The Improvement of Computer Algorithm for Forest Fire Model Based on Cellular Automata

Jian Wang
School of Safety science and engineering
Henan Polytechnic University
Jiaozuo, Henan, China
wjhpu@hpu.edu.cn

Xiaoyan Liu
School of Computer Science and Technology
Henan Polytechnic University
Jiaozuo, Henan, China
xyanliu@hpu.edu.cn

***Abstract*—The computer algorithm and its improvement of traditional forest fire model based on cellular automata are discussed in this paper. By optimizing algorithm of the model, dynamic updating the tree cluster, introducing the breadth-first search (BFS) algorithm and optimizing the internal memory using, the running time of the model is greatly shorten, and the occupation of internal memory is obviously reduced. The results show that the improvement makes the practical application of the model in forest fire forecasting and warning possible.**

Keywords-computer algorithm; cellular automata; breadth-first search; dynamic update; optimizing

I. INTRODUCTION

The traditional forest fire model (FFM) was proposed by Drossel-Schwabl and is used to explain the self-organized criticality (SOC) [1, 2] of the forest fire. This model is a cellular automata model and is used to simulate the fire occurrences and spread. In this model, the forest area is represented with a L×L lattice, in which trees grow with a probability p and fire occurs with a lesser probability $f<<p$. A burning tree will ignite all its neighboring trees (except those already burnt) so that a forest cluster will burn down in a single event [3]. The model can describe the spread process of the fire in a grid system in d-dimension. Usually, the model is calculated in 2-dimension in terms of the actual situation. In the model, there is a corresponding variable that indicates the current status of each grid, including "occupied" (there is a tree in this grid), "burning" (the tree in this grid is burning) or "empty" (no trees or the tree was burnt). When the model is realized on computer, in each time step, all the grids was updated as follows (it is called AL1 here):

1. If some grid A is occupied, and at least one of its neighbors is burning, then in the next moment, the status of the grid A should be changed to "burning" status.

2. If some grid A is occupied, and there is no one of its neighbors being burning, then in the next moment, there is a possibility f that grid A will be changed to "burning" status.

3. If some grid A is empty, then in the next moment, the grid will be changed to "occupied" status.

4. If some grid A has a status of "burning", then it will be changed to "empty" status in the next moment.

In the early realization of the FFM, the status of each grid occupied one byte, namely that the status was express by a 32 bit integer variable [4]. It resulted in a consequence that the calculation will occupy a lot of memory. Also, the traditional method spent a lot of time. These are the obstacle of the practical application of the model in forest fire prediction. So in this paper, the authors studied the optimizing of the algorithm and memory occupation to promote the computation speed.

II. OPTIMIZING OF THE ALGORITHM

As is well-known, time of fire spreading is much smaller than that of tree growing up and the interval of two forest fires, so in the optimized algorithm, the authors suppose that fire spread is completed in one time step. As a result, there are two statuses for a grid, "empty" and "occupied". So the status of each grid is represented by one bit of an integer variable. Because the change of each grid status can be resolved by means of simple logical calculus, which is the most efficient in current computer system, and the operation can be carried out with a set of grids simultaneously at the same time, the computation speed will be promoted by a wide margin. And, because each grid just need only one bit to indicate its status, the need of memory is reduced.

The specific algorithm is described as follows (for short AL2):

1. Select a grid randomly.

2. Determine whether the status of this grid is "empty". If the answer is positive, then the grid is changed to "be occupied" with possibility p, or else all the trees neighbored are burned with possibility f, namely being changed to "empty".

3. Repeat the two steps above.

Because the growing speed of tree is much smaller than burning speed, the burning of the cluster can be seen as an instantaneous process. So the algorithm can be updated as the following steps (for short AL3):

1. The following processes are conducted with possibility p:

(1) Select a grid randomly;

(2) Determine whether the status of this grid is "empty". If it is, then the status of this grid is changed to "occupied", otherwise, all the trees neighbored are burned with possibility f/p.

2. Repeat step 1.

Sponsored by:
the Natural Science Research Project of Education Department of Henan Province (2011B620002)
the Doctor Fund of Henan Polytechnic University (B2010-63)

978-1-4577-0536-6/11/$26.00 ©2011 IEEE

It can be seen that because of the reduction of the number of grids selected randomly and the calling times of the random number generator, the efficiency of the AL3 is higher than that of the AL2.

But one problem arises that whether the results of the calculation still keep the characteristic of SOC. Clar and Drossel [4] concluded that if the occupation of grids and the burning of the trees are completely independent, the results of AL3 and AL2 are equivalent. In these two algorithms, operation of burning the trees is just carried out in the "occupied" grids, while operation "occupy the grid" is conducted in the "empty" grids, so it is obviously that these two operations are independent absolutely. If the sample space is large enough, and the space scale is large enough too, from AL3 people can obtain the same results as AL2.

In the simulation the global time step is defined as the average time that all of the grids in the 2-dimension grid system are visited, so one global time step equals L×L loops. To reach the equilibrium state, all the data is gathered during 100 global time steps after running 20 global time steps.

III. TRACE AND DYNAMIC UPDATE OF CLUSTERS

With regard to a given grid, it's the key step of the algorithm that how to determine the tree clusters composed of its neighbors. This problem refers to two aspects, tracing and dynamic updating the clusters.

A. Trace of Clusters

To achieve trace of the tree clusters and then determine the set of them, the authors apply breadth-first search (BFS) [5, 6] accomplished based on Standard Template Library (STL) of C++. In this paper, the algorithm flow is as follows:

1. New a set "visited" represents all the nodes visited. Its initial value is null.

2. New a set "visiting" which represents the node being visited. Its initial value is the given burning source node.

3. If the set "visiting" is not null, repeat the following steps:

a) Select a node A from the set "visiting";

b) Delete node A from the set "visiting";

c) Insert node A in the set "visited";

d) With regard to all the neighbors B of node A, if its status is "occupied", and it is not in set "visited" nor in set "visiting", insert node B into set "visiting".

At the end, all the nodes in set "visited" represent all the trees in the cluster. In the initial simulation, the queue container and stack container are used. In these two containers the nodes searched off will be deleted, while all the nodes of the clusters are needed, another container, set container, is used to achieve the algorithm.

In set container, operations are completed by means of Red Black Tree or Balance Binary Tree (AVL Tree) [7], and its complexity is O(log*N*). It is obviously superior to linear search algorithm. Also, there are no functions to directly access the data, so the access of data can be just accomplished indirectly by iterator. The final data structure is shown in figure 1.

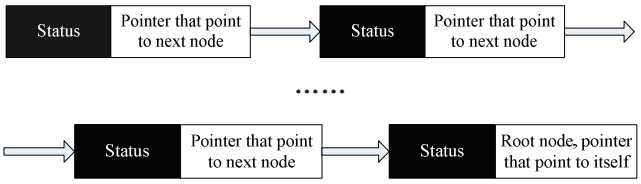

Figure 1. Structure of the data used

B. Dynamic Update of Cluster

To trace the size distribution of clusters and update them dynamically, the authors introduce another two variables: one represents the area of the cluster, and the other the pointer of a representative node in the cluster.

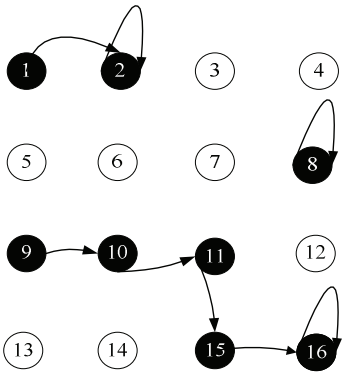

Figure 2. Structure of the girds

As shown in figure 2, black represents "occupied" status, and white "empty" status. Each "occupied" grid has a pointer pointing to representative node. These chain tables are formatted in the above BFS algorithm, and the pointer of the end node of the search points to itself. This node is defined as the root node. So each cluster can be defined by an exclusive root node.

In the early achievement of FFM, after updating the grids or the fire occurred, the parameters such as the size and amount of the tree cluster are recalculated. The efficiency is very low. So in this paper a new dynamic algorithm that the dynamic update of clusters can be completed by altering a few values is designed.

Calculation begins in an empty grid system. After some loops, one grid will be changed to "occupied" status from "empty" status, and because there is no neighbor being occupied, its representative node is itself. The root node is itself, and the area of the cluster is 1. Along with the continuous grow of trees, if

those new trees are still not neighboring to each other, repeat the steps above. If there are more than one neighbor are occupied, and these neighbors belong to different clusters, then,

(1)Search the root node of all the occupied neighbors;

(2)If the neighbor belongs to the same cluster with pre-neighbor, then neglect it.

(3)According to the area value stored in the root node, determine the largest cluster;

(4)Increase the area value of the root node of largest cluster. Its value equals sum of area values of all root nodes in the set plus one.

(5)Make the representative node of the root node of non-largest clusters point to the root node of largest cluster.

(6)Make the representative node of the new grid point to the root node.

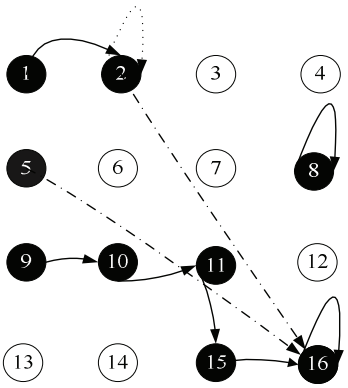

Figure 3. Transformation after inserting a new tree

(Dotted line represents pointer deleted. Dash and dot line represents new pointer.)

According to the initial situation shown in figure 2, if node 5 is the new node to be inserted, the following operation is shown as in figure 3. To reduce the height of binary tree used to store the tree clusters, node 1 can also point to node 16.

IV. Results and Discussions

This paper simulated three implementation method of the forest fire model as follows:

1) Method 1

a) Use the initial algorithm AL2;

b) Use a 32 bit integer but not one bit for each grid;

c) D o not update the cluster dynamically;

d) Use the BFS algorithm the authors compile themselves, but not by means of STL.

2) Method 2

a) Use the improved algorithm AL3;

b) Use 1 bit to represents each grid's status;

c) Do not update the cluster dynamically;

d) Use the BFS algorithm the authors compile themselves, but not by means of STL.

3) Method 3

a) Use the improved algorithm AL3;

b) Use a 32 bit integer to represents each grid according to the need of dynamic update;

c) Update the cluster dynamically;

d) Carry out the BFS algorithm using the set container based on STL.

In all these simulations, use f/p=0.005 as test value, and simulations are conducted in 100 global time steps. Time (in unit second) needed in each simulation is shown in table I.

TABLE I. Elapsed Time of Each simulation MethodTable Type Styles

Space Scale	Method 1 (s)	Method 2 (s)	Method 3 (s)
L=800	219	15	21
L=1600	3111	78	99
L=3200	Out of memory	410	657
L=6400	Out of memory	3995	6220

The results show that, with the initial algorithm, if people want to predict the forest fire spread behavior detailedly using software program, the elapsed time will be very large and even the process impossible because of out of memory. So this algorithm will be much useless in practical application. For example, when the spatial scale is 1600×1600, elapsed time reaches 3111 seconds, almost 1 hour. After the improvement to the algorithm, elapsed time is much reduced. In method 2, using one bit to represent the status of each grid, and neglecting the dynamic update of cluster, the elapsed time is about 1 hour for the spatial scale 6400×6400, and it is reduced to 78 seconds in the spatial scale 1600×1600. When dynamic update is referred to, and a 32 bit integer represents the status of each grid, the elapsed time is also much lower than the initial algorithm. In addition, the problem of memory overflow is also resolved through the improvement. These results indicate that the promotions of the algorithm are much efficient.

V. Conclusions

In this paper, the authors studied the computer algorithm and its improvement of forest fire model. By dynamic updating the tree cluster, introducing the breadth-first search (BFS) algorithm

and optimizing the internal memory using, the elapsed time is greatly reduced. The results show that these optimized measures are feasible and effective. These will produced contributes to the practical application of the forest fire model in the forest fire prediction and fire risk management.

REFERENCES

[1] P. Bak, C. Tang, K. Wiesenfeld, Self-Organized Criticality: An Explanation of 1/f Noise, Phys.Rev.Lett.1987, 59, 381-384.

[2] P. Bak, C. Tang, K. Wiesenfeld, Self-organized criticality, Phys.Rev.A 1988, 38, 364-374.

[3] W.G. Song, H.Y. Zheng, J. Wang, J. Ma, Weather-driven model indicative of spatiotemporal power laws, Phys. Rev. E. 2007, 75, 016109

[4] S. Clar, B. Drossel, F. Schwabl, Scaling Laws and Simulation Results for the Self-Organized Critical Forest-Fire Model, Phys.Rev.E 1994, 50, 1009-1018

[5] Ratan K. Ghosh, G. P. Bhattacharjee, Parallel Breadth-first Search Algorithms for trees and graphs. International Journal of Computer Mathematics, 1984, 15, 255-268.

[6] Luo Lijuan, Wong Martin, Hwu Wen-Mei, An Effective GPU Implementation of Breadth-first Search K, Proceedings-Design Automation Conference. Anaheim, CA, United states, June 2010, pp.52-55

[7] A. Gibbons, R. Ziani, Balanced Binary Tree Technique on Mesh-connected Computers, Information Processing Letters. 1991,37(2), 101-109