Computers & Geosciences 90 (2016) 41–48

Contents lists available at ScienceDirect

Computers & Geosciences

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

Research paper

A fast two-step algorithm for invasion percolation with trapping

Yder Masson

Institut de Physique du Globe de Paris, 1 rue Jussieu, 75005, Paris, France

ARTICLE INFO

Article history:
Received 22 September 2015
Received in revised form
2 February 2016
Accepted 3 February 2016

Keywords:
Invasion percolation
Trapping
Cluster labeling
Algorithm
Hoshen Kopelman
Two phase flow

ABSTRACT

I present a fast algorithm for modeling invasion percolation (IP) with trapping (TIP). IP is a numerical algorithm that models quasi-static (i.e. slow) fluid invasion in porous media. Trapping occurs when the invading fluid (that is injected) forms continuous surfaces surrounding patches of the displaced fluid (that is assumed incompressible and originally saturates the invaded medium). In TIP, the invading fluid is not allowed to enter the trapped patches. I demonstrate that TIP can be modeled in two steps: (1) Run an IP simulation without trapping (NTIP). (2) Identify the sites that invaded trapped regions and remove them from the chronological list of sites invaded in NTIP. Fast algorithms exist for solving NTIP. The focus of this paper is to propose an efficient solution for step (2). I show that it can be solved using a disjoint set data structure and going backward in time, i.e. by un-invading all sites invaded in NTIP in reverse order. Time reversal of the invasion greatly reduces the computational complexity for the identification of trapped sites as one only needs to investigate sites neighbor to the latest invaded/un-invaded site. This differs from traditional approaches where trapping is performed in real time, i.e. as the IP simulation is running, and where it is sometimes necessary to investigate the whole lattice to identify newly trapped regions. With the proposed algorithm, the total computational time for the identification and the removal of trapped sites goes as $O(N)$, where N is the total number of sites in the lattice.

1. Introduction

Invasion percolation (IP) is an algorithmic model introduced by Wilkinson and Willemsen (1983) to model biphasic fluid migration in porous materials. IP considers the scenario of two immiscible fluids. The first fluid is slowly injected inside a random porous medium and displaces the second fluid that originally saturates the pore space. This process occurs naturally in various contexts, for example, when oil, water or gas migrate through reservoir rocks. In IP, the pore space is modeled as a lattice of sites that corresponds to larger interstitial spaces or pores between grains, connected by bonds, that represent smaller throats connecting neighboring pores. The invasion proceeds by invading the sites one by one. As the invasion goes, it may happen that some clusters formed by the defending fluid become completely surrounded by the invading fluid, this is called trapping. There are two IP model variants: invasion percolation without trapping (NTIP), where trapped sites are allowed to be invaded, and, invasion percolation with trapping (TIP), where the defending fluid is assumed incompressible and sites belonging to trapped clusters cannot be invaded. From an algorithmic point of view, trapping brings additional complexity to the IP procedure as it requires extra operations to search for trapped clusters. The focus of this paper is to propose an efficient algorithm for identifying the trapped sites in TIP.

Regardless of trapping, one often distinguishes between site IP and bond IP that intend to model two different types of displacement named drainage and imbibition (e.g. Lenormand and Bories, 1980; Chandler et al., 1982). Site IP intends to model imbibition where a wetting fluid (e.g. water) is invading a medium originally saturated by a non-wetting fluid (e.g. oil). Bond IP intends to model the opposite situation called drainage where a non-wetting fluid is invading a medium originally saturated by a wetting fluid. In practice, IP models have mainly been used for slow drainage in porous and fractured media, this is because the use of the IP for imbibition is not well justifiable, due to additional physical mechanisms (e.g., film flow and snap-off) which often accompany imbibition and are not taken into account in IP. I refer the reader to Løvoll et al. (2005) and Toussaint et al. (2005, 2012) for extensive discussions of various invasion scenarios. From an algorithmic point of view, there is no difference between bond IP and site IP and they can be modeled using a unique IP algorithm on appropriate lattices (e.g. Patzek et al., 2001). For the sake of clarity, I only consider site IP in the present study.

The site TIP procedure (Wilkinson and Willemsen, 1983) consists of the following steps:

1. Define a lattice of sites connected by bonds, injection sites (i.e. sites at which the invading fluid is injected), sink sites (i.e. sites

E-mail address: yder.masson@cal.berkeley.edu

http://dx.doi.org/10.1016/j.cageo.2016.02.003
0098-3004/

at which the defending fluid is free to escape), and set all sites to invadable.
2. To all sites in the lattice, assign an invasion potential (e.g. Wilkinson, 1984; Meakin et al., 1992; Glass and Yarrington, 1996)

$$P_i = \frac{2\sigma \cos(\theta_c)}{a_i} - \Delta\rho g (L - z_i) \quad (1)$$

where a_i is the effective radius of site i, θ_c is the equilibrium contact angle between the wetting fluid and the solid, $\Delta\rho$ is the density contrast between the two fluids, g is the acceleration of gravity, L is the height of the system to be invaded, and z_i is the elevation of site i. Practically, the radii a_i are chosen randomly from a given probability distribution.
3. Invade the sites one by one until the invading fluid percolates, i.e. repeat the following three steps until the invading fluid reaches a sink site:
(a) Identify the trapped clusters formed by the defending fluid that are not connected to a sink and set all sites belonging to these clusters to un-invadable.
(b) Among all invadable sites, find the site that is neighbor to the invading fluid and that has the maximum invasion potential.
(c) Invade that site.

The site NTIP procedure is similar to the TIP procedure except that step 3(a) in the above sequence is not performed.

The NTIP problem can be solved efficiently using a binary tree data structure (e.g. Knuth, 1998) that maintains an up-to-date list of sites that neighbor the invader. This data structure permits us to find the site with the largest invasion potential in $O(1)$ operation and to insert new neighbor sites in $O(\log(n))$ operations, where $n \leq N$ is the number of active sites in the list (e.g. Schwarzer et al., 1999; Sheppard et al., 1999; Masson and Pride, 2014). I refer to Masson and Pride (2014) for a detailed implementation of NTIP using a perfectly balanced binary tree that strictly guarantees $O(M \log M)$ execution time, where M is the number of sites invaded at percolation time.

Algorithms for TIP can be constructed by modifying existing fast algorithms that solve the NTIP problem. A classic approach is to identify the trapped clusters formed by the defender at each time step. The trapped sites are then flagged to prevent further invasion (i.e. step 3(a) in the above procedure). This can be done using cluster labeling algorithms such as the classic (Hoshen and Kopelman, 1976) algorithm, or a similar but more generic disjoint-set data structure (e.g. Knuth, 1998; Cormen et al., 2001). A nice analysis of algorithms that label isolated clusters is given by Babalievski (1998). Labeling clusters at each time step is however highly inefficient because it requires to scan the entire lattice M times which gives a computation time of $O(MN)$ for the trapping part of TIP, where N is the total number of sites in the lattice. A better approach is to first investigate the neighbors of each newly invaded site to check for trapping which is ruled out in most cases. If trapping is possible, a different algorithm is used to update the cluster labeling as necessary (e.g. Schwarzer et al., 1999; Sheppard et al., 1999; Meakin, 1991). In this work, I propose an alternative approach where clusters are labeled only once at the end of the IP invasion.

The aim of this paper is to detail a comprehensive and computationally efficient algorithm for solving the TIP problem. A search through the recent literature (Chen et al., 2012; Yang et al., 2013) shows that less efficient algorithms with execution time $O(MN)$ are still widely used, further, I found no study that provides the reader with a detailed fast TIP algorithm that can easily be turned into code. The present paper together with Masson and Pride (2014) should allow the graduate student or people less familiar with IP to get started quickly with adequate algorithms. The solution I propose for TIP is based on the simple observation that trapping can be treated a posteriori through time-reversal of the invasion sequence obtained in NTIP. Surprisingly, to my knowledge, this simple and efficient approach has not been reported in the literature. Numerical results show better performance than previously proposed algorithms.

2. The proposed TIP algorithm

In this section, I first show that TIP can be solved by post-processing the NTIP solution. Then, I detail an efficient two-steps algorithm for solving TIP. Finally I analyze the performance of the proposed algorithm.

2.1. Algorithm principle

Consider a NTIP sequence as illustrated in Fig. 1 and imagine a site belonging to a trapped region is invaded, e.g. as in Fig. 1b. We observe that the interface between the two fluids outside the trapped region (i.e. which encloses the newly invaded site) is not modified by this invasion. Therefore, this will not affect the way sites will be invaded outside this trapped region, i.e. which one of these sites will be invaded and in what order. In other words, the sites invaded in NTIP are the same as those invaded in TIP plus some extra sites belonging to trapped regions at invasion time. Based on this observation, I formulate the following proposition:

Proposition 1. *Given two IP simulations, a NTIP simulation S_{NT} and a TIP simulation S_T, both performed using the exact same setup (i.e. lattice, realization of invasion potentials, injection sites, sink sites, and boundary conditions), let L_T be the list of sites invaded in S_T sorted by increasing invasion time and L_{NT} be the list of sites invaded in S_{NT} sorted by increasing invasion time.* When removing the sites that invaded trapped regions from L_{NT}, we obtain L_T.

It follows from Proposition 1 that the TIP problem can be decomposed into two sub-problems that can be solved sequentially: (1) Solve the NTIP problem. (2) Identify and remove the sites that invaded trapped regions in NTIP.

One inefficient way to obtain the TIP invasion sequence knowing the NTIP solution is to proceed as for traditional TIP. That is, redo the invasion and, at each invasion step, check whether or not the newly invaded site belongs to a trapped region. A simple improvement to this procedure is to identify all clusters connected to a sink at the end of NTIP. In this case, all sites connected to a sink do not need to be re-investigated when searching for trapped clusters. I now show that further improvement can be achieved through time-reversal of the NTIP simulation.

Consider the situation where a new site is invaded in NTIP. There are three and only three possible scenarios regarding the evolution of the clusters formed by the defending fluid:

1. The number of clusters stays the same (see e.g. Fig. 1a).
2. One cluster of size one is suppressed (see e.g. Fig. 1d).
3. One cluster is split into multiple clusters (see e.g. Fig. 1e).

Now imagine the opposite situation where we go backward in time and un-invade a site, there are again three possible scenarios:

1. The number of cluster stays the same (see e.g. Fig. 1a).
2. A new cluster of size one is created (see e.g. Fig. 1d).
3. Multiple clusters are merged together (see e.g. Fig. 1e).

Notice the important difference between the two situations (i.e. going forward in time and going backward in time): on the one