Computers & Geosciences 37 (2011) 1110–1121

Contents lists available at ScienceDirect

Computers & Geosciences

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

Parallel implementation of simulated annealing to reproduce multiple-point statistics

Oscar Peredo [a], Julián M. Ortiz [b,c,*]

[a] Department of Computer Sciences, University de Chile, Chile
[b] ALGES Lab, Advanced Mining Technology Center, University of Chile, Chile
[c] Department of Mining Engineering, University of Chile, Chile

ARTICLE INFO

Article history:
Received 12 March 2010
Received in revised form
16 October 2010
Accepted 20 October 2010
Available online 23 November 2010

Keywords:
Geostatistics
Stochastic simulation
Multipoint
Speculative computation
Parallel computing

ABSTRACT

This paper shows an innovative implementation of simulated annealing in the context of parallel computing. Details regarding the use of parallel computing through a cluster of processors, as well as the implementation decisions, are provided. Simulated annealing is presented aiming at the generation of stochastic realizations of categorical variables reproducing multiple-point statistics.

The procedure starts with the use of a training image to determine the frequencies of occurrence of particular configurations of nodes and values. These frequencies are used as target statistics that must be matched by the stochastic images generated with the algorithm. The simulation process considers an initial random image of the spatial distribution of the categories. Nodes are perturbed randomly and after each perturbation the mismatch between the target statistics and the current statistics of the image is calculated. The perturbation is accepted if the statistics are closer to the target, or conditionally rejected if not, based on the annealing schedule.

The simulation was implemented using parallel processes with C++ and MPI. The message passing scheme was implemented using a speculative computation framework, by which prior to making the decision of acceptance or rejection of a proposed perturbation, processes already start calculating the next possible perturbation at a second level; one as if the perturbation on level one is accepted, and another process as if the proposed perturbation is rejected. Additional levels can start their calculation as well, conditional to the second level processes. Once a process reaches a decision as to whether accept or reject the suggested perturbation, all processes within the branch incompatible with that decision are dropped. This allows a speed up of up to $\log_n(p+1)$, where n is the number of categories and p the number of processes simultaneously active.

Examples are provided to demonstrate improvements and speed ups that can be achieved.

1. Introduction

Numerical modeling with geostatistical techniques aims at characterizing natural phenomena by summarizing and using the spatial correlation for the statistical inference of parameters of the distribution of uncertainty at unsampled locations in space. In simulation techniques, this spatial correlation is imposed into a model commonly constructed on a regular lattice. The models must reproduce the statistical (histogram) and spatial distribution (variogram or other spatial statistics) and their quality is often judged in terms of the reproduction of geological features (Ortiz and Peredo, 2009).

Conventional techniques in geostatistics address the modeling using statistical measures of spatial correlation that quantify the expected dissimilarity (transition to a different category) between locations separated by a given vector distance, in reference to a given attribute, such as the facies, rock type, porosity, grade of an element of interest, etc. This is done using the variogram. Limitations of these techniques have been pointed out in that they only account for two locations at a time when defining the spatial structure. Much richer features can be captured by considering multiple-point statistics that consider the simultaneous arrangement of the attribute of interest at several locations, providing the possibility to account for complex features, such as hierarchy between facies, delay effects, superposition, curvilinearity, etc. There are several approaches to simulate accounting for multiple-point statistics. Modifications of conventional methods to impose local directions of continuity using the variogram is a simple approach to impose some of the complex geological features (Xu, 1996; Zanon, 2004). Object-based methods and methods inspired

* Corresponding author at: ALGES Lab, Advanced Mining Technology Center, University of Chile, Chile.
E-mail addresses: operedo@dcc.uchile.cl (O. Peredo), jortiz@ing.uchile.cl (J.M. Ortiz).

0098-3004/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2010.10.015

in the genetic rules and physics of the deposition of sediments in different environments also seek to overcome the limitations of conventional categorical simulation techniques with significant progress (Deutsch and Wang, 1996; Tjelmeland, 1996; Pyrcz and Strebelle, 2008).

Presently, the most popular method is a sequential approach based on Bayes' postulate to infer the conditional distribution from the frequencies of multiple-point arrangements obtained from a training image. This method, originally proposed by Guardiano and Srivastava (1993), and later efficiently implemented by Strebelle and Journel (2000), is called single normal equation simulation (snesim) (see also Strebelle, 2002). This method has been the foundation for many variants such as simulating directly full patterns (Arpat and Caers, 2007; Eskandari and Srinivasan, 2007) and using filters to approximate the patterns (Zhang et al., 2006). The use of a Gibbs sampling algorithm to account directly for patterns has also been proposed (Boisvert et al., 2007; Lyster and Deutsch, 2008). A sequential method using a fixed search pattern and a 'unilateral path' also provides good results (Daly, 2005; Daly and Knudby, 2007; Parra and Ortiz, 2009). Other approaches available consider the use of neural networks (Caers and Journel, 1998; Caers and Ma, 2002), updating conditional distributions with multiple-point statistics as auxiliary information (Ortiz, 2003; Ortiz and Deutsch, 2004; Ortiz and Emery, 2005) or secondary variable (Hong et al., 2008), and simulated annealing (Deutsch, 1992).

Simulated annealing provides a very powerful framework to integrate different types of statistics, and potentially, generate models subject to constraints that cannot be handled by other methods. This motivates studying approaches to speed up the iterative process involved in annealing simulation. The recent increase in availability of powerful multiple-processor computers and multiple-core central processing units (CPU), as well as the use of graphics processing units (GPU) for parallelizing the calculations required for some heavy computing tasks, motivates researching new applications in this framework and let us revisit algorithms that were too demanding for the technology existing a few years ago. In this paper, we explore a known paradigm in Computer Science and implement simulated annealing to impose multiple-point statistics. The approach is known as speculative computing, and works in a parallel computing setting, aiming at reducing the time of computation for complex problems. Performance is assessed regarding speedup in computation time and statistical reproduction of multiple-point frequencies in the simulated realizations.

2. Simulated annealing

Simulated annealing is a general optimization algorithm that can generate numerical models by reducing an objective function—usually minimizing a weighted sum of mismatch terms with respect to reference values—reproducing different spatial statistics and respecting constraints imposed in that objective function (Besag, 1986; Farmer, 1992; Geman and Geman, 1984; Kirkpatrick et al., 1983; Rothman, 1985).

In a spatial context, the algorithm can be used to generate models in a lattice, where each node of the grid has a value for a particular property being simulated and the objective function considers some statistical parameter that relates those nodes spatially. Typical applications have been the simulation of categorical variables such as facies or rock types, or the reproduction of continuous variables, such as petrophysical properties (porosity, permeability), conductivity or concentrations of elements (Goovaerts, 1996; Fang and Wang, 1997).

In essence, the algorithm works by perturbing one or more nodes at a time of an initial model, which usually is a model with a random spatial distribution of values. At every step of the process, the mismatch between the current statistics of the model and those required (target statistics) is quantified. If a perturbation reduces the mismatch, this means the simulated model has statistics closer to the target ones, hence the perturbation is kept. If a perturbation increases the mismatch, this means the model has statistics that are more different from the target ones, hence the perturbation should be rejected. However, in simulated annealing some of the unfavourable perturbations are kept, in order to allow the model moving away from local minima and reaching a lower mismatch later on. If a local minimum is reached with a given perturbation and all unfavourable changes were rejected, then it would be impossible to move out of that local minimum to get closer to the global minimum. Depending on the topology of the solution space, unfavourable changes should be rejected with a higher probability. The rate of acceptance of unfavourable perturbations is controlled by the "annealing schedule", which defines the probability of acceptance of bad changes and also controls the way this probability changes as the simulation progresses.

The annealing schedule is defined by an initial temperature and a procedure to lower that temperature as simulation progresses. In addition to these parameters, a perturbation mechanism is required, e.g. swapping nodes, randomly perturbing one or more nodes, etc. and one or more stopping criteria.

The general formulation of the algorithm considers an objective function of the following type:

$$O = \sum_{i=1}^{N_c} w_i O_i \tag{1}$$

where N_c is the number of components in the objective function, w_i are the weights assigned to each one of the components, and O_i is the mismatch value for component i. For example, this function could be composed by the mismatch in histogram reproduction, defined as the squared difference in the cumulative frequencies measured at some quantiles for the model simulated versus the target histogram, a mismatch in variogram reproduction, composed by squared differences between the target variogram model and the variogram calculated from the realization being perturbed, for a number of lag distances, and a mismatch in the reproduction of multiple-point statistics. In general, any constraint can be ensured in a similar fashion. Conditioning data can be imposed simply by not allowing the nodes to be perturbed at sample locations. An important requirement to achieve a low mismatch between model and target statistics is that all the statistics and constraints must be consistent. Since imposing the multiple-point statistics implicitly defines lower order statistics, if statistics are inferred from multiple sources, some inconsistencies are expected. Most often, we do not have control over the consistency of the statistics we try to impose, as these may come from different sources. The use of training images as the basis for inferring the multiple-point statistics can generate problems when its lower order statistics (histogram and variogram) are different than those on the sample data (Ortiz et al., 2007).

In general, the reproduction of a variogram map, indicator variograms, a histogram of multiple-point statistics for some pattern sizes and the requirement of honouring conditional information can be imposed through elements of the objective function.

A typical procedure for implementing the algorithm is:

1. Start with a spatially random distribution of values over all the nodes on the lattice to be simulated, but ensuring that conditioning values at sample locations are honoured.

2. Compute the global objective function, considering all target statistics and set the initial parameters of the annealing schedule.
3. Select a node at random and propose a different value for that location.
4. Evaluate the change in the objective function. This updating can be done by only computing statistics affected by the change. This avoids the lengthy calculation of a usually complex global objective function.
5. Calculate the probability of acceptance of the proposed perturbation. If the change is favourable, the perturbation is accepted; otherwise, the perturbation may be conditionally accepted as dictated by a probability distribution defined by the annealing schedule. The probability of acceptance is given by the Boltzmann distribution:

$$\mathbb{P}(\text{accept}) = \begin{cases} 1 & O_{new} \leq O_{old} \\ e^{(O_{old}-O_{new})/t} & \text{otherwise} \end{cases} \quad (2)$$

where t is a parameter equivalent to the product of the Boltzmann constant k_b and the temperature T in the application to the physical process. By analogy, t is called the temperature in simulated annealing; O_{old} and O_{new} are the values of the objective function before and after the perturbation, equivalent to the difference in Gibbs free energy ΔE in the physical process of annealing. In simulated annealing, the temperature must be lowered as the algorithm runs, emulating the cooling that occurs during the physical process that lets the molecules reorganize to lower energy states.
6. Update the objective function if the change was accepted.
7. Decide if the temperature should be lowered.
8. Check the stopping criteria and stop or go back to 3.

A variety of options regarding the perturbation mechanism (Deutsch and Wen, 2001; Deutsch, 2002), updating the objective function (Deutsch, 2002), and defining the initial temperature (Norrena and Deutsch, 2000) have been proposed. Usually, these parameters strongly depend upon the problem at hand and are set empirically after some trial and error runs.

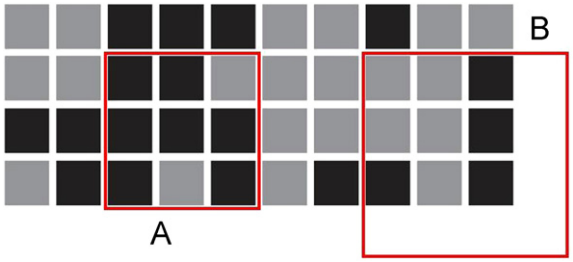

Fig. 1. Pixels and patterns.

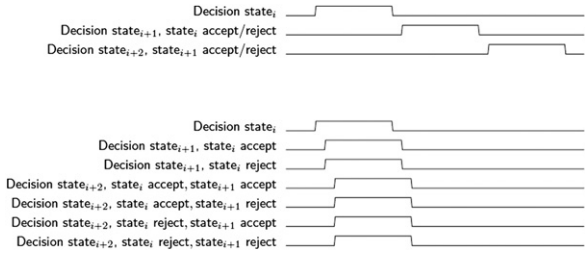

Fig. 2. Comparison of task execution between a sequential (top) and a 7-process parallel (bottom) simulated annealing.

3. Implementation

A 2D implementation of simulated annealing to impose multiple-point statistics in a parallel computation framework is presented. The tools used are C++ and MPI (Pacheco, 1996). This implementation could be extended to 3D and multiple classes, although dimensionality may be an issue as the pattern size and the number of categories increase.

3.1. 2D patterns

The 2D patterns used to calculate the multiple-point statistics are stored using the class `map<string,int>` from the C++ Standard Template Library. The implementation is made to

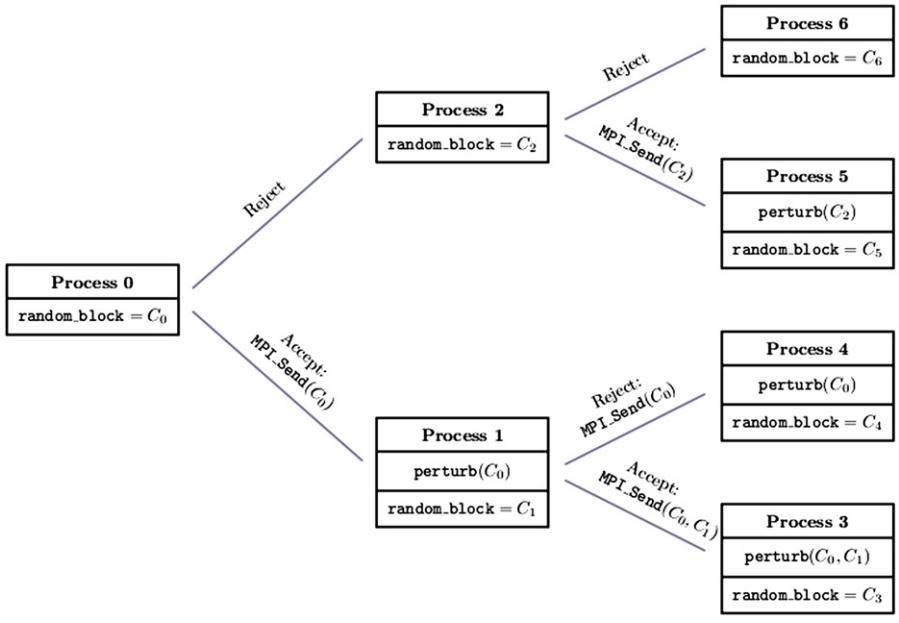

Fig. 3. First step in the message passing scheme.

calculate rectangular patterns and supports two categories of data (binary case), represented by black and grey pixels in Fig. 1. Each pattern is represented by a string of 1's, 0's and X's. The X indicates that there is no information available at that node.

For example, in Fig. 1, the pattern A is represented by the string

```
110111101
```

and the pattern B by

```
001X001X101XXXXX
```

The class `map<string,int>` is used to store the occurrence of the patterns represented by these strings in a training image. That class allows assignment of an integer to each string which acts as a counter. The counter is increased every time an occurrence of the string is detected in the training image.

If the class `map<string,int>` is instantiated by the variable `hash` and `code="110111101"` then, if it is the first occurrence of the pattern represented by that string code, an insertion is performed with frequency equal to 1: `hash.insert(make_pair(code,1))`, and in case it has already been detected, an increment of 1 in the frequency is performed: `hash[code]++`.

3.2. Parallel simulated annealing

Simulated annealing is an inherently sequential algorithm because a decision about accepting or rejecting the current state, defined by the proposed perturbation, must be made and this decision conditions all subsequent states of the model. Implementing simulated annealing in a framework called speculative computation (Burton, 1985; Witte et al., 1991) allows reducing the computation time to achieve models that converge to the required statistics. In that framework, a binary tree of communications is designed, where each node of the tree represents a process launched by an independent CPU. Every time a process

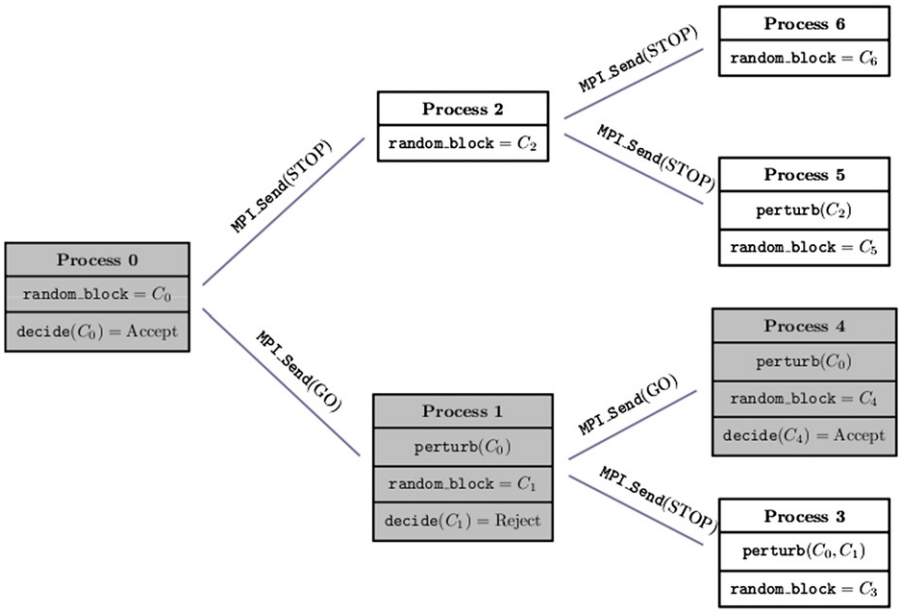

Fig. 4. Second step in the message passing scheme.

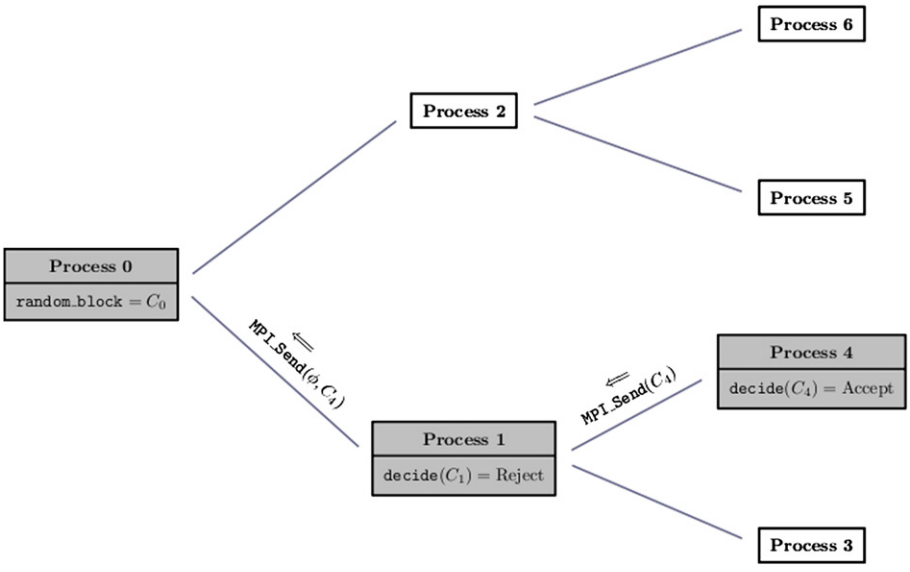

Fig. 5. Third step in the message passing scheme.

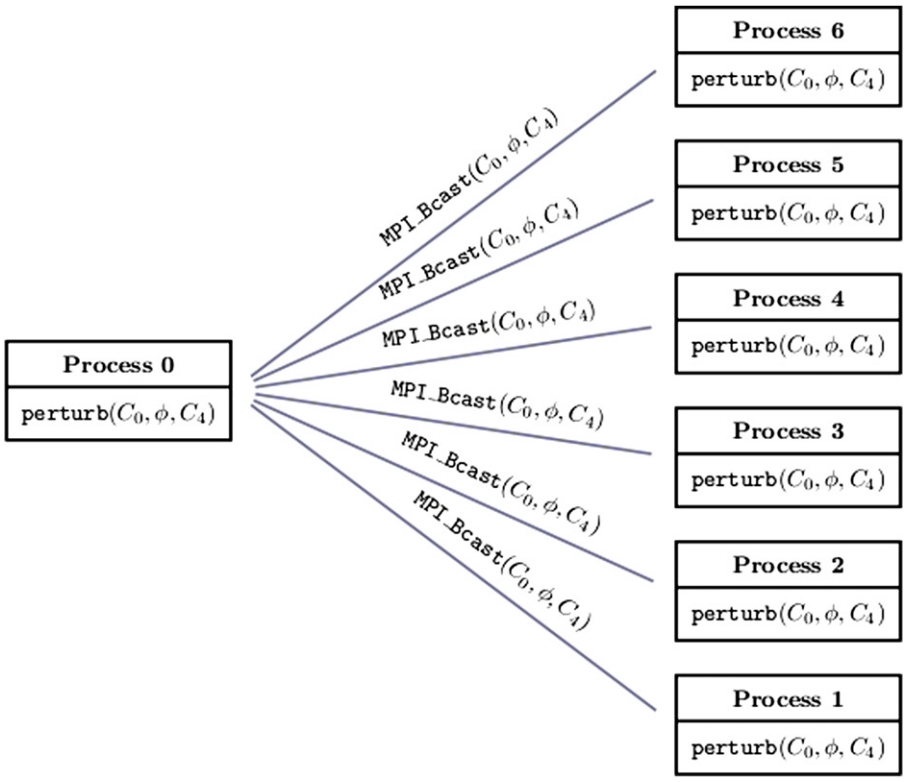

Fig. 6. Last step in the message passing scheme.

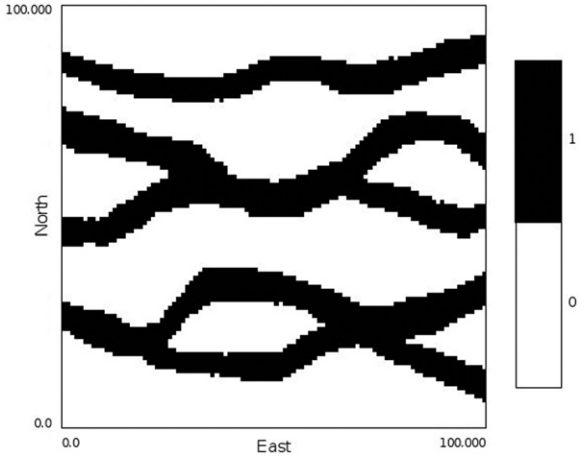

Fig. 7. Training image: channels.

Table 1
Average speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 7.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.75	1.80	1.73	3 processes	1.70	2.26	1.86
7 processes	3.98	1.92	2.24	7 processes	2.60	2.23	2.51

finishes the calculation step, it sends the information to the root process. This results in the evaluation of several states in a time close to the time required for computing a single state in a single process, but requires an additional processing capacity to be available.

Table 2
Maximum values for the speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 7.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.91	1.95	1.82	3 processes	2.47	2.59	1.88
7 processes	5.04	2.06	2.34	7 processes	4.09	2.48	2.54

Table 3
Minimum values for the speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 7.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.36	1.40	1.64	3 processes	1.42	1.69	1.84
7 processes	2.87	1.64	2.17	7 processes	1.49	1.55	2.45

Suppose that we are in the state i and the objective state is the state $i+3$. The path to be followed before reaching that state is $i \to i+1 \to i+2 \to i+3$ and for each step the decision of perturbing the state must be calculated, taking a considerable amount of time (it involves the calculation of the objective function). In this case, three decision evaluations must be calculated, so the decision tree has 2^3-1 nodes (balanced binary tree). In the sequential simulated annealing, the only process involved must evaluate three tasks sequentially. In the parallel simulated annealing, where each node of the decision tree if evaluated by a different process, the tasks are the following (one task per process):

- The decision for state i.
- The decision for state $i+1$, assuming that the decision for state i was accept.

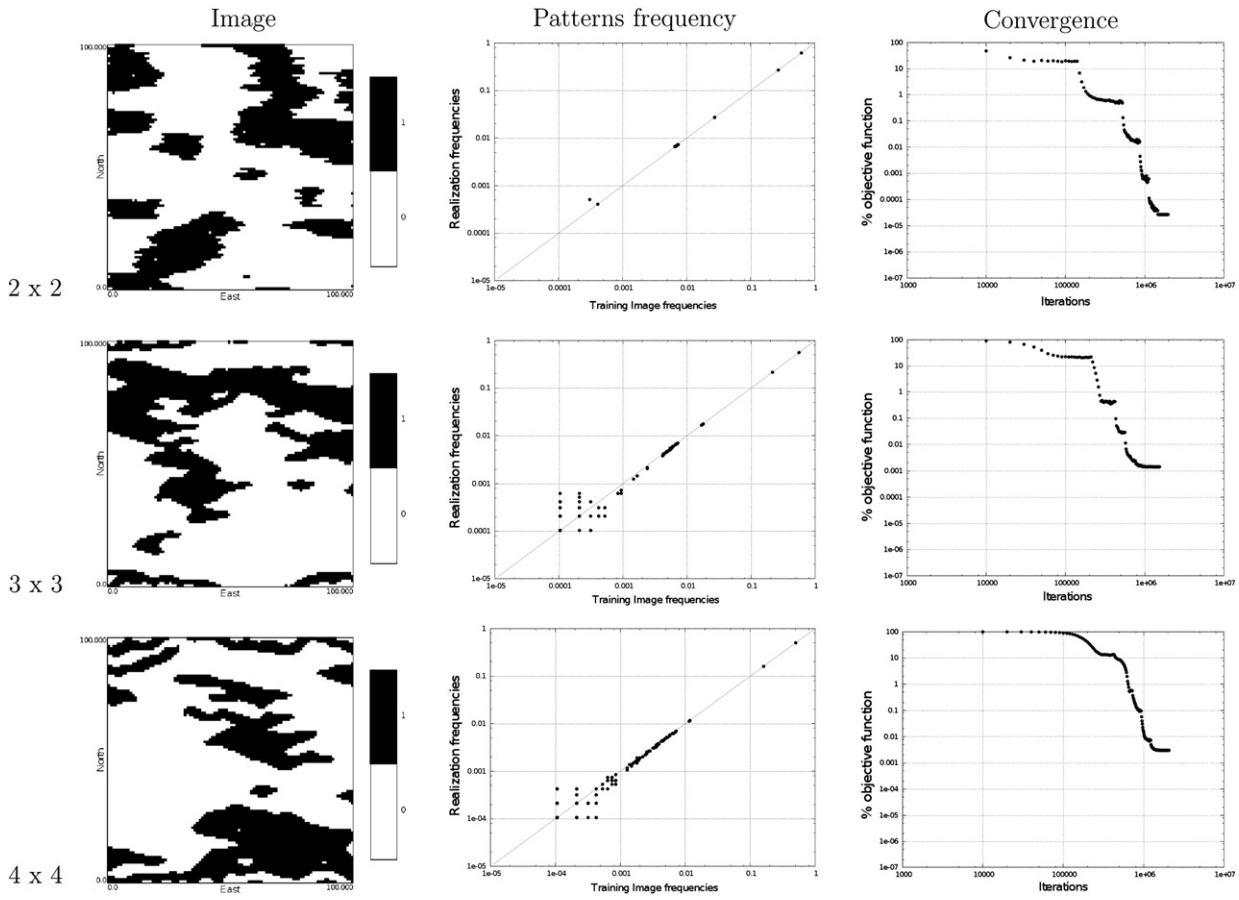

Fig. 8. Simulated images: channels (equal weighting).

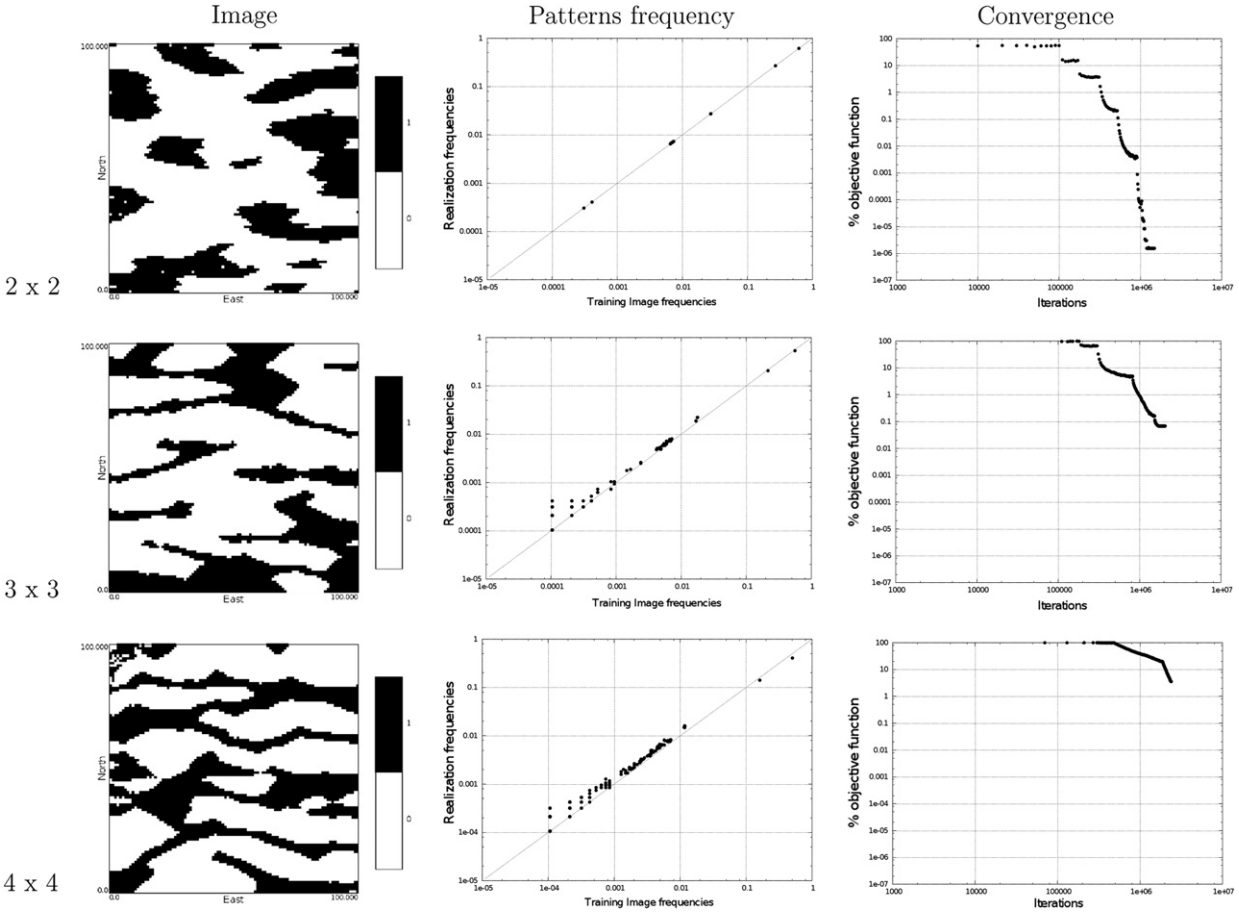

Fig. 9. Simulated images: channels (inversely proportional weighting).

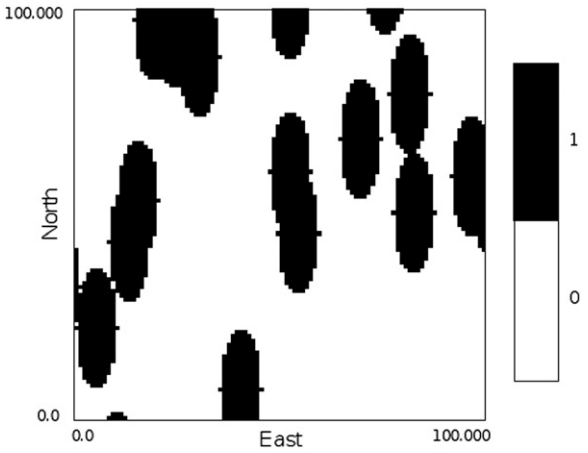

Fig. 10. Training image: ellipses 0.

Table 4
Average speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 10.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	2.19	2.38	1.71	3 processes	2.15	2.16	1.85
7 processes	3.71	2.82	2.22	7 processes	4.02	2.31	2.38

- The decision for state $i+1$, assuming that the decision for state i was reject.
- The decision for state $i+2$, assuming that the decision for state i was accept and assuming that the decision for state $i+1$ was accept.
- The decision for state $i+2$, assuming that the decision for state i was accept and assuming that the decision for state $i+1$ was reject.
- The decision for state $i+2$, assuming that the decision for state i was reject and assuming that the decision for state $i+1$ was accept.
- The decision for state $i+2$, assuming that the decision for state i was reject and assuming that the decision for state $i+1$ was reject.

With this scheme it is expected that the entire decision tree will be calculated in an amount of time similar to the time of evaluation of the first task (decision for state i), so when the first task is completed, giving a result of accept or reject, the path to the corresponding leaf of the tree will be already calculated and the new state will be $i+3$ (see Fig. 2).

3.3. Message passing scheme

An example with 7 processes in a balanced binary tree topology illustrates the message passing scheme. Considering an initial state of the model, the first step consists in applying perturbations in each node of the tree assuming that the corresponding parent node accepted the proposed perturbation. In Fig. 3, the process 0 obtains random coordinates C_0 (perturbation of process 0) and sends the coordinates to process 1 (accepter). The process 1 must update its configuration as soon as it gets the coordinates from its parent node. Then, the process 1 sends C_0 to the processes 3 and 4, and sends its random coordinates C_1 (perturbation of process 1) to process 3 (accepter). The process 3 must update its configuration in the coordinates C_0 and then C_1, in that order, and the process 4 must update its configuration in the coordinate C_0. A similar situation occurs in the other branch with the processes 2, 5 and 6 (rejectors).

In summary, each process performs the updates corresponding to the random coordinates obtained by its parent, and the parent of the parent and so on, depending on its condition of accepter or rejector, storing the coordinate set from its parent and its own coordinate, which will be evaluated in the next step.

Table 5
Maximum values for the speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 10.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	3.45	2.99	1.85	3 processes	3.94	2.98	1.90
7 processes	7.07	4.08	2.33	7 processes	9.17	2.91	2.55

Table 6
Minimum values for the speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 10.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.67	1.75	1.40	3 processes	1.27	1.68	1.79
7 processes	2.23	1.51	2.15	7 processes	1.31	1.64	2.13

In the second step, after updating the configurations for every node, the decision of accepting or rejecting the random coordinate chosen by each process must be computed. If the process 0 accepts a perturbation in C_0, the rejection branch (processes 2, 5 and 6) is discarded, sending a STOP signal to those processes (Fig. 4). The procedure is performed in the next node, in this case, the node associated with process 1. In Fig. 3, the final path is $C_0 \rightarrow \phi \rightarrow C_4$, where ϕ indicates that no perturbation is performed (a rejection branch).

The next step is to report every process about the path of perturbations, $C_0 \rightarrow \phi \rightarrow C_4$. This is done in two sub-steps: (1) the path must be sent back to the root node, associated with process 0. To send the path of perturbations to process 0, it has to travel backwards through all the nodes associated with the path to which a GO signal was sent. In this example, this occurs with processes 4 and 1. In Fig. 5, process 4 sends C_4 to process 1 because it was decided to accept C_4, then process 1 sends ϕ and C_4 to process 0 (ϕ represents a rejection). Once those values are received by process 0, a decision is made whether to include or not C_0 in the path. In this example, C_0 is accepted and the final path to be broadcasted is $C_0 \rightarrow \phi \rightarrow C_4$. And, (2) the path must be sent from the root to every other node using a broadcast function from MPI (`MPI_Bcast`) and then each process synchronizes its configuration with respect to the root node associated with process 0 (Fig. 6).

3.4. Speedup

3.4.1. Amdahl's law

To obtain a theoretical estimation for the speedup using the speculative framework, let us first recall Amdahl's law (Amdahl, 1967) which gives a theoretical bound for the speedup of a general potentially parallel application. The speedup is given by

$$S(P) = \frac{T_\sigma}{(1-r)T_\sigma + \frac{rT_\sigma}{P}} \tag{3}$$

where P is the number of processes, T_σ represents the sequential processing time (with one process) and r is the percent of possible parallel computations. $S(P)$ is the ratio between the sequential processing time and the parallel processing time. Cancelling T_σ and calculating the derivative with respect to P gives

$$\frac{dS(P)}{dP} = \frac{r}{((1-r)P+r)^2} \tag{4}$$

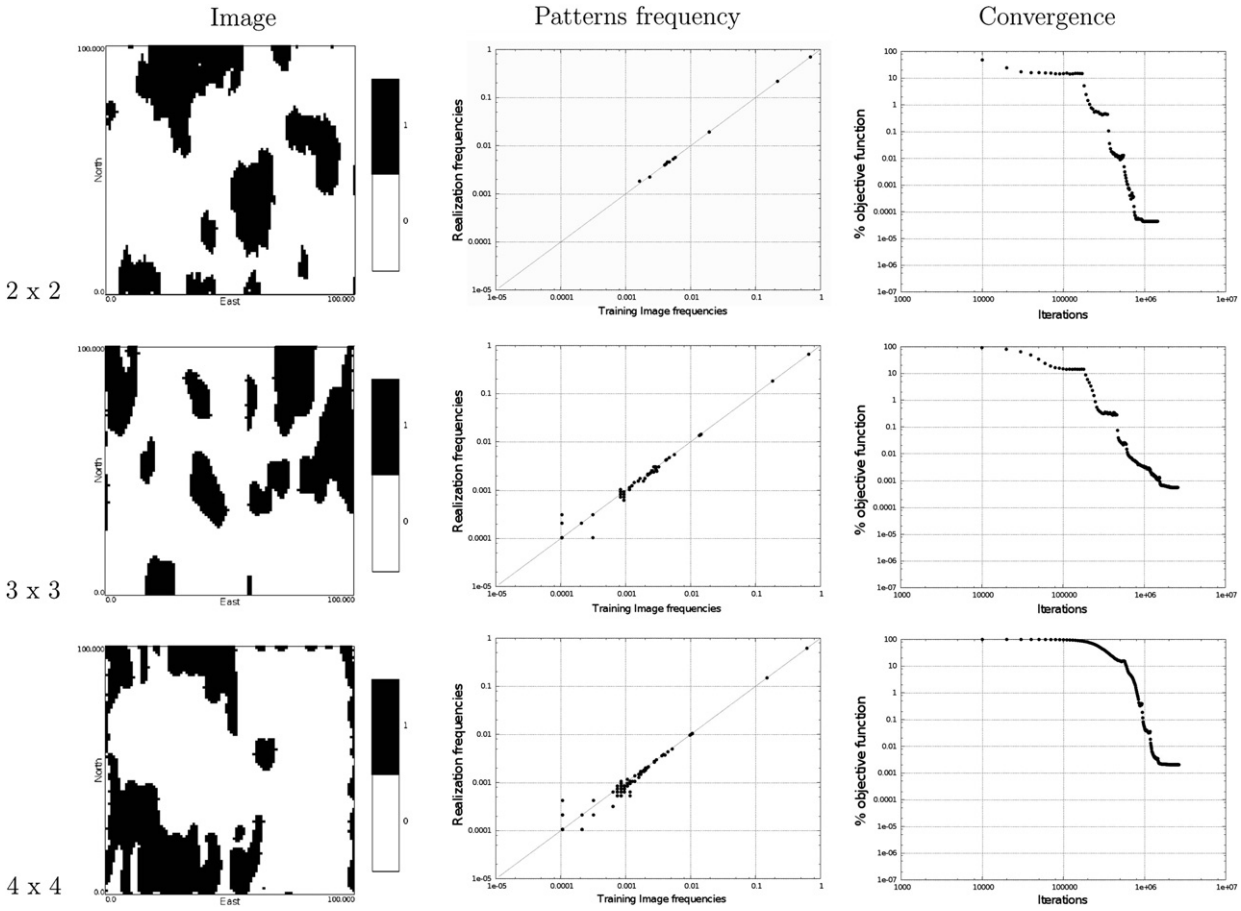

Fig. 11. Simulated images: ellipses 0 (equal weighting).

which is positive for all $r \in [0,1]$, then $S(P)$ is a monotone increasing function with limit $1/(1-r)$ when $P \to \infty$. Therefore, there is an upper bound for the speedup in a parallel application. This bound depends on the percentage of possible operations that can be calculated in parallel and the problem should be analyzed to characterize this bound.

3.4.2. Balanced trees

Using the speculative framework to parallelize simulated annealing with a balanced binary tree of processes yields a theoretical speedup of $S(P) = \log_2(P+1)$, where P is the number of processes/nodes. The theoretical speedup coincides with the number of levels in the tree. For example, if the tree has three nodes (3 processes), the number of levels is two ($\log_2(3+1) = 2$) which coincides with the speedup. The approach could be generalized to be implemented using a non-balanced tree (Witte et al., 1991), although this is not covered in this work.

Tests conducted with 3 and 7 processes, confirmed the theoretic speedup in the balanced scenario, giving evidence of the power of the method by increasing the number of processes in the tree.

4. Examples

Some examples are presented to illustrate the implementation and results over three simple cases. Several tests were carried out to evaluate convergence and speedup of the algorithm, in particular considering multiple-point statistics obtained for squared patterns of adjacent nodes of size 2×2, 3×3 and 4×4 pixels. This is equivalent to trying to reproduce the multiple-point histogram (Deutsch, 1992).

Two objective functions were tested, one that assigns equal importance to all patterns:

$$O = \sum_{i=1}^{N_{MPE}} (f_i^{target} - f_i^{model})^2 \tag{5}$$

where f_i^{target} are the frequencies of the multiple-point events from the training image, f_i^{model} are the statistics for the same multiple-point events in the current state of the simulated model, and N_{MPE} is the total number of possible multiple-point events for the pattern size considered. A second implementation is done assigning a weight inversely proportional to the frequencies of multiple-point events, in order to increase the relative importance of less frequent events in the objective function:

$$O = \sum_{i=1}^{N_{MPE}} \lambda_i (f_i^{target} - f_i^{model})^2 \tag{6}$$

where λ_i is a standardized weight inversely proportional to the target frequency of the multiple-point event:

$$\lambda_i = \begin{cases} \dfrac{\dfrac{1}{f_i^{target}}}{\sum_{i=1}^{N_{MPE}} \dfrac{1}{f_i^{target}}} & f_i^{target} \neq 0 \\ 0 & f_i^{target} = 0 \end{cases} \tag{7}$$

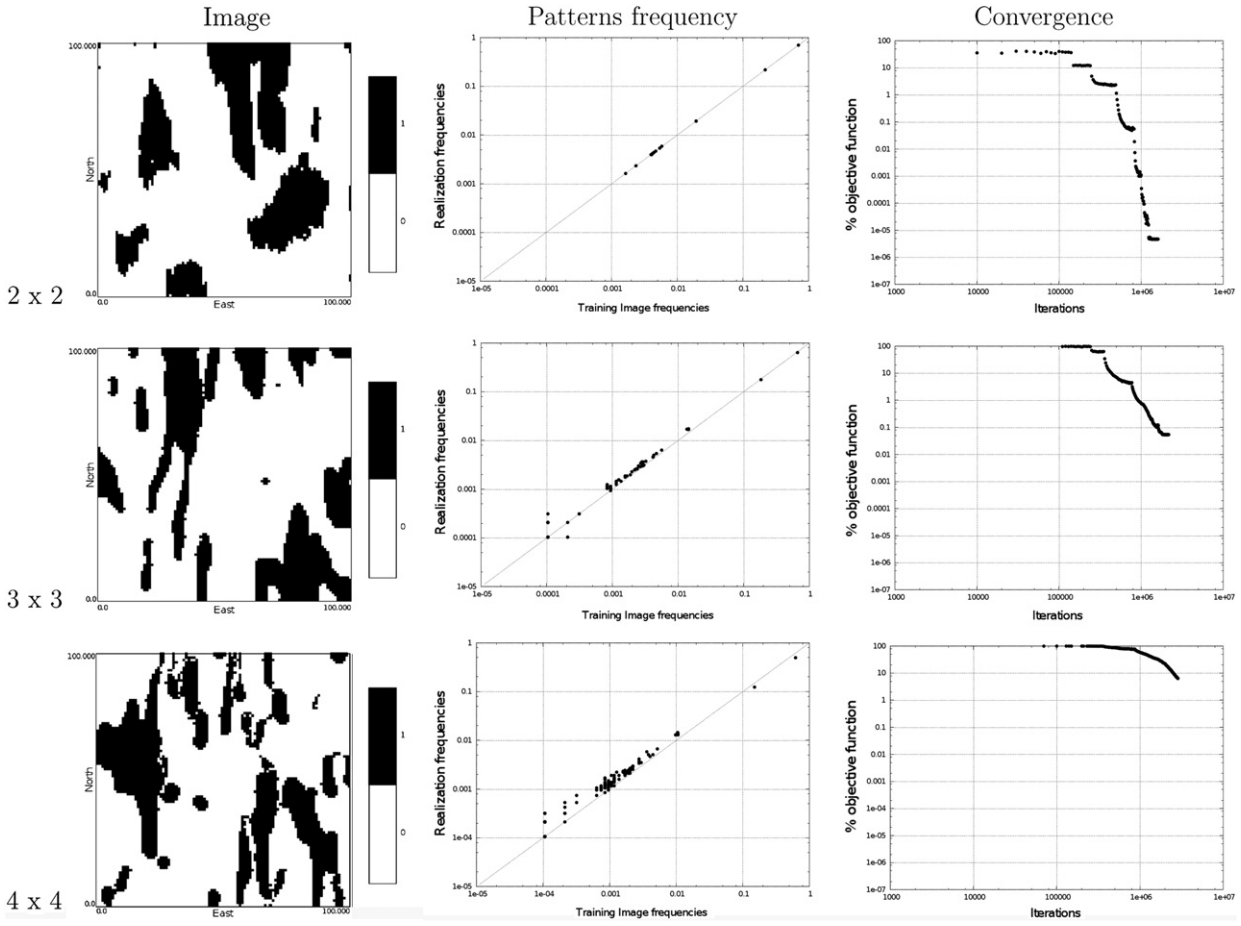

Fig. 12. Simulated images: ellipses 0 (inversely proportional weighting).

All tests were run 10 times (10 realizations) with 1 process (sequential implementation), 3 processes (2-level balanced tree) and 7 processes (3-level balanced tree), so a total amount of 180 tests were performed for each example (10 realizations $\times$ {1p, 3p, 7p} $\times$ {2×2, 3×3, 4×4} $\times$ {equal weight, inversely proportional weight}). The tests performed with one processor constitutes the sequential implementation (base case for measuring the speedup).

4.1. Example 1: channels

The first example consists of simulating a geological setting of sinuous channels in a background, using the training image presented in Fig. 7 (Strebelle, 2002).

The average speedup of the parallelization for this example is presented in Table 1, in which each test was performed 10 times and all tests are measured using the same annealing schedule. The maximum and minimum speedups are presented in Tables 2 and 3. An example of the simulated image, the statistical validation and the evolution (in percentage) of the objective function for a particular test is presented in Figs. 8 (case of equal weighting) and 9 (inversely proportional weighting).

The experimental speedup in the inversely proportional weighting case coincides more accurately with the theoretical speedup. The simulated images using this weighting more resemble the training image. It should be emphasized that the validation of the realizations is done from a statistical point of view, in terms of the reproduction of multiple-point frequencies. Visually, a better result would be obtained by increasing the pattern size or by implementing a multi-scale approach, such as a multiple grid

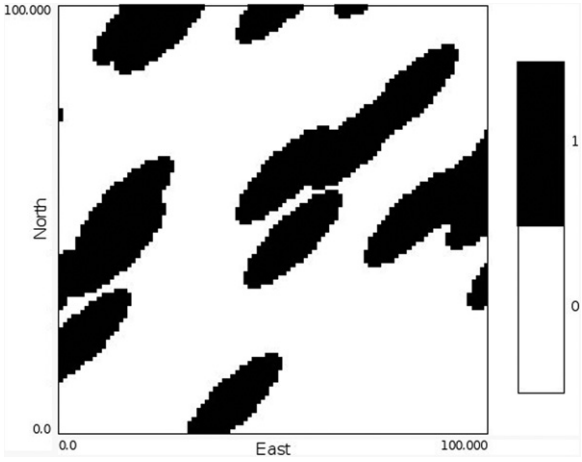

Fig. 13. Training image: ellipses 45.

Table 7
Average speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 13.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.71	1.58	1.88	3 processes	2.59	1.80	1.86
7 processes	2.59	3.04	2.25	7 processes	4.15	3.03	2.53

(Strebelle, 2002). A slower convergence of the objective function is the trade-off of this resemblance.

4.2. Example 2: ellipses 0

The second example consists in simulating an artificial setting of ellipses in a background, with their major semi-axis aligned with the vertical direction, using the training image presented in Fig. 10.

The average speedup of the parallelization for this example is presented in Table 4, in which each test was performed 10 times and all tests are measured using the same annealing schedule. The maximum and minimum speedups are presented in Tables 5 and 6. An example of the simulated image, the statistical validation and the evolution (in percentage) of the objective function for a particular test is presented in Figs. 11 (case of equal weighting) and 12 (inversely proportional weighting).

There are no major differences in the experimental speedup for equal and inversely weighted cases. The simulated images in both cases resemble the training image, except in the case of a 4×4 pattern using an inversely proportional weighting in which there are convergence problems. Also, a slower convergence and a slight bias in the multiple-point statistics reproduction occur when the inversely proportional weights are used.

Table 8
Maximum values for the speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 13.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.87	1.92	1.96	3 processes	3.41	2.12	1.88
7 processes	5.42	5.58	2.42	7 processes	6.42	6.92	2.63

Table 9
Minimum values for the speedup of the parallel implementation for the equal weighting (left) and inversely proportional weighting (right) for the examples using the training image in Fig. 13.

	2×2	3×3	4×4		2×2	3×3	4×4
1 process	1.00	1.00	1.00	1 process	1.00	1.00	1.00
3 processes	1.38	1.26	1.74	3 processes	1.12	1.27	1.80
7 processes	1.41	1.23	2.02	7 processes	1.69	1.13	2.47

4.3. Example 3: ellipses 45

The third example consists in simulating a setting of ellipses in a background, each of them rotated in 45° with respect to the horizontal axis, using the training image presented in Fig. 13.

The average speedup of the parallelization for this example is presented in Table 7, in which each test was performed 10 times and all tests are measured using the same annealing schedule. The maximum and minimum speedups are presented in Tables 8 and 9. An example of the simulated image, the statistical validation and the evolution (in percentage) of the objective function for a particular test is presented in Figs. 14 (case of equal weighting)

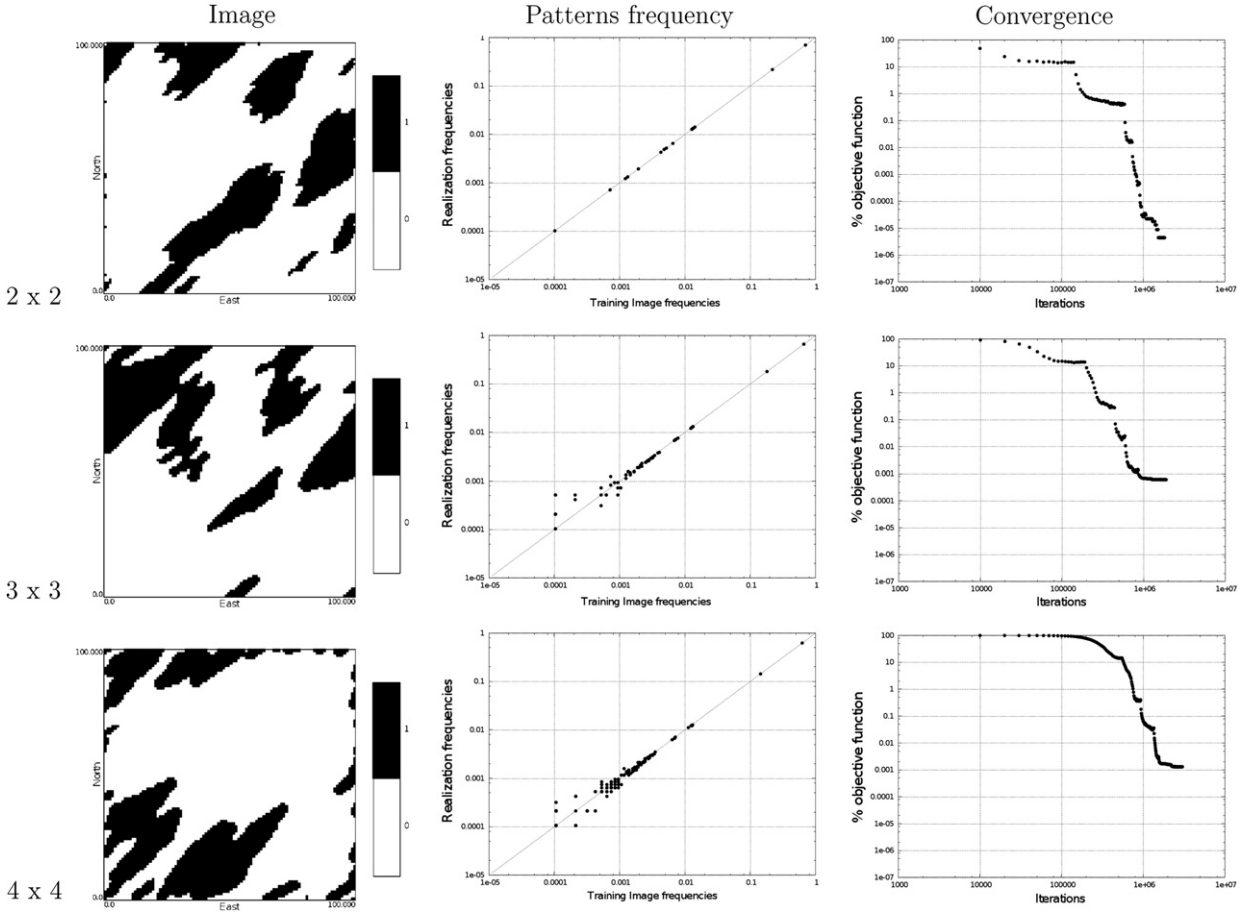

Fig. 14. Simulated images: ellipses 45 (equal weighting).

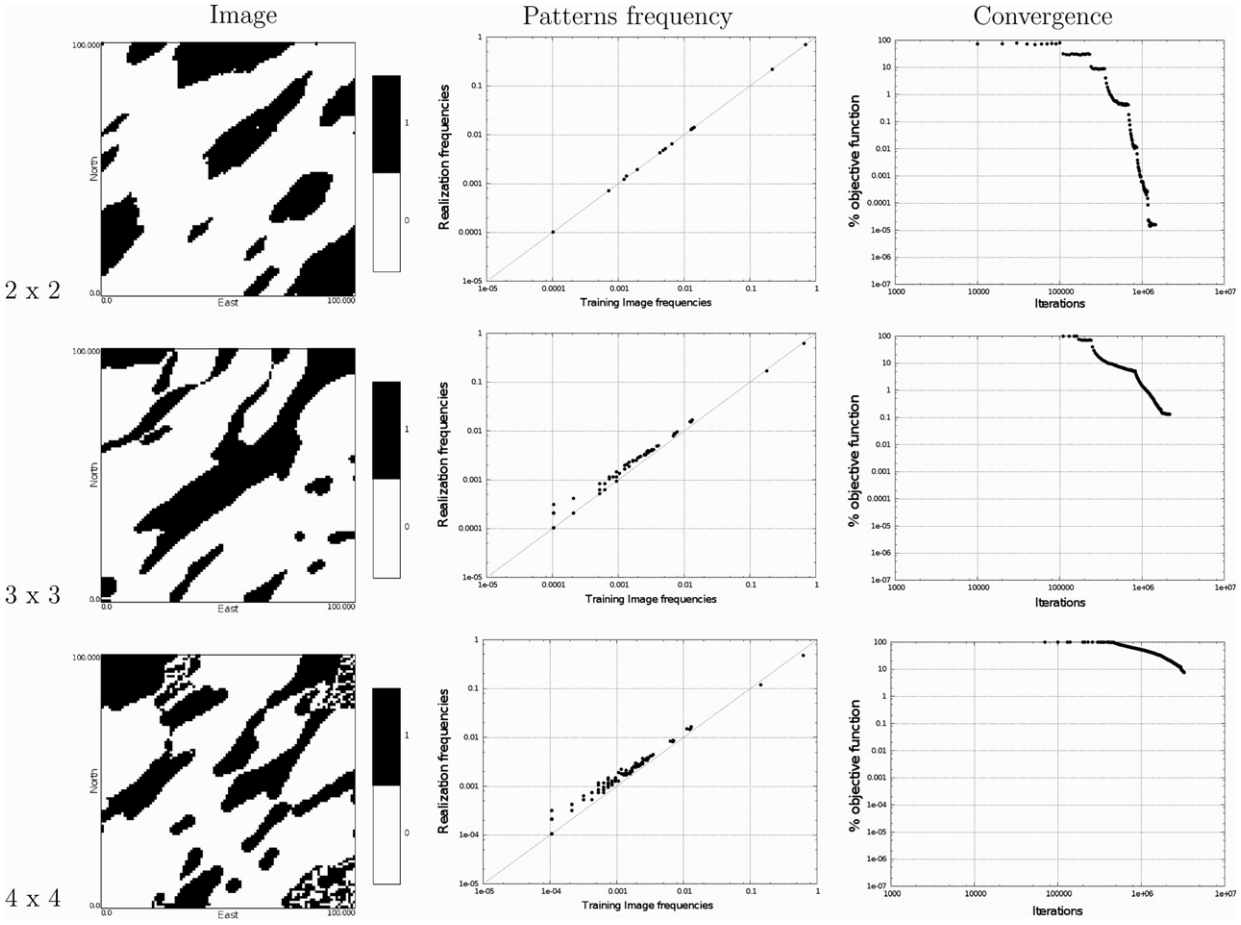

Fig. 15. Simulated images: ellipses 45 (inversely proportional weighting).

and 15 (inversely proportional weighting). Some problems related to edge effects are visible in these results. These can be explained by the statistical weight that a perturbation node has on an edge which should be equivalent to the one of a node at the center of the simulation domain. However, since a node near the edge may affect a smaller number of patterns when perturbed, in the current formulation of the objective function, it has a smaller influence in the overall performance causing the edge effects. This can be solved by modifying the objective function or by simulating a larger domain that is later eroded to the size of interest.

The experimental speedup in both cases coincides with the theoretical speedup, with small differences. As in example 2, the simulated images in both cases resemble the training image, with some convergence problems in the case of a 4×4 pattern considering the inversely proportional weighting case. Again, a slower convergence in the inversely proportional weighting case is observed.

5. Conclusions

We have shown a parallel implementation of simulated annealing under a speculative framework that allows speeding up sequential implementations.

Several conclusions can be drawn from this work. The most relevant result is that a significant speedup can be achieved using multiple processors to run the simulation under the speculative scheme. The theoretical speedup $\log_2(P+1)$ on the balanced binary tree topology was approximately obtained in all of the test cases performed. The inclusion of a weighted objective function slightly improved the results in terms of reproduction of multiple-point statistics. However, the trade-off is an increase in the simulation time for larger templates (4×4 or more). Also, a slight bias in the statistical reproduction of multiple-point events could be seen. It should be pointed out that this work aims at showing the potential of simulated annealing in terms of the statistical reproduction of multiple-point frequencies, rather than obtaining geologically appealing simulated images. This last point is relevant, as the final goal of the numerical algorithms is to assist in the study of the geological phenomenon. However, the geological quality of the resulting realizations depends on factors such as the pattern configurations and size used. Our focus was on the speedup that can be obtained with the speculative approach, considering that simulated annealing is powerful enough to include many more statistics and constraints in the objective function than presented in the examples provided here.

There are many avenues of further research. The optimization of data structures to manage the patterns and the templates is important. The time of simulation grows exponentially with the size of the template, so this topic is a main priority in the development of this technique. Also, continuing the research with non-balanced trees in the speculative scheme for simulated annealing is of interest.

Acknowledgments

This research was funded by the National Fund for Science and Technology of Chile (FONDECYT) and is part of project number 1090056.

References

Amdahl, G., 1967. Validity of the single processor approach to achieving large-scale computing capabilities. In: AFIPS Conference Proceedings, vol. 30, 1967, pp. 483–485.

Arpat, B., Caers, J., 2007. Stochastic simulation with patterns. Mathematical Geology 39 (2), 177–203.

Besag, J., 1986. On the statistical analysis of dirty pictures. Journal of the Royal Statistical Society B 48, 259–302.

Boisvert, J.B., Lyster, S., Deutsch, C.V., 2007. Constructing training images for veins and using them in multiple-point geostatistical simulation. In: Magri, E.J. (Ed.), 33rd International Symposium on Application of Computers and Operations Research in the Mineral Industry, APCOM 2007, pp. 113–120.

Burton, F.W., 1985. Speculative computation, parallelism, and functional programming. IEEE Transactions on Computers 34 (12), 1190–1193.

Caers, J., Journel, A.G., 1998. Stochastic reservoir simulation using neural networks trained on outcrop data. In: 1998 SPE Annual Technical Conference and Exhibition, New Orleans, LA, Society of Petroleum Engineers, SPE paper # 49026, pp. 321–336.

Caers, J., Ma, X., 2002. Modeling conditional distributions of facies from seismic using neural nets. Mathematical Geology 34 (2), 143–167.

Daly, C., 2005. Higher order models using entropy Markov random fields and sequential simulation. In: Leuangthong, O., Deutsch, C.V. (Eds.), Geostatistics Banff 2004. Springer, Dordrecht, The Netherlands, pp. 215–224.

Daly, C., Knudby, C., 2007. Multipoint statistics in reservoir modelling and in computer vision. Petroleum Geostatistics, A32.

Deutsch, C.V., 1992. Annealing techniques applied to reservoir modeling and the integration of geological and engineering (well test) data. Ph.D. Dissertation, Stanford University, 306pp.

Deutsch, C.V., 2002. Geostatistical Reservoir Modeling. Oxford University Press, New York, 376pp.

Deutsch, C.V., Wang, L., 1996. Hierarchical object-based stochastic modeling of fluvial reservoirs. Mathematical Geology 28 (7), 857–880.

Deutsch, C.V., Wen, X.H., 2001. Integrating large-scale soft data by simulated annealing and probability constraints. Mathematical Geology 32 (1), 49–68.

Eskandari, K., Srinivasan, S., 2007. Growthsim—a multiple point framework for pattern simulation. Petroleum Geostatistics, A06.

Fang, J.H., Wang, P.P., 1997. Random field generation using simulated annealing vs. fractal-based stochastic interpolation. Mathematical Geology 29 (6), 849–858.

Farmer, C.L., 1992. Numerical rocks. In: King, P.R. (Ed.), The Mathematics of Oil Recovery. Clarendon Press, Oxford, pp. 437–447.

Geman, S., Geman, D., 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (November), 721–741.

Goovaerts, P., 1996. Stochastic simulation of categorical variables using a classification algorithm and simulated annealing. Mathematical Geology 28 (7), 909–921.

Guardiano, F., Srivastava, M., 1993. Multivariate geostatistics: beyond bivariate moments. In: Soares, A. (Ed.), Geostatistics Troia '92, vol. 1. Kluwer, pp. 133–144.

Hong, S., Ortiz, J.M., Deutsch, C.V., 2008. Multivariate density estimation as an alternative to probabilistic combination schemes for data integration. In: Ortiz, J.M., Emery, X. (Eds.), Geostats 2008—Proceedings of the Eighth International Geostatistics Congress, Gecamin Ltda., Santiago, Chile, vol. 1, pp. 197–206.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Lyster, S., Deutsch, C.V., 2008. MPS simulation in a Gibbs sampler algorithm. In: Ortiz, J.M., Emery, X. (Eds.), Geostats 2008—Proceedings of the Eighth International Geostatistics Congress, vol. 1, Gecamin Ltda., Santiago, Chile, pp. 79–88.

Norrena, K., Deutsch, C.V., 2000. Using the critical temperature to improve the speed of geostatistical applications of simulated annealing. In: Kleingold, W.J., Krige, D.G. (Eds.), Proceedings of GEOSTATS 2000, Geostatistical Congress, vol. 1. Cape Town, South Africa, pp. 254–262.

Ortiz, J.M., 2003. Characterization of high order correlation for enhanced indicator simulation. Ph.D. Dissertation, University of Alberta, 246pp.

Ortiz, J.M., Deutsch, C.V., 2004. Indicator simulation accounting for multiple-point statistics. Mathematical Geology 36 (5), 545–565.

Ortiz, J.M., Emery, X., 2005. Integrating multiple point statistics into sequential simulation algorithms. In: Leuangthong, O., Deutsch, C.V. (Eds.), Geostatistics Banff 2004. Springer, Dordrecht, The Netherlands, pp. 969–978.

Ortiz, J.M., Lyster, S., Deutsch, C.V., 2007. Scaling multiple-point statistics to different univariate proportions. Computers & Geosciences 33 (2), 191–201.

Ortiz, J.M., Peredo, O., 2009. Multiple point geostatistical simulation with simulated annealing: implementation using speculative parallel computing. In: Atkinson, P.M., Lloyd, C.D. (Eds.), GeoENVVII—Geostatistics for Environmental Applications. Springer, Dordrecht, pp. 383–394.

Pacheco, P.S., 1996. Parallel Programming with MPI. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 500 pp.

Parra, A., Ortiz, J.M., 2009. Conditional multiple-point simulation with a texture synthesis algorithm. In: Proceedings of the 12th IAMG (International Association of Mathematical Geosciences) Conference, Stanford University, CA, USA, 5 pp.

Pyrcz, M.J., Strebelle, S., 2008. A showcase of event-based geostatistical models. In: Ortiz, J.M., Emery, X. (Eds.), Geostats 2008—Proceedings of the Eighth International Geostatistics Congress, Gecamin LTDA, Santiago, Chile, vol. 2, pp. 1143–1148.

Rothman, D.H., 1985. Nonlinear inversion, statistical mechanics and residual statics estimation. Geophysics 50, 2784–2796.

Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple-point statistics. Mathematical Geology 34 (1), 1–21.

Strebelle, S., Journel, A.G., 2000. Sequential simulation drawing structures from training images. In: Kleingold, W.J., Krige, D.G. (Eds.), Proceedings of GEOSTATS 2000, Geostatistical Congress, vol. 1. Cape Town, South Africa, pp. 381–392.

Tjelmeland, H., 1996. Stochastic models in reservoir characterization and Markov random fields for compact objects. Ph.D. Dissertation, Norwegian University of Science and Technology, Trondheim, Norway, 84 pp.

Witte, E., Chamberlain, R.D., Franklin, M.A., 1991. Parallel simulated annealing using speculative computation. IEEE Transactions on Parallel and Distributed Systems 2 (4).

Xu, W., 1996. Conditional curvilinear stochastic simulation using pixel-based algorithms. Mathematical Geology 28 (7), 937–949.

Zanon, S., 2004. Advanced aspects of sequential Gaussian simulation. M.Sc. Thesis, University of Alberta, 65pp.

Zhang, T., Switzer, P., Journel, A., 2006. Filter-based classification of training image patterns for spatial simulation. Mathematical Geology 38 (1), 63–80.