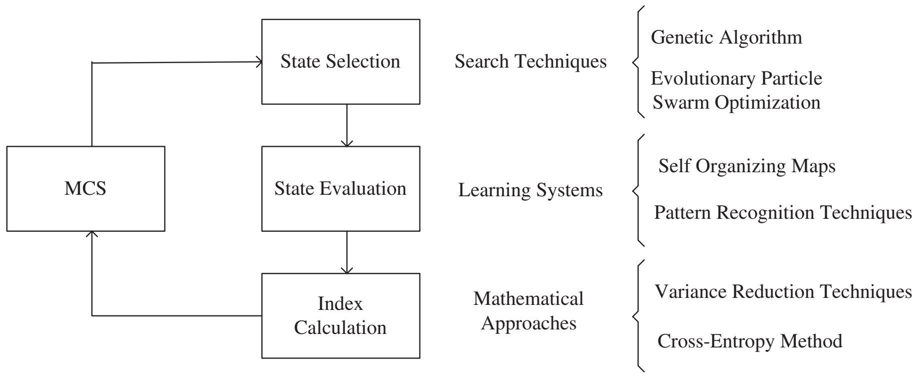

Fig. 2. MCS stages and most recent approaches.

3.2. Sequential Monte Carlo simulation

In order to propose a new way to approach reliability assessment using MAS technology, some basic concepts used by the sequential MCS are revisited. This is a very flexible simulation type, as it allows the representation of non-exponential residence times, useful when dealing with chronological processes. The operation history of system states, for a simulation period *T*, is based on stochastic models of the components and on the load model. The initial state is usually sampled from a non-sequential Monte Carlo simulation algorithm. After evaluating each state, performance indices are estimated using test-functions *G*(*t*):

$$E[G] = \frac{1}{T}\int_0^T G(t)dt \tag{1}$$

Each performance index can be estimated using a suitable test-function. The failure probability, for instance, corresponds to the expected value of an indicator function where $G(t) = 1$ if the system associated with time *t* is a failure state; otherwise $G(t) = 0$. Another way of estimating the expected value of *G*(*t*) is shown as follows:

$$\widetilde{E}[G] = \frac{1}{NY}\sum_{k=1}^{NY} G(y_k) \tag{2}$$

where *NY* is the number of simulated years and y_k is a sequence of system states in year *k*. For instance, the energy not supplied will be the summation of unsupplied energy associated with each interruption of a simulated year. The uncertainty around the estimated indices is given by the variance of the estimator

$$V(\widetilde{E}[G]) = \frac{V(G)}{NY} \tag{3}$$

where *V*(*G*) is the variance of the test-function. The convergence of the simulation process is tested using the coefficient of variation β [17–19].

$$\beta = \sqrt{V(\widetilde{E}[G])}/\widetilde{E}[G] \tag{4}$$

In this work, the problem in power system reliability assessment to be used as an example of the application of MAS technology is the generating capacity evaluation [22], although any similar reliability application (e.g. composite generation and transmission) could be exploited.

3.3. MAS applied to chronological MCS

It is important to observe that the concept of agent was designed for flexibility, not for speed. This means that communication speed must not be required within the algorithm, and all effort should be concentrated on the relevant local tasks for all agents, as to reduce network calling or sending of messages. Therefore, it will be possible to increase processing speed through flexibility. In order to describe the inherent tasks carried out for the agents to evaluate (1)–(4), bearing in mind a generating capacity reliability assessment, the following algorithm is used:

(i) Sample a sequence of generation states based on the failure and repair rates of the generating system units, until completing year *k*;
(ii) Select the chronological load levels for that period (load uncertainties can be included if necessary);
(iii) Identify the set of ordered points in time, which can be linked to system state transitions, to define sequence y_k;
(iv) Evaluate the generating system condition, if success or failure, for the state sequence y_k;
(v) Identify for sequence y_k the contributions for all reliability indices, expressed by the general function *G*;
(vi) Estimate $\widetilde{E}[G]$ in (2) for all generating capacity reliability indices;
(vii) Update the overall indices and estimate their β coefficients as in (4). If the estimate accuracies are acceptable, than stop; otherwise, return to step (i).

Therefore, it is possible to divide and organize tasks for agents as follows:

- Sequence producer (SP) agent: tasks number (i) to (iii);
- State evaluator (SE) agent: tasks number (iv) and (v);
- Index calculator (IC) agent: tasks number (vi) and (vii).

Although the tasks could be divided in another way, it is important to observe that this proposal provides more flexibility to run autonomous agents. All agents have their own time to work on the platform and carry out their tasks independently from each other, but all agents search a common sense, such as a target to achieve results in less time. The Java technology makes it possible to distribute and keep the control of the applications with flexibility and relative speed, improving the simulation process.

Two approaches will be designed to implement the proposed MAS based reliability methodology. The first is named non-synchronized approach, which uses JADE library, and the other one is called synchronized approach, which uses just Java technology.

3.4. Non-synchronized approach

There are different ways to apply the concept of agents to the reliability evaluation. Obviously, the idea is to concentrate computational expensive tasks on agents that can be replicated

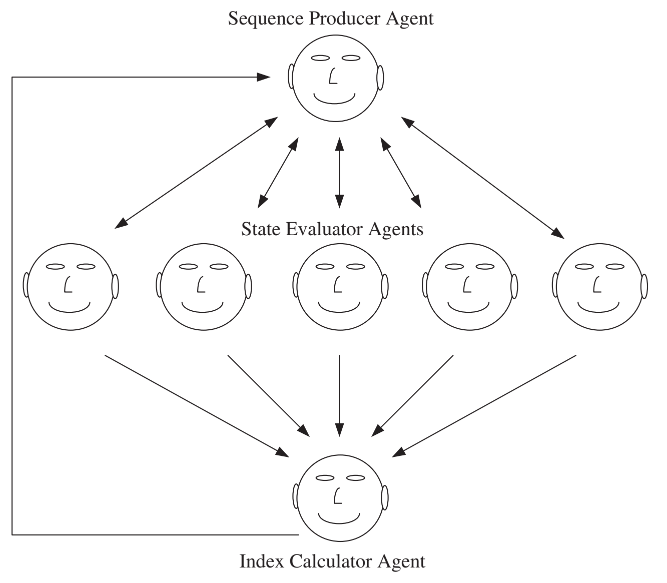

Fig. 3. Agent communication in the first approach.

Table 1
Measure average timing consuming tasks (50-year sequence).

Tool	Identification	Tasks	CPU time (s)	
			RTS	BSS
Power agent platform	SP agent	1–3	0.3050	0.8355
	SE agent	4–5	2.9424	19.1360
	IC agent	6–7	0.0110	0.0218

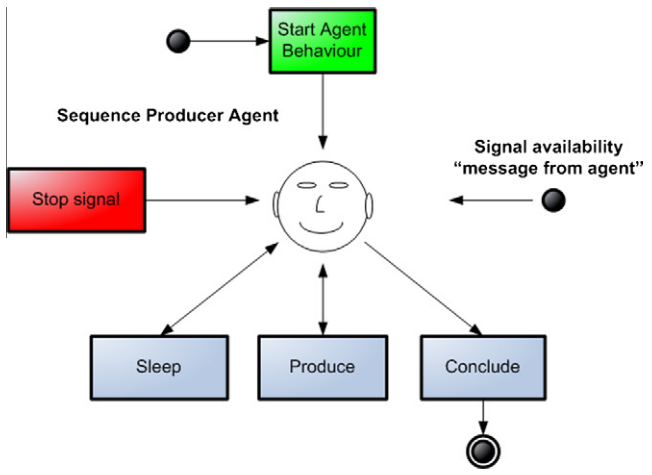

Fig. 4. SP agent behavior: non-synchronized approach.

and work on a parallel sense, that is, to make agents work together in different parts of the evaluation. Monte Carlo techniques allow this kind of approach because of their inherent characteristic to produce independent blocks of information. These blocks, as it will be discussed later in this work, are composed of yearly sequences of system states. These sequences can be separately generated, and, in this case, the initial states of the generating units (i.e. if up or down) are sampled according to the corresponding unavailabilities. This sampling process removes possible biases in the simulation. The other option is to connect chronologically all yearly sequences y_k within those blocks previously mentioned. This is the procedure adopted in this work. Thus, an agent produces as many blocks of system states as necessary to assess the reliability indices, these blocks are sent to other agents to carry out the evaluation of the associated state sequences, and then resent to another agent to estimate the indices and check the convergence. Fig. 3 shows the idea of this first approach.

It can be observed that the SP agent works in the third level of the platform producing sequence of states for the system. This agent uses the other two levels: MCS tools (second level) and the power systems representation (first level). The SP agent behaves differently bearing in mind the following two major tasks: first, to generate the state sequences, and second to communicate with the SE agents. In the second task, the SP agent keeps the communication and the control of the simulation, receiving information from other agents to manage the messages sent.

Note that there is a clear concurrent process for all tasks, in which the message exchanges must be minimized in order to maximize the simulation process efficiency and speed-up. One of the most important tasks of the SP agent is to distribute work (i.e. blocks of information through dynamic sequence matrices) to the SE agents. This important behavior ensures the complete control of the process, that is, when the SP agent sends a message to the SE agent, it receives two replies: one to confirm whether the message was sent correctly or not, and another one when the SE agent ends its task and states that it is free again to receive other dynamic sequence matrices to evaluate. Therefore, it is possible to control the availability of agents.

This approach was called non-synchronized because there are agents that end their tasks and remain idle, waiting to receive more work, thus wasting CPU time. Obviously, this means that the simulation process loses efficiency although this non-synchronized approach is very simple to be implemented. In the next sub-section, a more sophisticated approach to minimize waiting times will be presented.

In order to illustrate the importance of creating more agents for the SE, a straightforward test is carried out, where only one agent is used for each level. Table 1 presents the CPU time, in second (s), spent by each agent to assess generating capacity reliability indices for the IEEE-RTS [23] and BSS systems. The detail characteristics of these systems are provided in Section 4. As it can be observed, considering the IEEE-RTS, the CPU time spent by the SE agent is about 10 and 268 times greater, respectively, than the CPU time spent by the SP and IC agents. Considering the BSS system, these values become 23 and 877, respectively. For the RTS, 2500 years were used as the stopping criterion, which ensures a coefficient of variation (β), set for the expected energy not-supplied (EENS) index, of approximately 5%. For the BSS, the MCS stops at $\beta = 5\%$ for the EENS index (i.e. 1340 years). These cases were performed in an Intel Core 2 Duo with 2.66 GHz.

The convergence characteristics are not so much related with the system size, but mainly with the probability of system failure. The previous example has proved this well-known characteristic in power system reliability evaluation and, therefore, justifies the use of more agents to act as SE to improve the simulation process efficiency. However, if necessary, one can increase the number of SP and IC agents as well.

There are other variables that should also be considered when defining the main characteristics (including the number of agents per level) of any MAS application. In this particular reliability application involving sequential MCS, the size of the dynamic matrices (i.e. hash tables [12]) being exchanged among agent levels is quite relevant. Moreover, the hardware processing capacity associated with the agents must also be taken into account. Finally, when the SP agent receives the stop signal from the IC agent, it propagates this signal for all agents and so the work is completed.

As previously mentioned, the SP agent behaves in a different way as illustrated in Fig. 4. There are three possible statuses for this

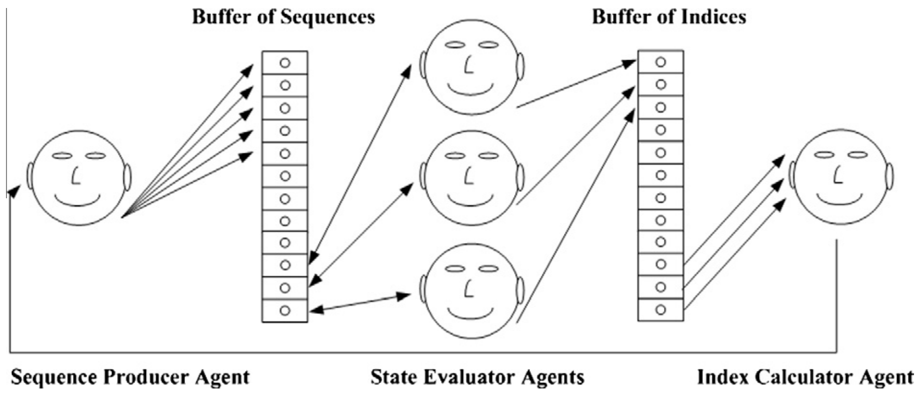

Fig. 5. Agent communication in the second approach.

agent at the beginning of its behavior. First of all, it looks for SE agents registered in the platform and verifies their availability to perform a task. If it does not find any SE agent available, the SP agent immediately enters in sleep mode, until it receives a new message from an SE agent. Once the SP agent perceives an availability signal, it starts generating yearly sequences of states until completing a dynamic sequence matrix to be promptly sent to the available SE agent. From time to time, this process is repeated until a stop signal is sent by the IC agent. The agent behavior is built on the threads process, which is offered within the Java technology. Threads in Java are processes that run in parallel within the Java Virtual Machine [12].

3.5. Synchronized approach

In this sub-section, another approach to deal with the division of tasks among agents is presented. The major idea is to minimize the waiting time between generations of yearly state sequences, through better synchronized actions among agents. This will be accomplished by avoiding the use of messages for all communication processes and allowing the agents to manage their own remote calls. Fig. 5 shows this new approach to communicate and includes in the process the concept of buffer: a device (computer area) that temporarily stores data that is being transferred between two machines that process data at different rates.

The buffer of sequences enables synchronized tasks between all agents, since the SP agent can produce its dynamic sequence matrix independently, and place it in the buffer of sequences, so any SE agent can get its tasks from this buffer. Therefore, more autonomy is being given to the agents. From Table 1, it can be observed that the SP agent is very fast and, most of the time, it ends its tasks before all SE agents. However, the use of buffers for storing state sequences and index updates may lead to some known problems, such as:

- Memory heap space (buffer);
- Size of dynamic matrices (remote calls);
- Serialized network communication.

These problems can be solved using some computational procedures available in Java Technology [12]. Basically, the agents must have the ability of automatically switching to the sleep mode, each time the buffer of sequences is full. The agents should remain in this mode during a certain period of time until the buffer has been partially emptied. This ability enables the optimization of the number of SE agents necessary to carry out the tasks, thus reducing the waiting time. Whenever an SP agent switches to sleep mode, a new group of SE agents must come to assist with the evaluation

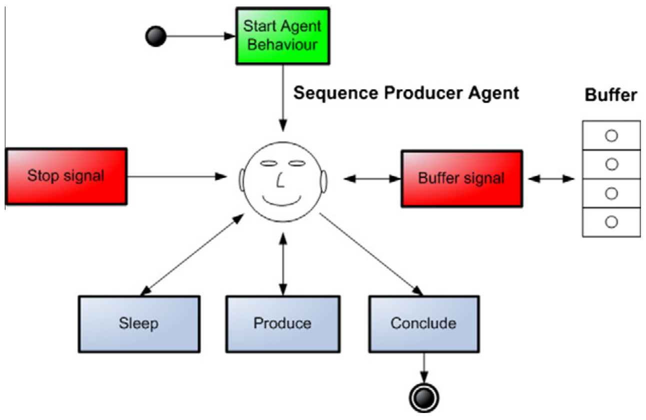

Fig. 6. SP agent behavior: synchronized approach.

tasks. This behavior is especially designed to provide more speed-up and efficiency to the sequential MCS process using intelligent agents.

The same principles applied to the buffer of sequences are also used for the buffer of indices. In this case, this task is simpler since only one IC agent is being shown but, as previously commented, more agents of this type can be employed. The most important task is to verify whether the estimate accuracies are acceptable or not, and send a single message to the SP agent to stop the simulation. For that, a conventional communication process (single message) is used.

It is not always possible to improve the computational efficiency by only increasing the number of SE agents. As discussed for the non-synchronized approach, the synchronized scheme also depends on the definition of some parameters to reach its best performance. The adequate dimensioning of the buffers for sequences and indices, the close monitoring of the sleep mode time, the careful choice of agents with different processing capacity, etc. will determine the computational efficiency, together with the quality of Java programming techniques used to solve the application.

It should be observed that the number of SP and IC agents can be eventually increased to improve the overall performance. The main concern is to maximize the usage of agents with a minimum number of them in order to reduce the CPU time to acceptable levels.

Similarly to the non-synchronized approach, the SP agent behaves in a more complex way as illustrated in Fig. 6. There are also three possible statuses for this agent at the beginning of its behavior. Here, however, the inclusion of the buffer concept allows the agent to work without looking for other agents, thus eliminating the use of messages. The SP agent can, therefore, work regardless of the SE agent availabilities.

The SP agent begins to produce the yearly sequences of states and sends them to the buffer through a dynamic sequence matrix. If the buffer has space, the SP agent keeps its status of generating sequences, until the buffer is full and then it signals to switch the SP agent status to the sleep mode. Although this action can indicate that time is being wasted, it can also be used to wake up other SE agent. The single message used here is the stop signal, which is sent by the IC agent to conclude the simulation process.

The SP agent behavior can be called synchronized because it has more autonomy and minimizes *waiting times* for all tasks among agents. Observe that the SE agents do not need to wait for their task; they only access the buffer through a remote call and get the dynamic sequence matrices to evaluate. Finally, the SP agent, before switching to the conclude status, propagates the stop signal to all agents. Undoubtedly, the synchronized approach is computationally more efficient than the non-synchronized approach, although the associate programming is indeed more complex.

4. Results

The evaluation of the proposed MAS technology-based platform to assess generating capacity reliability indices will be verified as follows. First, the proposed sequential MCS process is applied to the IEEE-RTS [23] and the obtained results are compared with two other programming Fortran standards: an analytical program [24] and a chronological MCS program [25]. The basic idea is to ensure that the Java language can be used for scientific applications. This comparison is followed by other two sub-sections where the results with the non-synchronized and synchronized approaches are shown and discussed.

4.1. Java environment calculations

The analytical program used is written in Fortran language and is based on very fast convolution techniques. It uses a generalized frequency and duration methodology, whose accuracy can be controlled by a *capacity rounding increment* (CRI), in MW, and also by a *truncation probability* (TProb) [24]. The other program used is also written in Fortran and has several modelling features to cope with hydrological inflows, wind power generation, etc. It is able to evaluate all sorts of reliability indices including those from the well-being analysis [25]. Obviously, the full potential of these two Fortran algorithms will not be completely exploited with the simple IEEE-RTS: 32 generating units, 3405 MW of installed capacity with a load model covering 8736 h. However, this system is a benchmark whose results can be easily replicated.

Three indices are chosen to carry out the comparisons: LOLE = loss of load expectation, EENS = expected energy not-supplied, and LOLF = loss of load frequency. For the analytical program, the following parameters are used: CRI = 1 MW and TProb = 10^{-16}. Considering the sequential MCS, 2500 years are used in the simulation process. Table 2 shows the results achieved with the three tools: the analytical [24], the sequential MCS [25], both using the conventional Fortran language, and the sequential MCS using the Java platform. Taking the analytical results as reference, the accuracies of both sequential simulations are within the coefficient of variation related with 2500 simulated years, i.e. $\beta \cong 5\%$.

Analytical generating capacity reliability assessment algorithms are very efficient tools from the computational point of view. However, they are based on Markovian models, which drastically reduce their ability to deal with several real applications [25]. Conversely, sequential or chronological MCS algorithms are very flexible tools, from the modelling point of view, but also very time consuming techniques. Therefore, it is inadequate the comparison between analytical and sequential MCS in terms of computational effort. However, the comparison between the two sequential MCS, i.e. the one from [25] written in Fortran and the one proposed here written in Java language and run under a MAS platform (25-year sequence), can be done.

In this first test, to provide a better idea in relation to the programming language, only one agent is used for each task. The proposed Java-based algorithm spent 241 s while the Fortran-based algorithm spent 124 s. All cases were performed in a Pentium IV processor with 2.4 GHz. It must be pointed out that the latter is a much more sophisticated program with several features available as compared with the Java-based program. Moreover, Fortran has more efficient libraries available for scientific developments than Java. However, the performance of the proposed Java-based algorithm is very good, bearing in mind that the language potential for flexible communication was not yet explored in this example, since only one SE agent was used.

In order to assess the full potential of the proposed MAS-based algorithm, the IEEE-RTS and the BSS systems are tested considering both the non-synchronized and the synchronized approaches. The configuration used for the BSS contains 413 buses, 685 circuits and 242 generating units. The installed capacity and annual peak load are equal to 46 GW and 41 GW, respectively. This was the configuration used in planning studies during the 1990s. A typical annual curve, with 8736 levels, is used to represent the behavior of the hourly load in all buses of the system.

4.2. Non-synchronized approach

Firstly, 15 personal computing machines, all connected in a local area network, with different processing capacities were chosen to participate in this test. One machine was selected to act as the SP agent, up to eight machines were chosen to work as SE agents, and one machine was also selected to perform as the IC agent. The generation of yearly sequences varied in sizes of 25, 50, 75 and 100 years. Fig. 7 shows sensitivity tests for the IEEE-RTS,

Table 2
Comparison results with three different tools (IEEE-RTS).

Indices	Analytical [24]	Sequential MCS [25]	Proposed MCS–MAS
LOLE (h/year)	9.3942	9.4984	9.3690
EENS (MWh/year)	1176.3	1162.3	1158.4
LOLF (occ./year)	2.0197	2.0732	2.0388

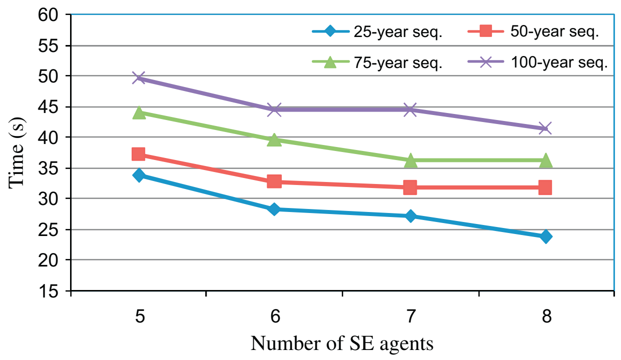

Fig. 7. Non-synchronized approach: sensitivity tests (IEEE-RTS).

Table 3
Comparison of CPU time (25-year sequence).

Approach	No. of SE agents	IEEE-RTS time (s)	BSS time (s)
Sequential	1	241	786.4
Non-synchronized	8	23.8	79.7
Synchronized	8	17.2	58.1

considering these parameter variations, i.e. the number of SP agents and the sequence sizes.

As it can be observed, the overall CPU times range between 23 and 50 s, depending on the number of SE agents and on the size of the sequences sent by the SP agent. Moreover, in all cases the CPU times saturate in a narrower range between 23 and 42 s. It can be concluded that it is not worth increasing the number of SE agents beyond a certain number. In a Java environment, the relationship between producers (i.e. SP agent) and consumers (i.e. SE agents) is extremely important in order to reach the best computational benefits [12]. Clearly, two explanations can be given for the observed saturation. First, the SP agent is running with full capacity, generating state sequences without ever entering in the sleep mode. This condition can only be improved by providing more SP agents to the proposed MAS-based framework. The second reason is that the relationship between producer and consumer is not enough organized, for instance due to the excessive number of message exchanges, which can be minimized by using the proposed synchronized approach to be shown in the next sub-section.

Considering 8 SE agents and a 25-year sequence, it can be observed in Table 3 that the total CPU times to assess the reliability indices are 23.8 s and 79.7 s for the IEEE-RTS and BSS systems, respectively. These values can be compared with those obtained with only one SE agent, also shown in Table 3. From these results, it is possible to evaluate the speed-up (S_u) and the computational efficiency (η), expressing the use of computing resources as compared to an ideal case of optimal performance [26]. For these systems: IEEE-RTS $\Rightarrow S_U \cong 10.1$ and $\eta \cong 71.2\%$, BSS $\Rightarrow S_U \cong 9.86$ and $\eta \cong 72.7\%$.

4.3. Synchronized approach

The MAS technology was initially based on the distributed computation concept and uses TCP/IP protocol, which does not allow fast remote call or fast interchange message. There are other communication techniques, which enable fast communication, or improve efficient remote call between computers, e.g. Java RMI (Remote Method Invocation) [27]. Java RMI enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and *unmarshal* parameters and does not truncate types, supporting true object-oriented polymorphism. Therefore, the proposed synchronized approach was designed using this technology.

Considering the same 15 machines used with the non-synchronized approach, Fig. 8 shows the same sensitivity performance tests using the synchronized approach. As it can be observed, the overall CPU times range between 15 and 50 s, depending on the number of SE agents and on the size of the sequences sent by the SP agent. Moreover, in all cases, the CPU times start saturating in a narrower range between 15 and 32 s. Cleary, it can be noted that the saturation point is shifted to 8 SE agents, as compared to the non-synchronized case. This proves the benefits of the buffers and the use of a more appropriate Java environment.

Considering 8 SE agents and a 25-year sequence, it can be observed in Table 3 that the total CPU times to assess the reliability

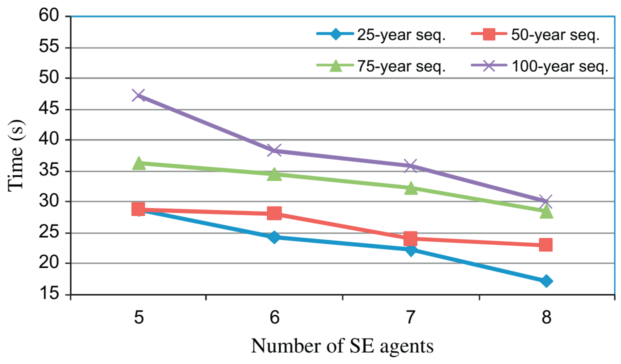

Fig. 8. Synchronized approach: sensitivity tests (IEEE-RTS).

indices are 17.2 s and 58.1 s for the IEEE-RTS and BSS systems, respectively. For these conditions, the speed-up and efficiency performance indices are: IEEE-RTS $\Rightarrow S_U \cong 14.0$ and $\eta \cong 98.8\%$, BSS $\Rightarrow S_U \cong 13.5$ and $\eta \cong 99.7\%$. Therefore, the speed-up and efficiency depend on the sequence size, number of agents per level and computational power of the involved processors.

5. Final remarks

Power systems analysis is known as a research subject strongly linked to computation and mathematical areas, but extremely rigid to change concepts and tools, particularly those related with programming paradigms. Bearing in mind this aspect, this paper presented a new way to approach one typical power system problem, the generating capacity reliability assessment, using MAS technology. It was proved that is possible to use this technology to achieve very good results based on two structures: non-synchronized and synchronized. Although the first one is very simple to handle, the latter usually achieves higher computing performance. However, there are several aspects related with the MAS environment that should be taken into account: organization of tasks, number of agents per task, size of the local network, information exchange, etc. The results were achieved using regular computers, that is, personal computers used everyday and linked to conventional networks.

The MAS technology is very attractive since it merges artificial intelligence and distributed computation areas, in a simple way. This technology is ready to be applied to complex power system evaluations, and can achieve good results due to its main characteristic: the communication flexibility. Java environment offers several features, including the possibility to encapsulate programs/codes written in other languages (e.g. Fortran or C). Thus, there should be no concern among power system researchers and engineers that well tested programs will have to be rewritten. Moreover, other methods to improve computational efficiency, such as variance reduction techniques usually used in problems involving Monte Carlo simulation, can be easily established in this new framework.

The extension of the proposed methodology to other power reliability problems or applications involving population-based heuristics is almost straightforward. This work opens the way to build models where more intelligence is added to the agents, so that they may autonomously organize themselves to better manage the computational efficiency and speed-up.

References

[1] McArthur SDJ, Davidson EM, Catterson VM, Dimeas AL, Hatziargyriou ND, Ponci F, et al. Multi-agent systems for power engineering applications – Part I: concepts, approaches, and technical challenges. IEEE Trans Power Syst 2007;22(4):1743–52.

[2] McArthur SDJ, Davidson EM, Catterson VM, Dimeas AL, Hatziargyriou ND, Ponci F, et al. Multi-agent systems for power engineering applications – Part II: technologies, standards, and tools for building multi-agent systems. IEEE Trans Power Syst 2007;22(4):1753–9.
[3] Koritarov VS. Real-world market representation with agents. IEEE Power Energy Mag 2004;2(4):39–46.
[4] Oo NW, Miranda V. Multi-energy retail market simulation with intelligent agents. In: Proc of the IEEE power technol; 2005.
[5] Nagata T, Sasaki H. A multi-agent approach to power system restoration. IEEE Trans Power Syst 2002;17(2):457–62.
[6] Dimeas AL, Hatziargyriou ND. Operation of a multiagent system for microgrid control. IEEE Trans Power Syst 2005;20(2):1447–55.
[7] Hopkinson K, Wang X, Giovanine R, Thorp J, Birman K, Coury D. EPOCHS: a platform for agent-based electric power and communication simulation build from commercial off-the-shelf components. IEEE Trans Power Syst 2006;21(2):548–58.
[8] Borges CLT, Falcão DM, Mello JCO, Melo ACG. Composite reliability evaluation by sequential Monte Carlo simulation on parallel and distributed processing environments. IEEE Trans Power Syst 2001;16(2):203–9.
[9] Biskas PN, Bakirtzes AG, Macheras NI, Pasialis NK. A decentralized implementation of DC optimal power flow on a network of computers. IEEE Trans Power Syst 2005;20(1):25–33.
[10] Shu J, Xue W, Zheng W. A parallel transient stability simulation for power system. IEEE Trans Power Syst 2005;20(4):1709–17.
[11] Bigus JP, Bigus J. Constructing intelligent agents using Java. 2nd ed. New York: John Wiley & Sons, Inc.; 2001.
[12] Deitel HM, Deitel PJ, Santry SE. Advanced Java 2 platform: how to program. New Jersey: Prentice-Hall; 2002.
[13] Bellifemine F, Caire G, Greenwood D. Developing multi-agent systems with JADE. Chichester UK: John Wiley & Sons, Ltd; 2007.
[14] Wooldridge M. An introduction to multiagent system. Chichester UK: John Wiley & Sons; 2002.
[15] Leite da Silva AM, Resende LC, Manso LAF, Miranda V. Composite reliability assessment based on Monte Carlo simulation and artificial neural networks. IEEE Trans Power Syst 2007;22(3):1202–9.
[16] Silva MG, Rodrigues AB, Castro CLC, Coelho NA, Moutinho EA, Correa AC. An application of predictive reliability analysis techniques in Brazil's NorthEast distribution networks. Int J Electric Power Energy Syst 2007;29(2):155–62.
[17] Rubinstein RY. Simulation and Monte Carlo method. New York: John Wiley & Sons; 1980.
[18] Anders GJ, Endrenyi J, Pereira MV, Pinto LMVG, Oliveira CG, Cunha SHF. A fast Monte-Carlo simulation technique for power system reliability studies. In: Proc CIGRE; 1990.
[19] Leite da Silva AM, Manso LAF, Mello JCO, Billinton R. Pseudo-chronological simulation for composite reliability analysis with time varying loads. IEEE Trans Power Syst 2000;15(1):73–80.
[20] Wang L, Singh C. Population-based intelligent search in reliability evaluation of generation systems with wind power penetration. IEEE Trans Power Syst 2008;23(3):1336–45.
[21] Samaan N, Singh C. Adequacy assessment of power system generation using a modified simple genetic algorithm. IEEE Trans Power Syst 2002;17(4):974–81.
[22] Billinton R, Allan RN. Reliability evaluation of power systems. New York: Plenum Press; 1996.
[23] IEEE APM subcommittee. IEEE reliability test system. IEEE Trans PAS 1979; 99(6): 2047–54.
[24] Leite da Silva AM, Mello ACG, Cunha SHF. Frequency and duration method for reliability evaluation of large-scale hydrothermal generating systems. IEE Proc C 1991;138(1):94–102.
[25] Leite da Silva AM, Manso LAF, Sales WS, Resende LC, Aguiar MJQ, Matos MA, et al. Application of Monte Carlo simulation to generating system well-being analysis considering renewable sources. Eur Trans Electric Power 2007;17(4):387–400.
[26] Kumm ET, Lea RM. Parallel computing efficiency: climbing the learning curve. In: Proc of the TENCON'94 – IEEE region 10's ninth annual international conference – frontiers of computer technology; 1990.
[27] Maassen J, Nieuwpoort RV, Veldema R, Bal H, Kielmann T, Jacobs C, et al. Efficient Java RMI for parallel programming. ACM Trans Progr Lang Syst 2001;23(6):747–75.