Computers & Geosciences 89 (2016) 57–70

Contents lists available at ScienceDirect

Computers & Geosciences

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

Research paper

GOSIM: A multi-scale iterative multiple-point statistics algorithm with global optimization

Liang Yang [a,b], Weisheng Hou [a,b,*], Chanjie Cui [a,b], Jie Cui [c]

[a] School of Earth Science and Geological Engineering, Sun Yat-sen University, Guangzhou, 510275, China
[b] Guangdong Provincial Key Laboratory of Mineral Resource Exploration & Geological Processes, Guangzhou, 510275, China
[c] Guangdong Techno-economy Research and Development Center, Guangzhou, 510070, China

ARTICLE INFO

Article history:
Received 10 July 2015
Received in revised form
19 December 2015
Accepted 29 December 2015
Available online 7 January 2016

Keywords:
MPS algorithm
Global optimization
Iterative algorithm
Geostatistical simulation

ABSTRACT

Most current multiple-point statistics (MPS) algorithms are based on a sequential simulation procedure, during which grid values are updated according to the local data events. Because the realization is updated only once during the sequential process, errors that occur while updating data events cannot be corrected. Error accumulation during simulations decreases the realization quality. Aimed at improving simulation quality, this study presents an MPS algorithm based on global optimization, called GOSIM. An objective function is defined for representing the dissimilarity between a realization and the TI in GOSIM, which is minimized by a multi-scale EM-like iterative method that contains an E-step and M-step in each iteration. The E-step searches for TI patterns that are most similar to the realization and match the conditioning data. A modified PatchMatch algorithm is used to accelerate the search process in E-step. M-step updates the realization based on the most similar patterns found in E-step and matches the global statistics of TI. During categorical data simulation, *k*-means clustering is used for transforming the obtained continuous realization into a categorical realization. The qualitative and quantitative comparison results of GOSIM, MS-CCSIM and SNESIM suggest that GOSIM has a better pattern reproduction ability for both unconditional and conditional simulations. A sensitivity analysis illustrates that pattern size significantly impacts the time costs and simulation quality. In conditional simulations, the weights of conditioning data should be as small as possible to maintain a good simulation quality. The study shows that big iteration numbers at coarser scales increase simulation quality and small iteration numbers at finer scales significantly save simulation time.

1. Introduction

As a developing technique originated about a decade ago, multiple-point statistics (MPS) has been applied in many fields, such as reservoir forecasting (Pyrcz and Deutsch, 2014), mineral resources (Jones et al. 2013; Boucher et al., 2014) and climate modeling (Jha et al., 2013). The main purpose of MPS algorithms is to simulate multiple models via capturing the spatial structure in training images (TIs), while simultaneously following the constraints of conditioning data (Journel and Zhang, 2006; Caers, 2011). The similarity between realizations and TI, also called pattern reproduction, is one of the major factors in evaluating MPS algorithms (Tan et al., 2014; Mariethoz and Caers, 2015). Although MPS algorithms have been studied for some time, further pattern reproduction quality improvements need to be achieved.

The first MPS algorithm presented by Guardiano and Srivastava (1993) is rarely used in practical applications due to its high CPU demand. Since then, various MPS algorithms have been proposed. Some algorithms that are called pixel-based algorithms simulate one pixel at a time (Strebelle, 2002; Mariethoz et al., 2010; Straubhaar et al., 2011). Some other algorithms are pattern-based algorithms because the entire pattern is pasted at a time during simulation (Zhang et al., 2006; Arpat and Caers, 2007; Honarkhah and Caers, 2010; Tahmasebi et al., 2012, 2014; Mahmud et al., 2014). Different from these TI-based algorithms, HOSIM uses spatial cumulants to represent high-order information that can be directly calculated from abundant data (Dimitrakopoulos et al., 2010; Mustapha and Dimitrakopoulos, 2011). HOSIM is promising in the mineral resources field, where numerous borehole data are available.

Both existing pixel-based and pattern-based MPS algorithms follow a sequential simulation procedure. In these MPS algorithms, the update of data event is confined by a local region and errors might occur. Because the grid is traversed only once, the error cannot be corrected and errors accumulate due to the sequential

* Corresponding author at: School of Earth Science and Geological Engineering, Sun Yat-sen University, B410 Dihuan Building, No. 135 Xingangxi Road, Guangzhou, Guangdong Province 510275, China.
E-mail address: houwsh@mail.sysu.edu.cn (W. Hou).

http://dx.doi.org/10.1016/j.cageo.2015.12.020
0098-3004/

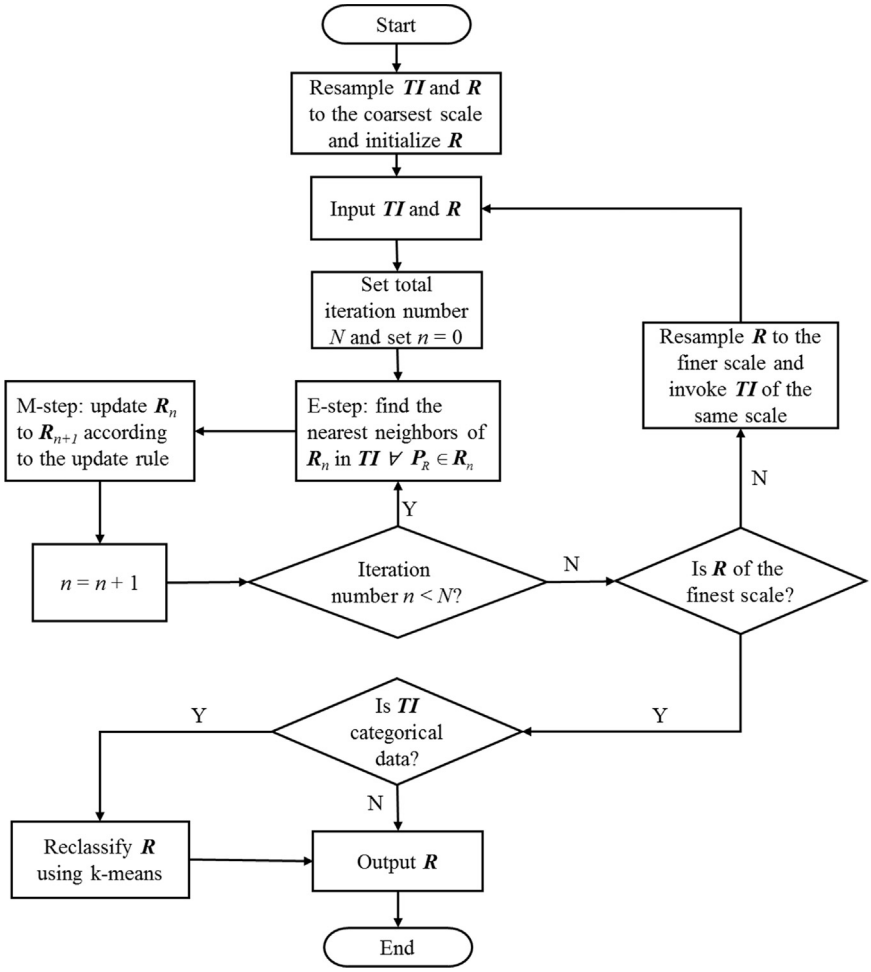


Fig. 1. The GOSIM algorithm flowchart.

process. After a sequence of error accumulations, the final realization may deviate from the TI. Although it has been noted that, a realization obtained with a raster path is less sensitive to the pixels or patterns simulated at the beginning compared to a random path (Parra and Ortiz, 2011; Tahmasebi et al., 2012), the issue of error accumulation remains a hindrance of MPS.

In the field of computer graphics, example-based texture synthesis has a goal that is similar to that of MPS (Wei et al., 2009; Mariethoz and Lefebvre, 2014). Current example-based texture synthesis algorithms can be classified as pixel-based methods (Efros and Leung, 1999; Wei and Levoy, 2000), patch-based methods (Efros and Freeman, 2001; Kwatra et al., 2003) and optimization-based methods (Kwatra et al., 2005; Kopf et al., 2007). The first two algorithm types are similar to the pixel-based and pattern-based methods of MPS, while the optimization-based methods have no analogous MPS algorithm. The optimization-based methods refine realizations using an iterative process, thus they avoid the issue of error accumulation. However, the original global optimization algorithm proposed by Kwatra et al. (2005) is relatively inefficient, and cannot be directly applied to categorical data simulations, 3D simulations and conditional simulations. Although Kopf et al. (2007) extends the optimization-based method to 3D cases and allows the addition of soft constraints to control the synthesis outcome, the algorithm cannot be directly used in 3D simulation cases with 3D TI because it utilizes 2D exemplar to synthesize 3D results, and is not suitable for hard point data conditioning. A thorough discussion of the similarities and differences between texture synthesis and MPS can be found in Mariethoz and Lefebvre (2014).

In this study, we present an MPS algorithm based on the global optimization scheme, hereafter GOSIM. GOSIM is adapted from the global optimization algorithm proposed by Kwatra et al. (2005). Different from the algorithm presented by Kwatra et al. (2005), GOSIM can be applied to categorical data simulations, 3D simulations and conditional simulations. In addition, GOSIM uses a modified version of PatchMatch (Barnes et al., 2009) to accelerate the pattern search process. This paper is organized as follows. Section 2 provides important background and terminology. The details of how GOSIM is implemented in different situations are explained in Section 3. Section 4 compares several simulation tests between GOSIM, MS-CCSIM and SNESIM, by means of human vision, ensemble averages and analysis of distance (ANODI). In addition, a sensitivity analysis of GOSIM is presented in Section 4.

2. Background and terminology

GOSIM is based on a simple assumption. If a realization ($\boldsymbol{R}$) visually resembles a training image ($\boldsymbol{TI}$), then as many patterns as possible in $\boldsymbol{R}$ come from $\boldsymbol{TI}$. Let $\boldsymbol{P}_{TI}$ denote a pattern in $\boldsymbol{TI}$ and $\boldsymbol{P}_R$ denote a pattern in $\boldsymbol{R}$, the distance (or dissimilarity) between $\boldsymbol{R}$ and $\boldsymbol{TI}$ is:

$$d(\boldsymbol{R}, \boldsymbol{TI}) = \sum_{\boldsymbol{P}_R \in \boldsymbol{R}} \min_{\boldsymbol{P}_{TI} \in \boldsymbol{TI}} D(\boldsymbol{P}_R, \boldsymbol{P}_{TI}) \tag{1}$$

The implication of Eq. (1) is that for each $\boldsymbol{P}_R \in \boldsymbol{R}$, its most

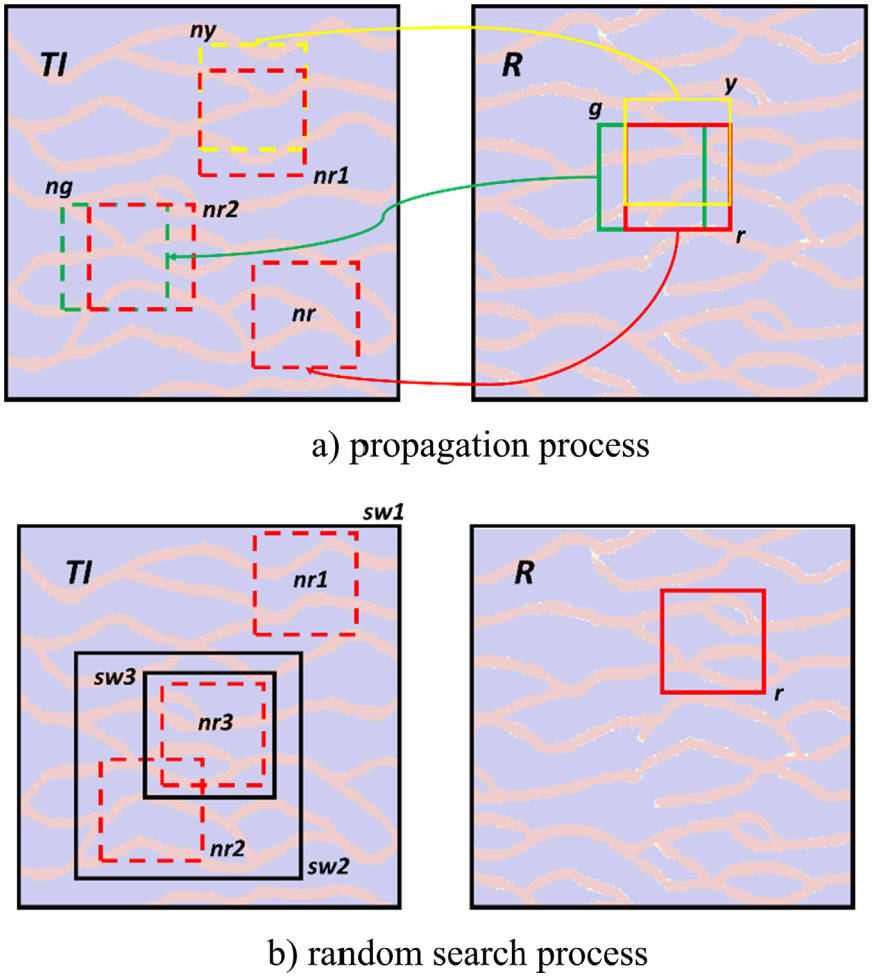


Fig. 2. The PatchMatch iteration process. The red solid rectangle in $\boldsymbol{R}$ represents the pattern of interest, and the red dashed rectangles in $\boldsymbol{TI}$ are candidates of the approximate nearest neighbors. The arrows start from patterns in $\boldsymbol{R}$ and point to the current approximate nearest neighbors of these patterns. The most similar of these candidates is chosen as the new approximate nearest neighbor. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

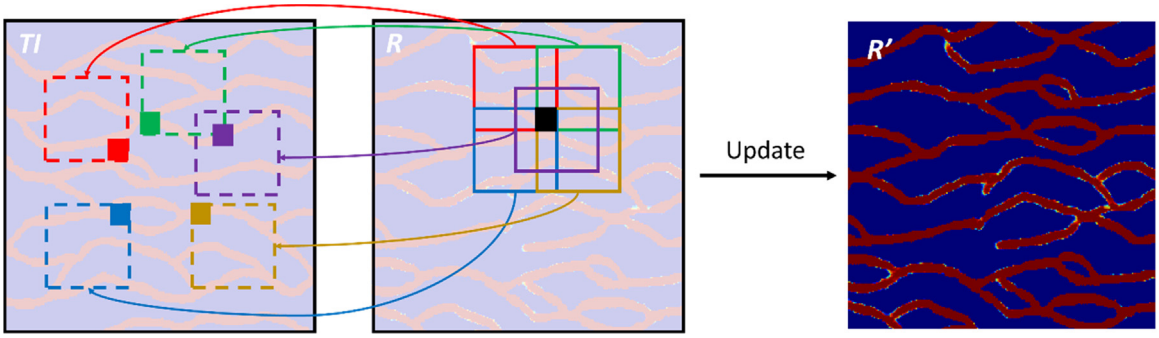


Fig. 3. Illustration of the process and result of M-step. The solid rectangles in $\boldsymbol{R}$ represent the patterns that contain the black cell, and the dashed rectangles in $\boldsymbol{TI}$ are the approximate nearest neighbors of these solid patterns. The colorful cells inside the dashed rectangles are cells that will be used to calculate the value of the black cell. $\boldsymbol{R}'$ is the realization that is updated from $\boldsymbol{R}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

similar (or nearest) $\boldsymbol{P}_{TI} \in \boldsymbol{TI}$ is found, then the sum of all of the distances between the corresponding $\boldsymbol{P}_R \in \boldsymbol{R}$ and $\boldsymbol{P}_{TI} \in \boldsymbol{TI}$ represents the global dissimilarity between $\boldsymbol{R}$ and $\boldsymbol{TI}$. The distance, D, represents the Euclidean distance in GOSIM. $d(\boldsymbol{R}, \boldsymbol{TI})$ is the objective function, which is minimized in the simulation process. Unlike the sequential simulation process, global optimization is a refinement process. In other words, after an initial guess, the realization is iteratively modified to produce a final realization $\boldsymbol{R}_{final}$, which satisfies:

$$\boldsymbol{R}_{final} = \arg\min_{\boldsymbol{R}} \; d(\boldsymbol{R}, \boldsymbol{TI}) \tag{2}$$

Let $\boldsymbol{P}_{TI}^1, \ldots, \boldsymbol{P}_{TI}^n$ denote all of the patterns in $\boldsymbol{TI}$. For $\boldsymbol{P}_R \in \boldsymbol{R}$ and $\boldsymbol{P}_{TI} \in \boldsymbol{TI}$, we conclude that $\boldsymbol{P}_{TI}$ is the "nearest neighbor" of $\boldsymbol{P}_R$ if they satisfy:

$$\boldsymbol{P}_{TI} = \arg\min_{\boldsymbol{P}_{TI}^i \in \boldsymbol{TI}} D(\boldsymbol{P}_R, \boldsymbol{P}_{TI}^i) \qquad i = 1, 2, \ldots, n \tag{3}$$

Eq. (3) suggests that $\boldsymbol{P}_{TI}$ is the most similar pattern to $\boldsymbol{P}_R$. Assuming that $\boldsymbol{P}_{TI}$ is the approximate nearest neighbor to $\boldsymbol{P}_R$, let

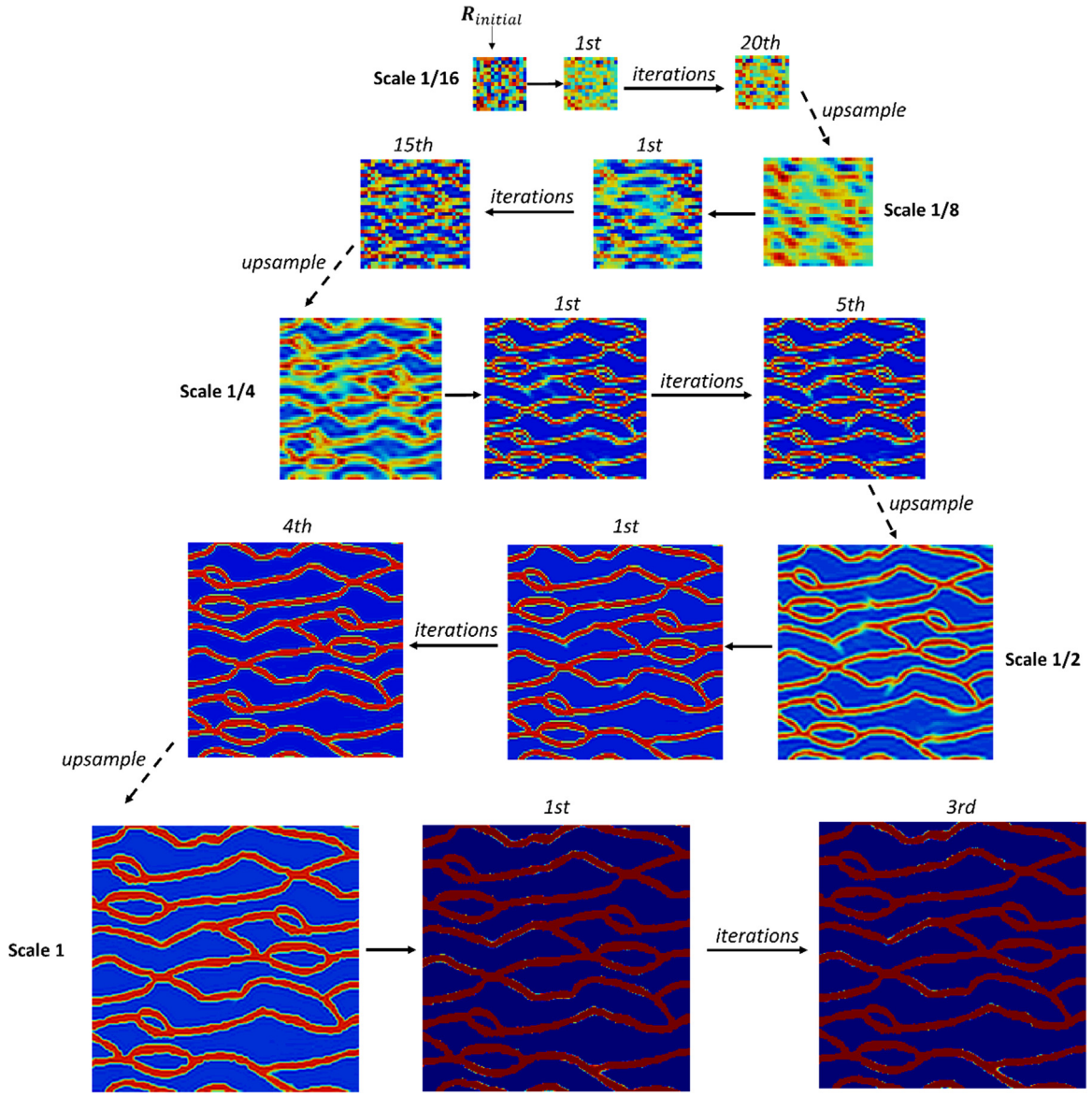


Fig. 4. Multi-scale strategy in GOSIM. The initial realizations, realizations obtained after the 1st iteration and realizations obtained after the last iteration in each scale are shown.

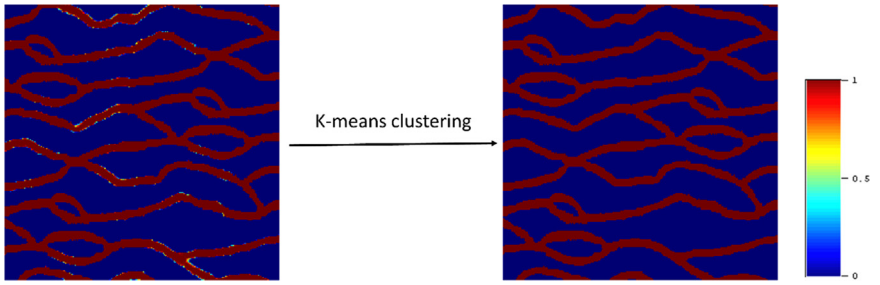


Fig. 5. Transformation results of the continuous variables to categorical variables using *k*-means clustering.

$\boldsymbol{u}_R$ be the center cell of $\boldsymbol{P}_R$ and $\boldsymbol{u}_{TI}$ be the center cell of $\boldsymbol{P}_{TI}$, then $\boldsymbol{u}_{TI}$ is called a "candidate" of $\boldsymbol{u}_R$. If (x, y, z) is the coordinate of $\boldsymbol{u}_R$ (in 3D), then the coordinate of $\boldsymbol{u}_{TI}$ can be defined as a vector function, $\boldsymbol{f}$, of $\boldsymbol{u}_R$:

$$\boldsymbol{u}_{TI} = \boldsymbol{f}(\boldsymbol{u}_R) = \boldsymbol{f}(x, y, z) \tag{4}$$

3. Methodology

We are supposed to know both the patterns in $\boldsymbol{R}$ and their most similar patterns in $\boldsymbol{TI}$ to obtain $\boldsymbol{R}_{final}$ in Eq. (2). This is similar to an optimization problem where the estimated variables and the parameters are both unknown (Kwatra et al., 2005). The Expectation–Maximization (EM) algorithm is suitable for solving this

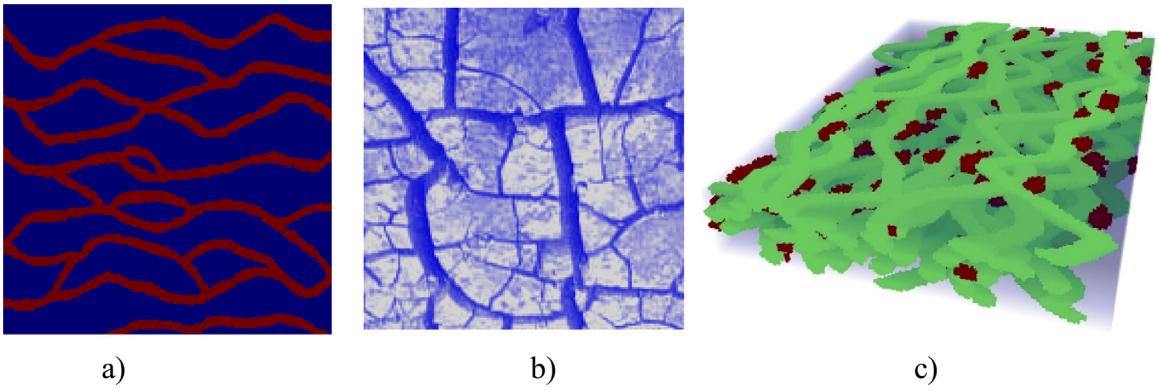


Fig. 6. TIs used in the simulation test. From left to right are the (a) 2D channel TI, (b) continuous crack TI and (c) 3D facies TI, respectively.

Table 1
Algorithm parameters for SNESIM.

Simulation cases	Conditioning data nodes	Search ellipsoid	Multi-grid	Sub-grid
Unconditional case 1 and Conditional case 1	100	[60 60 1]	3	No
Conditional case 2	60	[20 20 5]	4	Yes

Table 2
Algorithm parameters for MS-CCSIM.

Simulation cases	Search template	Overlap region	Multi-scale	Replicates no.
Unconditional case 1 and Conditional case 1	[30 30 1]	[8 8 1]	1	10
Unconditional case 2	[26 26 1]	[8 8 1]	1	5
Conditional case 2	[32 32 10]	[12 12 4]	1	1

type of problem (Mclachlan and Krishnan, 1997; Kwatra et al., 2005). Following the work of Kwatra et al. (2005), the EM-like algorithm is adopted in GOSIM. Fig. 1 illustrates the algorithm flowchart of GOSIM. First, a random initial guess is given to $\boldsymbol{R}$ at the coarsest scale. Next, several iterations are performed to refine the initial guess. Each iteration is divided into an E-step and an M-step. In the E-step, we search for the (approximate) most similar $\boldsymbol{P}_{TI} \in \boldsymbol{TI}$ for all of the $\boldsymbol{P}_R \in \boldsymbol{R}$. The M-step updates $\boldsymbol{R}$ using an update rule, which is based on the nearest patterns found in the E-step and minimizes Eq. (1). This process is conducted with a multi-scale strategy. Finally, the algorithm outputs $\boldsymbol{R}$ at the finest scale. If $\boldsymbol{TI}$ is based on categorical data, an additional step is needed to reclassify the realization to a categorical result before outputting the results.

3.1. E-step: approximate nearest neighbor search

The goal of the E-step is to find the nearest neighbors of every $\boldsymbol{P}_R \in \boldsymbol{R}$ in $\boldsymbol{TI}$. The pattern search efficiency is vital in the global optimization algorithm, because pattern search processes are performed on all iterations. We adapt an algorithm called PatchMatch (Barnes et al., 2009) to solve the pattern search issue. The original PatchMatch presented by Barnes et al. (2009) was designed for finding corresponding patterns in 2D textures. In this study, the PatchMatch algorithm is modified for conducting 3D and conditional simulations. PatchMatch finds the approximate nearest neighbors iteratively instead of finding the exact nearest neighbors of every $\boldsymbol{P}_R \in \boldsymbol{R}$. It contains two steps, an initialization step and an iteration step.

In the initialization step, random candidates are given to every $\boldsymbol{u}_R^i \in \boldsymbol{R}$, $i = 1, 2, \ldots, n$. Let $\boldsymbol{u}_{TI}^1, \ldots, \boldsymbol{u}_{TI}^n$ denote all cells in $\boldsymbol{TI}$. The initialization step is formulated as:

$$\boldsymbol{f}(\boldsymbol{u}_R^i) = \text{random } (\boldsymbol{u}_{TI}^i) \quad i = 1, 2, \ldots, n \tag{5}$$

Let $\boldsymbol{P}_R^1, \ldots, \boldsymbol{P}_R^n$ denote all of the patterns in $\boldsymbol{R}$, and $\boldsymbol{P}_{TI}^1, \ldots, \boldsymbol{P}_{TI}^n$ denote the approximate nearest neighbors in $\boldsymbol{TI}$ found by Eq. (5). The errors (distances) between the corresponding patterns in $\boldsymbol{TI}$ and $\boldsymbol{R}$ are:

$$\boldsymbol{Err}^i = \widetilde{D}(\boldsymbol{u}_{TI}^i) = D(\boldsymbol{P}_R^i, \boldsymbol{P}_{TI}^i) i = 1, 2, \ldots, n \tag{6}$$

Then, an iteration step is conducted to refine the initial guess of the approximate nearest neighbors. Each iteration contains a propagation process and a random search process, illustrated with a 2D example in Fig. 2. In Fig. 2, $\boldsymbol{R}$ is an intermediate realization from the $\boldsymbol{TI}$. Fig. 2a shows the propagation of candidates in one direction (downward and rightward). The green pattern ($\boldsymbol{g}$) is one cell left of the red pattern ($\boldsymbol{r}$) in $\boldsymbol{R}$, and yellow pattern ($\boldsymbol{y}$) is one cell above $\boldsymbol{r}$. The green dashed pattern ($\boldsymbol{ng}$) and yellow dashed pattern ($\boldsymbol{ny}$) in $\boldsymbol{TI}$ are the current approximate nearest neighbors of $\boldsymbol{g}$ and $\boldsymbol{y}$, respectively. One of the red dashed patterns, $\boldsymbol{nr}$, is the current approximate nearest neighbor of $\boldsymbol{r}$. The other two red dashed patterns, $\boldsymbol{nr1}$ and $\boldsymbol{nr2}$, are shifted one cell from $\boldsymbol{ng}$ and $\boldsymbol{ny}$, respectively. Then, $\boldsymbol{nr}$, $\boldsymbol{nr1}$ and $\boldsymbol{nr2}$ are compared with $\boldsymbol{r}$ and the most similar one is chosen as the new approximate nearest neighbor of $\boldsymbol{r}$. Fig. 2b illustrates the random search process. To find the new approximate nearest neighbor of the red pattern ($\boldsymbol{r}$) in $\boldsymbol{R}$, we first define a sequence of search windows, $\boldsymbol{sw1}$, $\boldsymbol{sw2}$ and $\boldsymbol{sw3}$ (the three black rectangles on the $\boldsymbol{TI}$, including the envelope of $\boldsymbol{TI}$). The sizes of $\boldsymbol{sw1}$, $\boldsymbol{sw2}$ and $\boldsymbol{sw3}$ decrease exponentially

Table 3
Algorithm parameters for GOSIM.

Simulation Cases	Pattern size	Multi-scale	Iteration number	w_{cd}	Histogram matching
Unconditional case 1	[11 11 1]	5	[20, 15, 5, 4, 3]	–	No
Unconditional case 2	[13 13 1]	4	[15, 10, 4, 3]	–	No
Conditional case 1	[11 11 1]	5	[20, 15, 5, 4, 3]	[0, 0.1, 0.15, 0.2, 0.2]	No
Conditional case 2	[7 7 5]	4	[20, 4, 3, 2]	[0.1, 0.2, 0.3, 0.3]	Yes

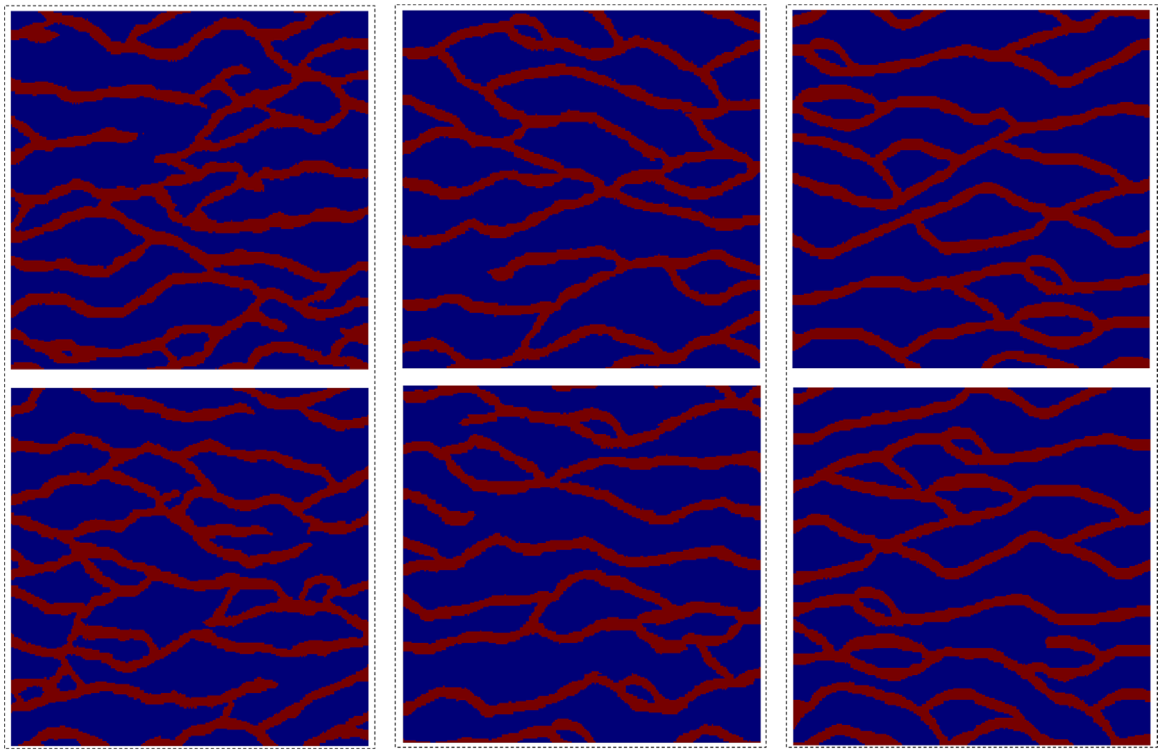


Fig. 7. Comparison of unconditional realizations of 2D channels.

(***sw1*** > ***sw2*** > ***sw3***), from the size of ***TI*** to a size that is slightly larger than the pattern size. From each search window, one candidate is randomly picked (***nr1***, ***nr2*** and ***nr3*** come from ***sw1***, ***sw2*** and ***sw3***, respectively). If the current approximate nearest neighbor of ***r*** is ***nr*** (not shown in Fig. 2b), then ***nr***, ***nr1***, ***nr2*** and ***nr3*** are compared with ***r*** and the most similar one is chosen as the new approximate nearest neighbor of ***r***. After the random search process, other iterations are executed until convergence or a fixed iteration number (such as five) is reached.

Eqs. (7)–(9) present a general description of PatchMatch in 3D case. When searching for the candidates of (x, y, z) on odd iterations during propagation, if a good candidate is found for $(x-1, y, z)$, $(x, y-1, z)$ or $(x, y, z-1)$, the candidate will be propagated to (x, y, z), i.e.,

$$\boldsymbol{f}(x,y,z) = \arg\min\begin{Bmatrix} \widetilde{D}(\boldsymbol{f}(x,y,z)+(0,0,0)) \\ \widetilde{D}(\boldsymbol{f}(x-1,y,z)+(1,0,0)) \\ \widetilde{D}(\boldsymbol{f}(x,y-1,z)+(0,1,0)) \\ \widetilde{D}(\boldsymbol{f}(x,y,z-1)+(0,0,1)) \end{Bmatrix} \tag{7}$$

On even iterations, propagation is conducted in a reverse direction. That means, a good candidate for $(x+1, y, z)$, $(x, y+1, z)$ or $(x, y, z+1)$ is propagated to (x, y, z), i.e.,

$$\boldsymbol{f}(x,y,z) = \arg\min\begin{Bmatrix} \widetilde{D}(\boldsymbol{f}(x,y,z)+(0,0,0)) \\ \widetilde{D}(\boldsymbol{f}(x+1,y,z)+(-1,0,0)) \\ \widetilde{D}(\boldsymbol{f}(x,y+1,z)+(0,-1,0)) \\ \widetilde{D}(\boldsymbol{f}(x,y,z+1)+(0,0,-1)) \end{Bmatrix} \tag{8}$$

During a random search, a sequence of candidates are tested. These candidates are located at an exponentially decreasing distance from $\boldsymbol{f}(x, y, z)$, which is found after propagation:

$$\boldsymbol{u}_{TI} = \boldsymbol{f}(x,y,z) + W\alpha^i(r_x, r_y, r_z) \qquad i = 0,\ 1,\ 2,\ \ldots \tag{9}$$

where r_x, r_y and r_z are all uniform random values in range $[-1, 1]$ and W is the width of the maximum search window (the entire ***TI*** in our simulation cases). α is a fixed ratio value (such as 1/2) and i is the exponent of α. Therefore, $W\alpha^i$defines the width of current search window and $\boldsymbol{u}_{TI}$ is the center of a pattern randomly picked inside that search window. The patterns for $i = 0, 1, 2, \ldots$ are examined until $W\alpha^i$ is smaller than the size of one cell.

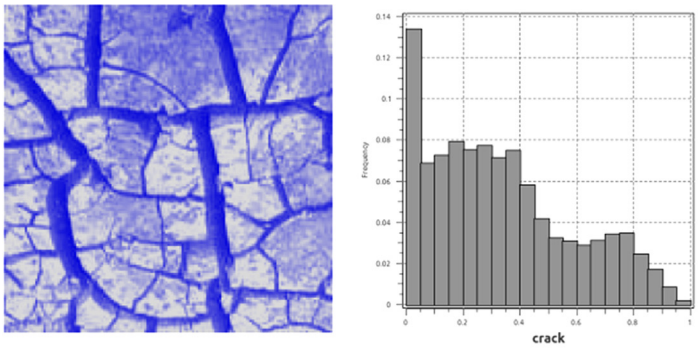
In the case of conditional simulation, it is not enough to only search for $\boldsymbol{P}_{TI} \in$ ***TI***that minimizes $\boldsymbol{Err}^i$ in Eq. (6). We should also search for the patterns that are consistent with conditioning data. Therefore, a new grid for conditioning data (***CD***) that has the same size as ***R*** is created. Notice that ***CD*** may contain cells with special values, which are used to indicate NOT DATA. Let $\boldsymbol{P}_{CD}^1, \ldots, \boldsymbol{P}_{CD}^n$ denote all of the patterns in ***CD***, $\boldsymbol{N}_p$ denote the number of cells in $\boldsymbol{P}_R \in \boldsymbol{R}$ and $\boldsymbol{N}_{CD}$ denote the number of cells that contain data values in $\boldsymbol{P}_{CD} \in \boldsymbol{CD}$. Then, Eq. (6) can be changed to:

$$\boldsymbol{Err}^i = w_{ps}\frac{D(\boldsymbol{P}_R^i, \boldsymbol{P}_{TI}^i)}{\boldsymbol{N}_P} + w_{cd}\frac{D'(\boldsymbol{P}_{TI}^i, \boldsymbol{P}_{CD}^i)}{\boldsymbol{N}_{CD}} \qquad i = 1, 2, \ldots, n \tag{10}$$

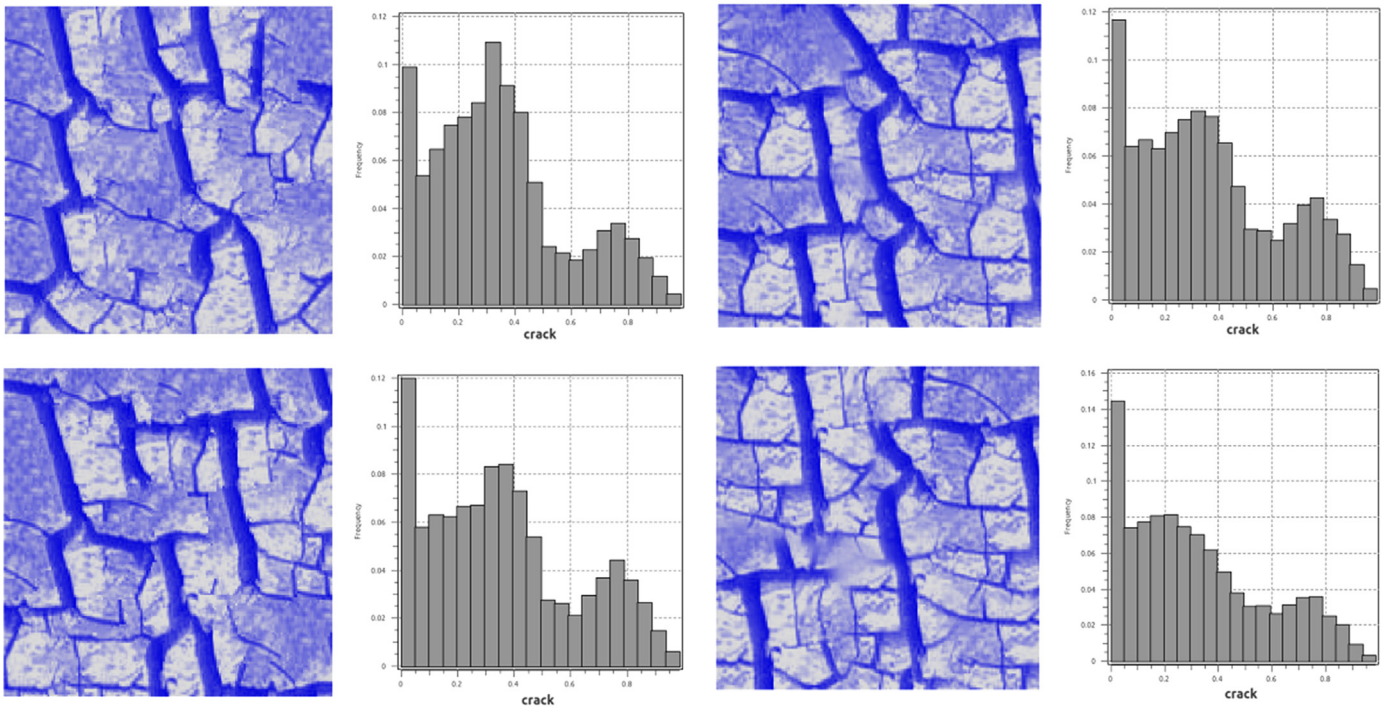
where w_{ps} and w_{cd} are the pattern similarity and conditioning data weights, respectively. $D'(\boldsymbol{P}_{TI}^i, \boldsymbol{P}_{CD}^i)$ is the Euclidean distance between $\boldsymbol{P}_{TI}^i$ and $\boldsymbol{P}_{CD}^i$. We use a different notation (D') to denote the distance between $\boldsymbol{P}_{TI}^i$and $\boldsymbol{P}_{CD}^i$because only cells that contain valid values in $\boldsymbol{P}_{CD}^i$ are taken into calculation, and cells with NOT DATA values are ignored. If w_{cd}=0, realizations are unconditional simulation results.

3.2. M-step: update

After the E-step search process, the simulation grid should be updated via the M-step (Fig. 3). The function of the M-step is to minimize the objective function based on the patterns found in the E-step. To update the value of the black cell in ***R***, all of the patterns in ***R*** that contain the black cell must be found (only a subset of all such patterns is drawn). Then, the nearest neighbors



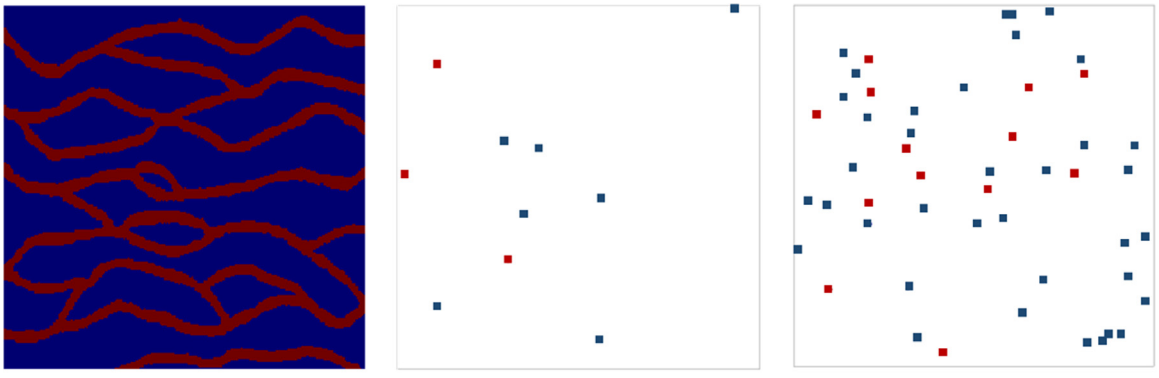
a)TI and its histogram



b) MS-CCSIM realizations and corresponding histograms

c) GOSIM realizations and corresponding histograms

Fig. 8. Comparison of unconditional realizations of 2D continuous cracks.



2D binary TI

10 conditioning data

50 conditioning data

Fig. 9. TI and conditioning data considered in 2D conditional simulation tests.

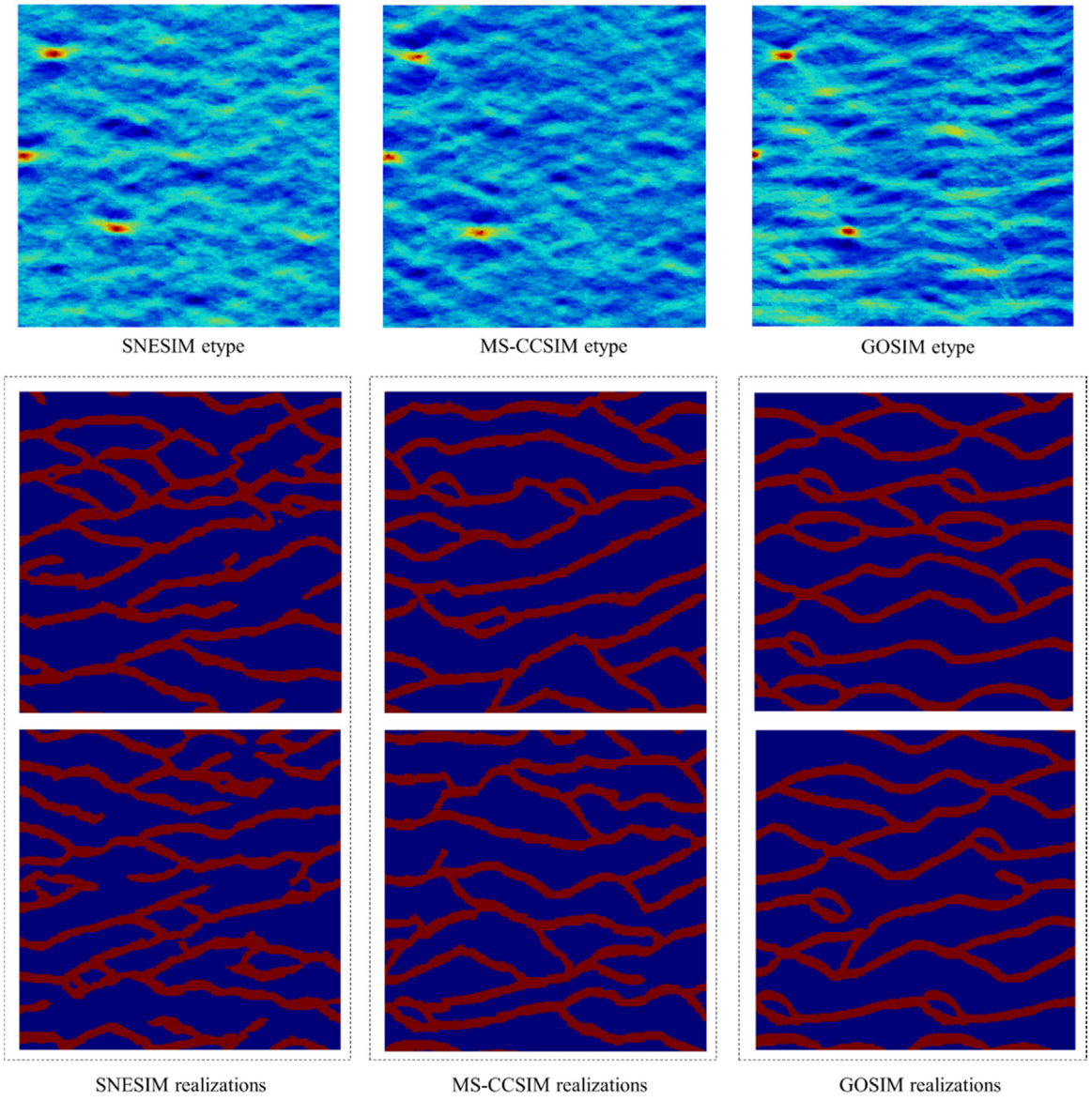


Fig. 10. Comparison of 2D conditional realizations with 10 hard point data.

of these patterns are found in $\boldsymbol{TI}$ (drawn as the dashed boxes). Lastly, according to the values of the corresponding cells (drawn as the cells filled with colors) in these nearest neighbors, the new value of the black cell is replaced by the weighted average of these corresponding cell values. After all of the cells in $\boldsymbol{R}$ are updated, a new realization, $\boldsymbol{R}'$, is obtained. The M-step can be described as follows.

Let $\boldsymbol{P}_R^1, \ldots, \boldsymbol{P}_R^m$ denote all of the patterns in $\boldsymbol{R}$ that contain cell $\boldsymbol{u}_R$, and $\boldsymbol{P}_{TI}^1, \ldots, \boldsymbol{P}_{TI}^m$ denote the nearest neighbors in $\boldsymbol{TI}$ found during the E-step, i.e.,

$$\boldsymbol{P}_{TI}^i = \arg\min_{\boldsymbol{P}_{TI} \in \boldsymbol{TI}} D(\boldsymbol{P}_{TI}, \boldsymbol{P}_R^i) \qquad i = 1, 2, \ldots, m \tag{11}$$

Let $\boldsymbol{u}_{TI}^1, \ldots, \boldsymbol{u}_{TI}^m$ represent the cells in $\boldsymbol{P}_{TI}^1, \ldots, \boldsymbol{P}_{TI}^m$ that are at the same relative position as $\boldsymbol{u}_R$ within $\boldsymbol{P}_R^1, \ldots, \boldsymbol{P}_R^m$. Let $\boldsymbol{R}(\boldsymbol{u}_R)$ denote the value of cell $\boldsymbol{u}_R \in \boldsymbol{R}$ and $\boldsymbol{TI}(\boldsymbol{u}_{TI}^i)$ denote the value of cell $\boldsymbol{u}_{TI}^i \in \boldsymbol{TI}$. Then, the new value of cell $\boldsymbol{u}_R \in \boldsymbol{R}$ is:

$$\boldsymbol{R}(\boldsymbol{u}_R) = \frac{\sum_{i=1}^m \boldsymbol{w}_i \boldsymbol{TI}(\boldsymbol{u}_{TI}^i)}{\sum_{i=1}^m \boldsymbol{w}_i} \tag{12}$$

where $\boldsymbol{w}_i$ represents the importance of $\boldsymbol{u}_{TI}^i \in \boldsymbol{TI}$. We can increase $\boldsymbol{w}_i$ when some cell values are more important to the realization, or decrease $\boldsymbol{w}_i$ if the opposite is true. Eq. (12) is the update rule. $\boldsymbol{w}_i = 1$ means that each cell has an equal contribution to the result.

In GOSIM, a histogram matching method proposed by Kopf et al. (2007) is provided as an optional function. In 3D simulations, the histogram matching function is turned on with a relatively small pattern to reduce the large time cost caused by a large number of $\boldsymbol{TI}$ cells.

3.3. Multi-scale strategy

The multi-scale strategy is illustrated in Fig. 4 with intermediate results from the 2D channel simulation case. The ratio between the grid sizes of adjoining scales is 1/2 (not displayed by scale in Fig. 4). The simulation grid is initialized at the coarsest scale. After a number of EM iterations at the coarsest scale, the realization is resampled to a finer scale. The resampled realization serves as the initial guess at the finer scale. Then, EM iterations are performed. This process is continued until all of the iterations at the finest scale have been completed. The Lancos filtering method (Burger and Burge, 2009) is suitable for image resizing and is used

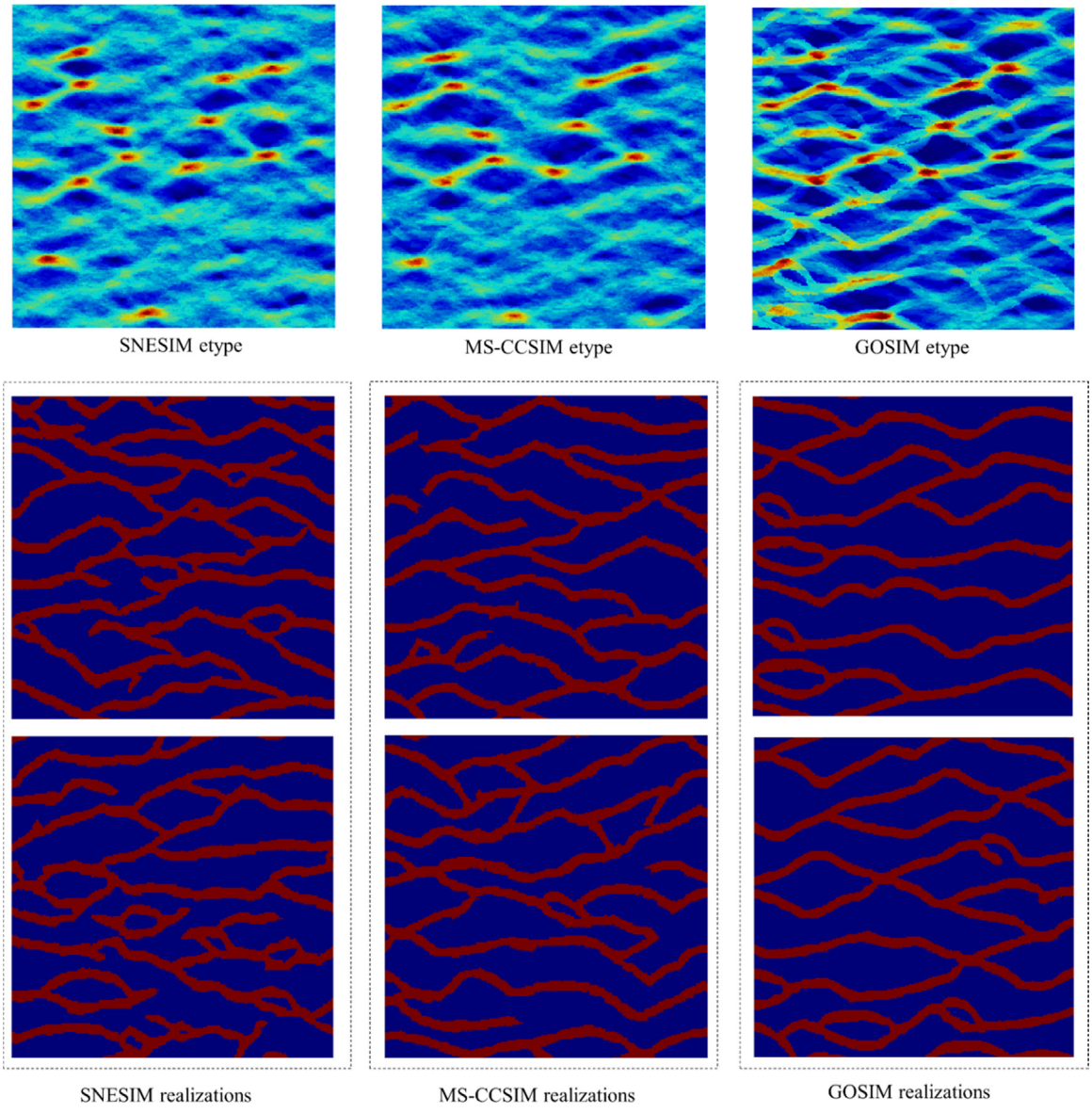


Fig. 11. Comparison of 2D conditional realizations with 50 hard point data.

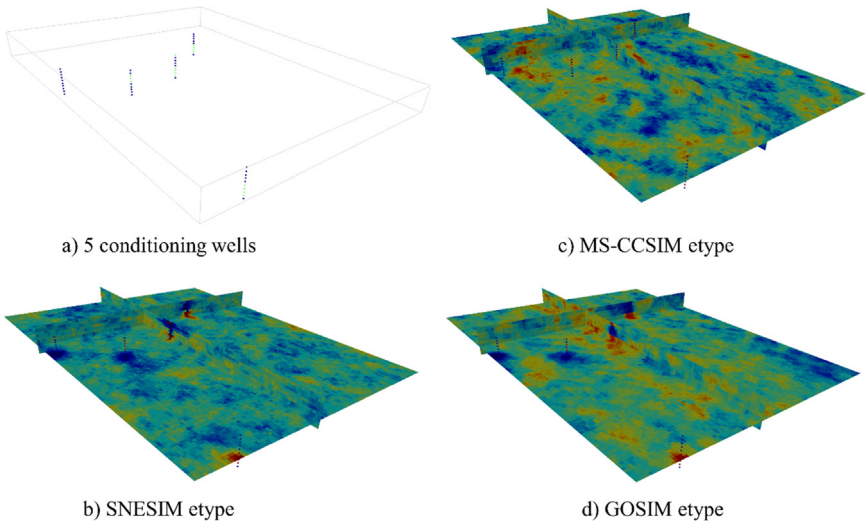


Fig. 12. 3D Conditioning data and etypes of SNESIM, MS-CCSIM and GOSIM.

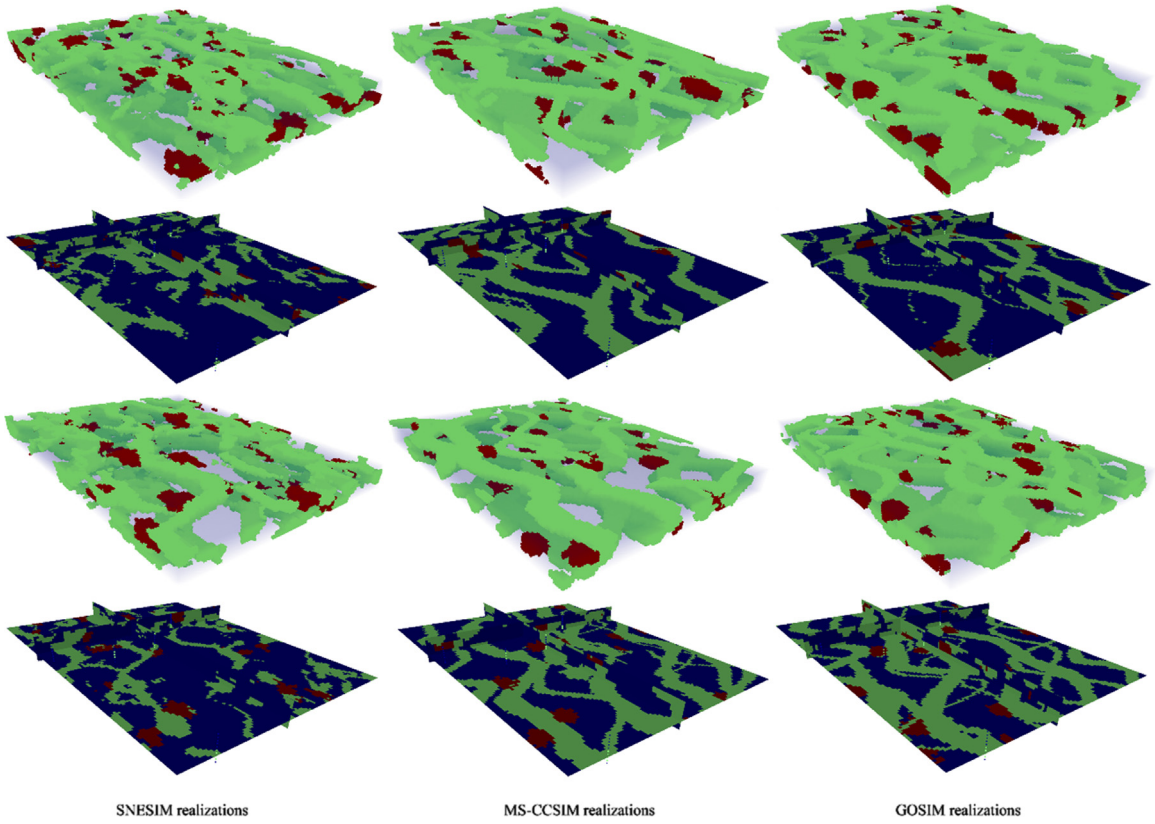


Fig. 13. Comparison of 3D conditional realizations.

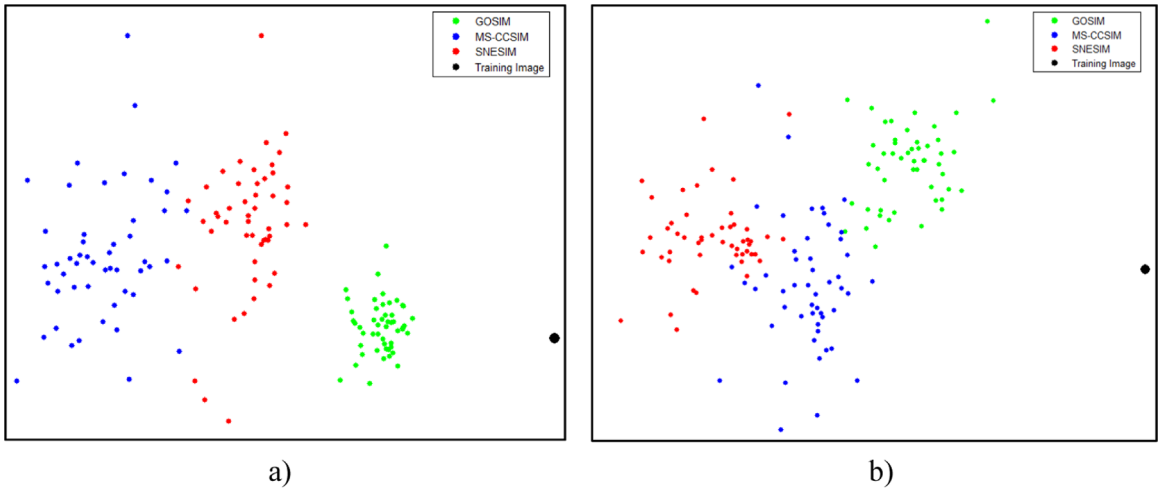


Fig. 14. MDS plot of TI and the SNESIM, MS-CCSIM and GOSIM realizations. (a) Original MDS plot; (b) New MDS plot after TI database enlargement.

Table 4
ANODI scores of SNESIM, MS-CCSIM and GOSIM.

Ranking	SNESIM	MS-CCSIM	GOSIM
Uncertainty space (between)	2.18	2.41	1
Pattern reproduction (within)	1.90	2.74	1
Total (between/within)	1.15	0.88	1

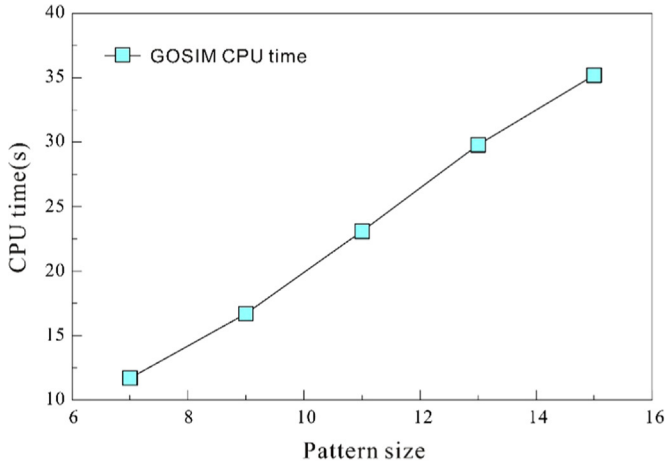
Table 5
ANODI scores of SNESIM, MS-CCSIM and GOSIM with enriched TIs.

Ranking	SNESIM	MS-CCSIM	GOSIM
Uncertainty space (between)	0.97	1.26	1
Pattern reproduction (within)	1.80	1.41	1
Total (between/within)	0.54	0.89	1

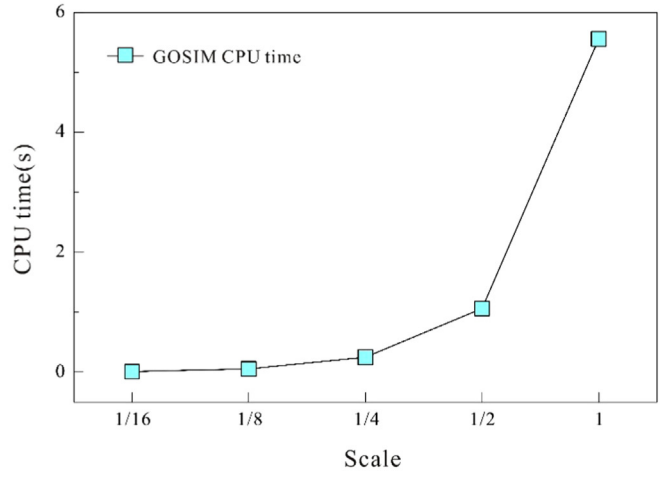
for resampling in our experimental GOSIM code.

3.4. Categorical data simulation

For categorical simulation, initially, each category is assigned an integer value and the simulation process for categorical data are the same as the process for continuous data. As shown in Fig. 4, cell values may end up with decimals after updating or resampling. In GOSIM, a variant of *k*-means clustering (Lloyd, 1982) called *k*-means++ (Arthur and Vassilvitskii, 2007) is utilized to



a) Response of CPU time to pattern size.



b) CPU time of one iteration at various scales.

Fig. 15. Sensitivity of GOSIM's time cost to pattern size and iteration number at various scales.

reclassify the final result back to categorical values. Fig. 5 shows the effect of k-means++ on the final realization in Fig. 4. The TI contains sand channels and mud background, and the integers 0 and 1 are assigned to indicate these two facies, respectively. The realization before reclassification contains a small proportion of continuous values around the channels (the left image in Fig. 5). The continuous values are turned into categorical indicators after transformation based on $k=2$ (the right image in Fig. 5).

4. Results and Discussion

4.1. Testing environment

In this study, three TIs (Fig. 6), a 2D channel TI (Strebelle, 2002), a continuous crack TI (Journel and Zhang, 2006; Honarkhah and Caers, 2010) and a 3D facies TI (Remy et al., 2009), are used to test the algorithm. The grid sizes of these three TIs are 250*250*1, 159*159*1 and 150*195*30, respectively.

Realizations produced by GOSIM, MS-CCSIM and SNESIM are compared with the visualization function of SGeMS (Remy et al., 2009). The current version of GOSIM is implemented in C++. SNESIM is the standard program of SGeMS. The MS-CCSIM algorithm is provided by Tahmasebi et al. (2014). The parameters for each algorithm are chosen according to the literature values that produce optimal simulation quality (Remy et al., 2009; Mahmud et al., 2014; Tahmasebi et al., 2014). All of the algorithm parameters used in our tests are listed in Tables 1–3.

4.2. Unconditional simulation

4.2.1. Case 1: categorical data simulation

Fig. 7 illustrates realizations based on the 2D channel TI (Fig. 6a) obtained by GOSIM, MS-CCSIM and SNESIM. In this case, the ideal simulation result would depict a realization with the same degree of connectivity and smoothness as the TI. Although we have set a sufficiently large search template for SNESIM, unconnected channels are widespread in the realizations. Compared to SNESIM, MS-CCSIM realizations have more connected channels. However, the channels in the MS-CCSIM realizations are not as smooth as the TI channels and the channel widths drastically vary. GOSIM realizations contain less hanging objects and reproduce smoother channels. However, the variability among GOSIM realizations is smaller than that of SNESIM and MS-CCSIM realizations.

4.2.2. Case 2: continuous data simulation

The simulation results of GOSIM and MS-CCSIM based on the continuous crack TI (Fig. 6b) are compared in Fig. 8. The major difficulty in this case is reproducing the completeness of both thick cracks and thin cracks. Each thick crack in TI is connected to other thick cracks, separating the TI into several blocks. Compared to GOSIM realizations, thick cracks in MS-CCSIM realizations are more fragmented. In addition, thin cracks in MS-CCSIM realizations are neither connected nor complete. GOSIM reproduces thin cracks more effectively. However, due to the weighted average update rule, blurry portions can be observed in some GOSIM realizations, such as the portion in the middle of the 2nd GOSIM realization. Less bias of histogram reproduction are observed in GOSIM realizations.

4.3. Conditional simulation

In conditional simulation, because of the multi-scale strategy, conditioning data are supposed to be relocated at coarse scales. In GOSIM, conditioning data are relocated to the nearest cell when the conditioning data grid is downsampled to a coarser scale. Other approaches, such as the dual-mesh method (Tahmasebi et al., 2014), are also potential choices. No matter what type of data relocation method is used, conditioning data at a coarse scale will not be as accurate as at a fine scale. In order to alleviate the influence of inaccurate conditioning data at coarse scales, smaller conditioning weights at coarser scales and larger conditioning weights at finer scales are used in GOSIM. The ensemble averages (etypes) of 50 realizations simulated using every algorithm are calculated to compare the performances of SNESIM, MS-CCSIM and GOSIM based on conditional simulation.

4.3.1. Case 1: 2D conditional simulation

The 2D channel (Fig. 6a) is used as TI in this test. 10 pixels and 50 pixels are randomly sampled from the TI and used as conditioning data for sparse and dense data simulation tests, respectively (Fig. 9).

Simulation results from SNESIM, MS-CCSIM and GOSIM based on 10 conditioning pixels and 50 conditioning pixels are compared in Figs. 10 and 11, respectively. The etypes of SNESIM and MS-CCSIM are similar, whereas some faint channels can be observed in the two GOSIM etypes in Figs. 10 and 11. Especially in Fig. 11, the etype of GOSIM maintains a stronger continuity between conditioning data than the etypes of SNESIM and MS-CCSIM. The GOSIM etypes show that some patterns tend to occur at certain

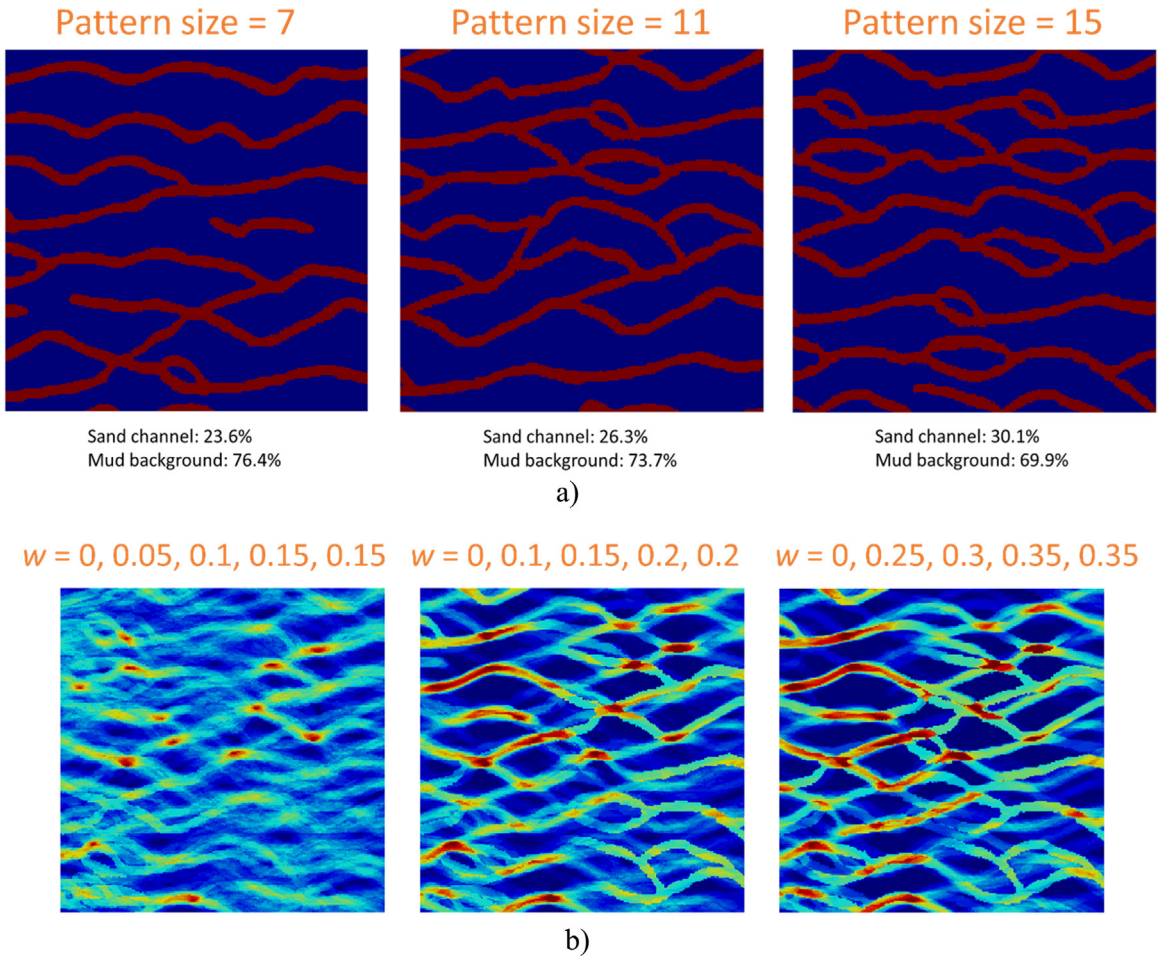


Fig. 16. Comparison of GOSIM's simulation quality with (a) realizations using different pattern sizes and (b) etypes using different conditioning data weights.

locations with the fixed conditioning data. It indicates a small variability among the GOSIM realizations. Compared to the SNESIM and MS-CCSIM realizations, GOSIM realizations contain more connected and smooth channels. Note that all conditioning data are matched in the GOSIM realizations, despite small conditioning weights at each scale. Some conditioning data are missed in MS-CCSIM realizations in Fig. 11.

4.3.2. Case 2: 3D conditional simulation

In this case, the 3D facies TI (Fig. 6c) is used. The size of the simulation grid is 100*130*10. Five virtual wells are randomly sampled from the TI as conditioning data (Fig. 12a). The etype of each algorithm is shown by 3 slices at $X=56$, $Y=38$ and $Z=3$ (Fig. 12b–d). Realizations produced by SNESIM, MS-CCSIM and GOSIM were shown in Fig. 13. All 5 wells are presented with etypes and realization slices to show the effect of conditioning.

According to the etypes (Fig. 12b–d) and realization slices (Fig. 13), the conditioning data are correctly matched by GOSIM realizations and SNESIM realizations, but MS-CCSIM realizations fail to condition to the majority of the 3D hard data. In the 3D facies TI, long channels are connected and wedge-shaped. The upper portions of a channel are always wider than those in the lower portions. Crevasses are approximately ellipsoid-shaped. In Fig. 13, the long channel connectivity and crevasse shapes are accurately reproduced via GOSIM and MS-CCSIM. However, in SNESIM realizations, some of the channels are unconnected, and shapes of channels and crevasses are irregular. Considering the effects of pattern reproduction and data conditioning, GOSIM performs better than SNESIM and MS-CCSIM in this case.

4.4. Trade-off between pattern reproduction and uncertainty space

Using the analysis of distance (ANODI), one can quantitatively assess the pattern reproduction and uncertainty space performance of different algorithms (Tan et al., 2014). ANODI has been proved to be an effective method for comparing the performance of MS-CCSIM, CCSIM and SNESIM (Tahmasebi et al., 2014). Fig. 14a illustrates an MDS plot of three groups of 50 unconditional realizations obtained by SNESIM, MS-CCSIM and GOSIM, where the 2D channel is used as the TI. The ANODI scores are listed in Table 4.

Based on the current parameters, the projected dots of GOSIM realizations are closer to the TI than MS-CCSIM and SNESIM, and these dots are close to each other (Fig. 14a). In Table 4, GOSIM has a higher between score and a lower within score than MS-CCSIM and SNESIM. Both the MDS plot and ANODI scores reveal that GOSIM possesses strength in pattern reproduction and limitations linked to uncertainty space. The limitation of GOSIM can also be observed in Fig. 7, in which some distinct patterns, such as the combination of two closed rings, appears in many of the GOSIM realizations. The total score suggests that SNESIM exhibits a better total performance than GOSIM and MS-CCSIM with current parameters.

A major cause of limited variability among GOSIM realizations is related to the fact that the TI is small and exhibits few patterns. This paper adopts a method similar to Rezaee et al. (2015) to enlarge the TI database, but we choose GOSIM realizations as the new TIs rather than CCSIM realizations in Rezaee et al. (2015). To compare the three algorithms in a similar condition, the same TIs are used for each algorithm. For each algorithm, 10 realizations are

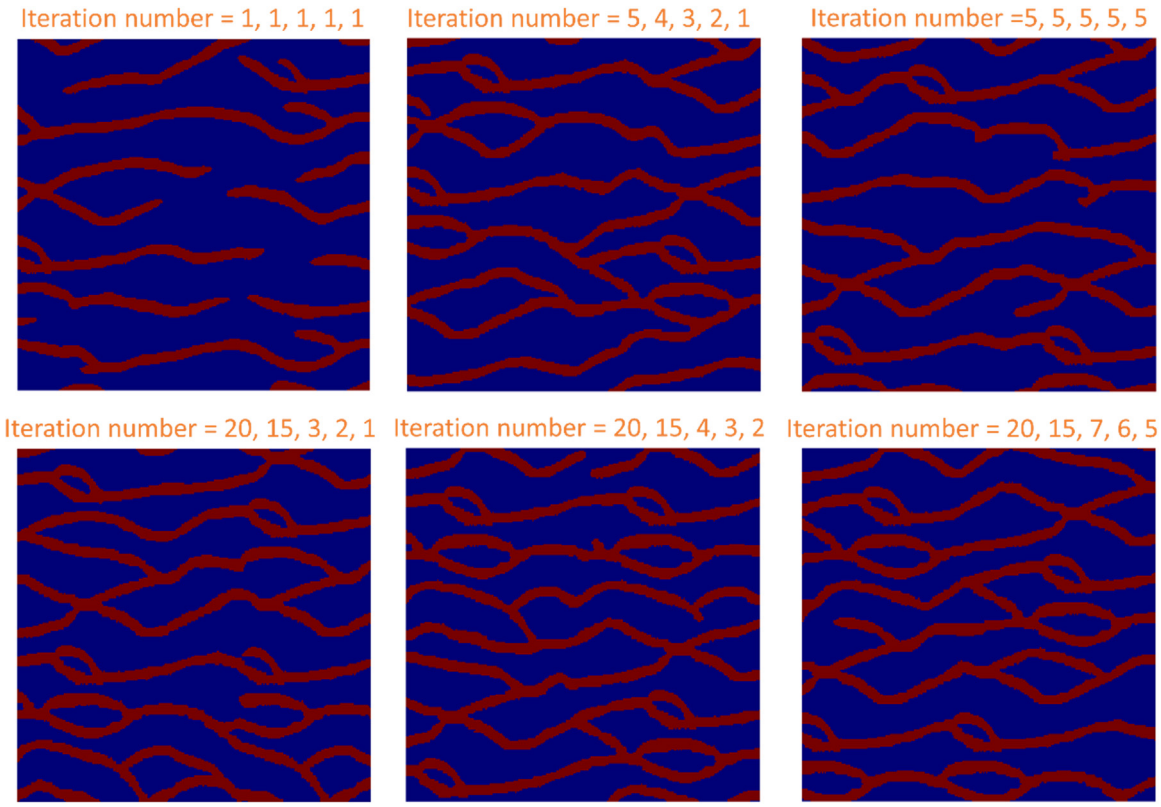


Fig. 17. Comparison of GOSIM's simulation quality with different iteration numbers at multi-scales.

produced with every TI including the original TI. New MDS plot (Fig. 14b) and ANODI scores (Table 5) that calculated after TI database enlargement illustrate that, via enlarging the TI database, the uncertainty space among GOSIM realizations increase dramatically and becomes similar to SNESIM. Simultaneously, GOSIM's pattern reproduction ability is still the best among the three algorithms. The total score indicates GOSIM has a better total performance than SNESIM and MS-CCSIM with enriched TIs.

4.5. Sensitivity analysis

The simulation quality and efficiency of GOSIM are subject to the algorithm parameters. We chose the 2D channel simulation as our sensitivity analysis case. The parameter setting in Section 4.2.1 is the reference setting for the unconditional simulation, and the parameter setting in Section 4.3.1 is the reference setting for the conditional simulation. If not specified, other parameters are set based on the reference setting.

4.5.1. Time cost sensitivity

Fig. 15a illustrates the time cost with different pattern sizes and Fig. 15b shows the time cost of one iteration at each scale. The simulation grid sizes at each scale are 15*15*1, 31*31*1, 62*62*1, 125*125*1 and 250*250*1, respectively. The simulation time approximately linearly increases as the pattern size increases. In Fig. 15b, the simulation time is trivial at the coarsest scale, but the time cost exponentially increases as the grid size increases. The GOSIM time cost is dominated by the time cost at finer scales. Thus, increasing the iteration numbers at finer scales will increase the time cost of GOSIM more significantly than at coarser scales.

4.5.2. Simulation quality sensitivity

Pattern size, conditioning data weight and iteration number contribute to the GOSIM simulation quality. The effect of pattern size has been discussed in previous pattern-based MPS algorithms, and the conditioning data weight effect has also been investigated in IQ (Mahmud et al., 2014). Fig. 16 depicts the GOSIM realizations and etypes based on different pattern sizes and conditioning data weights. In Fig. 16a, when pattern size is too small (pattern size=7), the proportion of channels in the realization is less than that in the TI. When the pattern size reaches 11, the pattern reproduction becomes acceptable. However, the simulation quality does not improve further when pattern size reaches 15. Fig. 16b shows that the variability among realizations decreases when the conditioning data weights increases.

With a multi-scale strategy, it is tedious to find the optimal iteration numbers at each scale. In Fig. 4, we notice that the shape and connectivity of channels are depicted in the initial image at the 3rd scale (scale 1/4), and does not significantly change at finer scales. We also know that increasing the iteration number at a coarse scale will not significantly increase the time cost, but it will at finer scales (Fig. 16b). Therefore, to obtain a good initial guess and save CPU time, we intend to assign large iteration numbers (typically larger than 15) at coarser scales and small iteration numbers (typically smaller than 5) at finer scales.

For a systematic discussion of the effect of iteration number, the 5 scales of the 2D channel TI are divided into two groups: coarser scales (scale 1/16 and scale 1/8) and finer scales (scale 1/4, scale 1/2 and scale 1). Fig. 17 illustrates six unconditional realizations with different iteration number settings. The first row shows the results with small iteration numbers at coarser scales. The second row illustrates the results with large iteration numbers at coarser scales. Fig. 17 shows that with small iteration numbers (smaller than or equal to 5) at coarser scales, the patterns in realizations are generally simple or incomplete. Iteration number change at finer scales does not significantly affect the simulation quality. This implies that the algorithm's performance at coarse scales is important. GOSIM is not sensitive to the iteration numbers at fine scales and fine scale iterations only slightly refine the realization.

5. Conclusions

The main contribution of this paper is adapting a global optimization algorithm for geostatistical stochastic simulations. Different from the sequential simulation process used in existing pixel-based or pattern-based algorithms, GOSIM overcomes the error accumulation issue with iterative global optimization. Compared with SNESIM and MS-CCSIM, GOSIM is more powerful at reproducing patterns from the TI in the case of both unconditional simulation and conditional simulation. GOSIM is suitable for applications that require high pattern reproduction quality.

One drawback of GOSIM is the inherent parameter tuning difficulty, especially related to iteration numbers. The sensitivity analysis shows that pattern size influences the pattern reproduction and algorithm efficiency of GOSIM. In conditional simulations, under the premise of matching all conditioning data, the conditioning data weight should be as small as possible to maintain a high simulation quality. GOSIM is more sensitive to the iteration number at coarse scales than at fine scales. A sufficiently large number is useful at the coarsest scale, while the iteration number at finer scales can be as small as possible to reduce time cost. The MDS plot and ANODI scores indicate that GOSIM has a small variability issue. This issue can be alleviated using TI database enlargement, but more approaches that can increase the variability between realizations should be investigated.

Because the E-step and M-step are separate, GOSIM shows promise when only lower-dimension TIs are available (Kopf et al., 2007). For example, in the case of 3D simulation with 2D TIs, it is possible to independently search for the nearest neighbors in each plane direction in the E-step. Then, the realization can be updated by averaging all of the corresponding cell values in the M-step. The difference between 3D simulations based on the 2D TI and 3D TI is that the three dimensional pattern searching is not required with 2D TI.

Acknowledgments

We thank the editor and anonymous reviewers for their critical review of the paper and constructive comments. This paper was presented at the IAMG 2015 annual conference held in Freiberg, Germany, September 5–13, 2015. This research was jointly funded by the NSFC Program (41102207, 41472300), the Fundamental Research Funds for the Central Universities (12lgpy19, 15lgjc45) and the Specialized Research Fund for the Doctoral Program of Higher Education of China (RFDP) (20100171120001).

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2015.12.020.

References

Arpat, G., Caers, J., 2007. Conditional simulation with patterns. Math. Geol. 39 (2), 177–203.

Arthur, D., Vassilvitskii, S., 2007. K-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.

Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D., 2009. PatchMatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. 28 (3), 1–24.

Boucher, A., Costa, J., Rasera, L., Motta, E., 2014. Simulation of geological contacts from interpreted geological model using multiple-point statistics. Math. Geosci. 46 (5), 561–572.

Burger, W., Burge, M., 2009. Principles of Digital Image Processing: Core Algorithms. Springer Ltd., UK, p. 329.

Caers, J., 2011. Modeling Uncertainty in the Earth Sciences. Wiley-Blackwell, John Wiley & Sons Ltd., UK, p. 229.

Dimitrakopoulos, R., Mustapha, H., Gloaguen, E., 2010. High-order statistics of spatial random fields: exploring spatial cumulants for modeling complex non-Gaussian and non-linear phenomena. Math. Geosci. 42 (1), 65–99.

Efros, A., Freeman, W., 2001. Image quilting for texture synthesis and transfer. In: Proceedings of ACM SIGGRAPH 2001, 12–17 August, pp. 341–346.

Efros, A., Leung, T., 1999. Texture synthesis by non-parametric sampling. In: Proceedings of IEEE International Conference on Computer Vision, 1999, Kerkyra, Greece, vol. 2, pp. 1033–1038.

Guardiano, F., Srivastava, R., 1993. Multivariate geostatistics: beyond bivariate moments. In: Soares, A. (Ed.), Geostatistics-Troia 1. Kluwer Academic, Dordrecht, pp. 133–144.

Honarkhah, M., Caers, J., 2010. Stochastic simulation of patterns using distance-based pattern modeling. Math. Geosci. 42 (5), 487–517.

Jha, S., Mariethoz, G., Evans, J., McCabe, M., 2013. Demonstration of a geostatistical approach to physically consistent downscaling of climate modeling simulations. Water Resour. Res. 49, 245–259.

Jones, P., Douglas, I., Jewbali, A., 2013. Modeling combined geological and grade uncertainty: Application of multiple-point simulation at the Apensu Gold Deposit, Ghana. Math. Geosci. 45 (8), 949–965.

Journel, A., Zhang, T., 2006. The necessity of a multiple-point prior model. Math. Geol. 38 (5), 591–610.

Kopf, J., Fu, C., Cohen-Or, D., Deussen, O., Lischinski, D., Wong, T., 2007. Solid texture synthesis from 2D Exemplars. ACM Trans. Graph. 26 (3).

Kwatra, V., Essa, I., Bobick, A., Kwatra, N., 2005. Texture optimization for example-based synthesis. ACM Trans. Graph. 24 (3), 795–802.

Kwatra, V., Schodl, A., Essa, I., Turk, G., Bobick, A., 2003. Graphcut textures: image and video synthesis using graph cuts. ACM Trans. Graph. 22 (3), 277–286.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Trans. Inf. Theory 28 (2), 129–137.

Mahmud, K., Mariethoz, G., Caers, J., Tahmasebi, P., Baker, A., 2014. Simulation of earth textures by conditional image quilting. Water Resour. Res. 50, 3088–3107.

Mariethoz, G., Renard, P., Straubhaar, J., 2010. The direct sampling method to perform multiple-point geostatistical simulations. Water Resour. Res. 46, W11536.

Mariethoz, G., Lefebvre, S., 2014. Bridges between multiple-point geostatistics and texture synthesis: review and guidelines for future research. Comput. Geosci. 66, 66–80.

Mariethoz, G., Caers, J., 2015. Multiple-point Geostatistics: Stochastic Modeling with Training Images. Wiley-Blackwell, John Wiley & Sons Ltd., UK, p. 364.

Mclachlan, G., Krishnan, T., 1997. The EM Algorithm and Extensions. Wiley-Blackwell, John Wiley & Sons Ltd., UK.

Mustapha, H., Dimitrakopoulos, R., 2011. HOSIM: a high-order stochastic simulation algorithm for generating three-dimensional complex geological patterns. Comput. Geosci. 37 (9), 1242–1253.

Parra, A., Ortiz, J., 2011. Adapting a texture synthesis algorithm for conditional multiple point geostatistical simulation. Stoch. Environ. Res. Risk Assess. 25 (8), 1101–1111.

Pyrcz, M., Deutsch, C., 2014. Geostatistical Reservoir Modeling, 2nd ed. Oxford University Press, New York (US), p. 431.

Remy, N., Boucher, A., Wu, J., 2009. Applied Geostatistics with SGeMS: A User's Guide. Cambridge University Press, New York (US), p. 259.

Rezaee, H., Marcotte, D., Tahmasebi, P., Saucier, A., 2015. Multiple-point geostatistical simulation using enriched pattern databases. Stoch. Environ. Res. Risk Assess. 29 (3), 893–913.

Straubhaar, J., Renard, P., Mariethoz, G., Froidevaux, R., Bessen, O., 2011. An improved parallel multiple-point algorithm using a list approach. Math. Geosci. 43 (7), 305–328.

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

Tahmasebi, P., Hezarkhani, A., Sahimi, M., 2012. Multiple-point geostatistical modeling based on the cross-correlation functions. Comput. Geosci. 16 (3), 779–797.

Tahmasebi, P., Sahimi, M., Caers, J., 2014. MS-CCSIM: accelerating pattern-based geostatistical simulation of categorical variables using a multi-scale search in Fourier space. Comput. Geosci. 67, 75–88.

Tan, X., Tahmasebi, P., Caers, J., 2014. Comparing training-image based algorithms using an analysis of distance. Math. Geosci. 46 (2), 149–169.

Wei, L., Levoy, M., 2000. Fast texture synthesis using tree-structured vector quantization. In: Proceedings of SIGGRAPH 00, New Orleans, LA USA, pp. 479–488.

Wei, L., Lefebvre, S., Kwatra, V., Turk, G., 2009. State of the art in example-based texture synthesis. In: Proceedings of EUROGRAPHICS 2009, 30 March–3 April, Munich, Germany, pp. 93–117.

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