IDENTIFYING CRITICALITY VIA PERCOLATION TRANSITION IN ANISOTROPIC SYSTEMS

SIMON SUCHAN

Bachelor's Thesis

Institute for Theoretical Solid State Physics
Faculty of Mathmatics, Computer Science and Natural Sciences
RWTH-Aachen University
First examiner: Prof. Dr. Stefan Weßel
Second examiner: Prof Dr. Moritz Helias
Aachen, 25th July 2022

ABSTRACT

In this thesis, we study the general Potts model and how we can determine critical properties by applying computational algorithms. Most of these algorithms were implemented in isotropic systems. The main aim of this thesis is to generalise these algorithms for anistropic systems and to analyze, if the obtained data matches with theoretical solutions.

CONTENTS

1 INTRODUCTION

The general Potts model is a mathematical model used in different fields of studies like computer science, biology and physics. In physics, it is applied to classical statistical mechanics, it is used to describe the mixture of different fluids and it can be employed to describe the behavior of magnets. The main topic of this thesis, is to apply different algorithms to solve for quantities of the Potts model. We need to apply algorithms since we can not solve the general Potts model for all different configurations. There are many different algorithms, which have been applied to the Potts model. Most of them, are discussed in isotropic systems. Therefore it would be interesting to know if these algorithms can be generalized for anisotropic systems. The main algorithm we want to generalize, uses the percolation property of the Potts model around the critical temperature. Percolation describes the presence of a dominating cluster. In certain physical systems, there exists a temperature at which these dominating clusters exist. For example: These clusters could be a group of classical spins which are pointing in the same direction. This leads to a magnetisation which would only exist below the critical temperature. The general idea of the algorithm is to check if the system percolates or not. Depending if it does or not, we increase or decrease the temperature. With the resulting distribution of temperatures we gain information about the critical temperature. Since we are only able to simulate finite systems we can take advantage of the fact that these systems show a certain scaling pattern by which we simulate the temperature for different system sizes. We then can extrapolate the critical temperature for the thermodynamic limit, that is for very large systems. Before we can implement the algorithm, which is called the probability cluster changing algorithm [1] (PCC), we have to discuss how we can efficiently identify the clusters and determine if they are percolating or not. To do that, we use the Hoshen–Kopelman [2] and Newman-Ziff [3] algorithm.

2

INTRODUCTION TO PERCOLATION THEORY

The aim of this chapter is to give a brief overview to percolation theory and its properties. On a simple system, we want to present finite size scaling to understand how it can be useful to extrapolate quantities in the thermodynamic limit.

A visual introduction to percolation: Consider a 2D square-lattice. We will name the "dots" of every lattice "sites" and every connection of these dots we will call "bonds". If we randomly connect lattice sites, we will at some point create a connection between 2 opposing sites of the lattice. An example is visualised in Fig. 1

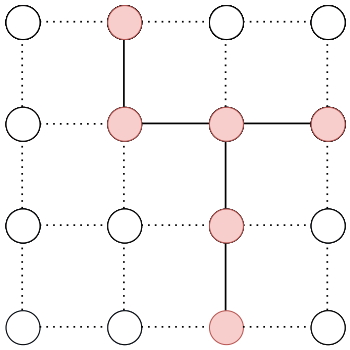
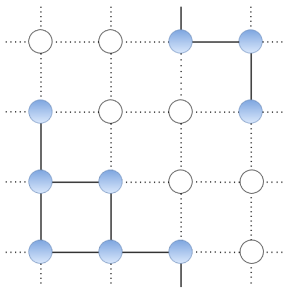


Figure 1: Bonds placed in a way, that connects the top and bottom part of the lattice

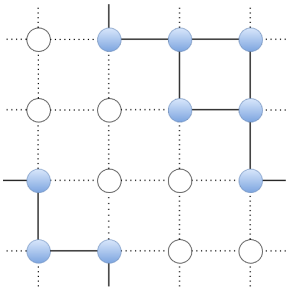
This system could be applied to conductivity or many other systems. So, a definition of percolation could be that the system is in a percolated state, if and only if two opposing sites are connected. We note here that there is no unique definition for percolation. In our example, we implicitly assumed open boundary conditions. But what if we use periodic boundary conditions? We are not able to define two opposing sites anymore. For periodic systems we will consider two different percolation rules.

1. The extension rule: If there exists a cluster of maximum extend L in one dimension, the system is in a percolated state.

2. The topological rule: There exists a cluster, which wraps around the entire system in at least one dimension.



(a) Example for percolation by the extension rule excluding the topological rule.



(b) Example for a cluster percolating via the topological rule and therefore by the extension rule.

Figure 2: Exemplary percolating clusters

An interesting property of these definitions is that if a cluster percolates following the topological rule, it implies that it percolates following the extension rule as seen in Fig. 2. The other direction of this statement does not hold true.

The next question would be: At which probability p do we have to allow bonds, at which we could almost certainly say that there exists such a percolating cluster. There are two approaches: This probability has been proven to be $p = 1/2$. But we could also simulate the system with a certain system size L, many times for different values of p and track the relative amount of systems, which end up in a percolated state. In the thermodynamic limit, the relative amount of percolated systems plotted against the bond placing probability p should become a step function. For finite systems, the function becomes sharper as the system sizes increase. This can be observed in Fig. 3, 4 and 5.

An interesting observation is that the finite percolation behavior is different for different percolation definitions. As we can see, the connecting probability p of the extension rule is lower than the connecting probability p of the topological rule. That is because it is more probable for a system to percolate in the sense of the extension rule. It is quite interesting that for each system size, the curves intersect at the critical probability. This observation was also made by Newman and Ziff [3] for various percolation definitions. By choosing a relative amount of percolated systems and tracking the resulting probability for different system sizes, we can use another useful property. Finite size scaling describes the convergence of quantities by sam-

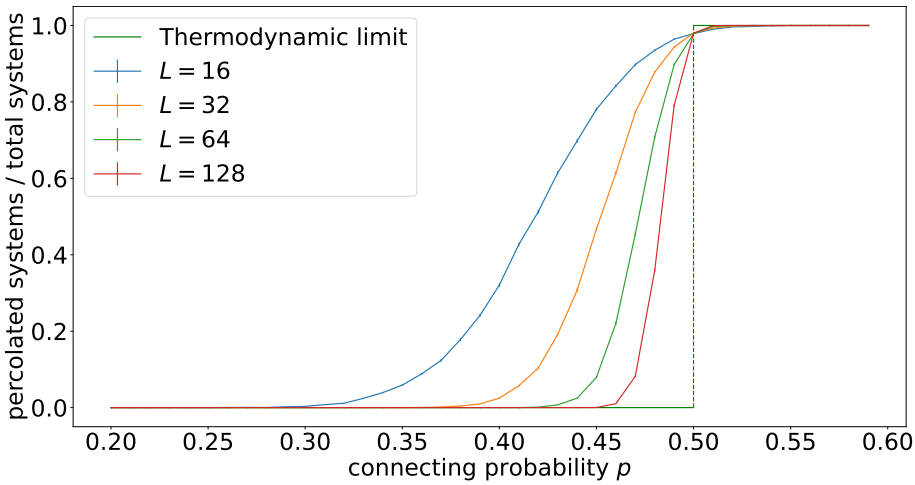


Figure 3: Percolation probability plotted against the connecting probability p for different system sizes using the extension rule

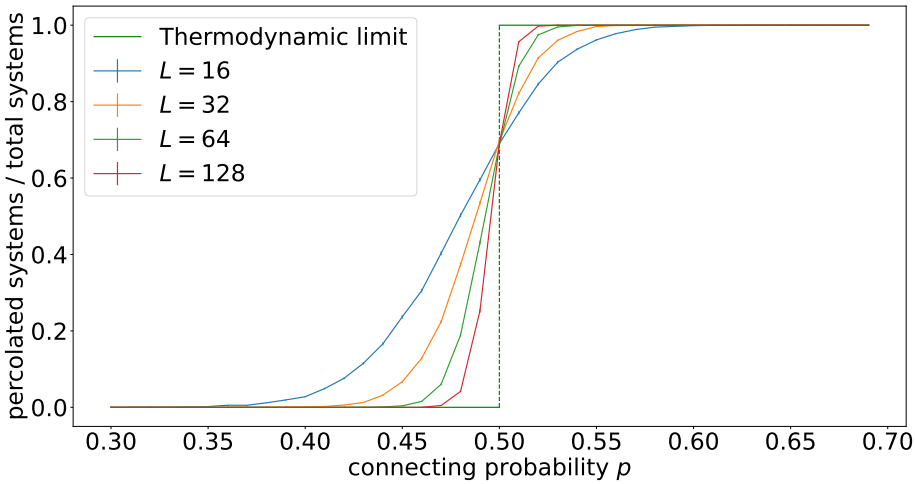


Figure 4: Percolation probability plotted against the connecting probability p for different system sizes using the topological rule

pling them against bigger and bigger systems. If we estimate the critical percolation probability $p_c(L)$ for each system by the probability, at which the percolation probability p_p(relative amount of percolated systems) is equal to 0.5 or 0.15, we can estimate the connecting probability $p_c(\infty)$ by finite size scaling. It may be not intuitive at first why we may choose any value of $p_p \in (0, 1)$. But since we obtain a step function in the thermodynamic limit, each scaling series of $p_c(L)$ will scale to our desired limit. The scaling factor for this system is known, therefore we can fit to the following:

$$p_c(L) = a \cdot L^{-\left(\frac{1}{\nu}\right)} + b \tag{1}$$

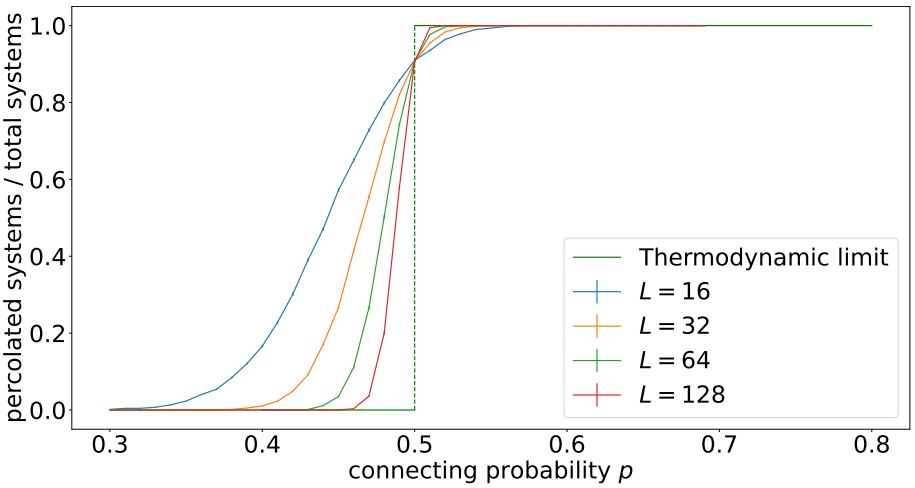


Figure 5: Percolation probability plotted against the connecting probability p for different system sizes using the open boundary conditions and connecting 2 opposite sites

with $\nu = 4/3$. These scaling properties are typical phenomena for systems around the critical percolation threshold. The parameter b estimates our $p_c(\infty)$ and we expect it to be close to $p_c = 1/2$. In Fig. 6, we have plotted the series for the topological rule of our estimator $p_c(L)$. In Fig. 3 or 4, we see that the statistical errors, visualized by error bars, are very small. Therefore the dominating error on our estimator is determined by the sample increment of the connecting probability p. This systematic error was used for the fit.

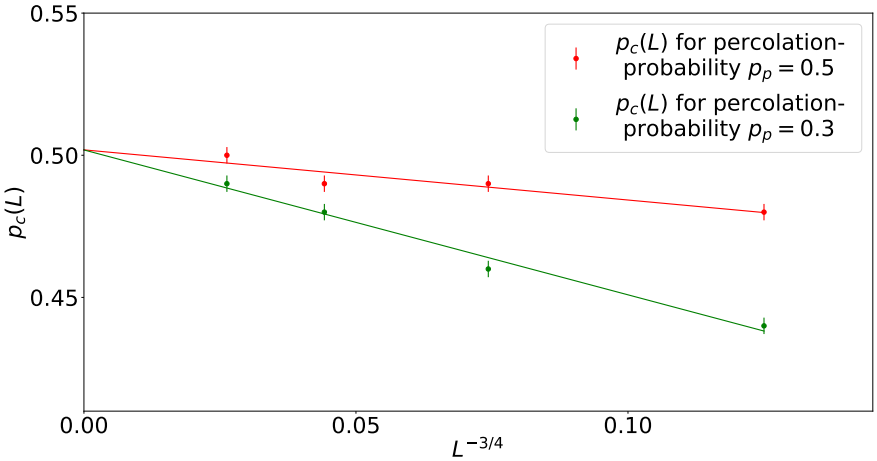


Figure 6: The fit of equation 12 for 2 different criteria

The results can be seen in Fig. 6 and in Table 1. As we expected, b is a single standard deviation away from the theoretical value.

p_p	0.5	0.15
a	-0.17 $\pm$ 0.04	-0.50 $\pm$ 0.04
b	0.501 $\pm$ 0.003	0.501 $\pm$ 0.003
χ^2/dof	1.54	1.32

Table 1

Lastly we showed that finite size scaling can be a very useful concept to determine quantities such as the critical probability. The algorithms used to simulate the systems will be presented in the next chapter.

3

ALGORITHMS TO IDENTIFY CLUSTERS AND PERCOLATION

3.1 MODIFIED HOSHEN–KOPELMAN ALGORITHM FOR IDENTIFYING CLUSTERS AND DETECTING PERCOLATION

We want to describe the algorithms used to identify clusters on a lattice which has connected sites. Since we are later going to use this algorithm on a deformed triangular lattice (visualised in Fig. 7), we will describe all algorithms for this more general case. The additional possible connection direction will be referred to as the d-direction. We may always ignore the additional possible bonds and recreate the square lattice bond percolation problem.

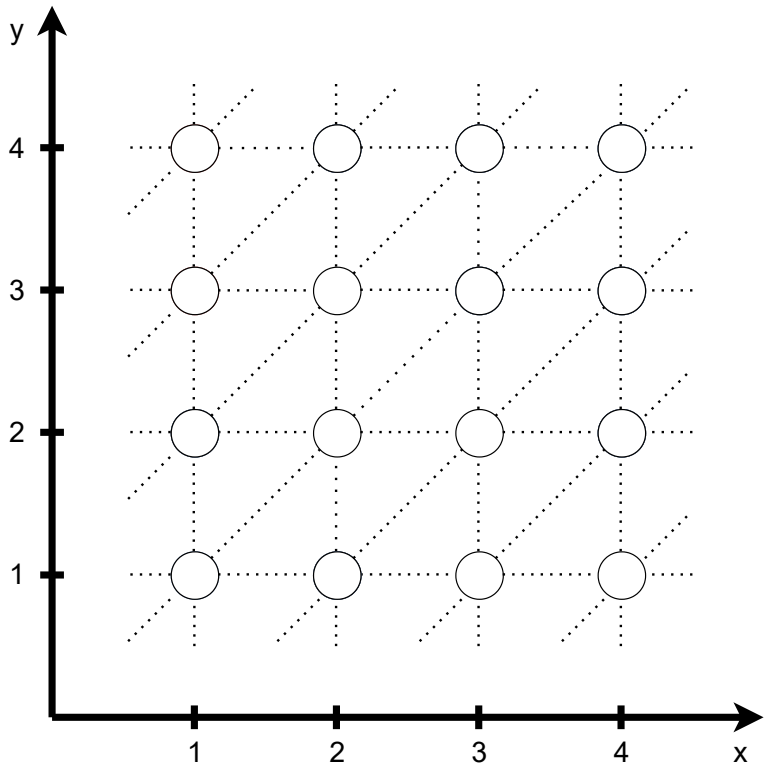


Figure 7: Sample grid with all possible bonds

3.2 THE HOSHEN-KOPELMAN ALGORITHM

The goal of this algorithm is to label each connected site together. For site percolation we can use the Hoshen-Kopelman [2] algorithm. To apply this algorithm for bond percolation, we needed to do some minor modifications. The algorithm works as follows: We check every bond. This bond can either be connecting or non-connecting. If a bond does not connect, we will not have to do anything. If a bond is connecting, there will be three cases which have to be distinguished:

1. Both sites have not been labeled yet: We simply assign a new label to each of the sites.
2. One of the sites has been labeled: We merge the unlabeled one to the labeled one.
3. Both sites have been labeled: We merge both clusters.

3.2.1 *Example on a given bond configuration*

We will present the algorithm on an example. Assume the following bond configuration:

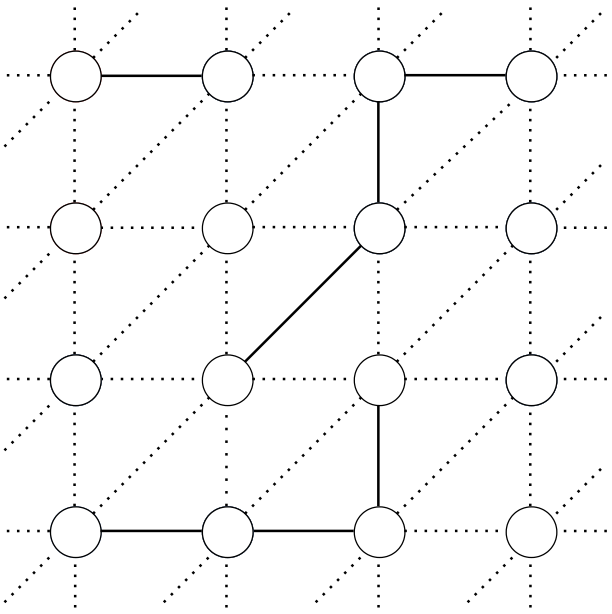


Figure 8: Sample bond configuration in which we will identify all clusters

First, we traverse all bonds in the x-direction and use the rules we stated before. We then obtain the following configuration:

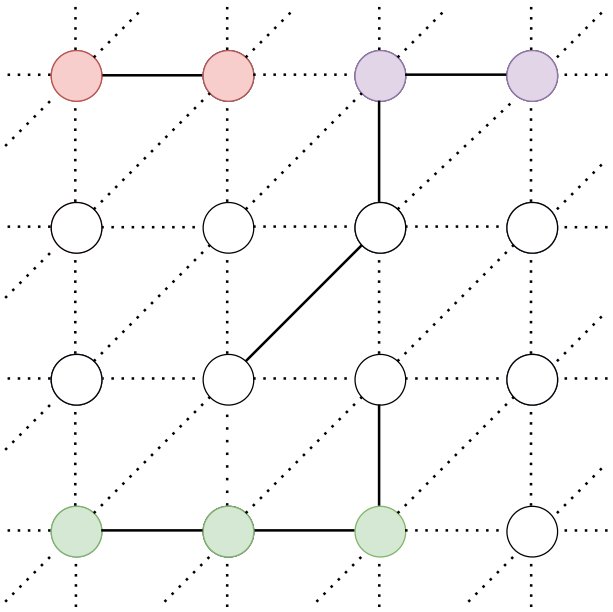


Figure 9: All temporary clusters after traversing all bonds in the x-direction

Then we traverse all bonds in the y-direction:

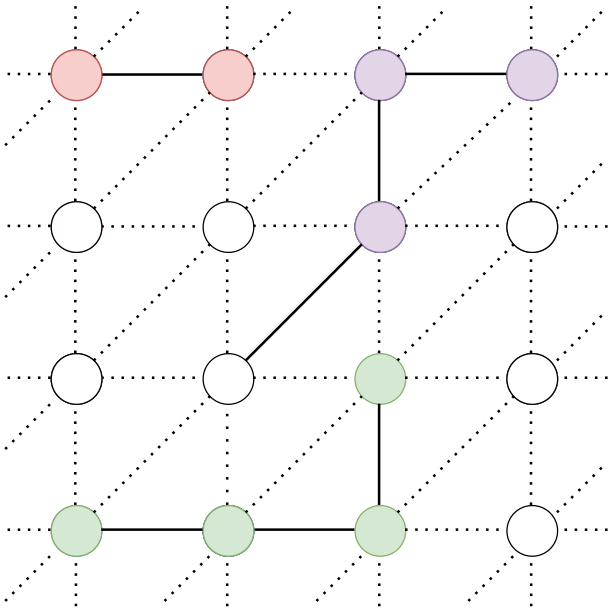


Figure 10: All temporary clusters after traversing all bonds in the y-direction

Finally in the d-direction:

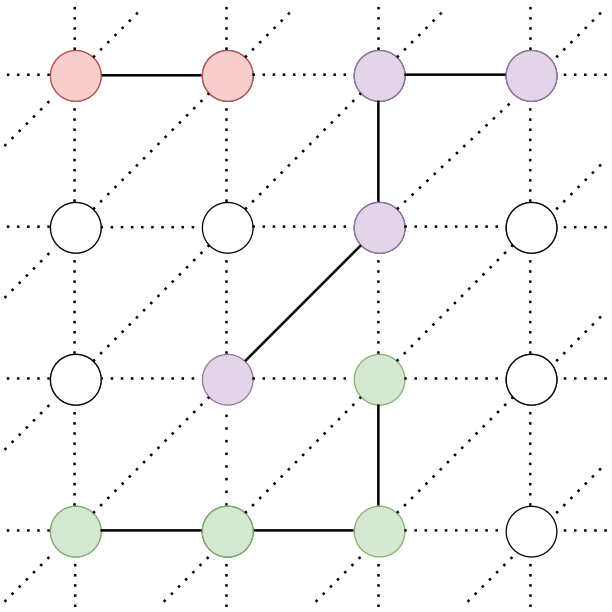


Figure 11: Final state of the Hoshen-Kopelman algorithm

3.2.2 *Tracking percolation via the extension rule*

After traversing through every possible bond, we obtain the labeled lattice. The method to check for percolation via the extension rule is to use the gathered information of the labeled lattice. To search for percolation we just check if any cluster has been labeled throughout the x-direction or the y-direction.

3.2.3 *Modified Newman-Ziff algorithm for identifying clusters and detecting percolation*

The Newman-Ziff algorithm [2] identifies clusters and percolation via the topological rule. To explain this algorithm, we define the following:

- Every cluster has only **one unique** root site. In the beginning, every site is a cluster with size 1 and therefore its own root site.
- Every site of a cluster, which is not the root site, is assigned a displacement vector to another site of the cluster. This site could be the root site or a site which also has a displacement vector.
- If we were to traverse all displacements of a chain of sites of a cluster, we would always end up at the root site.

- We identify two sites to be in the same cluster if each root site is equal.
- Additionally, we track the size of each cluster.

An example of the state of the algorithm is shown in Fig. 12

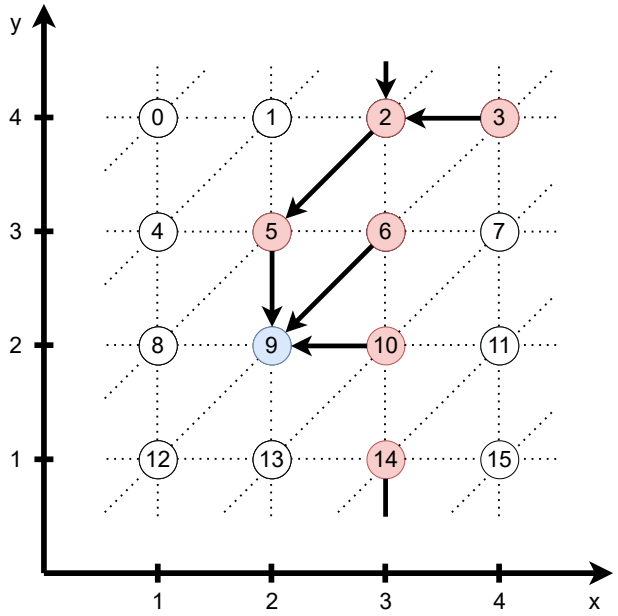


Figure 12: Exemplary state of the Newman-Ziff algorithm.

We observe site 9 as the root site because it does not contain a displacement vector. Sites 5, 6 and 10 point directly to the root site. To obtain the root site of sites 2, 3 and 14 we have to traverse the chain of displacement vectors. The cluster is not percolating via the topological rule. If we add a bond to site 2 and 6, the cluster would obviously still not be percolating. The algorithm checks that by calculating the total displacement to the root site. The total displacement of site 6 is $\Delta_6 = (-1,-1)$. The displacement of site 2 has to be calculated, by adding all displacements in the chain up to the root. In this case we obtain: $\Delta_2 = (-1,-1) + (0,-1) = (-1,-2)$. We now compare the absolute difference in the displacement: $|\Delta_2 - \Delta_6| = (0,1)$. As we observe, this difference is equal to one lattice spacing, where one lattice spacing is defined as $(1,0)$, $(0,1)$ and $(1,1)$. If we add a bond, which would lead to percolation, the difference in displacement would not be one lattice spacing. For example: Assume we add a bond between site 10 and 14. For the displacements we obtain:

$$\Delta_{10} = (-1,0) \quad \Delta_{14} = (-1,-3) \Rightarrow |\Delta_{45} - \Delta_{01}| = (0,3) \qquad (2)$$

The approach to merge clusters is to change the root site of one cluster to the root site of another cluster. Let us consider the configuration of Fig. 13.

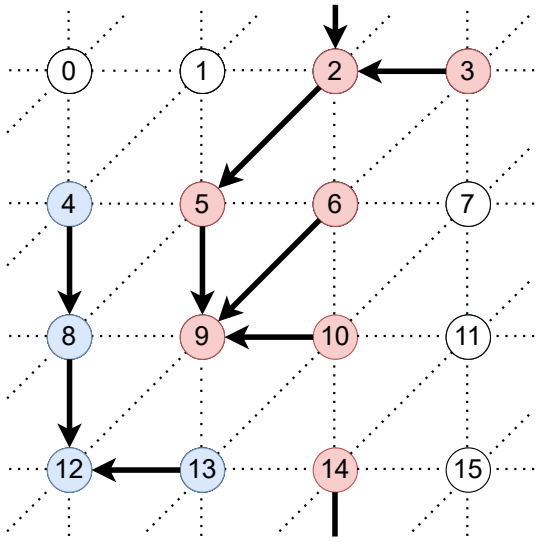


Figure 13: Exemplary configuration, in which we want to place a bond between sites 4 and 5.

If we place a bond between sites 4 and 5, we would have to merge the clusters. We will always change the root of the cluster with less sites. In this case we will change the root of the blue cluster to the root of the red one. First we obtain the root sites of site 4 and 5, which are sites 12 and 9 respectively. Then we just add a displacement vector from site 12 to site 9. The obtained configuration is shown in Fig. 14.

The algorithm for identifying clusters and percolation via the topological rule can be summarize as follows:

1. Every cluster has only one unique root site. In the beginning every site is a cluster with size 1 and therefore its own root site

2. We traverse every bond and add them with probability p. There are two possibilities when a bond is placed:

   a) Both elements are in different clusters meaning they have different roots: We change the root of the cluster with fewer elements to the root of the other cluster.

   b) Both elements are in the same cluster meaning they have the same root: We check if the absolute difference in displacement is not one lattice spacing. If so, we detect percolation.

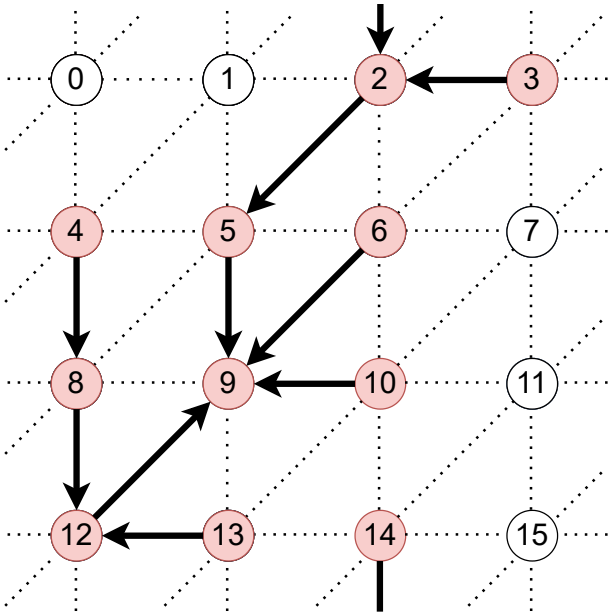
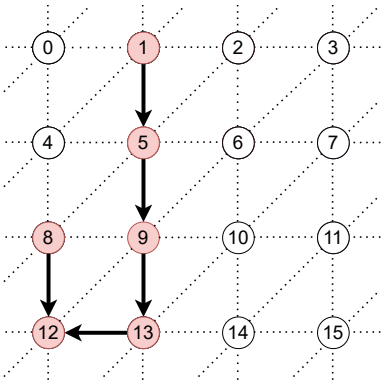


Figure 14: State of the merged configuration after adding a bond.

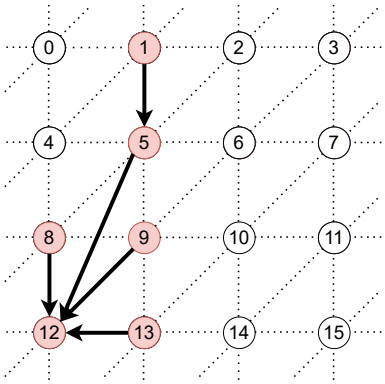
It has to be noted that the original Newman-Ziff algorithm was used for square-lattice bond-percolation. We generalised it for the triangular lattice. To efficiently implement this algorithm we need one main function. The main root-find-function can be described as follows:

1. A function which recursively calls all sites which are chained together by displacement vectors. When the root is found the recursion ends and every displacement vector is updated to point to the root.

An example on how this function works is shown in Fig. 15. After calling this function on site 5 we then call this function on site 9 and after that on site 13. Now we end up at the root site 12 and update the displacement on each site.



(a) State of the Newman-Ziff algorithm, in which we want to find the root site of site 5.



(b) State of the Newman-Ziff algorithm, after applying the root-find-function on site 5.

Figure 15: Example clusters

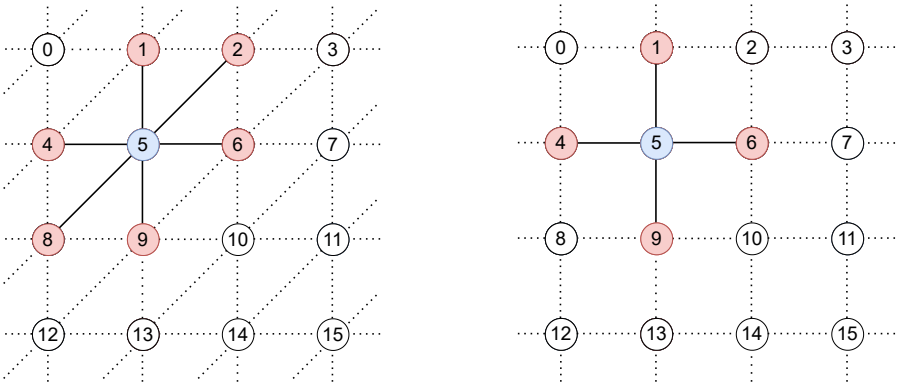
4

THE GENERAL Q-STATE POTTS MODEL AND MARKOV CHAINS

The Potts model is a simple, yet still one of the most important models of classical statistical physics. At each of the N sites of a d−dimensional lattice, we consider q states $s_i \in \{s_1, s_2, ..., s_q\}$. A set configuration of states on a 2-dimensional lattice is then defined as:

$$\sigma = (s_1, s_2, ..., s_N) \tag{3}$$

We will only discuss nearest-neighbor interactions which are visualized in Fig. 16. The Hamiltonian is given by:



(a) Nearest neighbors in the deformed triangular lattice.

(b) Nearest neighbors in the square lattice.

Figure 16: Nearest neighbors of site 6 in the triangular lattice and the square lattice.

$$H(\sigma) = \sum_{<i,j>} J_{i,j}\delta_{s_i,s_j} \quad s_i \in \{s_1, s_2, ..., s_q\} \tag{4}$$

In this defined system the Hamiltonian is equal to the energy of the system. For $J_{i,j}$ we will only distinguish between three different possibilities: J_x, J_y and J_d where each of these indices stand for the uniform coupling in each lattice direction. We define systems as isotropic, if $J_x = J_y = J_d$ or if $J_x = J_y$, $J_d = 0$. Otherwise the system is called anisotropic. We consider the system coupled with a heat-bath at temperature T. Therefore the

statistical weight of a configuration σ is given by the Boltzmann weight:

$$P(\sigma) \propto e^{-\beta E(\sigma)} \tag{5}$$

with $\beta = 1/(k_B T)$ and $k_B = 1$. Therefore the partition function is given by:

$$Z = \sum_{\{\sigma\}} e^{-\beta E(\sigma)} \tag{6}$$

As we can see, the energy $E(\sigma)$ of the system depends on the values of $J_{i,i} \in \{x, y, d\}$. We will focus on the ferromagnetic case $J_i < 0$. In this case the energy is minimized if neighboring spins point in the same direction. If we are given a temperature T, the expectation value of an observable $< A >$ can be calculated by:

$$\langle A \rangle = \frac{1}{Z} \sum_{\sigma} A(\sigma) e^{-\beta E(\sigma)} \tag{7}$$

Now we will analyze the high and low temperature limit of the system.

- For $T \to \infty$ we can approximate $P(\sigma) \to \frac{1}{Z}$ and we can conclude that no particular configuration is preferred.
- For $T \to 0$ we already concluded that ordered configurations are preferred by the system.

At a certain temperature T, we observe a transition at which the system changes from an ordered to an unordered state. This temperature is called critical temperature T_c. For the $2-d$ Potts model we observe first and second order phase transitions. For $q > 4$ the transition is of first order and for $q \geqslant 4$ it is of second order [4]. First and second order phase transitions are defined as the behavior of an order parameter around the critical temperature. A continuous transition is of second order, and a discontinuous one is of first order. This temperature T_c is solved analytically for any set of $J_x, J_y and J_d$ for $q = 2$ and $q \geqslant 4$. For $q = 3$ there is an assumption about the theoretical solution. This assumption has not yet been proved. The solution [5] for $q = 2$ is the following: Let $S(J) = \sinh(2J\beta_c)$ then

$$S(J_x) \cdot S(J_y) + S(J_d) \cdot (S(J_x) + S(J_y)) = 1 \tag{8}$$

For $q \geqslant 4$ we have [4]: Let $x_i = (e^{J_i \cdot \beta} - 1)/\sqrt{q}$ then T_c can be obtained by solving:

$$\sqrt{q} \cdot x_x x_y x_d + x_x x_y + x_x x_d + x_y x_d = 1 \tag{9}$$

Equation 9 is also the assumed solution for the 3-state Potts model. These equations can be solved numerically and will be the foundation to determine the effectiveness of the algorithms we will apply.

4.1 THE MARKOV PROCESS

As we can see in equation 7, the expectation value of an observable is given by using a sum over configuration space $\Omega = \{\sigma\}$. If we generated every possible configuration for a given system, this would be a computation demanding task. For example: If we assume a system of size 100x100, we would end up with $2^{10.000}$ possible configurations. Another ansatz would be to generate a random set of N configurations. This would also end up being quite inefficient. That is the case since we would mainly generate unordered systems at low temperatures and their contribution would be negligible. Therefore it would be useful if we could generate configurations σ which follow the distribution of $P(\sigma)$. This can be achieved by a Markov chain. We generate a sequence of configurations using a stochastic process. Starting from an initial configuration $\sigma_i \in \Omega$ we generate:

$$\sigma_0 \to \sigma_1 \to \sigma_2 \to ... \to \sigma_N \tag{10}$$

If we start at a random configuration σ, we have to thermalise the system first. In the initial phase of these chains, we first have to obtain an equilibrium. After that we can start to use the Markov chain to measure certain quantities.

4.2 THE SWENDSEN-WANG ALGORITHM

The Swendsen-Wang Algorithm [6] uses cluster methods to achieve efficient Monte Carlo dynamics. The Swendsen-Wang update algorithm can be descibed as follows:

1. We begin with an arbitrary configuration σ.
2. If the state of two neighboring sites are equal, we cast a bond with probability $p = 1 - e^{-2\beta|J_r|} \quad r \in \{x, y, d\}$ or otherwise we will never cast a bond.
3. We identify all clusters and collectively flip the states of a cluster to a random state $s \in \{s_1, s_2, ...s_q\}$ with uniform probability $p = 1/q$.
4. We repeat step 2 with the new obtained configuration.

4.3 SWENDSEN-WANG ALGORITHM ON THE 2-STATE POTTS MODEL

The 2-state Potts model can be used to describe classical spins. Therefore the states are defined as $s_i \in \{\pm 1\}$. We were given data of a Metropolis algorithm run, with a system size of $L = 8$. Now we will compare this data to a Swendsen-Wang update run. We have simulated the system for different temperatures and we have measured the average Energy per spin $E(T)$ at each update step. The result can be seen in Fig. 17. As we can see, the data we generated and the data of the Motropolis algorithm match. We are not able to determine any differences. If we subtract the data of each other, the resulting samples scatter around 0 and the relative deviation is less than 1%.

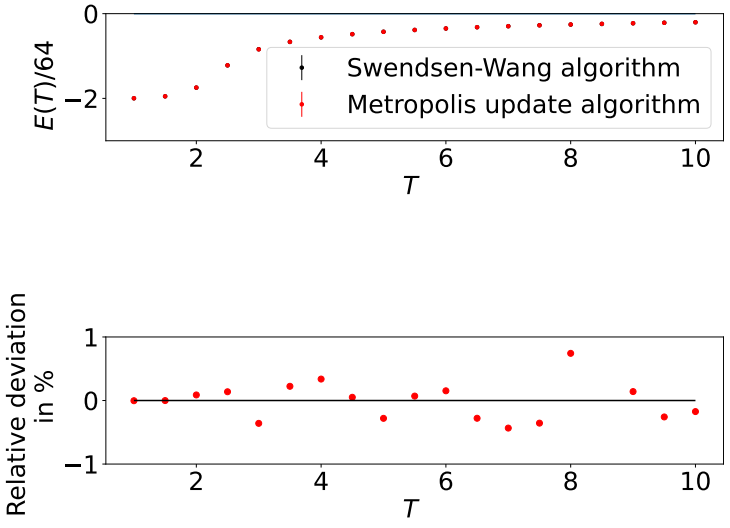
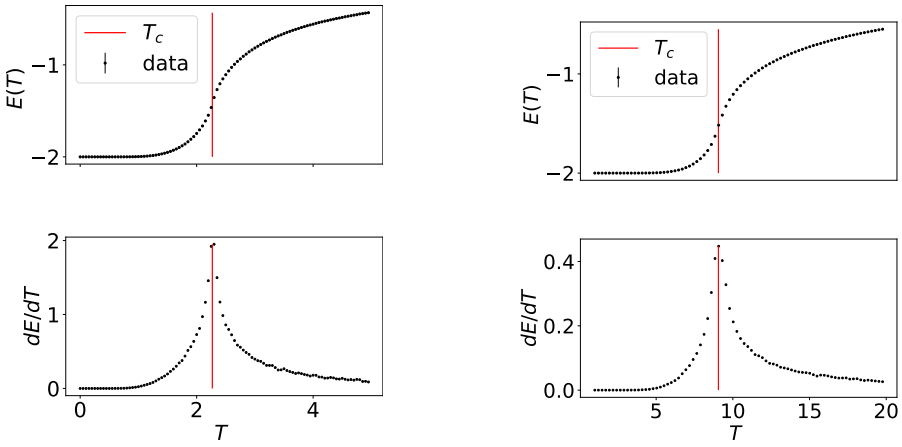


Figure 17: Comparison of data for the Energy for simulating a system of size $L = 8$ for 10000 thermalisation steps and 100.000 measuring samples.

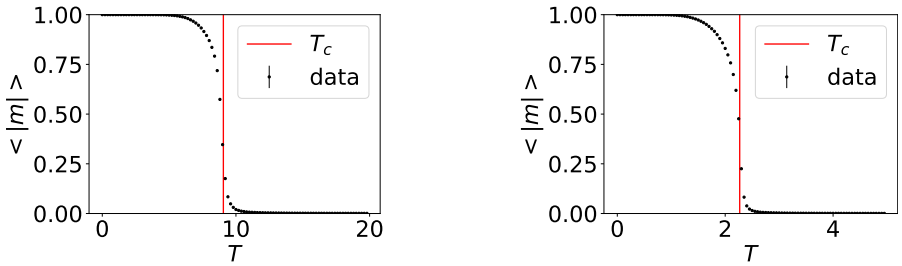
Since we can sample $E(T)$ for different values of T, we can numerically estimate the specific heat $C_v = dE/dT$ and the maximum of this curve should be around the critical Temperature T_c. Therefore we can get a rough estimation of T_c. The data we generated is shown in Fig. 18.



(a) $J_x = J_y = 1$ (b) $J_x = 10$ and $J_y = 1$

Figure 18: Sample Swendsen-Wang data for isotropic and anisotropic systems. We used a system size of $L = 64$, 1000 thermalisation steps and 5000 measuring samples.

Another observable which has a relation to the critical temperature T_c is the average of the absolute magnetisation $< |m| >$ where $m = \sum_i \sigma_i$. The results are plotted in Fig. 19



(a) $J_x = J_y = 1$ (b) $J_x = 10$ and $J_y = 1$

Figure 19: Sample Swendsen-Wang data for isotropic and anisotropic systems. We used a system size of L=64, 1000 thermalisation steps and 5000 measuring samples.

The absolute magnetisation for the 2-state Potts model can be defined as an order parameter. It is zero above T_c and nonzero under T_c. If the majority of the spins point in the same direction or not, we would expect a connection to the percolation property at different temperatures.

As we could show, Markov Chains are an effective tool to obtain quantities for given temperatures. In the following chapter we want to discuss how we can combine these concepts to estimate the critical temperature.

5

THE PCC ALGORITHM

The Probability-Changing-Cluster [1] algorithm can be summarized as follows:

1. We begin with an arbitrary configuration σ.
2. Construct clusters and flip them according to step 2 and 3 of the Swendsen-Wang algorithm.
3. Increase or decrease p depending if the system is percolating or not.
4. Go back to step 2.

The obtained histogram of the obtained values for p should be Gaussian distributed. This algorithm assumes an isotropic system. Because only then we can collectively change p by Δp and still keep an unique temperature.

We would like to apply this algorithm for anisotropic systems. So for a fixed set of J_x, J_y and J_d we would end up with different probabilities $p_x(T)$, $p_y(T)$ and $p_d(T)$ which can not be transformed by a fixed Δp. The approach we made is, that we define a Δp_x. We increase or decrease p_x by Δp_x and then calculate the resulting Temperature T. Then we can just calculate $p_y(T)$ and $p_d(T)$ with the given temperature. Therefore we define the modified PCC algorithm as:

1. We begin with an arbitrary spin configuration σ and a fixed set of J_x, J_y and J_d.
2. Construct clusters and flip them according to step 2 and 3 of the Swendsen-Wang algorithm.
3. Increase or decrease p_x by Δp_x depending if the system is percolating or not.
4. Calculate the corresponding temperature T and get p_y and p_d.
5. Go back to step 2.

For the algorithm to work properly we have to decide how many Monte Carlo steps we gather. Additionally we have to choose Δp_x. Since we may not know the critical temperature beforehand, we start with an initial guess and a rough $\Delta p_x = 0.01$. We do 10.000 initial steps. After that we adapt $\Delta p_x = 1/(20 \cdot L^2)$ and take 100.000 Monte Carlo samples. Lastly we calculate the mean of the temperature distribution. We will repeat that process ten times to estimate the error.

5.1 ERROR ANALYSIS ON THE PCC ALGORITHM

The resulting distribution of T seems to approach a Gaussian function. This was to be expected, since the resulting histogram of p_x approaches a Gaussian curve and when looking at

$$T = \frac{-2J}{\ln(1 - p_x)} \approx T_0 + \alpha \cdot p_x \tag{11}$$

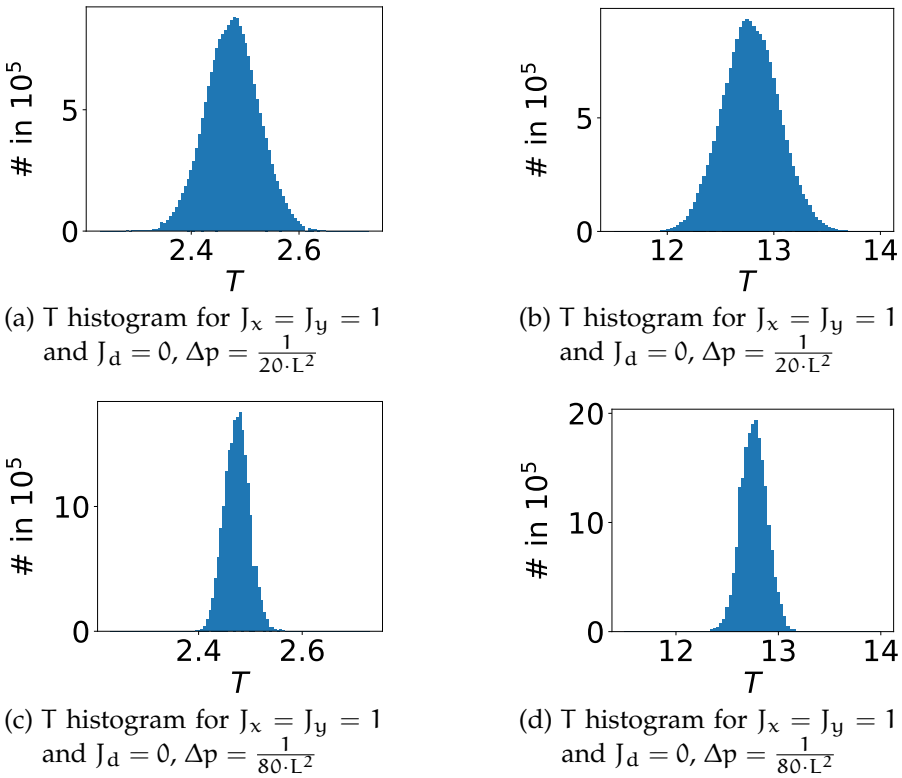
we will find the same distributions for small changes in p_x. This result can be seen for an isotropic and anisotropic example in Fig. 20. The mean of these samples will be at the critical Temperature $T_c(L)$. Additionally we have included the behavior, if we further decrease Δp_x. We can see that the distribution is sharper around the critical value.

We could verify the exact same resolution of $p_c(64)$ as Tomita and Okabe [1]. They determined $p_c(64) = 0.581\,954\,6 \pm 0.000\,013$ by gathering 100.000 MC samples and repeating that process 10 times. The error obtained using this process contains the systematic and statistical error. We want to analyse how the total error behaves if we increase the Monte Carlo samples or if we adjust Δp_x.

$T_c(L)$		Measuring steps	
		100.000	200.000
Stepsize Δp	$1/(20 \cdot 64^2)$	$2.293\,170 \pm 0.000\,081$	$2.293272 \pm 0.000\,032$
	$1/(40 \cdot 64^2)$	$2.293\,358 \pm 0.000\,147$	$2.293\,173 \pm 0.000\,047$

Table 2: $T_c(L)$ for $L = 64$. We repeated 10 measurements with 100.000 and 200.000 samples and different step sizes Δp respectively. We then formed the mean and standard deviation.

In Table 2 we have measured $p_c(64)$ for different parameters. By simply increasing the amount of samples, we can conclude



(a) T histogram for $J_x = J_y = 1$ and $J_d = 0$, $\Delta p = \frac{1}{20 \cdot L^2}$

(b) T histogram for $J_x = J_y = 1$ and $J_d = 0$, $\Delta p = \frac{1}{20 \cdot L^2}$

(c) T histogram for $J_x = J_y = 1$ and $J_d = 0$, $\Delta p = \frac{1}{80 \cdot L^2}$

(d) T histogram for $J_x = J_y = 1$ and $J_d = 0$, $\Delta p = \frac{1}{80 \cdot L^2}$

Figure 20: Temperature histogram of an isotropic and isotropic system for different values of Δp. We have taken 200.000 monte carlo samples for a system size $L = 8$.

that the precision will be increased. After observing that the Gaussian curve is more sharply peaked in Fig. 20 we would assume that the error on p would be smaller for the same amount of samples. But this does not seem to be the case. The reason why this happens, is that in the "prescan" we have chosen to take samples of $\Delta p_x = 0.001$ for 10.000 steps. This "prescan" gives us an imprecise initial guess. Therefore the values of T have to converge to the region of T_c. This behavior is shown in Fig. 21.

The resulting "tail" of T values results in an additional error. If we decrease the stepsize Δp_x the amount of steps needed to be around the peak, will increases also. This may explain why decreasing the stepsize leads to a bigger error.

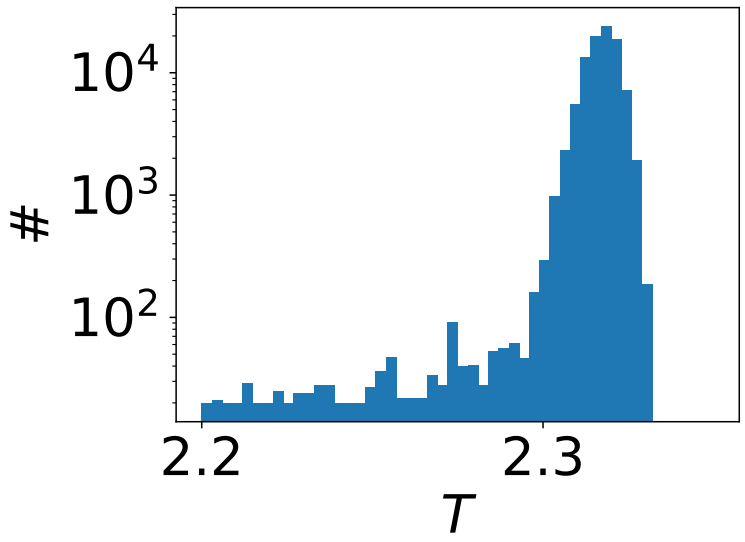


Figure 21: Histrogram of the 100.000 Temperature samples.

6

DATA ANALYSIS OF THE PCC ALGORITHM FOR Q=2

6.1 ISOTROPIC CASE FOR $J_x = J_y = 1$ AND $J_d = 0$

For the isotropic case where $J_x = J_y = 1$ and $J_d = 0$ we can fit $T_c(L)$ to

$$T_c(L) = a \cdot L^{-\left(\frac{1}{\nu}\right)} + b \tag{12}$$

as we did in chapter 2 with $p_c(L)$. The scaling factor for $q = 2$ is $\nu = 1$. By sampling 100.000 Monte Carlo samples 10 times for various system sizes.

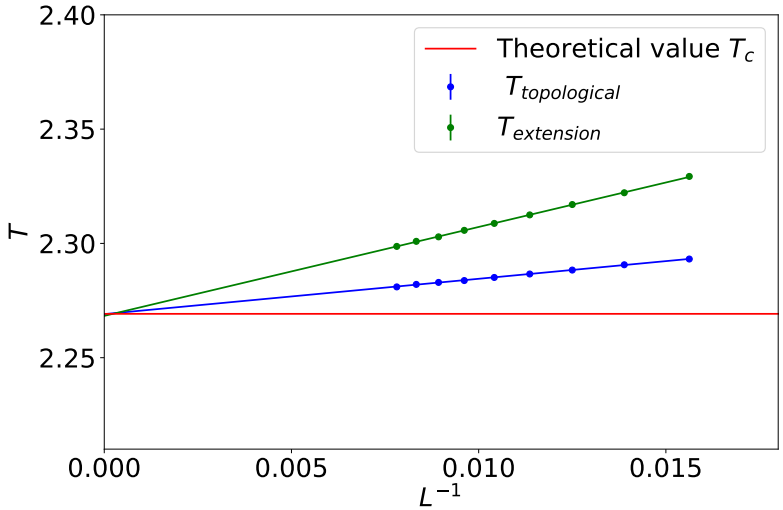
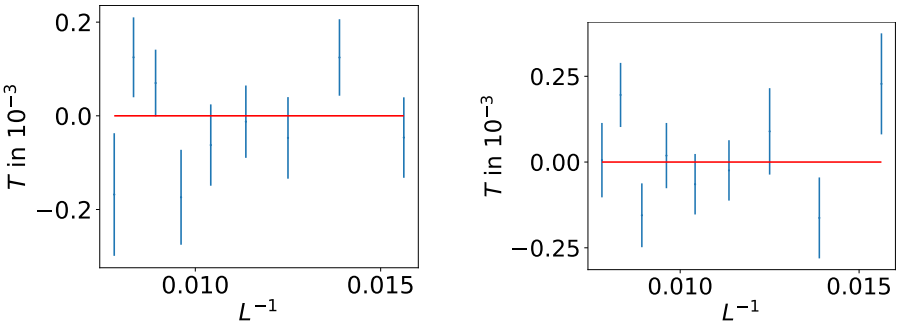


Figure 22: T_c for the topological rule and the extension rule.

	T_c	T_c by Tomita and Okabe	χ^2/dof	$\lvert T_c - T_{theo}\rvert$
Topological rule	$2.269\,06 \pm 0.000\,17$	$2.269\,2 \pm 0.000\,2$	1.59	0.69σ
Extension rule	$2.268\,29 \pm 0.000\,23$	$2.268\,8 \pm 0.000\,2$	1.80	3.88σ

Table 3: Data for the isotropic case

As we can see, the correction terms of the extension rules are bigger. That means for this system the topological rules is a better estimator for the system in general. We can also conclude this from Fig. 22 Table 3. Here we can see that the fit of the topological rule is more precise. In Fig. 23 we can see that the fit to the data does not have a dominating systematic deviation in the residual plots.



(a) Residual plot of the data generated with the topological rule.

(b) Residual plot of the data generated with the extension rule.

Figure 23: $T_c(L) - T_{fit}(L)$

From now on we will mainly focus on the topological rule. Tomita and Okabe used system sizes $L = (64, 128, 256, 512)$. We used rather "small" systems $L = (64, 72, 80, 88, 96, 104, 112, 120, 128)$, but we obtained the same precision for the critical temperature. The main difference in the approaches is that we sampled more smaller systems. Since simulating bigger system sizes demand a lot more computational power, it seems that sampling "smaller" systems is sufficient.

6.2 THE GENERAL ANISOTROPIC CASE

For this case we can analyse another possible variation of the PCC algorithm.

1. We begin with an arbitrary spin configuration σ and 2 fixed probabilities $p_j, p_i \quad i, j \in \{x, y, d\} \quad i \neq j$
2. Construct clusters and flip them, according to step 2 and 3 of the Swendsen-Wang algorithm.
3. Increase or decrease p_x by Δp_x depending if the system is percolating or not.
4. Go back to step 2.

So we just fixate, for example, p_x and p_d to a certain value. Then we can obtain a resulting distribution of p_y. Therefore we gain the following curve while sampling for different system sizes as seen in Fig. 24 and 25.

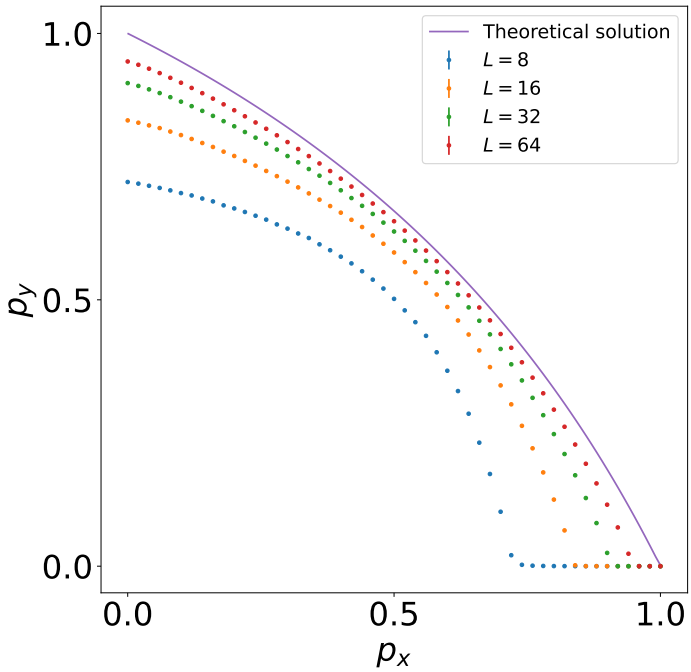


Figure 24: Scan of p_y for fixed p_x and $p_d = 0$.

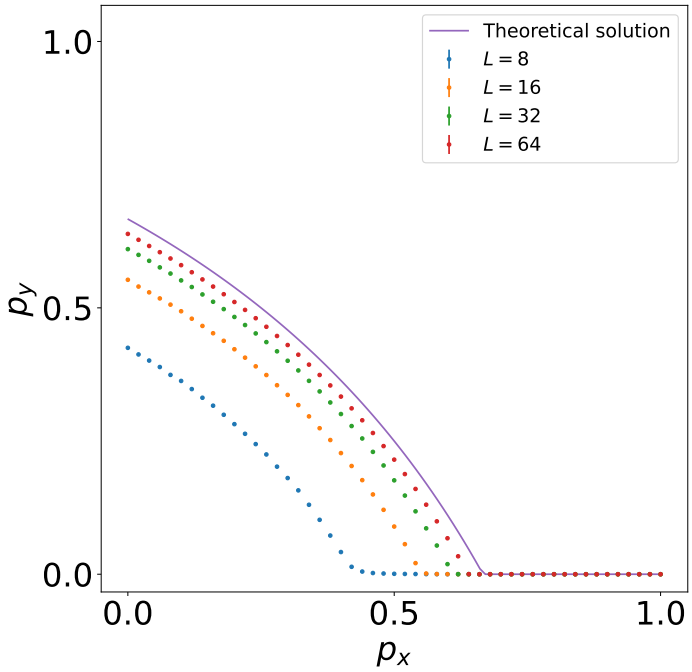


Figure 25: Scan of p_y for fixed p_x and $p_d = 0.5$.

The exact solution to this curve can be obtained by using Formula 8. We have to rely on a numerical solution for $J_d \neq 0$. For $p_d = 0$ we can solve the equation:

$$\sinh(2J_x\beta)\sinh(2J_y\beta) = 1 \tag{13}$$

$$\rightarrow p_x = 1 - \exp\left(\frac{-\operatorname{arsinh}}{\sinh(-\ln(1-p_y))}\right) \tag{14}$$

The context of T_c in this representation of the data is given by looking at the graph of $(x, y) = (p_x(T), p_y(T))$. The intersection of this graph for a given tuple of J_x, J_y gives us $T_c(L)$. This behavior is exemplary presented in Fig. 26.

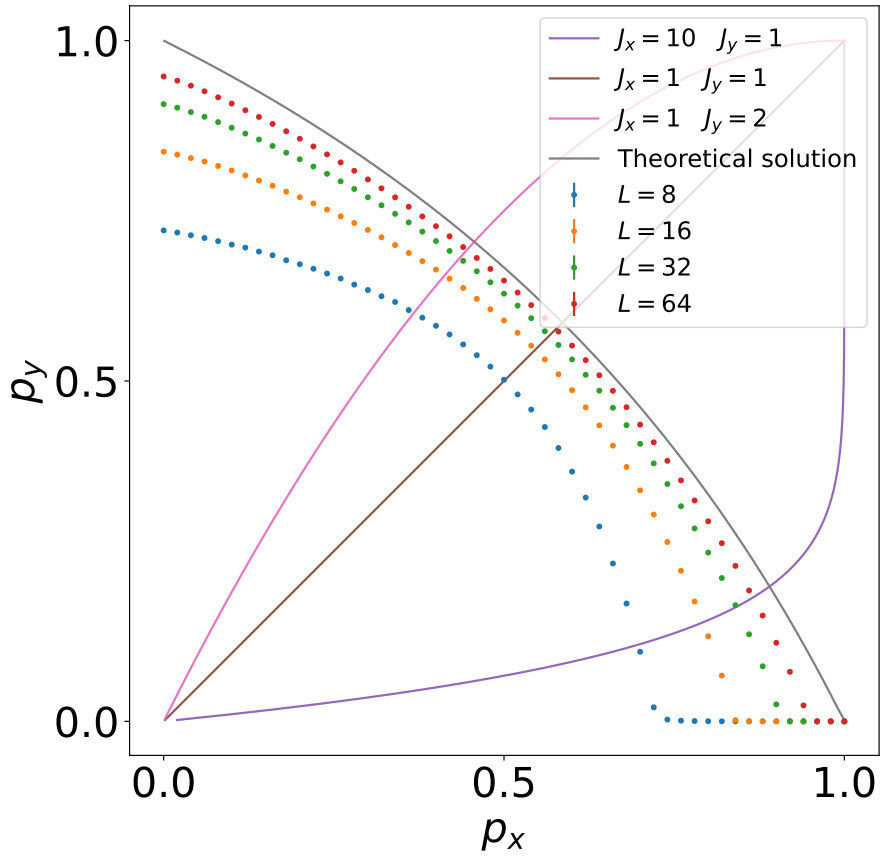


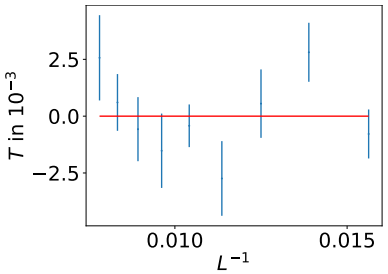
Figure 26: Scan of p_y. The coupling (J_x, J_y) of the graphs $(p_x(T), p_y(T))$ is $(10, 1)$,$(1, 1)$ and $(1, 2)$. Sample for temperatures $T \in (1, 1000)$.

J_x	J_y	J_d	T_c simulation	T_c theoretical	χ^2/dof	$T_c - T_{theo}$ in σ
1	5	50	48.043 ± 0.003	48.047	1.64	1.5σ
1	1	1	3.6408 ± 0.0002	3.6409	3.05	0.53σ
1	10	0	9.0585 ± 0.0004	9.0588	1.05	0.63σ
1	17	13	35.442 ± 0.002	35.443	2.53	0.35σ

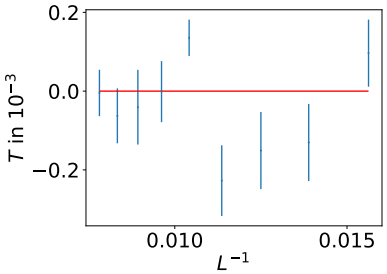
Table 4: Critical temperatures estimated by simulating the system sizes $L \in \{64, 72, 80, 88, 96, 104, 112, 120, 128\}$ for 200.000 MC samples 10 times.

Now we will apply the FSS on this anistrisotripic case: We will consider the following set of $J_i, i \in \{x, y, d\}$.

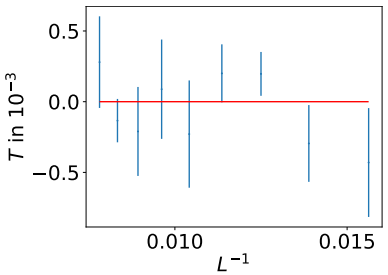
As we can observe in Table 4, the data matches the theoretical estimations. The fit value does not exceed the theoretical value above a few standard deviations. In Fig. 27 we can observe, that there is no dominating systematic error. In general we obtained an accuracy of around 0.01%. Therefore we may conclude that this approach to estimate the critical temperature could work on the 3-state Potts model.



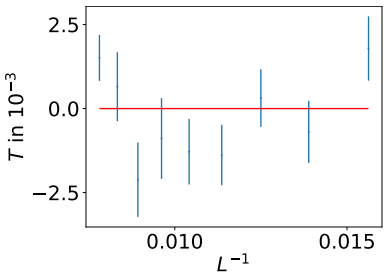
(a) Residue for $J_x = 1$ $J_y = 5$ $J_d = 50$



(b) Residue for $J_x = 1$ $J_y = 1$ $J_d = 1$



(c) Residue for $J_x = 1$ $J_y = 10$ $J_d = 0$



(d) Residue for $J_x = 1$ $J_y = 17$ $J_d = 13$

Figure 27: Residuals of the fit. We used the data we generated for the 2-state Potts model for various coupling constants.

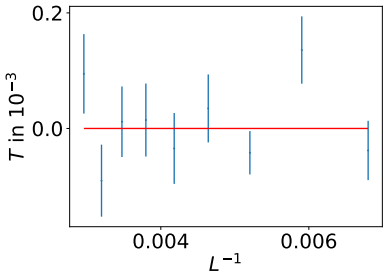
7

DATA ANALYSIS OF THE PCC ALGORITHM FOR q = 3

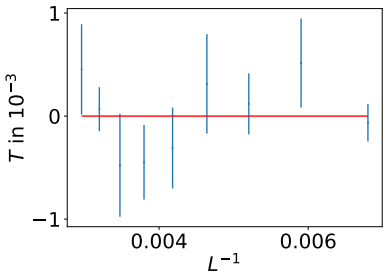
The data we obtained applying the PCC algorithm to the 3-state Potts model is shown in Table 5. The quality of the fit to the data is as good as for the 2-state Potts model. The residuals, which are shown in Fig. 28, do not show a dominiating systematic error. Tomita and Okabe [1] obtained the following result for the 3-state Potts model using the topological rule $T_c = 1.989\,88 \pm 0.000\,12$. The obtained data matches with the theoretical solutions.

J_x	J_y	J_d	T_c simulation	T_c theoretical	χ^2/dof	$\lvert T_c - T_{\mathrm{theo}}\rvert$
1	1	0	1.9901 ± 0.0004	1.9899	1.99	0.41σ
1	1	1	3.1703 ± 0.0005	3.1698	0.92	1.01σ
1	2	3	6.204 ± 0.001	6.205	0.85	0.74σ
1	3	7	10.876 ± 0.002	10.874	1.39	0.74σ
1	10	0	8.117 ± 0.001	8.119	1.05	1.25σ

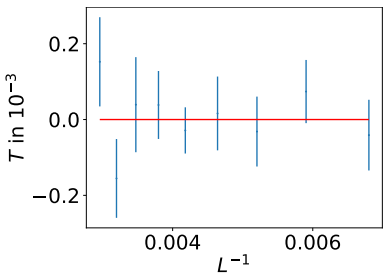
Table 5: PCC algorithm on the 3-state Potts model



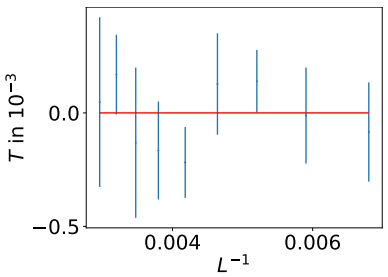
(a) Residue for $J_x = 1$ $J_y = 5$ $J_d = 50$



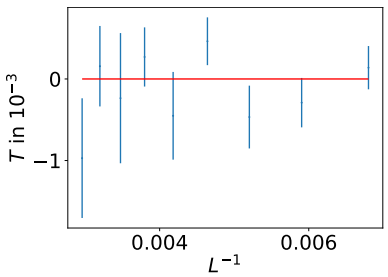
(b) Residue for $J_x = 1$ $J_y = 1$ $J_d = 1$



(c) Residue for $J_x = 1$ $J_y = 10$ $J_d = 0$



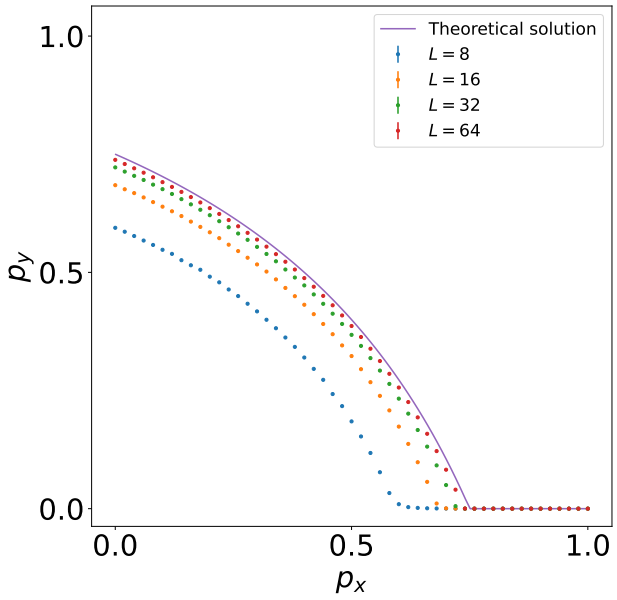
(d) Residue for $J_x = 1$ $J_y = 17$ $J_d = 13$



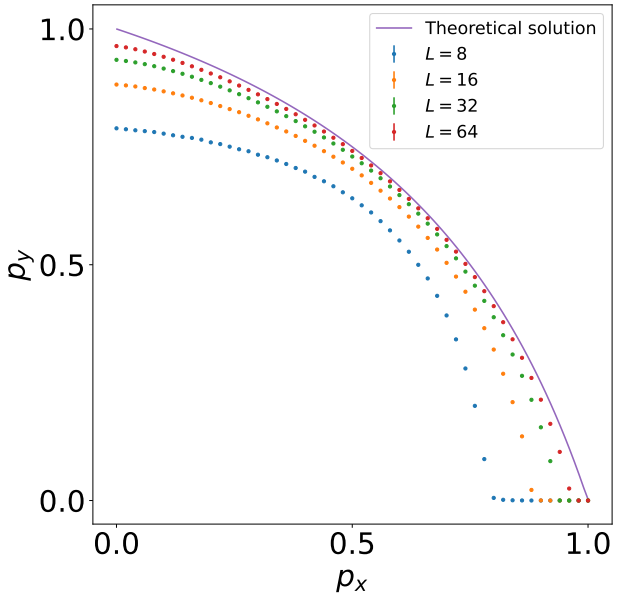
(e) Residue for $J_x = 1$ $J_y = 17$ $J_d = 13$

Figure 28: Residuals of the fit. We used the data we generated for the 3-state Potts model for various coupling constants.

For the 3-state Potts model we can use the variation of the PCC algorithm which we discussed in chapter 6. These scans are visualized in Fig. 29



(a) Scan of p_y for different values of p_x and $p_d = 0.5$



(b) Scan of p_y for different values of p_x and $p_d = 0$

Figure 29: Different scans for p_y using the 3-state Potts model.

We can also observe that the curves approach the theoretical solution.

8

IMPLEMENTATION

The entire code for the data was written in Python. Python is an interpreted language, therefore it is slow compared to C. We used a package called Numba which lets us compile the written Python code to get similar performance to C. It is a JIT (just-in-time) compiler for Python. There are some restrictions, such as that we have to be careful with types of variables. But the advantages are that we could use pure Python code to debug our algorithms and as soon as we want simulate big systems, we compile the code. Numba also lets us parallelise the code so that we can take advantage of the computational power of multiple cores. Simulations were performed with computing resources granted by RWTH Aachen University under project thes1243. The code is uploaded to Github [7].

9

CONCLUSION

Summing up, we can conclude that the PCC algorithm paired with finite size scaling is an effective tool to determine the critical Temperature T_c of anisotropic systems. We have shown that the assumption of the solution of the 3-state Potts model for square and triangular lattices matches with the simulated data. The Potts model is solved for $q = 2$ and $q \geqslant 4$ for the square-lattice, honeycomb and the triangular-lattice. Further, it could be analyzed if the algorithms are useful for unsolved lattices.

BIBLIOGRAPHY

[1] Yusuke Tomita and Yutaka Okabe. "Probability-Changing Cluster Algorithm for Potts Models." In: *Physical Review Letters* 86.4 (2001), pp. 572–575. DOI: 10.1103/physrevlett.86.572. URL: https://doi.org/10.1103%2Fphysrevlett.86.572.

[2] J. Hoshen and R. Kopelman. "Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm." In: *Phys. Rev. B* 14 (8 1976), pp. 3438–3445. DOI: 10.1103/PhysRevB.14.3438. URL: https://link.aps.org/doi/10.1103/PhysRevB.14.3438.

[3] M. E. J. Newman and R. M. Ziff. "Fast Monte Carlo algorithm for site or bond percolation." In: *Physical Review E* 64.1 (2001). DOI: 10.1103/physreve.64.016706. URL: https://doi.org/10.1103%2Fphysreve.64.016706.

[4] F. Y. Wu. "The Potts model." In: *Rev. Mod. Phys.* 54 (1 1982), pp. 235–268. DOI: 10.1103/RevModPhys.54.235. URL: https://link.aps.org/doi/10.1103/RevModPhys.54.235.

[5] R.M.F. Houtappel. "Order-disorder in hexagonal lattices." In: *Physica* 16.5 (1950), pp. 425–455. ISSN: 0031-8914. DOI: https://doi.org/10.1016/0031-8914(50)90130-3. URL: https://www.sciencedirect.com/science/article/pii/0031891450901303.

[6] Robert H. Swendsen and Jian-Sheng Wang. "Nonuniversal critical dynamics in Monte Carlo simulations." In: *Phys. Rev. Lett.* 58 (2 1987), pp. 86–88. DOI: 10.1103/PhysRevLett.58.86. URL: https://link.aps.org/doi/10.1103/PhysRevLett.58.86.

[7] Simon Suchan. *2d-ising-swendsen-wang*. Version 1.0.0. July 2022. DOI: 10.5281/zenodo.1234. URL: https://github.com/somsonson/2d-ising-swendsen-wang.