Hybrid cluster identification

J. Phys. A: Math. Gen. **37** (2004) 9377–9386 PII: S0305-4470(04)83525-X

Hybrid cluster identification

J Martín-Herrero

Department of Signal Theory and Communications, ETSIT, University of Vigo, E36200, Spain

E-mail: julio@uvigo.es

Received 24 June 2004, in final form 24 June 2004
Published 22 September 2004
Online at stacks.iop.org/JPhysA/37/9377
doi:10.1088/0305-4470/37/40/004

Abstract
I present a hybrid method for the labelling of clusters in two-dimensional lattices, which combines the recursive approach with iterative scanning to reduce the stack size required by the pure recursive technique, while keeping its benefits: single pass and straightforward cluster characterization and percolation detection parallel to the labelling. While the capacity to hold the entire lattice in memory is usually regarded as the major constraint for the applicability of the recursive technique, the required stack size is the real limiting factor. Resorting to recursion only for the transverse direction greatly reduces the recursion depth and therefore the required stack. It also enhances the overall performance of the recursive technique, as is shown by results on a set of uniform random binary lattices and on a set of samples of the Ising model. I also show how this technique may replace the recursive technique in Wolff's cluster algorithm, decreasing the risk of stack overflow and increasing its speed, and the Hoshen–Kopelman algorithm in the Swendsen–Wang cluster algorithm, allowing effortless characterization during generation of the samples and increasing its speed.

PACS numbers: 05.50.+q, 07.05.Kf

1. Introduction

Cluster identification is a low level technique fundamental in spatial analysis. Its purpose is assigning the same label to every connected site having the same state in a lattice. Connectivity is defined according to a neighbourhood rule, which depends on lattice dimension and geometry, usually a four-nearest-neighbour or eight-nearest-neighbour rule in a square two-dimensional lattice. Several algorithms for serial machines have been described for this task, which can be broadly classified between recursive and iterative techniques. Parallel implementations are not discussed here; the interested reader is directed to Alnuweiri and Prasanna (1992).

The best known iterative cluster labelling algorithm in physics is the Hoshen–Kopelman algorithm (HK) (Hoshen and Kopelman 1976), a two-pass algorithm that first propagates cluster labels sequentially to the forward neighbours while constructing a global equivalence table for conflicting labels, and then translates all labels to their definitive values according to the equivalence table on a second pass. It is a variation of the well-known Rosenfeld and Pfaltz 1966 iterative connected components algorithm that takes advantage of Tarjan's almost linear Union-Find algorithm (Tarjan 1975) to handle the equivalence table. Union-Find algorithms allow efficient construction and manipulation of equivalence classes using data trees, as first suggested by Galler and Fischer (1964). HK is widely used, and the most popular algorithm for cluster labelling in the field of percolation. The Swendsen–Wang (SW) cluster update algorithm (Swendsen and Wang 1987) uses HK to update cluster states in the generation of samples from the Ising model (Ising 1925) and the generalization for more than two states, the Potts model (Potts 1952). See Wu (1982) or Grimmet (1987) for a review.

On the recursive side, the pure recursive technique was already formalized in Tarjan (1972). It labels recursively every cluster in a single pass over the lattice, via a recursive call for each site in each cluster. Leath (1976) applied the idea to his cluster growth algorithm, to build a random binary lattice with a given occupation density. Wolff uses the same technique in his cluster Monte Carlo algorithm (Wolff 1989) for the Potts model, to update the spin state of an entire cluster at a time. Explicit algorithms and detailed treatment of the pure recursive technique for cluster labelling and percolation detection in small and huge lattices can be found in Martín-Herrero and Peón-Fernández (2000) where, incidentally, the basic algorithm was wrongly described as 'new'. The major drawback of the recursive technique is the intensive use of the stack, which threatens simulations with being interrupted by a stack overflow. This is due to the high number of consecutive recursive calls (recursion depth) issued by the algorithm when labelling big clusters. The advantages of the recursive technique are that it performs cluster labelling and, optionally, straightforward cluster characterization and percolation detection during the labelling in a single pass over the lattice, and its comparative simplicity, because it does not need handling any additional data structure as is the case with the two-pass iterative algorithms. It also allows the labelling and characterization of a given cluster without having to label all the lattice, an advantage in certain applications, as image processing or Wolff's algorithm.

Percolation theory is probably the field of physics that has been more prolific in cluster identification and related algorithms. There have been recent advances in simulation techniques for the study of percolation at different occupation densities and lattice geometries, with considerable gains in performance at the cost of increased specificity. Works like those of Paul *et al* (2001) or Babalievski (1998) focus on high-dimensional lattices. The former use a pure Leath technique to grow the sites on the spot; they do not label pre-built clusters. Babalievski compares spanning-tree approaches with HK, and presents a depth-first method for identifying loops of occupied sites. Sheppard *et al* (1999) report an efficient algorithm for simulating invasion percolation in big lattices. Rappaport (1986), Moukarzel (1998), Tiggemann (2001), or Moloney and Pruessner (2003) have also devised methods to deal with huge lattices, some of them implementing HK on parallel machines. Parallel implementations of HK can also be found in Flanigan and Tamayo (1995), Constantin *et al* (1997), or Teuler and Gimel (2000). Al-Futaisi and Patzek (2003) present an extended version of HK to deal with non-lattice environments. Newman and Ziff (2001) provide a method (including code) for studying percolation for all possible occupation densities, p, at a time in lattices built during the process, which outperforms the classical methods by several orders of magnitude (as a marginal note, I find their average 4.5 s for each possible value of p on a 1000^2 lattice using a typical depth-first search on a 2001 computer considerably high). However, this performance comes from the specific purpose of the method, as the authors recognize when they state that typical depth-first or breadth-first methods are optimal for a single value of p.

Algorithmic efficiency has many readings. Algorithms may exist which may be faster for a given task, but using quite complex data structures, rather obtrusive for the non-specialized computer programmer. Or the task may be so specifically constrained that the algorithms cannot be used for general purposes. Thus everything depends on what is expected from a given technique. The fact is that HK is used by a great number of researchers, to the point of being 'still the standard technique for identifying clusters in percolation' (Moloney and Pruessner 2003), in spite of the aforementioned works. This reflects the fact that not everybody has access to a parallel supercomputer, the convenience of simple, understandable, customizable code, and that many applications require the labelling of a pre-existent lattice, such that 'growing' the lattice is unsuitable.

Therefore, typical general purpose cluster labelling algorithms are still a useful tool, and the simplicity and elasticity of recursive labelling would make it a good alternative to HK if its stack limitations were overcome. The aim is maintaining the simplicity of the recursive approach, without any other data structure than the lattice itself, while dealing with the major drawback of recursive techniques, which is stack use; also preserving the single pass, one cluster at a time, character of the recursive technique, which makes it so flexible and suitable for many purposes.

Freed of the tight stack limitation, the hybrid technique I propose allows the labelling of moderately sized lattices. There are people interested in big lattices within percolation theory; some examples have been just mentioned. Besides, cluster labelling and percolation checking are useful techniques not only within percolation theory *per se*, but also in many fields where this theory is of use and huge lattices are common, for instance landscape analysis, where satellite images and mosaics of satellite images have to be dealt with. Moreover, countries do exist where the available hardware is well below the standards of wealthy countries. Size is relative. What is huge for a workstation 10 years old may be small for a modern desktop PC. In any case, the pure recursive technique needs a stack of about 5 Mb to prevent overflow when processing a lattice of 512^2 sites. That is too big a stack for not such a big lattice.

In the following, section 2 describes the hybrid approach to cluster labelling. Section 3 gives some results of testing the hybrid technique together with the pure recursive and iterative techniques on a set of uniform random lattices and on a set of Ising samples, regarding stack use and speed. Section 4 explains how to use the hybrid approach in Wolff's cluster algorithm and in the Swendsen–Wang algorithm, and gives some comparative results. Section 5 summarizes the paper and extracts the opportune conclusions.

2. Description of the method

A lattice is assumed to be stored into the computer random access memory as an integer array, where sites on a given state are assigned negative labels and the rest 0. The pure recursive technique scans the lattice in search of a negative label and calls the recursive function, which assigns the current cluster label to the site and explores its neighbours. If a negative label is found, the recursive function calls itself for that neighbour and the process is repeated. When the last site of the cluster has been thus labelled, the recursion ends, the current label is increased and the scan along the lattice is continued in search of the next cluster. Thus, a single pass over the lattice suffices to label all clusters with definitive consecutive labels, and each cluster is wholly labelled at one stroke, allowing cluster characterization (computation of cluster parameters) and percolation detection (checking if opposite ends of the lattice are reached) during the labelling. It also permits labelling a single cluster without having to label the whole lattice, which is, for instance, the idea behind Wolff's cluster update algorithm.

The drawback is that it requires a recursive call for each site in each cluster, many of them consecutive, and, therefore, the recursion depth can be very high, thus demanding a big stack, even in moderately sized lattices if a high occupation density or very compact clusters are expected.

The solution: let us use iterative scanning to explore along a direction, say along rows, and use recursion for the rest. Thus, on a two-dimensional lattice, the number of recursive calls, now used only to change row, is drastically reduced. Let a *burst* be a set of connected sites along a row having the same state bounded by sites having a different state (or the lattice borders, if reflecting boundaries are assumed). I use a recursive function which iteratively labels every site in a burst before exploring the adjacent rows. If another site on the appropriate state is found, a recursive call is issued for the burst on that row. Thus, the number of recursive calls is reduced from one per site in the cluster to one per burst in the cluster. Small clusters may have very few sites per burst, and thus the improvement may not be that significant for them, but small clusters never cause recursion depth troubles. Big clusters, more prone to requiring a high number of consecutive recursive calls, will also have a greater number of sites per burst, and, therefore, the decrease of the recursion depth is very significant. Regarding speed, the overhead of repeated function calls is higher than that of iterative scanning, and, therefore, the decrease in the number of recursive function calls balances the slightly increased complexity of the recursive function. Thus, the relaxation on stack requirements should come at no cost in overall performance.

The following pseudocode implements hybrid cluster labelling on a two-dimensional lattice $\mathbf{site}_{i,j}$:

```
Main
    $0 \rightarrow$ Label
    $\forall$ i,j | $\mathrm{site}_{i,j}<0$ do {increase Label, Hb(i,j)}
Hb(i,j)
    $i \rightarrow f$
    while $\mathrm{site}_{i-1,j}<0$ do {decrease i, Label $\rightarrow \mathrm{site}_{i,j}$}
    while $\mathrm{site}_{f,j}<0$ do {Label $\rightarrow \mathrm{site}_{f,j}$, increase f}
    while $i<f$ do {if $\mathrm{site}_{i,j\pm1}<0$ do Hb(i,j±1), increase i}
```

The recursive function first scans back to the beginning of the burst, labelling the scanned sites, then scans forward from the initial point to the end of the burst, labelling the remaining sites, and then scans the entire burst searching for adjacent bursts on the neighbouring rows. If any is found, a recursive call is issued to repeat the process on that row.

If characterization of the clusters during the labelling is required, the appropriate computations are included in the back scan and the first forward scan. Checking for percolation just requires checking if the extreme coordinates, **i** and **f**, reach opposite boundaries of the lattice within a cluster. This may be advantageously used if percolation check is the only purpose of the labelling: scanning the last column in the lattice is enough; if **i** reaches the first column during the labelling, the lattice percolates and the labelling can be halted.

3. Benchmark results

I have tested the performance of the hybrid (Hb) technique versus the pure recursive (PR) technique, versus a typical implementation of HK (Stauffer and Aharony 1994), and versus

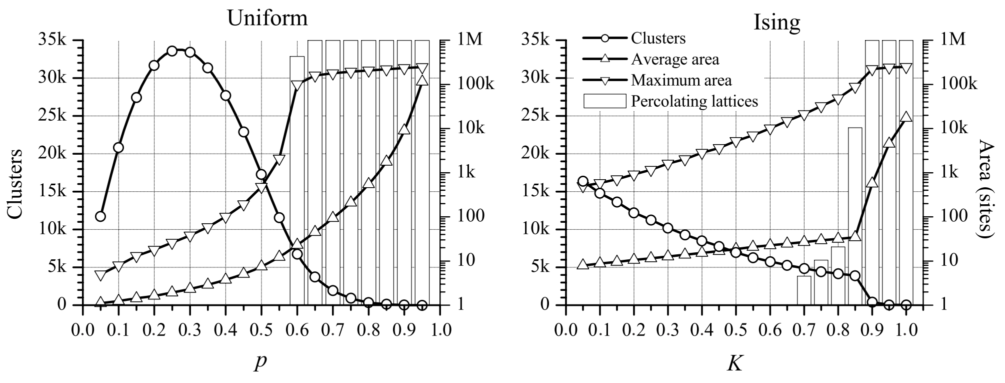

Figure 1. Number of clusters, average cluster area, average maximum cluster area and probability of percolation for the Uniform (left) and Ising (right) sets. The higher bars indicate 100%.

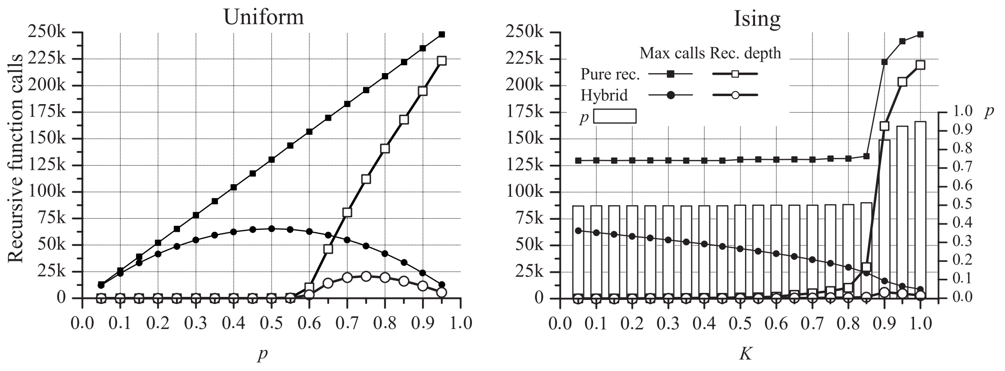

Figure 2. Number of recursive calls and recursion depth for the Uniform (left) and Ising (right) sets. The thicker lines with hollow symbols show recursion depth (maximum consecutive recursive calls). The bars show the average density of occupation in the Ising set.

the fastest iterative Union-Find (UF) implementation I have been able to write. For that purpose, I constructed two sets of pseudorandom two-dimensional lattices. One, which I call the Uniform set, is made of samples of a pseudorandom uniform distribution of occupied sites in a 512^2 lattice, 1000 samples per occupation density, p, ranging from 5% to 95% in steps of 5%. The other, which I call the Ising set, is made of samples of size 512^2 of the Ising model $P(\sigma) \propto \exp\left(K \sum \delta[\sigma_\mathrm{i}, \sigma_\mathrm{j}]\right)$, where $K = J/k_B T$, 1000 samples per value of K, ranging from 0.05 to 1.00 in steps of 0.05. Figure 1 characterizes the test sets.

Figure 2 shows recursive call data. In the Uniform set, the maximum recursion depth of PR was 223351 consecutive recursive calls at the highest p, while Hb issued just a maximum 20 820 consecutive recursive calls at $p = 0.75$, which is 9%. This implies that Hb required a stack 11 times smaller than PR. In the Ising set, figure 2(*b*), PR results are more or less the same than those for the Uniform set if the figures are compared using the occupation density, which in figure 2(*b*) is shown by the bars in the background. The maximum recursion depth was 219513 calls for PR, at $K = 1.00$, while Hb went down to a maximum 6037 ($K = 0.90$), which is less than 3%, thus requiring a stack 36 times smaller, i.e. Hb is able to label lattices greater than 3000^2 with the same stack required by PR for lattices of size 512^2.

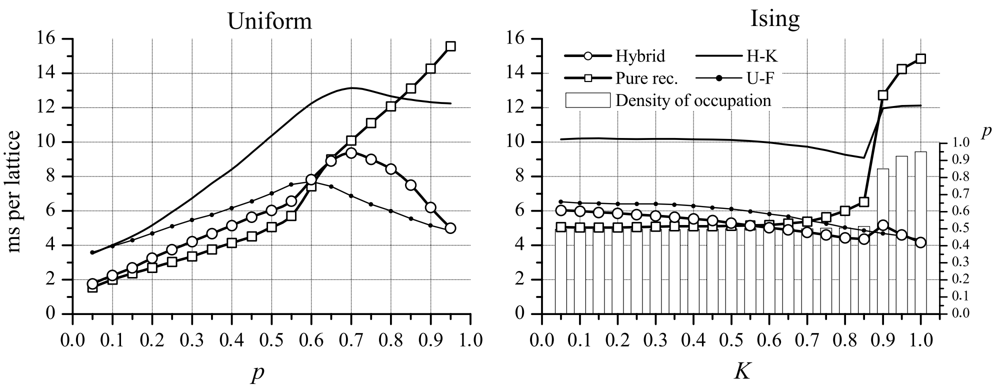

Figure 3. Average time per lattice used by HK, UF, PR and Hb in the labelling of the Uniform (left) and Ising (right) lattices. The bars show the average density of occupation in the Ising set.

Figure 3 shows the time spent in the labelling of the lattices by the four algorithms. Their behaviour is clearly different along the range of p in the Uniform set, but the overall results are favourable to Hb and UF, such that Hb averaged a 83% of the time consumed by PR, 61% of the time consumed by HK and virtually the same time (99%) consumed by UF. In the Ising set the results are clearly favourable to Hb: it averaged 74% with respect to PR, 51% with respect to HK and 91% with respect to UF. I repeated the same test with Uniform and Ising sets with different lattice size (100^2, 200^2, 300^2, 400^2 and 500^2), and the trend is similar to the behaviour shown in figure 3.

I also timed the algorithms while performing exclusively percolation detection. In the case of HK and UF this implies the labelling of the entire lattice and an additional scan of the opposite borders of the lattice in search of coincident labels. In the case of Hb and PR, I implemented the straightforward percolation detection scheme just described in section 2. This is not the only alternative to labelling the entire lattice. Ziff *et al* (1984) proposed a hull method, only for two-dimensional lattices, that surrounds the spanning cluster, thus checking for percolation without having to label all the lattice. Depending on the specific lattice configuration, this may be a faster approach, but whenever a direct path, or a quasi-direct path, from border to border exists, more probable at higher densities, a Hb percolation check just cuts through from border to border in one or a few iterative `for` loops. Thus, while the hull method runs in time $O(N^{7/8})$ in the criticality region, Hb could be running in time $O(N^{1/2})$ or close.

In the Uniform set, Hb required 1.8% of the HK time, 3% of the UF time and 70% of the PR time for percolation checking. In the Ising set, Hb required 1.6% of the HK time, 2.8% of the UF and 60% of the PR time.

4. 'Hybridizing' the Wolff cluster and Swendsen–Wang algorithms

The Wolff cluster algorithm generates samples of the Potts model using the pure recursive technique to flip the spin of part of a cluster. This is achieved by introducing a probability into the decision of entering (issuing the corresponding recursive call) a neighbouring site with the same spin. The major inconvenience of the algorithm is the possibility of stack overflow when big clusters and high probabilities are involved. Therefore, substituting the pure recursive technique by the hybrid technique should improve the algorithm. The hybrid technique is also faster than the pure recursive technique, an additional argument for introducing the hybrid

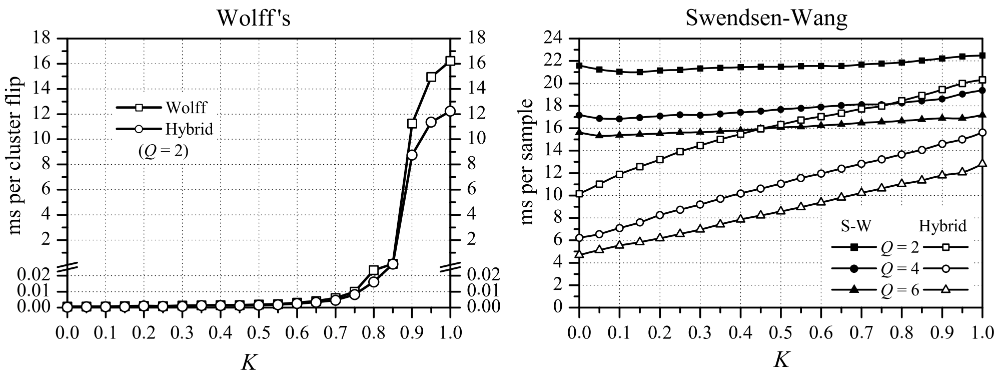

Figure 4. Left: average time per cluster flip with typical Wolff and hybrid Wolff in the generation of Ising ($Q = 2$) samples of size 512^2 for a range of values of K. Right: average time per Monte Carlo step (after convergence) with Wang's SW and hybrid SW for $Q = 2$, 4, 6 and K from 0.00 to 1.00.

approach into the Wolf algorithm. Periodic boundaries are usually assumed in Potts models to minimize edge effects. This, together with the probability check (the typical 'coin toss' of Monte Carlo simulations) are the only modifications required in the pseudocode above. Periodic boundaries are obtained by replacing the boundary checks needed every time a site coordinate is incremented or decremented (not explicit in the pseudocode above) by the appropriate conditional increments or decrements. Probability checks are ANDed to every check of the state of a lattice site (spin).

I introduced Hb with these modifications into Wolff's algorithm, and obtained a 25% time decrease with respect to the pure recursive version in the generation of Ising samples of size 512^2. The considerations of section 3 regarding the decrease in the risk of a stack overflow in the Ising set, or the capability to generate 36 times bigger samples with the same stack size apply exactly the same. Figure 4 (left) details the results.

The Swendsen–Wang algorithm uses another approach to Ising simulations, which involves bond connectivity. Bonds are located between sites sharing the same state with a given probability (function of $K = J/k_BT$). Then, the spin of the clusters defined by the bonds is randomly updated and the procedure is repeated for the new configuration. The implementation of Wang (2002) uses HK to identify the clusters and update their state. To replace HK with the hybrid recursive approach, Hb has to be modified to work with bond connectivity instead of site connectivity, and then the entire SW has to be adapted to work with the recursive technique. In Wang's implementation, first all clusters in the lattice are labelled with HK, then the equivalence table is rearranged (conflicting labels 'translated' to their corresponding 'canonical' labels, in usual notation), and then the labels in the equivalence table are randomly assigned the new states, such that, last, a scan over the lattice assigning the 'labels' to the lattice sites performs the cluster update. With the recursive labelling technique, randomly assigning one of the permitted states to the current label previous to labelling each cluster suffices to perform cluster identification (and characterization, if necessary) and update at the same stroke.

To work with bond connectivity, first an array of bonds has to be used to store the bonds, and then the site checks have to be replaced with bond checks. Also the bond which takes us to a site has to be removed after entering the site, or the labelling will endlessly recurse to the same sites.

The pseudocode below is an implementation of SW with hybrid cluster identification:

```
Main
    $\forall$ i<N initialize $spin_i$
    repeat MC times {
        $\forall$ i<N do {
            ($spin_i = spin_{i+1}$ and Rand(1) <P) $\rightarrow$ $hz_i$
            ($spin_i = spin_{i+L}$ and Rand(1) <P) $\rightarrow$ $vt_i$
        }
        $\forall$ i<N | ($hz_i$ or $vt_i$) do {Rand(Q) $\rightarrow$ Label, Hb(i)}
    }
Hb(i)
    i $\rightarrow$ j
    while $hz_{i-1}$<0 do {decrease i, Label $\rightarrow$ $spin_i$, 0 $\rightarrow$ $hz_i$}
    Label $\rightarrow$ $spin_j$
    while $hz_j$<0 do {0 $\rightarrow$ $hz_j$, increase j, Label $\rightarrow$ $spin_j$}
    while i<>j do {
        if $vt_i$ do {0 $\rightarrow$ $vt_i$, Hb(i + L)}
        if $vt_{i-L}$ do {0 $\rightarrow$ $vt_{i-L}$, Hb(i − L)}
    }
```

Decrements and increments of the location index are of course conditioned by the usual periodic boundary assumption, but this is not explicit in the pseudocode above for clarity. The lattice is stored into a one-dimensional array $\mathbf{spin}_i$, and two auxiliary binary arrays, $\mathbf{hz}_i$ and $\mathbf{vt}_i$, are used to store the horizontal and vertical bonds. **Rand(sup)** is a pseudorandom generator for a uniform distribution in [0, **sup**). **N** is the number of sites in the lattice, **L** is the length of a row, **Q** is the number of states, **P** is the bond probability and **MC** is the number of Monte Carlo steps.

I ran Wang's SW implementation (particularized for 2D lattices) and my hybrid implementation for Potts with two, four and six states, and the hybrid version took just an average 63% of the time consumed by Wang's (73% with two states, 61% with four states and 53% with six states). Figure 4 (right) details the results. Wang's SW has a flatter behaviour along *K*, but hybrid SW performs significantly faster. Besides, cluster characterization and percolation check are far simpler with the hybrid version than with HK SW.

5. Conclusions

Recursive cluster identification is a valuable alternative to HK if its tight stack constraints are relaxed, because it allows easy cluster characterization and straightforward percolation detection with quite simple code, and it is fast. Besides, it may be the only reasonable alternative when only one cluster has to be labelled, as it is the case with Wolff's cluster update algorithm. Hybrid cluster identification preserves the advantages of pure recursive labelling while greatly relaxing the stack requisites, as I have just shown.

Within percolation theory, some fast, efficient algorithms have been described that outperform any traditional cluster identification technique, but at the cost of highly specialized goals. If a given study does not fit into their constraints, traditional techniques like HK or the pure recursive technique are optimal. When this is the case, we have seen some examples of how the hybrid technique may substitute either the pure recursive technique or HK to perform the same task with improved performance and increased flexibility.

Specifically, I have shown results from the labelling and percolation detection in random 2D lattices and Ising 2D samples where the hybrid technique outperforms HK and the pure recursive technique. I have also presented hybrid versions of Wolff's algorithm and of the Swendsen–Wang algorithm, in both cases with an increase in speed. The hybrid Wolff's algorithm benefits also of the corresponding relaxation on the stack requirements without increasing the complexity of the algorithm, and the hybrid Swendsen–Wang algorithm has considerably simplified code and easier cluster characterization as additional advantages when compared to the typical implementation.

In this paper I have only dealt with two-dimensional lattices, which is the case where hybrid labelling presents a greater advantage: half the directions are changed from recursive to iterative. The higher the number of dimensions, the lesser the improvement, because every additional dimension has to be dealt with using recursive calls. In the limit, hybrid labelling converges to pure recursive labelling.

Recursive cluster identification has the advantage over HK-like techniques that it does not require any additional data structure, resorting instead to a built-in hardware feature, the system stack. The stack is an inherent characteristic of the hardware, therefore very fast and efficient, transparently accessed via recursive function calls through the built-in *push* and *pop* instructions, without any further consideration for the programmer than its limited size. Thus, recursive labelling does not need additional data structures, such as HK's equivalence table, with the overhead of complex, when compared to the built-in processor instructions, handling routines. The major improvement of the hybrid technique with respect to the pure recursive technique is the drastic reduction in the recursion depth, and therefore of the risk of a stack overflow. Thus, the hybrid technique, while keeping the same approach of the pure recursive technique, and thus its advantages, optimizes stack use and, therefore, performance.

References

Al-Futaisi A and Patzek T W 2003 Extension of Hoshen–Kopelman algorithm to non-lattice environments *Physica* A **321** 665

Alnuweiri H M and Prasanna V K 1992 Parallel architectures and algorithms for image component labelling *IEEE Trans. Pattern Anal. Mach. Intell.* **14** 1014

Babalievski F 1998 Cluster counting: the Hoshen–Kopelman algorithm vs. spanning tree approaches *Int. J. Mod. Phys.* C **9** 43

Constantin J M, Berry M W and Van der Zanden B T 1997 Parallelization of the Hoshen–Kopelman algorithm using a finite state machine *Int. J. Supercomput. Appl. High Perf. Comput.* **11** 34

Flanigan M and Tamayo P 1995 Parallel cluster labelling for large-scale Monte Carlo simulations *Physica* A **215** 461

Galler A B and Fischer M J 1964 An improved equivalence algorithm *Commun. ACM* **7** 301

Grimmet G 1987 Interacting particle systems and random media: an overview *Int. Stat. Rev.* **55** 49

Hoshen J and Kopelman R 1976 Percolation and cluster distribution: cluster multiple labelling technique and critical concentration algorithm *Phys. Rev.* B **14** 3438

Ising E 1925 Beitrag zur theorie des ferromagnetismus *Z. Phys.* **31** 253 (English translation available at http://www.fh-augsburg.de/~harsch/anglica/Chronology/20thC/Ising/isi_fm00.html)

Leath P L 1976 Cluster size and boundary distribution near percolation threshold *Phys. Rev.* B **14** 5046

Martín-Herrero J and Peón-Fernández J 2000 Alternative techniques for cluster labelling on percolation theory *J. Phys. A: Math. Gen.* **33** 1827

Moloney N R and Pruessner G 2003 Asynchronously parallelized percolation on distributed machines *Phys. Rev.* E **67** 37701

Moukarzel C 1998 A fast algorithm for backbones *Int. J. Mod. Phys.* C **9** 887

Newman M E J and Ziff R M 2001 A fast Monte Carlo algorithm for site or bond percolation *Phys. Rev.* E **64** 16706

Paul G, Ziff R M and Stanley H E 2001 Percolation threshold, fisher exponent, and shortest path exponent for 4 and 5 dimensions *Phys. Rev.* E **64** 26115

Potts R B 1952 Some generalized order-disorder transformations *Proc. Camb. Phil. Soc.* **48** 106

Rappaport D C 1986 Cluster number scaling in two-dimensional percolation *J. Phys. A: Math. Gen.* **19** 291

Rosenfeld A and Pfaltz J 1966 Sequential operations in digital picture processing *J. Assoc. Comput. Mach.* **13** 471

Sheppard A P, Knackstedt M A, Pinczewski W V and Sahimi M 1999 Invasion percolation: new algorithms and universality classes *J. Phys. A: Math. Gen.* **32** L521

Stauffer D and Aharony A 1994 *Introduction to Percolation Theory* (London: Taylor and Francis)

Swendsen R and Wang J 1987 Nonuniversal critical dynamics in Monte Carlo simulations *Phys. Rev. Lett.* **58** 86

Tarjan R E 1972 Depth-first search and linear graph algorithms *SIAM J. Comput.* **1** 146

Tarjan R E 1975 Efficiency of a good but not linear set union algorithm *J. Assoc. Comput. Mach.* **22** 215

Teuler J M and Gimel J C 2000 A direct parallel implementation of the Hoshen–Kopelman algorithm for distributed memory architectures *Comput. Phys. Commun.* **130** 118

Tiggemann D 2001 Simulation of percolation on massively-parallel computers *Int. J. Mod. Phys.* C **12** 871

Wang J 2002 http://www.cz3.nus.edu.sg/~wangjs/Bworkshop/sw-oner-gg.c

Wolff U 1989 Collective Monte Carlo updating for spin systems *Phys. Rev. Lett.* **62** 361

Wu F Y 1982 The Potts model *Rev. Mod. Phys.* **54** 235

Ziff R M, Cummings P T and Stell G 1984 Generation of percolation cluster perimeters by a random walk *J. Phys. A: Math. Gen.* **17** 3009