UNIVERSITEIT UTRECHT

MASTER THESIS

The Ramsey number as a statistical physics problem

Author:
Aris GIOTIS

Supervisor:
Dr. Lars FRITZ

May 8, 2021

Universiteit Utrecht

"There is no substitute for experience."

Marilyn L. Rice

UNIVERSITEIT UTRECHT

Abstract

Faculty of Science
Department of Physics

Master in Theoretical Physics

The Ramsey number as a statistical physics problem

by Aris GIOTIS

In this thesis we are going to study the Ramsey numbers. Our way of doing that differs from the way mathematicians do. Specifically we consider Ramsey numbers as physical systems that interact thermally with an environment. This way we can make use of Statistical Physics in combination with Monte Carlo simulations to answer some of the questions of this project. Using simulations is a different way instead of using analytical methods that are not so effective after all when Ramsey numbers get more complicated. Some of the goals of this thesis are to verify that our methodology works out and thus can be used to improve on the estimation of Ramsey numbers. Last but not least we make a thermodynamic study of the physical system we use to model the search for the Ramsey numbers.

Acknowledgements

First of all for this project I would like to thank my supervisor Lars. His approach, patience and the way he explained some concepts was extremely valuable. He always discussed with excitement motivating me and at the same time showing me how intuitive things are, which is a very important characteristic for a theoretical physicist. He also gave me access to the Gemini server of the Theoretical Physics department thus taking some load off the hardware I was using at home. I would also like to thank the whole group with which we discussed many interesting topics while at the same time we were hanging out as friends enjoying delicious food. This project was a challenge for my supervisor and me since the coronavirus pandemic made in person contact impossible. Nevertheless I had answers to my questions and online meetings that made this project happen.

Last but not least I express my appreciation to my family. My beloved parents and brother are always by my side, help me with issues I face, and remind me how important family is.

Contents

List of Figures

Chapter 1

Introduction

1.1 The party problem

"What is the least amount of people you should invite so that at least 3 people know each other or at least three people do not know each other?". Even though it is a very simple question the answer is not straightforward.

In mathematics, especially in combinatorics and graph theory, there is a concept under the name **Ramsey number**. In order to get a good grasp on what a Ramsey number is we will describe a popular problem named **the party problem**. Later on we will try to generalize the concept to the Ramsey numbers.

The best way to solve this problem is to use a graph representation[1]. Let us assume a *fully connected graph* where each invited person is represented by a *vertex* and every mutual connection between two guests is represented by a colored edge. If the edge is blue the corresponding guests do not know each other but if the edge is red colored the corresponding guests know each other. The way we should color the graph should be abstract so that there is no bias or in other words the guests are chosen in a unbiased manner. Now that we visualised the situation the next step is to find the answer starting from the simplest case and by trial and error increase the number of guests until we come into accordance with one or both statements.

It is obvious that starting with one or two people will never answer the question to the party problem. Next, let us think about 3 guests in total. In this case we can always dissatisfy both statements by simply choosing a fully connected graph using two colors for the edges. Moving on to 4 or 5 guests we can avoid the statement by coloring appropriately the graph. For example as we can see below figure 1.1 describes all the cases described before.

Continuing the search for the answer to the initial question we take a look at $n = 6$ guests. And here a strange but at the same time interesting thing happens. No matter how we color the edges for the party problem with 6 people invited there will always be at least one blue colored "triangle"[2] or

[1] Graphs have vertices and edges and as it will be explained later the only type of graphs we are going to talk about in this thesis is the complete graphs.

[2] A triangle, square, pentagon or whatever will be defined as a 3, 4, 5 or whatever fully connected monochromatic graph from now on.

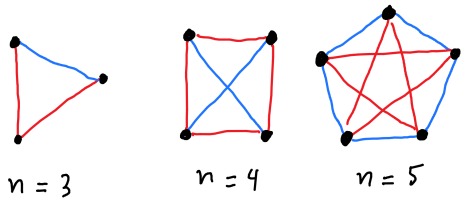


FIGURE 1.1: Party problem for 3,4 or 5 guests in total.

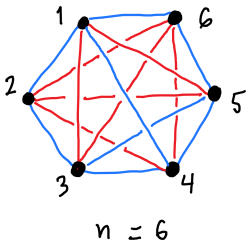


FIGURE 1.2: Party problem for 6 guests.

one red-colored "triangle"[3]. Figure 1.2 shows explicitly what is going on for a specific coloring of a party with 6 guests.

We can see that structures $2-6-4$ and $3-4-5$ are monochromatic. In other words we found the least populated party so that at least 3 people know each other or, in this case both statements are true, at least 3 people are completely strangers.

It is not hard to see that in case we further increase the number of guests, e.g. 7,8 onwards, we cannot avoid the monochromatic structures that already appeared for 6 guests. For this reason the *least amount of people* part of the statement is required since there is a lower bound over which there is no need to discuss further.

Before moving on to the more general concept we need to see some terminology originating from graph theory so that this thesis is more accessible to fields other than physics.

1.2 Small manual from graph theory

Let us present some useful terminology[4]. A **graph** is a set of vertices and edges connecting these vertices. It can be directed or undirected but in our case we will deal exclusively with **undirected graphs**, so there is no need to draw arrows. In the simplest case the edges have one color but in general

[3]Recall that monochromatic triangles in the graph representation refer to 3 people (vertices) mutually knowing or not knowing at all each other (blue or red edges connecting them). We do not refer to the usual meaning of triangles from geometry.

[4]For the interested reader a good source about combinatorics and graph theory is [2, section 1.8]. It discusses Ramsey Theory.

we can use many colors depending on the problem at hand. Also, graphs are classified in complete and incomplete graphs. A **complete** graph is one where every vertex is connected to every other vertex, so in other words it is a **fully connected graph**. In this thesis we are going to discuss only complete graphs since Ramsey numbers deal only with that. One very important concept that we need to highlight is the clique. The **clique** is a complete subgraph of a larger graph. In our case, since we are going to color the edges in more than just one color, we will need to consider **monochromatic cliques** only[5]. Last but not least, one more term that is going to be extremely useful in our study is the **adjacency matrix**. This is a square matrix used to represent a graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph. For example in a graph with two colors allowed if two vertices are connected by an edge then the input can be 1 or -1 depending on the convention of which color goes accordingly with the edge. Since in our case we study the relation between distinct elements (e.g different people, doesn't make sense to say I know me) the diagonal of this matrix is zero or irrelevant. Also since the graph is undirected the adjacency matrix is *symmetric*.

Adjacency matrix will be very useful since it is the matrix representation of a graph that we are going to use in our simulations later on and is easy to implement in our coding scripts.

1.3 Generalization of Ramsey numbers & notation

After discussing and solving the party problem let us introduce a concise notation for the Ramsey numbers. The party problem is no other thing than asking what is $R(3,3) =?$. We saw that $R(3,3) = 6$. If we want to be more detailed we could write $R(3,3) = 6$. Now let us remind once again what this expression means before we proceed. The last expression says the following: *6 is the least number of vertices of a complete graph where no matter how we color, by either blue or red, the edges there will be at least a blue clique with 3 vertices or a red clique with 3 vertices*.

Accordingly if we see something like $R(n,k) = x$ then this means: *x is the least number of vertices of a complete graph where no matter how we color, by either blue or red, the edges there will be at least a blue clique with n vertices or a red clique with k vertices*.

Generally, we can have $R(x_1, x_2, ..., x_n) = K$. Let us give the meaning of this last notation and then move on to the next section. The last expression simply means that: *K is the least number of vertices of a complete graph where no matter how we color, by either blue or red or... or green the edges, there will be at least a blue clique with x_1 vertices or a red clique with x_2 vertices* or ... or green clique with x_n vertices.

As you can guess on your own there can be many more Ramsey numbers just by using more colors (which means different types of connections between atoms) or by making statements which discuss about "3 or 3", "3 or

[5]To be more exact we will consider blue or red colored cliques.

TABLE 1.1: Some Ramsey numbers. Unknown Ramsey numbers are estimated in terms of intervals. $[x, y]$ is a closed interval from x up to y.

(x, y)	$R(x, y)$
(3,3)	6
(3,10)	[40,43]
(4,4)	18
(5,5)	[43,49]
(6,6)	[102,165]

4", "5 or 5", or whatever possible combination of two or even more natural numbers. So far we have described Ramsey numbers, but what is so special about them? The answer is simple and the reason being that mathematicians have not solved the Ramsey numbers problem for every possible combination you can think of. To be more precise there is a table where mathematicians indicate which Ramsey numbers are known and which are not known but residing in some speculated intervals[6]. Table 1.1 shows some of these numbers to get an idea.

1.4 Purpose of this project

Our initial goal is to *reproduce some known Ramsey numbers by using a physics methodology using Statistical Mechanics and Monte Carlo simulations*. After checking that this methodology works the other goal is to *tackle an unknown Ramsey number and study this "**system**"*[7] *from a thermodynamic perspective*. Before moving on to try and achieve the goals mentioned before we need to cover some background regarding Monte Carlo simulations and Statistical Physics.

[6]For further details you can check the following source link: https://mathworld.wolfram.com/RamseyNumber.html

[7]By saying systems we can think of interacting systems of many particles such as an Ising system. In complete analogy a Ramsey system will be treated the same way we treat an Ising model where spins are corresponding to edges of a graph. In the next chapter this analogy will be discussed in more detail.

Chapter 2

Modeling the Ramsey numbers

2.1 Analogy to Ising model

In order to proceed and introduce the technique from physics we remind ourselves of the Ising model. Basically we try to map the Ramsey number problem to an Ising type problem. The reason being the Ising model is relevant to Ramsey numbers of two arguments, has been widely studied both analytically and numerically, and in our case we are going to make an analogy to the numerical approach of this model. The initial step to try and create this analogy is by observing that in the Ramsey number of two colors we can assign each color a specific number. This is similar to assigning each of the 2 spin states of a spin $1/2$ in the Ising model a number. From now on whenever we refer to the color blue we assign the number -1 and to the color red the number $+1$. Ignoring the Ising model dynamics, which in this case are simply neighboring interactions, the general characteristics are more or less the same. An Ising system is a collection of many units where each unit has two possible degrees of freedom known as $1/2$ **spins**. We will not get into the specific physics of spins but we have to keep in mind that there are two possible values of a spin $1/2$ unit. Either spin up, lets us say number 1, or spin down, -1. Recall from chapter 1 that Ramsey numbers with two arguments have edges that are blue or red. So we will assume a *one to one correspondence between 2 colored edges and 1/2 spins*.

Now that we have the aforementioned convention at hand we know how to *represent a graph properly by using the adjacency matrix*. Let us see an example from the graph in figure 2.1. Since we have already labeled all vertices of

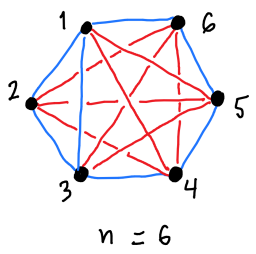


FIGURE 2.1: Graph with 6 vertices in $R(3,3)$ case.

the graph by numbers then the adjacency matrix will be the following 6×6

symmetric[1] matrix[2]:

$$\begin{pmatrix} 0 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & 1 & -1 & -1 & -1 \\ 1 & 1 & 0 & 1 & -1 & -1 \\ -1 & -1 & 1 & 0 & 1 & -1 \\ -1 & -1 & -1 & 1 & 0 & 1 \\ 1 & -1 & -1 & -1 & 1 & 0 \end{pmatrix}.$$

Practically the matrix representation, as well as the graph representation for bigger graphs, is useless to the common eye but it can be of great importance when this representation is read by a computer. This way studying even much larger graphs can be realised using simulations. Now that we know how to represent a graph for Ramsey numbers the next thing we need to tackle is actually trying to solve it. The idea is simple in its sense and it has to do directly with the definition we discussed previously.

Regarding a specific Ramsey number the way to proceed is *working from the bottom up*. In other words we will test a graph of some size[3] by using Monte Carlo simulations and in case we *do not find any monochromatic cliques* for this specific size, this immediately tells us that the lower bound of our estimation is the present size that we already studied. However to make sure that this is indeed the Ramsey number we are looking for, we need to run simulations at greater sizes and look out for graphs with no monochromatic structures. If we cannot find these non-monochromatic states then we can assume as the Ramsey number the lower bound we already found. It may be the case that our algorithm is not as efficient as it should be and thus finding non-monochromatic states is almost impossible. This results in not being sure that the lower bound might be higher in practice. For example suppose that we are trying to find $R(3,3)$ without knowing that the true answer is 6. If our simulation methodology finds zero energy states up to 4 vertices then this does not necessarily mean that this is the real value for $R(3,3)$ since in principle could be 6 or 7 or 17. I.e. 4 is just the first lower bound we found and may be replaced by a greater lower bound[4]. It could be that our simulations are not efficient enough to find non-monochromatic states[5] or that the true value is greater. So the *best estimation we can make is to find a lower bound*. However, since mathematicians provide a table with values using analytic methods then we can cross our results with theirs and make more concrete conclusions.

[1] The upper triangular part of this matrix is relevant to our problem and is going to be the state of our Ramsey system.

[2] Recall that the $A(i,j)$ entry of the adjacency matrix A is simply the color of the edge connecting the i-th vertex with j-th one.

[3] Obviously this size is going to be the maximum number between the two arguments of the Ramsey number we focus on. E.g. for $R(m,n)$ the starting point would be $max(m,n)$ number of vertices.

[4] In this example 4 gets replaced by the true answer 6.

[5] We should remind ourselves that an algorithm that we use is not always the most efficient. So other approaches may come in handy.

2.2 Hamiltonian of a Ramsey system

Counting monochromatic cliques is an important aspect of our model. But where are we going to use this counting we referred to? The idea, which is the inspiring point of the thesis, comes from using the number of monochromatic cliques in our Hamiltonian. In other words we are going to penalize our system for having too many monochromatic cliques and the goal of our simulations is no other than an optimization problem[6] for *finding the graph configuration*[7] *which has the least possible amount of monochromatic cliques*. Thus the definition of a Hamiltonian for a Ramsey system in some state s[8] will simply be:

$$H(s) = \sum_{\forall cliques} (n_{blue}(s) + n_{red}(s)). \tag{2.1}$$

By definition this Hamiltonian is positive definite, since $n \in N$. In practical terms for a specific Ramsey system our goal is to find the state of the system which minimizes this Hamiltonian. In case we find at least one state which has zero energy[9] we have a lower bound for calculating the Ramsey number we are looking for. On the other hand if we are not able to find the ground state this means either we found the Ramsey number, which is quite ambitious as we will see later, or the algorithm we are using in our simulations is not the optimal. Going back to figure 2.1 as an example we notice that there are 2 monochromatic cliques, one is blue $(1-2-3)$ and one red $(2-4-6)$, so the energy of this graph is 2.

Even though the Hamiltonian we introduced does not contain any interaction terms we stress that the system by itself is non-local by definition and this will make things more difficult when tackling higher Ramsey numbers. In contrast the Hamiltonian of the Ising model most of the times includes interactions between neighboring spins thus the dynamics is much simpler.

2.3 Perturbations of a Ramsey system

Continuing the analogy with the Ising model we mention the **excitations**. For an Ising model studied by Monte Carlo simulations, and more specifically to simulations that use the Metropolis algorithm, the excitations are simply done by flipping spins, one at a time. The equivalent to do for a Ramsey system will be "flipping" the color of a random edge. Thus in general our way of exploring the different states in a Ramsey simulation is through randomly[10] flipping edge colors and check how the energy varies along the computation.

[6] To be more specific it is going to be a minimization problem.

[7] From now on we are going to call it a **state** to go along with the physics terminology.

[8] As we saw before a state is represented by a matrix. Here this matrix is denoted by s with **s** standing for the word **state**.

[9] In other words and from now on we will say **ground state**.

[10] In the next chapter we will see that the algorithm we implement is not completely random.

2.4 Observables

Last but not least we need to discuss what observables we are going to encounter in a bit. We already mentioned the **Hamiltonian** which is very important and in it's essence embodies the statement of how to find a Ramsey number just by:

- counting monochromatic cliques
- penalizing states with many cliques, i.e. high energy states.

Other observables that will be quite useful will be the **specific heat** c, the **residual entropy** S_{res} and the **ground state degeneracy** $g(E=0)$. The reason being that later on we will make a thermodynamic study on the model in case we observe any possible phase transitions and the specific heat is going to help us with that. For the specific heat the formula we are going to use is the following[11]:

$$c(T) = \frac{\langle E^2 \rangle - \langle E \rangle^2}{NT^2}, \tag{2.2}$$

where

$$N = \binom{n}{x}, \tag{2.3}$$

is the total number of all x-cliques for a $R(x,x)$ system with n vertices. The reason for using this instead of the total number of edges, i.e. $\binom{n}{2}$, is because the energy scales with $\binom{n}{x}$. Moving on, since we need to check the degeneracy of the ground state the way to do that is by first calculating the residual entropy per clique $\tilde{S}_{res}$ and then estimate the ground state degeneracy. I.e.

$$\tilde{S}_{res} \equiv \frac{S_{res}}{N} = \ln 2 - \int_0^\infty dT \frac{c(T)}{T}\,, \tag{2.4}$$

from which the ground state degeneracy is obtained by

$$g(E=0) = \frac{\tilde{S}_{res}}{\ln 2}. \tag{2.5}$$

A concise description based on the Ising model for the residual entropy and the ground state degeneracy can be found in [1, chapter 9].

So far we saw how a graph can be represented by using the adjacency matrix. This matrix subsequently can be used to extract information about the specific state of a graph. For example what the energy of the graph is and how far from reaching the ground state the state is. Before diving into the actual simulations we conduct for the Ramsey numbers on the next chapter we

[11] $\Delta E = \langle E^2 \rangle - \langle E \rangle^2$ is the energy variance, T is the dimensionless temperature and N is the **energy scaling factor**. For the Ising model the energy scales by the total number of spins. Later on we will see that the energy scales by the total number of monochromatic cliques for the Ramsey system.

will explain roughly how Monte Carlo simulations help us explore the phase space of all possible states by simple examples. Then we will see how using statistical mechanics we can study Ramsey numbers using the Metropolis algorithm which is widely used for the Ising model[12].

[12] A quick review of how the Ising model is studied using the Metropolis Monte Carlo simulation might be helpful for the reader who is not familiar with this topic. A very good source is [3].

Chapter 3

Monte Carlo Simulations

3.1 What is a Monte Carlo simulation?

Let us describe a Monte Carlo simulation first to understand how it works and later on we will try to combine it with statistical mechanics. Monte Carlo comes from the city of Monte Carlo where casinos and gambling are what characterize this city. Basically, this name stands for randomness and this is exactly what we are utilizing except for the fact that we operate on systems under the canonical ensemble. So in some words, they are simulations that evolve randomly.

Imagine that we have a square and inside a circle which is tangent to the four edges as depicted in figure 3.1. Assume that the edge of the square is 2 so the area of the circle is $\pi \times 1^2 = \pi$. Without using any geometry knowledge we can make an estimation for π just by using **random sampling**. Let's say that we are throwing darts randomly onto the square, for a large number of times, and then count how many darts reside inside the circle with respect to how many were thrown in total. After the experiment, if we count the fraction of darts in the circle over the darts in the whole square, what we get is an estimate of π since:

$$\frac{\pi * a^2}{(2a)^2} = \frac{\pi}{4} \approx \frac{n_c}{n_s}. \tag{3.1}$$

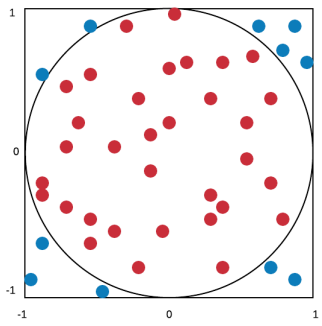


FIGURE 3.1: Estimating Pi using Monte Carlo Method. http://selkie-macalester.org/csinparallel/modules/CrossPlatformProgramming/build/html/MonteCarloPi/Pi.html

Basically what we are doing, ignoring π as a known irrational number, is to determine the area of a circle through random sampling with a simple division, just by probing if specific locations of darts are inside (hit) or outside (miss) of an area. By taking enough samples we get a good idea of how big an area is.

Now let's give a similar example from a simple calculation of the average body weight of humans in a specific area like in a real world study. Let's say we want to estimate the average body weight of all humans in Europe. Ideally we should be able to weigh each human in this continent and then average over all measurements taken. Practically this is not an easy task because the population is enormous so we should try doing something different. Instead, we can choose a much smaller group of humans and expect that their average body weight is a good estimate for all European citizens. But we should take into account some things.

- The first thing is that the smaller group we chose has to be unbiased and there are many reasons for this. If this group resides in the same city, for example, the weights tend to be quite different than other cities due to different quality of life from area to area or resources or even personal habits. In other words this small group might not be a good representative for the estimation of the average value. So in order to be **unbiased** we should choose people **randomly** through the whole Europe.

- The next thing we need to consider is the size of the group. If we only weigh 50 people out of millions the chances of picking heavy or light ones is very high. To be more confident about the estimation we need to pick a much much **larger group** so that variety is guaranteed. In mathematics and especially in probability and statistics this is called the **law of large numbers**. As a result the deviation from the mean value tends to zero[1].

The same is valid for Monte Carlo simulations if you think about it. The main idea is that we can obtain a representative group of samples of some large population of possibilities if we allow the simulation to evolve randomly.

The first example with the darts, we would need to check every possible point in the square so that we determine if a dart resides in or out the circle to calculate precisely the area. Just like we, in principle, would need to measure the body weight of each citizen to calculate the average weight accurately. Instead, we can rely on randomly selected samples, and according to the law of large numbers we can be more and more confident about the result the more samples we take.

[1] More samples means less variation.

3.2 Energy and Statistical Mechanics

Monte Carlo simulations are widely used in physics too. But there is a significant difference regarding the previous explanation. In physics, there is an important quantity called energy. Every physical system can be in a variety of states and every state is characterised by it's energy. The natural evolution of a physical system is going towards the state of minimum energy which is known as ground state. However in order to reach this ground state the system must be perturbed in the least possible way. The role of perturbation is played by temperature. And this is where statistical mechanics comes into play. In terms of simulations we can achieve that by using the canonical ensemble[2]. We need to ensure that the perturbations are low enough so that the system's energy is not increased high enough and goes away from the ground state energy. So we try to siumulate using low temperature values hoping that we do not get stuck in a metastable minimum during the simulation.

Assume that a system is in equilibrium at some temperature T. Then from statistical mechanics we know that a state s has a probability P_s to occur:

$$P_s = \frac{e^{-E_s/kT}}{Z} \tag{3.2}$$

where Z is the partition function of the system in the canonical ensemble[3] which is a normalization factor and is the sum of weights of all possible states:

$$Z = \sum_{i=1}^{n} e^{-E_i/kT}. \tag{3.3}$$

Having a Hamiltonian assigned to our system then we can proceed to do a study and even try to find a ground state. Going back to the discussion of Monte Carlo simulations in case we have a physical system we can make use of random sampling but with a slight difference now. Every time we generate a random number, which subsequently is going to be translated to a new state, we need to take into account the perturbations of a heat bath at temperature T that our system resides. For example we can generate a new state which is totally fine from the random aspect but it might violate the Boltzmann distribution which is essential to be satisfied at all times. Thus in Monte Carlo simulations we will make a slight modification by satisfying the Boltzmann distribution by using something known as Metropolis algorithm.

[2]In the canonical ensemble a system is part of a larger system with which is in equilibrium in the sense that it can only exchange energy but nothing else. Equilibrium then is achieved when the exchange of energy is almost zero.

[3]Also known as Boltzmann distribution. $k = 1.380649 \times 10^{-23} J \times K^{-1}$ is the Boltzmann constant in SI units. Later on we will consider $k = 1$ using reduced units for simplicity.

3.2.1 Energy and Ramsey numbers

In section 1.3 it was made obvious that in order to find the Ramsey number for a specific case, e.g. $(3,3)$ what we need to do is to find a graph configuration which has no monochromatic cliques involved. Thus, if we want to define some kind of energy of this Ramsey system then this energy should simply count the number of monochromatic cliques, i.e.

$$H = \sum_{cliques} (n_{blue} + n_{red}),$$

as we already saw in section 2.2.

Whenever this quantity becomes zero for a specific graph generated by Monte Carlo simulations then we move on to the next graph size until we can no longer find a zero-energy ground state[4]. The largest graph size for which a ground state of zero energy has been found will be our estimation for a particular Ramsey number.

3.2.2 Fixed temperature Metropolis algorithm for Ramsey numbers

Having in mind how the Metropolis algorithm works for the Ising model we can proceed in describing how the algorithm works for Ramsey systems. Let us start by illustrating how the algorithm works for a $R(k,l)$ system.

Initially we have to create the smallest size of a graph which is no other than the maximum number between k or l. Before moving on we define the temperature under which the Ramsey system is going to be perturbed. This value should be low enough so that perturbations are not high enough and ground state is more easily obtained. Next, we flip the color of an edge which is chosen randomly. Then, we calculate the energy of the new state. Every time we calculate the energy we check if this energy is zero. If the energy is zero we stop our simulation and we proceed to the next graph size and repeat the steps of the simulation. Otherwise, with this finite energy we found we calculate the difference between the energy of the new random state and the previous one and we estimate the Boltzmann factor $e^{-\Delta E/T}$, where $\Delta E \equiv E_{new} - E_{prev}$, and we accept the new state with probability $e^{-\Delta E/T}$ or we reject it with probability $1 - e^{-\Delta E/T}$. In case the new state is accepted we keep the new state and ignore the previous one. Then we update the state of the system and repeat the procedure from the beginning till we reach a ground state. Below we can see a flowchart where the algorithm is presented.

[4]To avoid any confusion we should point out that there will always be a ground state. However, in our case there can be ground states of energy higher than zero.

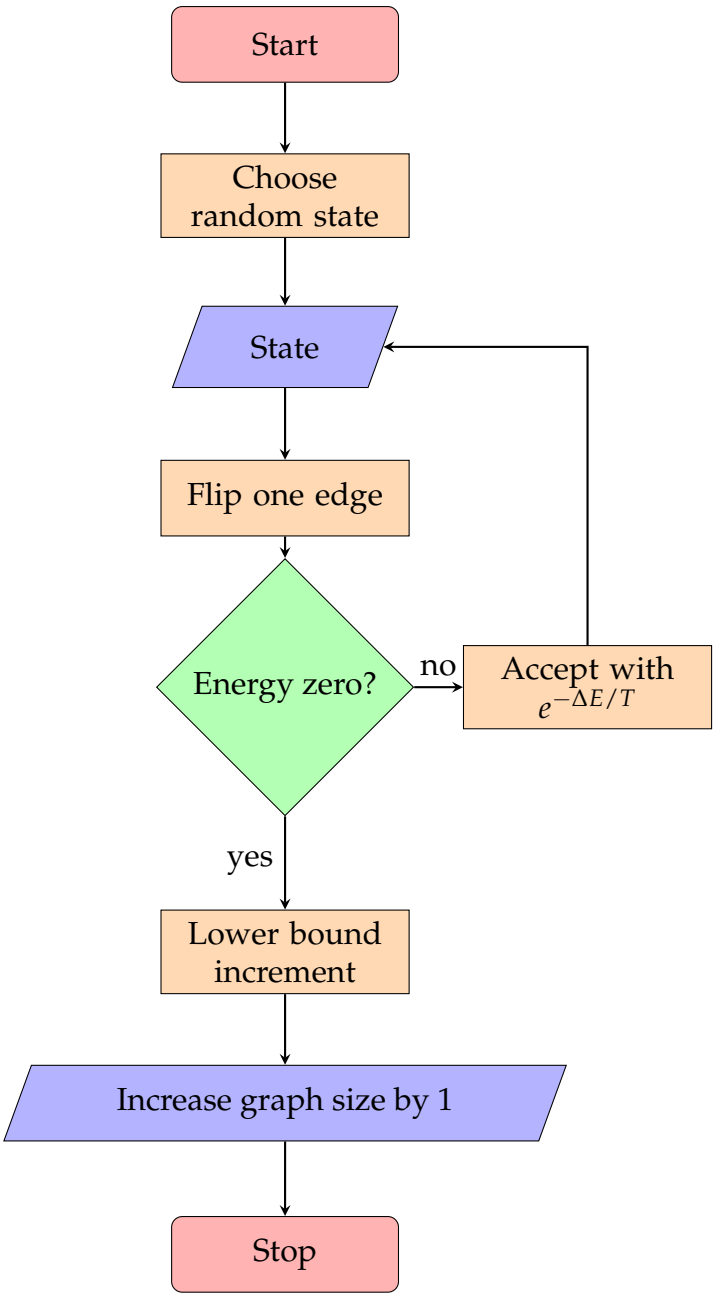


FIGURE 3.2: Fixed temperature Metropolis algorithm

There are some things we need to keep in mind. First of all we cannot simulate at zero temperature. This is easy to see when looking at the Boltzmann factor. When temperature is zero then the Boltzmann factor becomes zero thus not accepting any moves. What we can do however is to conduct simulations at temperatures just above zero. For example at temperatures 0.1 or 0.01 etc. However, we need to be careful of what temperature we are using. The reason being that the energy landscape of a physical system might be very complex so that small temperatures do not perturb strongly enough the system so that it explores the whole phase space in order to find the point or points of minimum energy, i.e. the ground state or ground states. Below you can see two examples of energy landscapes where the ground states are depicted as the global minima of their respective graphs. The one on the top is a simple landscape while on the bottom the situation is far more difficult for a simulation at a fixed temperature to reach the ground state.

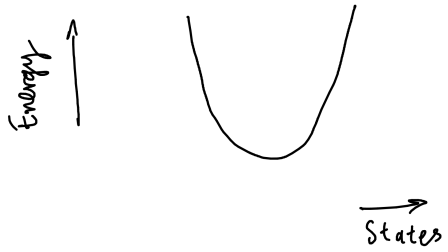


FIGURE 3.3: Simple energy landscape with a global minimum

FIGURE 3.4: Complex energy landscape with a global minimum and many metastable regions

The dilemma is the following: Should we use high enough temperature to explore the whole phase space of the system or use a small value and try to find the minimum instead hoping we do not get stuck in a metastable region? Although the question is simple the answer is not so easy to find. That is where many different algorithms appear in the literature to tackle these

kind of situations. One of these algorithms is known as Simulated Annealing algorithm. The way the algorithm works is quite simple but for now we will postpone the discussion for later on when we will discuss the thermodynamic aspect of Ramsey systems.

Now that we have the Metropolis algorithm we can start doing some sanity checks for some known Ramsey numbers. Then we will try to attack one unknown Ramsey number which is $R(5,5)$.

Chapter 4

Attacking $R(3,3)$, $R(4,4)$, $R(5,5)$

After discussing what the Ramsey number is by definition and the party problem example then we described how someone can calculate it by using an approach inspired from physics. The energy minimization where the ultimate goal is to find the ground state or states, in general. Now let us implement this approach in practice starting from two easy cases, i.e. $R(3,3) = 6$ & $R(4,4) = 18$. Then after this double check we attack the unknown $43 \leq R(5,5) \leq 49$.

4.1 First sanity check of $R(3,3)$

The first check we need to make in order to see in practice if the fixed temperature Metropolis algorithm works is to study the easiest non trivial Ramsey system $R(3,3)$. We note that our simulations were done in MATLAB programming language and we will include our scripts written in MATLAB for the interested reader to try on his/her own. In Appendix A you can find the script for the $R(3,3)$ system. Starting from the smallest graph size up to the desired size, which in this case is 6 vertices we get the following results shown in figure 4.1.

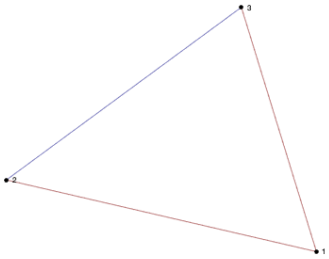
As indicated before, recall that we use the convention where **blue** edges are represented by -1 and **red** by $+1$.

The running time was pretty short, as expected, and the temperature we used for this simulation was $T = 0.1$[1], as indicated in the respective script in appendix A. For completeness, in figure 4.2 we demonstrate one ground state of an $R(3,3)$ Ramsey system for 6 vertices. Challenge yourself by trying to decrease even further the system's energy with 6 vertices and you will be convinced in practice that $R(3,3) = 6$.

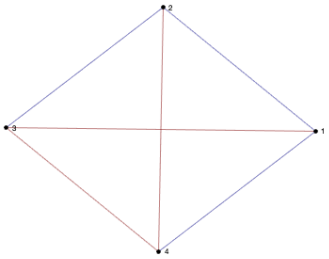
Reassuring that $R(3,3) = 6$ is the first out of 2 sanity checks in total. We reconfirmed that $R(3,3) = 6$ by using Monte Carlo simulation implementing the Metropolis algorithm. The next step is to do the same for a bit heavier task, i.e. to reconfirm $R(4,4) = 18$. Notice that the approach we follow using energy minimization through thermal equilibrium at "low" temperatures[2] is

[1] We are using reduced units where temperature is dimensionless.

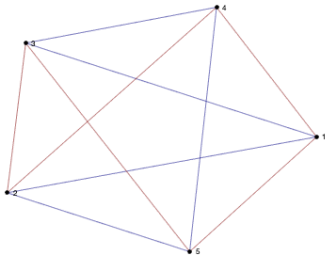
[2] By the term "low" we simply mean a temperature value which is capable of giving us the ground state. For $R(3,3)$ we used $T = 0.1$ but 0.01 or 1 or even greater values work fine. The reason being that the energy landscape of $R(3,3)$ is very simple.



(A) 3 vertices



(B) 4 vertices [$\sim$ 0.005 sec]



(C) 5 vertices [$\sim$ 0.005 sec]

FIGURE 4.1: $R(3,3)$ ground states for 3,4 and 5 vertices with **zero** energies.

working fine to begin with. Later on we will encounter $R(5,5)$ where searching through phase space for ground states becomes extremely difficult. As you can already see, depicting ground states by graphs becomes quite difficult to read. For the suspicious reader in the Appendix C we provide the corresponding adjacency matrices for ground states we found with the largest graph size for each $R(x,x)$ system[3]. At the very least we can expect that complexity becomes even more significant when we proceed to greater Ramsey indices $(4,4)$, $(5,5)$, $(6,6)$ etc.

4.2 Second sanity check of $R(4,4)$

For the $R(4,4)$ case the total number of computations that need to be done by the computer are much more. In fact, the **total number of edges for a 4-vertices clique** are

$$\binom{4}{2} = 6. \tag{4.1}$$

[3]It is meaningless to provide ground states for all graph sizes if we provide only those that correspond to the largest graph sizes.

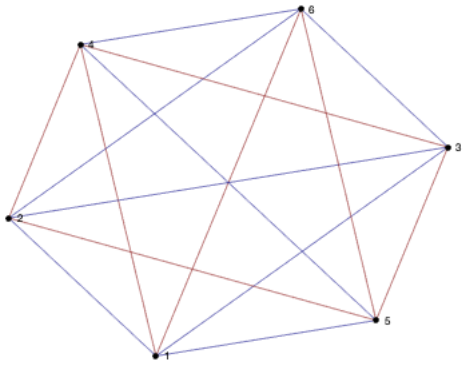


FIGURE 4.2: 6 vertices & ground state with energy greater than 0. [$\sim$ 0.05 sec]

Thus the program needs to take into account 3 vertices more than previously when calculating the energy of a specific clique. Moreover, the **total number of cliques** in a n-vertices $R(4,4)$ system are

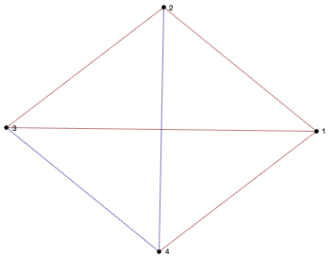
$$\binom{n}{4} = \frac{n!}{4!(n-4)!}. \tag{4.2}$$

However this relation is not enlightening enough to get an idea of what values we are encountering, so below there is a table showing the fast increase of the binomial coefficient.

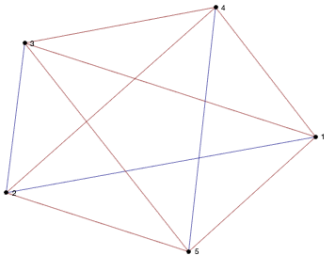
# vertices	$\binom{n}{4}$
4	4
5	5
6	15
7	35
8	70
9	126
10	210
11	330
12	495
13	715
14	1001
15	1365
16	1820
17	2380
18	3060

Now let us see the graphs we obtained by finding ground states of $R(4,4)$ system[4] starting from 4 vertices and ending at 18 vertices.

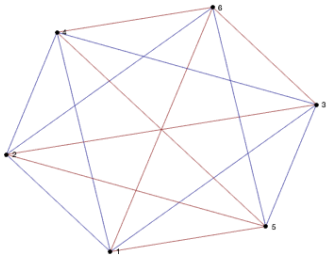
[4]In contrast with the previous $R(3,3)$ system now we increase the total number of Monte Carlo steps to 10^4 because the energy landscape is more complex. The temperature was kept same with value $T = 0.1$.



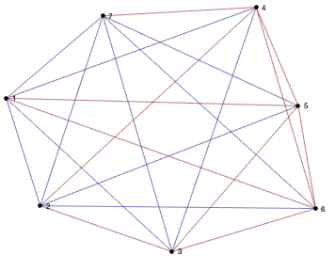
(A) 4 vertices



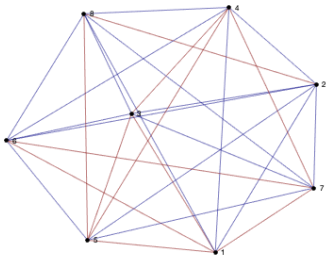
(B) 5 vertices



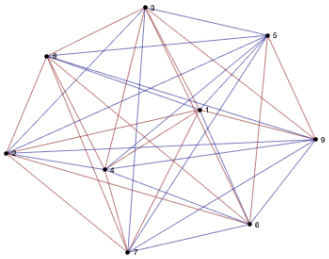
(C) 6 vertices



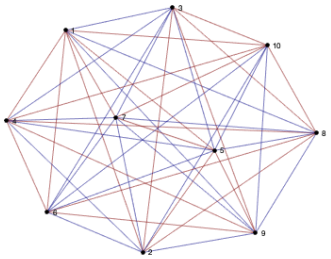
(D) 7 vertices



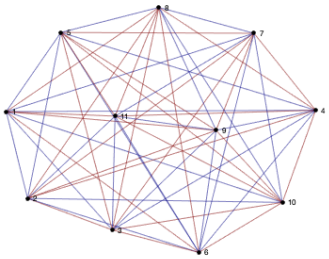
(E) 8 vertices



(F) 9 vertices

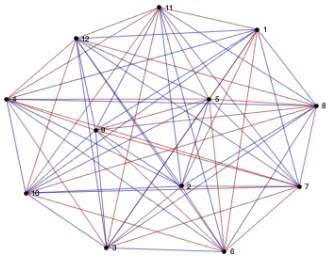


(G) 10 vertices

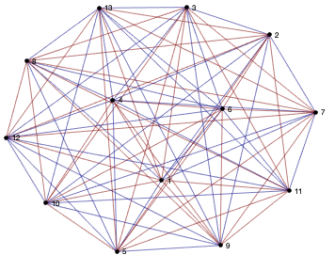


(H) 11 vertices

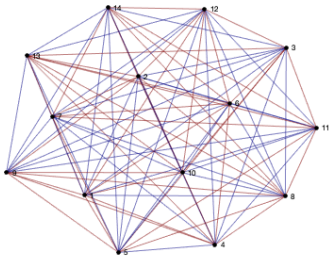
FIGURE 4.3: $R(4,4)$ ground states for $4 \leq n \leq 11$ vertices with **zero** energies.



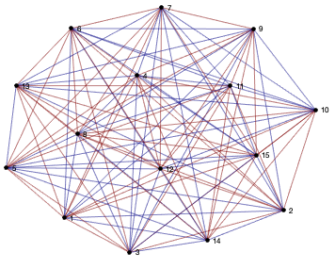
(A) 12 vertices



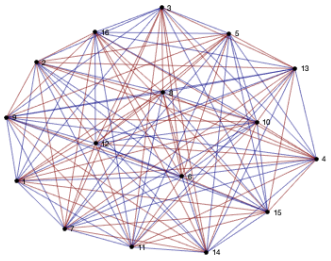
(B) 13 vertices



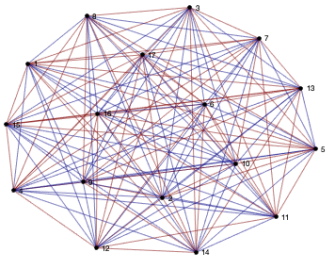
(C) 14 vertices



(D) 15 vertices



(E) 16 vertices



(F) 17 vertices

FIGURE 4.4: $R(4,4)$ ground states for $12 \leq n \leq 17$ vertices with **zero** energies.

Simulating $R(4,4)$ system we reconfirmed that $R(4,4) = 18$ as discovered by mathematicians. However the simulations are getting more difficult since the binomial coefficients we encountered are much greater than the binomial coefficients involved for the $R(3,3)$ case. That is why we increased the total Monte Carlo steps from 10^3 to 10^4 since the phase space, i.e. all possible states, is much larger. The graphical representation is becoming even more difficult to read by naked eye but at least we can agree on the fact that complexity increases significantly.

Since our algorithm works for the known cases $R(3,3)$ and $R(4,4)$ let us try attacking the unknown case of $43 \leq R(5,5) \leq 49$. In the next section we will try to study the $R(5,5)$ system.

4.3 Attacking the unknown $R(5,5)$

To begin with let us mention firstly the total number of cliques for a $R(5,5)$ system with n-vertices, which is:

$$\binom{n}{5} = \frac{n!}{5!(n-5)!}. \tag{4.3}$$

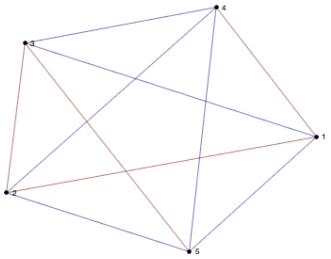
To get a grasp on what order of magnitude we are dealing with we present the table below.

# vertices	$\binom{n}{5}$	# vertices	$\binom{n}{5}$
5	5	19	11628
6	6	20	15504
7	21	21	20349
8	56	22	26334
9	126	23	33649
10	252	24	42504
11	462	25	53130
12	792	26	65780
13	1287	27	80730
14	2002	28	98280
15	3003	29	118755
16	4368	35	324632
17	6188	43	962598
18	8568	49	1906884

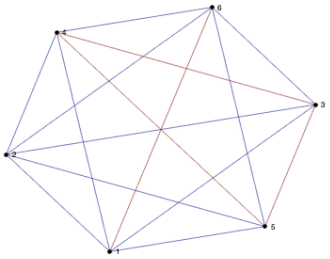
TABLE 4.1

According to what we did in the two previous sections we will implement the fixed temperature Metropolis algorithm. However, as we saw from $R(4,4)$ case we are expecting higher complexity in the $R(5,5)$ case thus running times will increase significantly as well as the algorithm will not be so

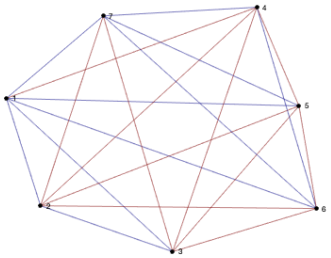
effective. Using $T = 0.1$ and 10^4 total Monte Carlo steps we obtain the following results.



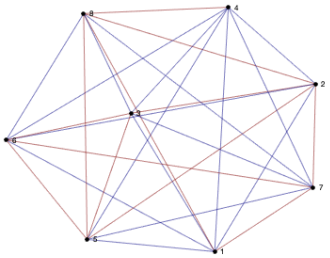
(A) 5 vertices



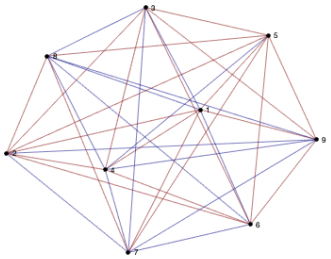
(B) 6 vertices



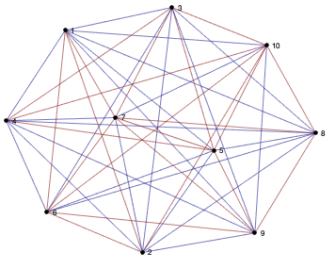
(C) 7 vertices



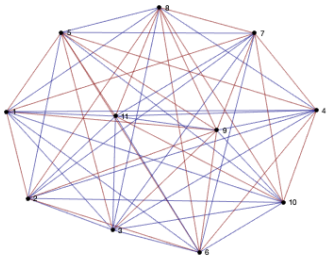
(D) 8 vertices



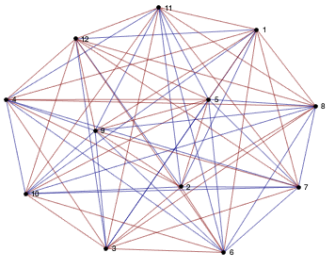
(E) 9 vertices



(F) 10 vertices

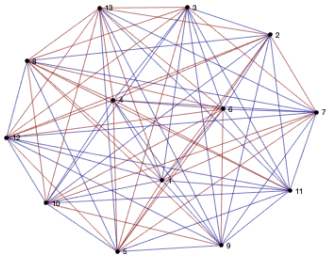


(G) 11 vertices

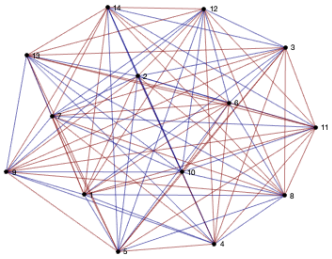


(H) 12 vertices

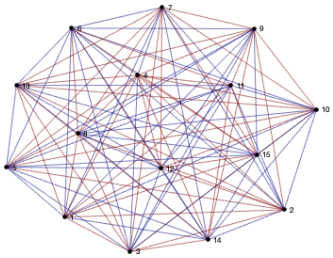
FIGURE 4.5: Some of the $R(5,5)$ ground states for $5 \leq n \leq 12$ vertices.



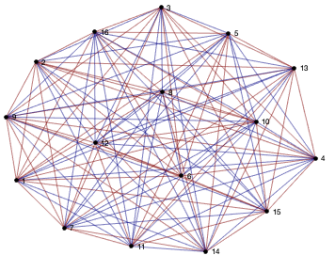
(A) 13 vertices



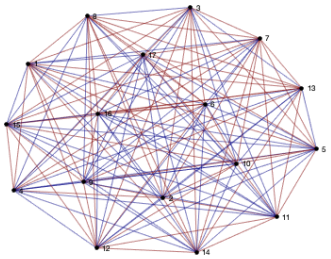
(B) 14 vertices



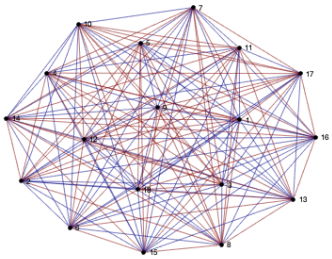
(C) 15 vertices



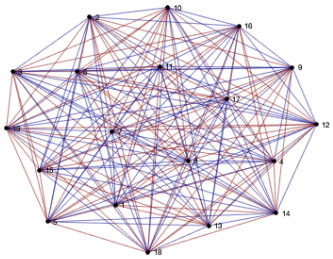
(D) 16 vertices



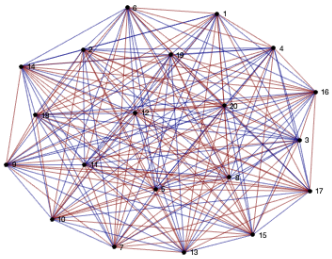
(E) 17 vertices



(F) 18 vertices

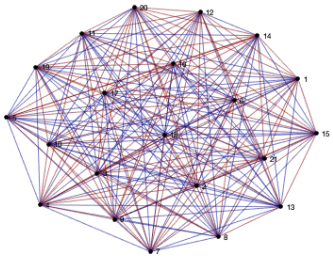


(G) 19 vertices

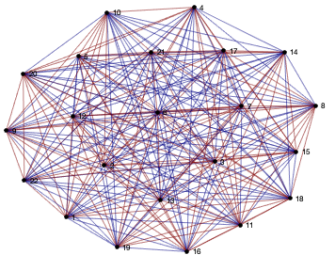


(H) 20 vertices

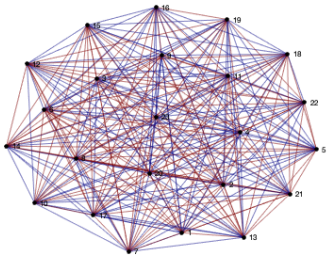
FIGURE 4.6: Some of the $R(5,5)$ ground states for $13 \leq n \leq 20$ vertices.



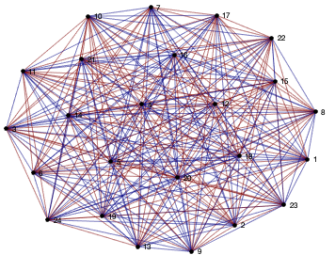
(A) 21 vertices



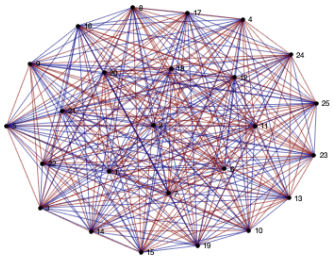
(B) 22 vertices



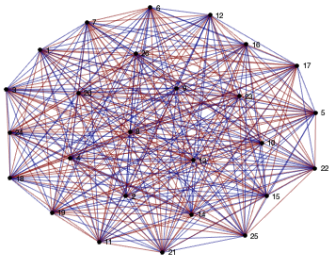
(C) 23 vertices



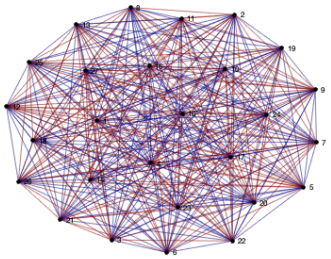
(D) 24 vertices



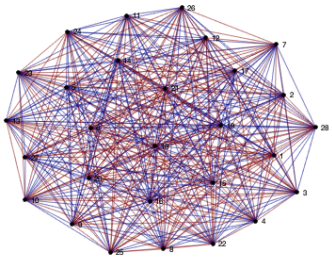
(E) 25 vertices



(F) 26 vertices

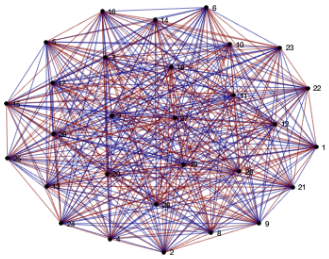


(G) 27 vertices

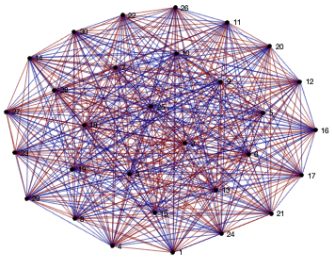


(H) 28 vertices

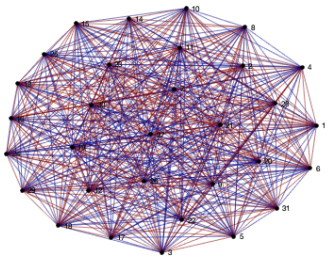
FIGURE 4.7: Some of the $R(5,5)$ ground states for $21 \leq n \leq 28$ vertices.



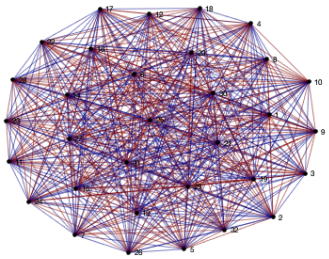
(A) 29 vertices



(B) 30 vertices



(C) 31 vertices



(D) 32 vertices

FIGURE 4.8: Some $R(5,5)$ ground states for $29 \leq n \leq 32$ vertices.

Until 32 vertices the total number of Monte Carlo steps we used were enough for the simulation to find ground states[5]. However, at some point we need either increase the total number of Monte Carlo steps or use a different approach. Let us increase the Monte Carlo steps to 10^5. We expect this to have more chances finding a ground state because the total number of cliques for 33 vertices is 237,336. Figure 4.9 shows a ground state we found for 33 vertices by simply increasing the total steps of the simulation.

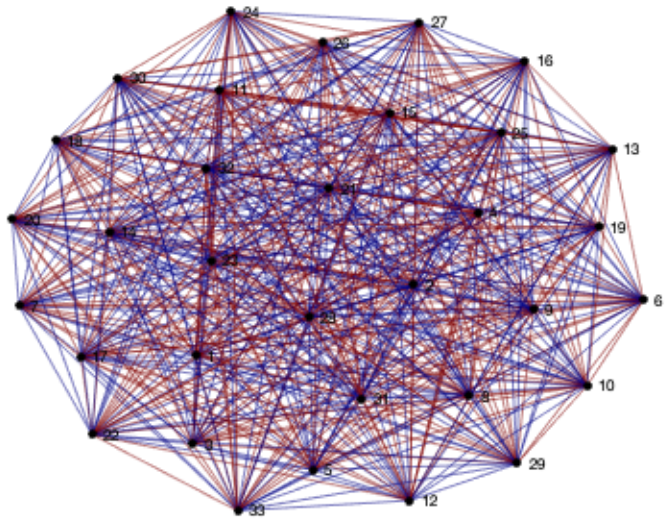


FIGURE 4.9: $R(5,5)$-system ground state with 33 vertices.

We noticed that the running time for the figure 4.9-simulation increased significantly in relation to the previous sizes. At this point we realised that we needed to make a crucial modification to our script which has to do with the energy calculation on the run. When running a simulation at the point where we flip the color of a randomly chosen edge when we compute the energy of the "flipped" configuration we consider all possible cliques and count how many of them are monochromatic. It is easy to see that when the size of a graph increases then the total number of monochromatic cliques increases exponentially thus making our current simulation much slower. However we can overcome this computational obstacle by changing the way we calculate the energy.

What we can do is the following: after flipping the color of one edge we focus only on the cliques that include the flipped edge and ignore the rest. Let

[5]Since 32 vertices are less than the estimated 43 these ground states have zero energy as expected.

us calculate how many cliques are affected in principle by a simple example. Assume that we have a graph of n vertices in some random state. Now let us consider a specific edge of the aforementioned graph and try to count how many cliques are connected to this edge. Since we are in the $R(5,5)$ case we know that we examine cliques of 5 vertices. From the specific edge two vertices have already been taken into consideration which gives a freedom for the rest 3 vertices chosen out of all $n-2$ vertices. So we can choose from $n-2$ vertices for the 3 free vertices[6], which is

$$\binom{n-2}{3} = \frac{n!}{3!(n-5)!}. \tag{4.4}$$

In practice this means that the number of computations done by the computer reduce by a significant amount. Table 4.2 shows the values we get by this modification.

# vertices	$\binom{n-2}{3}$	# vertices	$\binom{n-2}{3}$
5	1	19	680
6	4	20	816
7	10	21	969
8	20	22	1140
9	35	23	1330
10	56	24	1540
11	84	25	1771
12	120	26	2024
13	165	27	2300
14	220	28	2600
15	286	29	2925
16	364	35	5456
17	455	43	10660
18	560	49	16215

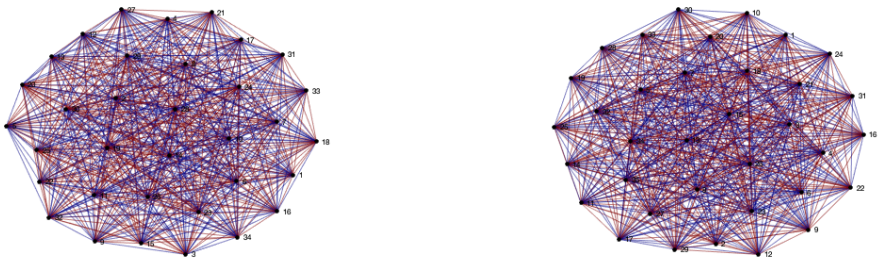
TABLE 4.2

Comparing to table 4.1 we see a difference of 2 orders of magnitude. This is going to make our simulations run faster.

One more thing we need to point out is that since the energy landscape becomes more and more complex we need to change the way we treat thermally our system. Since we are not getting a ground state as easy as for the smaller graphs instead of using a fixed low temperature algorithm we can use a simulated annealing approach. Namely, we start heating our system in an initial temperature and then we evolve it by lowering the temperature until we reach a final value. Throughout this cooling process we keep track

[6]The order does not matter in our case. Strictly speaking we choose without repetition with no order out of a set of numbers, i.e. vertices.

of the energy and if it becomes zero then we terminate the simulation outputting the corresponding state. Now we can move on to the 34 vertices by using the accelerating trick and the simulated annealing algorithm. Figure 4.10 shows two ground states for graphs of 34 and 35 vertices.



(A) 34 vertices; Simulation used 50 Monte Carlo Sweeps, 1 Touch per sweep, 25 temperatures with high temperature 5 and low temperature 0.2. It lasted for 82 seconds.

(B) 35 vertices; Simulation used 100 Monte Carlo sweeps, 1 touch per sweep, 25 temperatures with high temperature 5 and low temperature 0.2. It lasted 188 seconds.

FIGURE 4.10: $R(5,5)$ ground states for $34 \leq n \leq 35$ vertices.

The largest graph size whose a ground state has been found with our approach is 35 vertices and the corresponding adjacency matrix can be found in Appendix C. For $n \geq 36$ vertices it is very difficult to find a ground state. The simulations get very slow and the energy landscape becomes even more complex.

Chapter 5

Thermodynamic Study and Conclusions

At some point the energy minimization approach attempting to estimate $R(5,5)$ is not as efficient as we expected. The greatest graph size for which we found a state with no monochromatic structures is 35 vertices. For 36 vertices onward it is almost impossible to track any graphs with zero energy. Even though we cannot proceed any further with the minimization problem we can, however, study the Ramsey system $R(5,5)$ from a thermodynamic point of view.

5.1 Thermodynamics of $R(5,5)$

Going back to the analogy of a Ramsey model to the Ising model to make a thermodynamic study and see the behavior over temperature, if there is a phase transition or not we need to calculate the **specific heat**. Here we are not going to elaborate on how the specific heat formula is derived, for those who want to see a detailed derivation they can check [3, subsection 1.2.1], but we are going to use the formula and create plots that will give us a better understanding on what is happening over a wide range of temperatures. It would be very interesting to know if a phase transition is happening or if the Ramsey system behaves thermodynamically as a frustrated magnetic system[1].

5.1.1 Thermal equilibrium

In order to estimate the specific heat for the $R(5,5)$-system we need to stress some things first. Since we need to measure the system's energy at a fixed temperature we need to make sure that it has reached **thermal equilibrium**. The only way to ensure equilibrium at a fixed temperature is by checking the average value of the system's energy and see if it is the same for two different initial states that we start our simulation with. There is no observable like the magnetization for the Ising model which helps us characterize if we have equilibrium or not. The system will achieve thermal equilibrium if we let it

[1] This is a good guess that actually makes sense from the fact that non-locality that Ramsey systems have can behave as frustrated anti-ferromagnetic systems where it is way far from obvious what the ground states look like.

in thermal contact for a good amount of time, i.e. for sufficient Monte Carlo steps. In our case we are going to let our system heat up for 100 Monte Carlo sweeps[2] and then we are going to start measuring the energy.

5.1.2 Statistical independence

After the system reaches equilibrium we ignore the previous part of the simulation tending to the equilibrium and with the rest part we take energy measurements. Since these measurements are derived by Markov Chain Monte Carlo simulations many of these measurements are statistically correlated. In order for our specific heat estimation to be unbiased right before calculating the energy variance we need to consider only statistically independent measurements. To do that we have to calculate the **time displaced auto-correlation function** $\chi(t)$ of the energy at a specific temperature:

$$\chi(t) = \int dt^{'}[E(t^{'}) - \langle E \rangle][E(t^{'}+t) - \langle E \rangle] \tag{5.1}$$

$$= \int dt^{'}[E(t^{'})E(t^{'}+t) - \langle E \rangle]^2. \tag{5.2}$$

Remember however that time is discrete in simulations therefore we use the discrete version:

$$\chi(t) = \frac{1}{t_{max}-t}\sum_{t^{'}=0}^{t_{max}-t} E(t^{'})E(t^{'}+t) \tag{5.3}$$

$$- \frac{1}{t_{max}-t}\sum_{t^{'}=0}^{t_{max}-t} E(t^{'}) \times \frac{1}{t_{max}-t}\sum_{t^{'}=0}^{t_{max}-t} E(t^{'}+t), \tag{5.4}$$

where t_{max} is the duration of the simulation[3]. The auto-correlation function will give us the amount of time we need to wait between two measurements so that subsequent measurements are statistically independent. In other words the amount of time we need to wait is also known as **correlation time** τ. To be more specific the correlation time is given by:

$$\tau = \int_0^{\infty} dt \frac{\chi(t)}{\chi(0)}, \tag{5.5}$$

whose discrete version is simply the following sum,

[2] When doing simulations with systems that are made up of many components then we flip, on average, at least once each component so that after touching all of them the new state of the system can be statistically independent from the previous state. Which gives us the right to make measurements without worrying about statistical correlations and biased measurements. Recall the discussion about Monte Carlo simulations on the Chapter 2.

[3] After reaching equilibration. From now on time is measured in sweeps. This way we can compare systems of different sizes in case it's needed.

$$\tau = \sum_{t=0}^{\infty} \frac{\chi(t)}{\chi(0)}. \tag{5.6}$$

In our case after doing the calculation of the correlation time we found that it is always zero. Thus, we get uncorrelated states before and after each sweep, which gives us the right to do statistics right on. This is also a sign of how easy it is for a Ramsey system to go from one state to a completely different one. It might be that these states have the same energy but the chances of these configurations to be exactly the same or at least correlated after each sweep[4] are very low.

5.1.3 Specific heat

The next thing in line is the calculation of the specific heats using equation 2.2. At a fixed temperature we collect all the energy measurements and we calculate the variance $\sigma^2(E) =< \Delta E >^2$. The tricky part has to do with the error of our $c(T)$ estimation at each respective temperature. And for this. we use the **bootstrap method**. This error estimation method is described in detail in [3, subsection 3.4.3] so we sketch the outline of it. At a fixed temperature after collecting the energy measurements[5] we resample a new collection of measurements out of which we extract a new value for c. After doing this resample for quite many times, in our case we chose to resample 200 times using a collection of 1000 energy measurements, we get the estimation for the error of the specific heat. I.e.

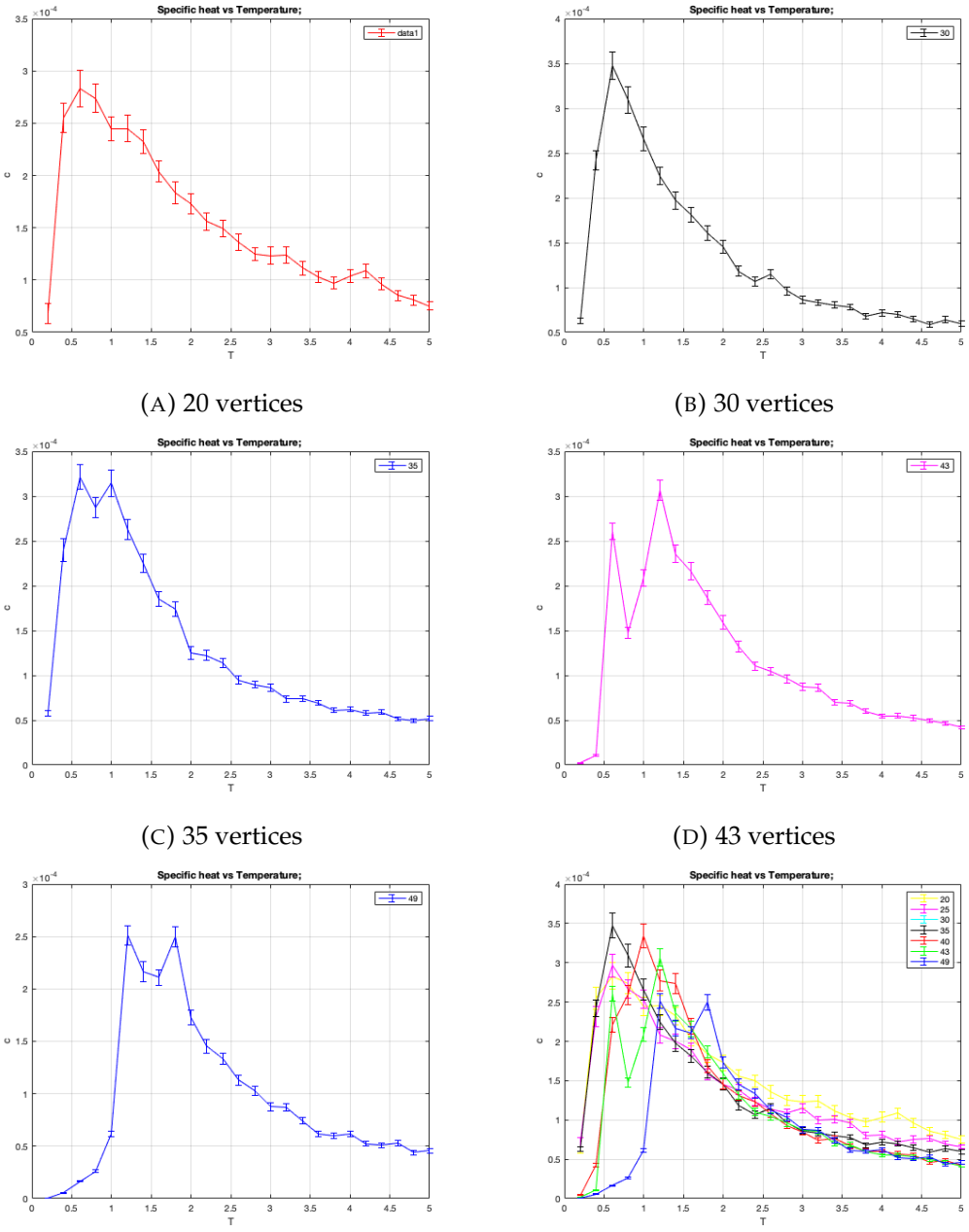
$$\sigma = \sqrt{\bar{c}^2 - \bar{c}^2}, \tag{5.7}$$

where $\bar{X}$ denotes the *average value over a set of resamples* at a fixed temperature.

Finally we are ready to see the results for the specific heat for systems of different sizes.

[4] We forgot to mention that for a graph of n vertices the sweep is equal to the total number of edges $\binom{n}{2}$.

[5] Which ideally should be statistically independent, or unbiased in other words.



(A) 20 vertices

(B) 30 vertices

(C) 35 vertices

(D) 43 vertices

(E) 49 vertices

(F) Altogether

FIGURE 5.1: Specific heat for various graph sizes.

Examining the specific heat plots we can say that for every size there is a region where a peak shows out. This peak behaves as an obstacle and tells us that around the respective temperature region there are many states having the same energy thus making it difficult for us to reach the ground state starting from a high temperature simulated annealing cooling procedure.Furthermore it suggests that there might be a phase transition of second order.

Chapter 6

Conclusions

6.1 Conclusions

After looking at the results we derived from calculating the specific heat for Ramsey graphs of different sizes for the $R(5,5)$ type we notice peaks that are indicative for a second order phase transition. Methods such as finite-size scaling are needed to make a more detailed analysis on the transition that is observed. In this project this analysis is not executed. One more point we need to stress is the algorithm we are using to track the ground states for different types of Ramsey graphs. We saw in the previous chapters that the algorithm we implemented in our simulations is the Simulated Annealing algorithm but this does not seem to be the optimal algorithm for our purposes. At this point we need to mention that we tried to make use of the Parallel Tempering algorithm[1] with no success. The reason was that some fine tuning of temperatures for the Parallel Tempering parameters was needed which was not obvious at all how to do. Even though Parallel tempering algorithm is very promising the implementation is not obvious at all.

Ramsey numbers are complex by definition. The way these graphs behave as physical systems is not easy to understand. Dealing with fully connected graphs from a physicist's perspective means to tackle a non-local systems. In contrast the Ising model we used to make the analogy is a local system thus giving us limitations on what we can achieve. So far the methodology we introduced in this project seems to work up to some point. However, after this point we need to make use of a more efficient algorithm. And if this algorithm is found then we can double-check the analytical results derived for the Ramsey numbers or even better make some corrections to them.

[1] Also known as Replica Exchange MCMC sampling.

Appendix A

Fixed temperature Metropolis algorithm scripts

A.1 Fixed temperature Metropolis algorithm for $R(3,3)$

The script using the fixed temperature Metropolis algorithm is shown below. Lines of code that need to be explained are commented on the side so that reading the script makes sense.

```
% R(3,3) system
% Fixed temperature Metropolis algorithm

rng("shuffle") % random number generator
n=5; % # vertices [INPUT]
edges = nchoosek(n,2); % # edges
mc_steps=1000; % # Monte Carlo steps [INPUT]
beta = 10;  % Inverse temperature [INPUT]
de = 1; % Energy of monochromatic clique [INPUT]
% energy = -ones(1,mc_steps); % Preallocation
energy = NaN(1,mc_steps); % Preallocation
% Random initial Adjacency matric (Zero trace & Symmetric)
B = unidrnd(2,[n,n])*2-3;
B = triu(B,1);
B = B + B.';
first = cc_energy(n,B,de); % Initial total energy

% If initial energy is zero we terminate
if first==0
    fprintf("\n\n case 01: success \n\n"); % first senario
    return
end

% Otherwise we proceed with the simulation
for i=1:mc_steps
    % calculate energy of old state
    energy(i) = cc_energy(n,B,de);
    % we search in the upper triangular part of
    % the adjacency matrix
    row = randi(n-1); % choose a random row
    col = randi([min(row+1,n),max(row+1,n)]); % same for column
    B(row,col) = B(row,col)*(-1); % flip random matrix entry
    B(col,row) = B(row,col); % symmetrize adjacency matrix
    % calculate energy of NEW state
    energy(i+1) = cc_energy(n,B,de);
```

```
    % if new energy is zero terminate
    if energy(i+1)==0
        fprintf("\n\n case 02: success \n\n"); % second senario
        break
    % else perturb thermally the system
    else
        % energy difference (new-old)
        delta_energy = energy(i+1) - energy(i);
        % Boltzmann factor acceptance rule
        if rand > acceptance(beta, delta_energy)
            B(row, col) = B(row, col)*(-1);
            B(col,row) = B(row,col);
            energy(i+1) = energy(i);
        end
    end
end

[red,blue] = cc_r_b(n,B); % count red & blue cliques
fprintf("\n\n case 03: failure \n\n"); % third senario
fprintf('\n - For %d vertices we obtain:\n',n);
fprintf('\n - final energy = %d\n\n',energy(i+1));

% GRAPH
% Input graph weights: Bonds matrix will give us the weights
figure(2)
G=graph(B,'upper');
plot(G,'NodeColor','k','EdgeCData',G.Edges.Weight)
c = colorbar;
w = c.LineWidth;
c.LineWidth = 1.5;
colormap cool

% FUNCTIONS
% - R(3,3) system energy calculation
function e = cc_energy(n, B, de)
    e = 0;
    for i=1:n
        for j=(i+1):n
            for k=(j+1):n

                if ( abs( B(i,j) + B(j,k) + B(k,i) ) == 3 )
                    e = e + de;
                end
            end
        end
    end
end

% - Acceptance rule calculation
function a = acceptance(beta, delta_energy)
    a = min(1,exp(-beta*delta_energy));
end

% - Counting red and blue cliques separately
function [red,blue] = cc_r_b(n,B)
    red = 0;
    blue = 0;
```

```
    for i=1:n
        for j=(i+1):n
            for k=(j+1):n
                if ( ( B(i,j) + B(j,k) + B(k,i) ) == 3 )
                    red = red + 1;
                elseif ( ( B(i,j) + B(j,k) + B(k,i) ) == -3 )
                    blue = blue + 1;
                end
            end
        end
    end
end
```

A.2 Fixed temperature Metropolis algorithm for $R(4,4)$

The script using the fixed temperature Metropolis algorithm is shown below. Lines of code that need to be explained are commented on the side so that reading the script makes sense.

```
% R(4,4) system
% Fixed temperature Metropolis algorithm

rng("shuffle") % random number generator
n=18; % # vertices [INPUT]
edges = nchoosek(n,2); % # edges
mc_steps=10000; % # Monte Carlo steps [INPUT]
beta = 10;  % Inverse temperature [INPUT]
de = 1; % Energy of monochromatic clique [INPUT]
% energy = -ones(1,mc_steps); % Preallocation
energy = NaN(1,mc_steps); % Preallocation
% Random initial Adjacency matric (Zero trace & Symmetric)
B = unidrnd(2,[n,n])*2-3;
B = triu(B,1);
B = B + B.';
first = dd_energy(n,B,de); % Initial total energy

tic

% If initial energy is zero we terminate
if first==0
    fprintf("\n\n case 01: success \n\n"); % first senario
    return
end

% Otherwise we proceed with the simulation
for i=1:mc_steps
    % calculate energy of old state
    energy(i) = dd_energy(n,B,de);
    % we search in the upper triangular part of
    % the adjacency matrix
    row = randi(n-1); % choose a random row
    col = randi([min(row+1,n),max(row+1,n)]); % same for column
    B(row,col) = B(row,col)*(-1); % flip random matrix entry
    B(col,row) = B(row,col); % symmetrize adjacency matrix
    % calculate energy of NEW state
```

```
    energy(i+1) = dd_energy(n,B,de);
    % if new energy is zero terminate
    if energy(i+1)==0
        fprintf("\n\n case 02: success \n\n"); % second senario
        break
    % else perturb thermally the system
    else
        % energy difference (new-old)
        delta_energy = energy(i+1) - energy(i);
        % Boltzmann factor acceptance rule
        if rand > acceptance(beta, delta_energy)
            B(row, col) = B(row, col)*(-1);
            B(col,row) = B(row,col);
            energy(i+1) = energy(i);
        end
    end
end

toc

[red,blue] = dd_r_b(n,B); % count red & blue cliques
fprintf("\n\n case 03: failure \n\n"); % third senario
fprintf('\n - For %d vertices we obtain:\n',n);
fprintf('\n - final energy = %d\n\n',energy(i+1));

% GRAPH
% Input graph weights: Bonds matrix will give us the weights
figure(2)
G=graph(B,'upper');
plot(G,'NodeColor','k','EdgeCData',G.Edges.Weight)
c = colorbar;
w = c.LineWidth;
c.LineWidth = 1.5;
colormap jet
colorbar off
axis off
box off

% FUNCTIONS
% - R(4,4) system energy calculation
function e = dd_energy(n, B, de)
    e = 0;
    for i=1:n
        for j=(i+1):n
            for k=(j+1):n
                for c=(k+1):n
                    if ( abs( B(i,j) + B(j,k) + B(k,c) +...
                            B(c,i) + B(i,k) + B(j,c) ) == 6)
                        e = e + de;
                    end
                end
            end
        end
    end
end

% - Acceptance rule calculation
```

```
function a = acceptance(beta, delta_energy)
    a = min(1,exp(-beta*delta_energy));
end

% - Counting red and blue cliques separately
function [red,blue] = dd_r_b(n,B)
    red = 0;
    blue = 0;
    for i=1:n
        for j=(i+1):n
            for k=(j+1):n
                for c=(k+1):n
                    if ( ( B(i,j) + B(j,k) + B(k,c) +...
                            B(c,i) + B(i,k) + B(j,c) ) == 6 )
                        red = red + 1;
                    elseif ( ( B(i,j) + B(j,k) + B(k,c) +...
                            B(c,i) + B(i,k) + B(j,c) ) == -6 )
                        blue = blue + 1;
                    end
                end
            end
        end
    end
end
```

A.3 Fixed temperature Metropolis algorithm for $R(5,5)$

The script using the fixed temperature Metropolis algorithm is shown below. Lines of code that need to be explained are commented on the side so that reading the script makes sense.

```
% R(5,5) system
% Fixed temperature Metropolis algorithm

rng("shuffle") % random number generator
n=5; % # vertices [INPUT]
edges = nchoosek(n,2); % # edges
mc_steps=10000; % # Monte Carlo steps [INPUT]
beta = 10;  % Inverse temperature [INPUT]
de = 1; % Energy of monochromatic clique [INPUT]
% energy = -ones(1,mc_steps); % Preallocation
energy = NaN(1,mc_steps); % Preallocation
% Random initial Adjacency matric (Zero trace & Symmetric)
B = unidrnd(2,[n,n])*2-3;
B = triu(B,1);
B = B + B.';
first = ee_energy(n,B,de); % Initial total energy

tic

% If initial energy is zero we terminate
if first==0
    fprintf("\n\n case 01: success \n\n"); % first senario
    return
end
```

```
25
26 % Otherwise we proceed with the simulation
27 for i=1:mc_steps
28     % calculate energy of old state
29     energy(i) = ee_energy(n,B,de);
30     % we search in the upper triangular part of
31     % the adjacency matrix
32     row = randi(n-1); % choose a random row
33     col = randi([min(row+1,n),max(row+1,n)]); % same for column
34     B(row,col) = B(row,col)*(-1); % flip random matrix entry
35     B(col,row) = B(row,col); % symmetrize adjacency matrix
36     % calculate energy of NEW state
37     energy(i+1) = ee_energy(n,B,de);
38     % if new energy is zero terminate
39     if energy(i+1)==0
40         fprintf("\n\n case 02: success \n\n"); % second senario
41         break
42     % else perturb thermally the system
43     else
44         % energy difference (new-old)
45         delta_energy = energy(i+1) - energy(i);
46         % Boltzmann factor acceptance rule
47         if rand > acceptance(beta, delta_energy)
48             B(row, col) = B(row, col)*(-1);
49             B(col,row) = B(row,col);
50             energy(i+1) = energy(i);
51         end
52     end
53 end
54
55 toc
56
57 [red,blue] = ee_r_b(n,B); % count red & blue cliques
58 fprintf("\n\n case 03: failure \n\n"); % third senario
59 fprintf('\n - For %d vertices we obtain:\n',n);
60 fprintf('\n - final energy = %d\n\n',energy(i+1));
61
62 % GRAPH
63 % Input graph weights: Bonds matrix will give us the weights
64 figure(2)
65 G=graph(B,'upper');
66 plot(G,'NodeColor','k','EdgeCData',G.Edges.Weight)
67 c = colorbar;
68 w = c.LineWidth;
69 c.LineWidth = 1.5;
70 colormap jet
71 colorbar off
72 axis off
73 box off
74
75 % FUNCTIONS
76 % - R(5,5) system energy calculation
77 function e = ee_energy(n, B, de)
78     e = 0;
79     for i=1:n
80         for j=(i+1):n
81             for k=(j+1):n
```

```
                for c=(k+1):n
                    for w=(c+1):n
                        if ( abs( B(i,j) + B(j,k) + B(k,c) +...
                                B(c,w) + B(w,i) +...
                                B(i,k) + B(j,c) +...
                                B(k,w) + B(i,c) +...
                                B(j,w) ) == 10)
                            e = e + de;
                        end
                    end
                end
            end
        end
    end
end

% - Acceptance rule calculation
function a = acceptance(beta, delta_energy)
    a = min(1,exp(-beta*delta_energy));
end

% - Counting red and blue cliques separately
function [red,blue] = ee_r_b(n,B)
    red = 0;
    blue = 0;
    for i=1:n
        for j=(i+1):n
            for k=(j+1):n
                for c=(k+1):n
                    for w=(c+1):n
                        if ( ( B(i,j) + B(j,k) + B(k,c) + ...
                                B(c,w) + B(w,i) +...
                                B(i,k) + B(j,c) +...
                                B(k,w) + B(i,c) +...
                                B(j,w) ) == 10)
                            red = red + 1;
                        elseif ( ( B(i,j) + B(j,k) + B(k,c) +
   ...
                                B(c,w) + B(w,i) +...
                                B(i,k) + B(j,c) +...
                                B(k,w) + B(i,c) +...
                                B(j,w) ) == -10)
                            blue = blue + 1;
                        end
                    end
                end
            end
        end
    end
end
```

A.4 Simulated annealing Metropolis algorithm for $R(5,5)$

The script using the simulated annealing Metropolis algorithm is shown below. Lines of code that need to be explained are commented on the side so that reading the script makes sense. We remind to the reader that the two differences with the previous script are the implementations of the **flip_energy** function and the use of a cooling down process.

```
% R(5,5) system
% Simulated Annealing Metropolis algorithm

rng("shuffle") % random number generator
n=33; % # vertices [INPUT]
edges = nchoosek(n,2); % # edges
mcs=20; % Monte Carlo Sweep; 300; 50; [INPUT]
touches=1; % X (# touches or attempt frequency); [INPUT]
sweep=edges*touches;
temps=25; % # temperatures; [INPUT]
energy=NaN(1,sweep*mcs*temps+1); % preallocation
measure_e=NaN(temps,mcs); % energy measurements
Ts=NaN(1,temps); % temperatures

t=1; % index for counting Monte Carlo steps realized
T_i=5;   % initial temperature; 1; 0.5; [INPUT]
T_f=0.2;   % final temperature; 0.1; 0.5; [INPUT]
de = 1; % Energy of monochromatic clique [INPUT]

% Random initial Adjacency matric (Zero trace & Symmetric)
B = unidrnd(2,[n,n])*2-3;
B = triu(B,1);
B = B + B.';

energy(1) = ee_energy(n,B,de); % energy initialization @ T_i
measure_e(1,1)=energy(1); % measure_e initialization

tic

% If initial energy is zero we terminate
if energy(1)==0
    toc
    fprintf("\n\n case 01: success \n\n"); % first senario
    return
end

% Otherwise we proceed with the simulation
for k=1:temps
    T = (T_f-T_i) * (k-1) / (temps-1) + T_i; % linear decrease
    beta = 1/T; % inverse temperature
    Ts(k)=T; % temperature vector
    for w=1:mcs % monte carlo sweep
        for i=1:sweep % sweeping all edges times touches
            % we search in the upper triangular part of
            % the adjacency matrix
            row = randi(n-1); % choose a random row
```

```
47              col = randi([min(row+1,n),max(row+1,n)]); % same for
    column
48
49              % calculate energy of cliques connected to row-col
    edge
50              flip = flip_energy(n, B, row, col, de);
51              % calculate energy of the rest cliques
52              rest = energy(t) - flip;
53
54              B(row,col) = B(row,col)*(-1); % flip random matrix
    entry
55              B(col,row) = B(row,col); % symmetrize adjacency
    matrix
56
57              flap = flip_energy(n, B, row, col, de);
58              % calculate energy of NEW state
59              energy(t+1) = rest + flap;
60
61              % if new energy is zero terminate
62              if energy(t+1)==0
63                  fprintf("\n\n case 02: success \n\n"); % second
    senario
64                  toc
65                  return
66              % else perturb thermally the system
67              else
68                  % energy difference (new-old)
69                  delta_energy = energy(t+1) - energy(t);
70                  % Boltzmann factor acceptance rule
71                  if rand > acceptance(beta, delta_energy)
72                      B(row, col) = B(row, col)*(-1);
73                      B(col,row) = B(row,col);
74                      energy(t+1) = energy(t);
75                  end
76                  t = t + 1;
77              end
78          end
79          measure_e(k,w)=energy(t);
80      end
81  end
82
83  toc
84
85  [red,blue] = ee_r_b(n,B); % count red & blue cliques
86  fprintf("\n\n case 03: failure \n\n"); % third senario
87  fprintf('\n - For %d vertices we obtain:\n',n);
88  fprintf('\n - final energy = %d\n\n',energy(i+1));
89
90  % GRAPH
91  % Input graph weights: Bonds matrix will give us the weights
92  figure(2)
93  G=graph(B,'upper');
94  plot(G,'NodeColor','k','EdgeCData',G.Edges.Weight)
95  c = colorbar;
96  w = c.LineWidth;
97  c.LineWidth = 1.5;
98  colormap jet
```

```
99  colorbar off
100 axis off
101 box off
102
103 % FUNCTIONS
104 % - R(5,5) system energy calculation
105 function e = ee_energy(n, B, de)
106     e = 0;
107     for i=1:n
108         for j=(i+1):n
109             for k=(j+1):n
110                 for c=(k+1):n
111                     for w=(c+1):n
112                         if ( abs( B(i,j) + B(j,k) + B(k,c) +...
113                                 B(c,w) + B(w,i) +...
114                                 B(i,k) + B(j,c) +...
115                                 B(k,w) + B(i,c) +...
116                                 B(j,w) ) == 10)
117                             e = e + de;
118                         end
119                     end
120                 end
121             end
122         end
123     end
124 end
125
126 % - Acceptance rule calculation
127 function a = acceptance(beta, delta_energy)
128     a = min(1,exp(-beta*delta_energy));
129 end
130
131 % - Counting red and blue cliques separately
132 function [red,blue] = ee_r_b(n,B)
133     red = 0;
134     blue = 0;
135     for i=1:n
136         for j=(i+1):n
137             for k=(j+1):n
138                 for c=(k+1):n
139                     for w=(c+1):n
140                         if ( ( B(i,j) + B(j,k) + B(k,c) + ...
141                                 B(c,w) + B(w,i) +...
142                                 B(i,k) + B(j,c) +...
143                                 B(k,w) + B(i,c) +...
144                                 B(j,w) ) == 10)
145                             red = red + 1;
146                         elseif ( ( B(i,j) + B(j,k) + B(k,c) +
    ...
147                                 B(c,w) + B(w,i) +...
148                                 B(i,k) + B(j,c) +...
149                                 B(k,w) + B(i,c) +...
150                                 B(j,w) ) == -10)
151                             blue = blue + 1;
152                         end
153                     end
154                 end
```

```
155             end
156         end
157     end
158 end
159 
160 % - Finding all combinations of cliques connected keeping fixed
    a specific
161 % - edge
162 function e = flip_energy(n, B, x, y, de)
163     % flipped edge (x,y) where x<y<=n
164     % 2 fixed indices {x,y}
165     % 3 free indices {i,j,f}
166     % 5 vertices cliques
167     % n: # total vertices
168     % B: Adjacency matrix
169     e = 0;
170     % 01
171     for i=(y+1):n
172         for j=(i+1):n
173             for f=(j+1):n
174                 if (abs(B(x,y)+B(y,i)+B(i,j)+B(j,f)+B(f,x)...
175                     +B(x,i)+B(y,j)+B(i,f)...
176                     +B(x,j)+B(y,f))==10)
177                     e = e + de;
178                 end
179             end
180         end
181     end
182     % 02
183     for i=1:(x-1)
184         for j=(y+1):n
185             for f=(j+1):n
186                 if (abs(B(i,x)+B(x,y)+B(y,j)+B(j,f)+B(f,i)...
187                     +B(i,y)+B(x,j)+B(y,f)...
188                     +B(i,j)+B(x,f))==10)
189                     e = e + de;
190                 end
191             end
192         end
193     end
194     % 03
195     for i=1:n
196         for j=(i+1):(x-1)
197             for f=(y+1):n
198                 if (abs(B(i,j)+B(j,x)+B(x,y)+B(y,f)+B(f,i)...
199                     +B(i,x)+B(j,y)+B(x,f)...
200                     +B(i,y)+B(j,f))==10)
201                     e = e + de;
202                 end
203             end
204         end
205     end
206     % 04
207     for i=1:n
208         for j=(i+1):n
209             for f=(j+1):(x-1)
210                 if (abs(B(i,j)+B(j,f)+B(f,x)+B(x,y)+B(y,i)...
```

```
                    +B(i,f)+B(j,x)+B(f,y)...
                    +B(i,x)+B(j,y))==10)
                    e = e + de;
                end
            end
        end
    end
    % 05 ATTENION NOW TAKE (y,x)
    for i=(x+1):n
        for j=(i+1):n
            for f=(j+1):(y-1)
                if (abs(B(x,i)+B(i,j)+B(j,f)+B(f,y)+B(y,x)...
                    +B(x,j)+B(i,f)+B(j,y)...
                    +B(x,f)+B(i,y))==10)
                    e = e + de;
                end
            end
        end
    end
    % 06
    for i=(x+1):(y-1)
        for j=(y+1):n
            for f=(j+1):n
                if (abs(B(x,i)+B(i,y)+B(y,j)+B(j,f)+B(f,x)...
                    +B(x,y)+B(i,j)+B(y,f)...
                    +B(x,j)+B(i,f))==10)
                    e = e + de;
                end
            end
        end
    end
    % 07
    for i=1:(x-1)
        for j=(x+1):(y-1)
            for f=(y+1):n
                if (abs(B(i,x)+B(x,j)+B(j,y)+B(y,f)+B(f,i)...
                    +B(i,j)+B(x,y)+B(j,f)...
                    +B(i,y)+B(x,f))==10)
                    e = e + de;
                end
            end
        end
    end
    % 08
    for i=1:n
        for j=(i+1):(x-1)
            for f=(x+1):(y-1)
                if (abs(B(i,j)+B(j,x)+B(x,f)+B(f,y)+B(y,i)...
                    +B(i,x)+B(j,f)+B(x,y)...
                    +B(i,f)+B(j,y))==10)
                    e = e + de;
                end
            end
        end
    end
    % 09
    for i=(x+1):n
```

```
268         for j=(i+1):(y-1)
269             for f=(y+1):n
270                 if (abs(B(x,i)+B(i,j)+B(j,y)+B(y,f)+B(f,x)...
271                     +B(x,j)+B(i,y)+B(j,f)...
272                     +B(x,y)+B(i,f))==10)
273                     e = e + de;
274                 end
275             end
276         end
277     end
278     % 10
279     for i=1:(x-1)
280         for j=(x+1):n
281             for f=(j+1):(y-1)
282                 if (abs(B(i,x)+B(x,j)+B(j,f)+B(f,y)+B(y,i)...
283                     +B(i,j)+B(x,f)+B(j,y)...
284                     +B(i,f)+B(x,y))==10)
285                     e = e + de;
286                 end
287             end
288         end
289     end
290 end
```

Appendix B

Brief discussion on observables

B.1 Residual entropy & ground state degeneracy

For the Ising model the energy scales by the total number of spins. Instead for a Ramsey system, let us assume $R(x, x)$, the energy scales as

$$\binom{n}{x},$$

which is the total number of cliques. Thus the specific heat can takes the following form:

$$c = \frac{\Delta E}{\binom{n}{x} k T^2}.$$

Moreover, the **residual entropy** S_{res} is:

$$S_{res} = \binom{n}{x} \left(\ln 2 - \int_0^\infty dT \frac{c}{T} \right),$$

thus the **residual entropy per clique** $\tilde{S}_{res}$ will be:

$$\tilde{S}_{rex} \equiv S_{res} / \binom{n}{x} = \ln 2 - \int_0^\infty dT \frac{c}{T},$$

from which the **ground state degeneracy** $g(E = 0)$ is obtained by

$$g(E = 0) = \frac{\tilde{S}_{res}}{\ln 2}.$$

B.2 Counting states

The way we count states for a Ramsey system in order to do correctly the calculation of the 3 aforementioned observables, c, S_{res}, $\tilde{S}_{res}$, takes into account the energy of a clique. From the energy perspective a clique can be excited or not. Thus, ignoring any additional degeneracies that may affect our counting, the total number of states with respect to energy are:

$$2^{\binom{n}{x}},$$

where n is a specific graph size, x is the argument of $R(x, x)$ giving us the $x-$cliques for the energy calculation. 2 comes from the fact that each clique can be in two states energetically speaking, i.e. excited or not, as discussed before. However there are many details that need to be considered in case we want to do a strict count of states when we are not interested energetically. For example each excited state has two possibilities, i.e. can be either blue or red, if we have two arguments for the Ramsey number, thus for energies greater than zero we need to correct by a factor of 2. If we are more cautious we observe that there is 2-fold degeneracy because energetically there is no difference if we inverted the colors from blue to red or vice versa. This is similar to Ising model with no external magnetic field.

Appendix C

Adjacency matrices

C.1 A ground state of a $R(4,4)$-system with 17 vertices

Below you can find the adjacency matrix of a ground state for a graph with 17 vertices for a $R(4,4)$-system. Recall $R(4,4) = 18$. Therefore you can copy and paste the following configuration on your own to double check that this is indeed one ground state.

0,-1,1,1,-1,1,1,-1,-1,-1,-1,1,1,-1,1,1
-1,0,-1,-1,1,1,-1,-1,1,-1,1,1,1,1,-1,1,-1
1,-1,0,1,-1,-1,1,-1,1,1,1,1,-1,1,-1,-1,-1
1,-1,1,0,1,-1,-1,1,-1,-1,1,1,-1,-1,-1,1,1
-1,1,-1,1,0,1,1,-1,-1,-1,1,1,-1,-1,1,-1,1
1,1,-1,-1,1,0,1,-1,1,1,-1,-1,-1,-1,-1,1,1
1,-1,1,-1,1,1,0,-1,-1,1,1,-1,1,-1,1,-1,-1
-1,-1,-1,1,-1,-1,-1,0,1,1,1,-1,1,-1,1,1,1
-1,1,1,-1,-1,1,-1,1,0,1,1,-1,-1,1,-1,-1,1
-1,-1,1,-1,-1,1,1,1,1,0,-1,1,-1,-1,1,1,-1
-1,1,1,1,1,-1,1,1,1,-1,0,-1,1,-1,-1,-1,-1
-1,1,1,1,1,-1,-1,-1,-1,1,-1,0,-1,1,1,1,-1
1,1,-1,-1,-1,-1,1,1,-1,-1,1,-1,0,1,1,1,-1
1,1,1,-1,-1,-1,-1,-1,1,-1,-1,1,1,0,1,-1,1
-1,-1,-1,-1,1,-1,1,1,-1,1,-1,1,1,1,0,-1,1
1,1,-1,1,-1,1,-1,1,-1,1,-1,1,1,-1,-1,0,-1
1,-1,-1,1,1,1,-1,1,1,-1,-1,-1,-1,1,1,-1,0

C.2 A ground state of a $R(5,5)$-system with 35 vertices

Below you can find the adjacency matrix for oe ground state of a graph with 35 vertices for a $R(5,5)$-system. Recall $43 \leq R(5,5) \leq 49$. Again you by copying the matrix below you can check on your own that it is a ground state.

0,1,-1,1,-1,-1,1,1,-1,-1,1,-1,1,1,-1,-1,1,1,1,-1,1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,-1,1,-1
1,0,1,1,1,1,-1,1,-1,-1,-1,-1,-1,1,1,1,-1,1,-1,1,1,1,1,-1,1,1,-1,-1,-1,1,-1,1,-1,-1,-1
-1,1,0,-1,-1,-1,-1,-1,1,-1,1,1,1,1,1,1,1,1,-1,-1,-1,1,1,1,-1,1,-1,-1,1,1,1,1,-1,-1,-1,-1,-1
1,1,-1,0,-1,-1,1,-1,-1,-1,-1,-1,-1,1,-1,1,-1,-1,1,1,1,-1,1,-1,1,-1,1,1,1,1,1,1,1,1,-1,1
-1,1,-1,-1,0,-1,-1,1,1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,-1,-1,-1,-1,1,1,1,1,-1,1,-1,-1,-1,1,1,1
-1,1,-1,-1,-1,0,-1,-1,1,1,1,-1,-1,1,-1,1,1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,-1,1,1,1,1,1,-1
1,-1,-1,1,-1,-1,0,-1,1,1,1,-1,1,1,1,-1,1,1,-1,-1,1,-1,-1,1,-1,-1,-1,-1,1,-1,1,-1,1,-1,1
1,1,1,-1,1,-1,-1,0,1,-1,-1,1,1,1,-1,-1,-1,1,1,1,-1,-1,1,-1,1,-1,1,-1,-1,-1,1,-1,-1,1,-1
-1,-1,-1,-1,1,1,1,1,0,-1,1,-1,1,-1,-1,1,-1,-1,-1,1,1,1,-1,-1,-1,1,1,1,-1,1,1,-1,1,-1,1
-1,-1,1,-1,-1,1,1,-1,-1,0,1,1,-1,-1,1,1,-1,1,1,-1,1,1,-1,-1,-1,1,-1,1,1,1,-1,1,1,-1,-1,1
1,-1,1,-1,1,1,1,-1,1,1,0,-1,-1,1,1,-1,1,-1,1,1,1,-1,-1,1,-1,1,1,-1,1,1,-1,-1,1,1,-1,-1
-1,-1,1,-1,-1,-1,-1,1,-1,1,-1,0,-1,1,1,-1,1,-1,1,-1,-1,1,-1,1,-1,1,1,1,-1,1,1,1,1,1,1
1,-1,1,-1,1,-1,1,1,1,-1,-1,-1,0,1,-1,-1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,1,1,-1,-1,1
1,1,1,1,1,1,1,1,-1,-1,1,1,1,1,0,-1,1,-1,-1,-1,-1,-1,1,1,1,-1,-1,-1,-1,-1,1,-1,1,-1,-1,1,-1
-1,1,1,-1,-1,-1,1,-1,-1,1,1,1,1,-1,-1,0,-1,-1,1,-1,1,1,1,-1,1,1,1,1,1,-1,-1,-1,-1,1,1,-1,-1
-1,1,1,1,1,1,1,-1,-1,1,1,-1,-1,-1,1,-1,0,1,-1,-1,1,-1,-1,1,-1,1,-1,1,-1,1,-1,1,-1,1,-1,1,-1
1,-1,-1,-1,-1,1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,1,0,1,1,1,-1,1,-1,1,1,1,1,1,-1,-1,-1,1,-1,1
1,1,-1,-1,1,1,1,1,1,-1,1,-1,-1,1,-1,1,-1,1,-1,0,-1,1,-1,-1,-1,1,1,-1,-1,1,-1,-1,1,1,-1,1
1,-1,-1,1,1,-1,-1,1,-1,1,1,1,1,1,-1,-1,-1,1,-1,0,-1,1,-1,1,-1,-1,1,-1,1,-1,-1,-1,1,1,1,-1
-1,1,1,1,-1,1,-1,1,1,1,-1,1,-1,-1,-1,1,1,1,1,-1,0,-1,-1,1,1,-1,-1,1,1,1,1,-1,-1,-1,-1,-1
1,1,1,-1,-1,1,-1,1,1,1,-1,-1,-1,-1,1,1,1,-1,-1,1,-1,0,1,-1,1,-1,1,1,1,-1,-1,-1,-1,1,-1,1,1
1,1,1,-1,-1,1,-1,-1,1,1,-1,1,-1,1,-1,-1,1,-1,-1,-1,1,0,-1,-1,1,-1,-1,-1,1,1,1,-1,1,1,1,1
-1,1,-1,1,-1,1,-1,1,-1,-1,1,-1,-1,1,1,-1,-1,1,-1,1,-1,1,0,1,-1,1,-1,1,1,-1,1,1,1,-1,1,-1
-1,-1,1,-1,-1,-1,1,1,1,-1,-1,1,-1,1,1,1,-1,1,1,1,-1,1,-1,1,0,1,1,-1,-1,-1,-1,1,1,1,1,1
1,1,-1,1,1,1,-1,-1,-1,-1,1,1,-1,1,-1,1,1,1,1,-1,-1,1,-1,1,1,0,1,1,-1,-1,-1,-1,-1,1,1,-1,1
-1,1,-1,-1,-1,1,-1,-1,1,1,1,1,1,-1,-1,1,-1,1,-1,1,-1,1,-1,1,1,1,0,1,1,-1,-1,1,-1,-1,-1,1
1,-1,1,1,1,1,-1,-1,-1,1,-1,-1,1,1,1,1,-1,-1,1,1,1,-1,-1,-1,1,1,1,0,-1,-1,1,1,1,-1,1,-1
1,-1,1,1,-1,-1,-1,1,1,1,1,1,-1,-1,-1,1,1,1,-1,1,-1,-1,-1,-1,-1,1,1,1,0,-1,1,-1,-1,1,1
-1,-1,1,1,1,-1,1,-1,-1,1,1,-1,1,1,1,-1,1,-1,-1,-1,-1,-1,1,-1,1,1,1,1,-1,0,1,-1,-1,-1,1
1,1,1,1,-1,-1,-1,-1,1,-1,-1,1,1,1,-1,1,-1,1,-1,1,-1,-1,-1,1,1,1,1,1,-1,-1,0,-1,1,-1,1,1
-1,-1,-1,1,-1,1,1,1,1,1,1,-1,1,1,1,-1,1,-1,1,-1,-1,-1,-1,1,-1,1,1,1,1,-1,-1,0,1,-1,-1,-1
-1,1,-1,1,-1,1,-1,-1,-1,1,1,1,1,1,-1,1,1,1,-1,1,-1,-1,1,-1,1,1,1,1,-1,-1,1,-1,0,1,-1,1
-1,-1,-1,1,1,1,1,1,-1,1,-1,1,1,1,-1,1,1,-1,-1,1,-1,-1,1,-1,-1,1,-1,1,-1,1,-1,-1,1,0,-1,1
1,-1,-1,-1,1,1,-1,1,-1,-1,-1,1,-1,1,-1,1,1,1,1,1,-1,1,1,1,1,-1,-1,1,-1,1,-1,-1,1,-1,0,1
-1,-1,-1,1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,1,-1,-1,1,-1,1,1,1,-1,1,1,1,-1,1,1,1,-1,-1,1,0

Bibliography

[1] S. Buhrandt. "Magnetic monopoles in chiral magnets frustrated magnetism on the swedenborgite lattice". In: (Feb. 2015). URL: http://dspace.library.uu.nl/handle/1874/305850.

[2] John Harris, Jeffry L. Hirst, and Michael Mossinghoff. "Combinatorics and Graph Theory". In: *Springer-Verlag New York* (2008). URL: https://www.springer.com/gp/book/9780387797106.

[3] M. E. J. Newman and G. T. Barkema. "Monte Carlo Methods in Statistical Physics". In: *Oxford University Press, USA* (1999). URL: https://www.amazon.com/Monte-Carlo-Methods-Statistical-Physics/dp/0198517971.