Universidade de Aveiro Departamento de Física
2019

José Vítor Correia Rendeiro Vieira

Bipartição de redes de pequeno mundo

Bipartition of small-world networks

Universidade de Aveiro Departamento de Física
2019

José Vítor Correia Rendeiro Vieira

Bipartição de redes de pequeno mundo

Bipartition of small-world networks

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Física, realizada sob a orientação científica de António Luís Campos da Sousa Ferreira, Professor Associado do Departamento de Física da Universidade de Aveiro, e Manuel António dos Santos Barroso, Professor Auxiliar do Departamento de Física da Universidade de Aveiro.

o júri / the jury

presidente / president	**Vítor Hugo da Rosa Bonifácio**
	Professor Auxiliar da Universidade de Aveiro
vogais / examiners committee	**António Luís Campos da Sousa Ferreira**
	Professor Associado da Universidade de Aveiro
	João Manuel Viana Parente Lopes
	Investigador da Faculdade de Ciências da Universidade do Porto

agradecimentos / acknowledgements

Este trabalho não teria sido possível sem o apoio do meu orientador, o Prof. António Luís Ferreira, e o meu co-orientador, o Prof. Manuel Barroso. Fico-lhes particularmente grato pela sua incansável paciência e disponibilidade. Agradeço ao Bruno Faria e ao Rui Américo, por toda a sua ajuda ao longo deste processo. Não posso deixar de agradecer ao Prof. Vítor Bonifácio, cujos conselhos, sinceridade e duvidável gosto literário deixaram uma marca indelével.

Tenho que agradecer à FISUA, por me permitir a utilização abusiva das suas instalações todos estes anos, assim como os meus colegas de mesa, João de Oliveira, Francisco Teixeira, Diana Pereira e Pedro Manuel. A sua curiosidade e entusiasmo nunca deixam de me surpreender.

Agradeço os meus camaradas noctívagos, Gonçalo Nunes, Tiago Freitas, João Pedro, Sara Mantey, Augusto Santos, Telma Xavier, Guilherme Borges e Jaime Andrade. Fizeram-me compreender o valor que um sorriso e uma piada podem ter, mesmo nas situações mais difíceis. São as minhas funções de densidade de probabilidade favoritas.

Agradeço os meus amigos de longa data, Nuno Fortunato, Sara Sousa, Daniel Pereira, Ricardo Batalha, Diamantino Silva, Raul Costa, Jorge Laranjeira, João Duarte, João Pinheiro, Michael Russo e Miguel Cardoso. Espero vir a poder devolver o seu apoio, boa disposição e compreensão ao longo deste percurso, assim como as horas das suas vidas gastas à conta da minha ineptidude.

Em último lugar, agradeço as estrelas guias da minha vida, a minha mãe, a minha irmã e a minha avó. Sem o seu infindável amor e carinho, estaria para sempre à deriva.

Resumo

Neste estudo, investigámos a bipartição de redes de pequeno mundo. Utilizámos um modelo de Watts-Strogatz modificado para a geração de redes de pequeno mundo a partir de redes quadradas. Comparámos vários algoritmos de partição, tais como Monte Carlo com dinâmica Kawasaki e Simulated Annealing, Extremal Optimization e Multilevel K-way Based Partitioning.

Obtivemos os valores críticos do grau médio das redes na transição de percolação e os valores limite na transição de partição para redes ao longo do espetro entre uma rede quadrada e uma rede aleatória. Obtivemos os expoentes de variação do custo de partição mínimo em função da grau médio da rede na proximidade do valor limite de partição, assim como o expoente de escalonamente em função do tamanho da rede. Este é o primeiro trabalho a estudar este problema, tanto quanto sabemos.

Observou-se que as redes modificadas apresentam os mesmos expoente, independendemente do número de arestas modificadas, enquanto que a rede quadrada tem um comportamento distinto de todas as redes de pequeno mundo. Os valores dos expoentes para a rede aleatória estão em concordância com resultados prévios.

Abstract

We studied the bipartitioning of small-world networks. We generated small-world networks from a square lattice via a modified Watts-Strogatz algorithm. We compared several partitioning algorithms, such as Monte Carlo with Kawasaki dynamics and Simulated Annealing, Extremal Optimization and Multilevel K-way partitioning.

We obtained the critical percolation values of the mean degree and the threshold partition values for several networks in the continuum between a square lattice and a random network. We obtained the exponents of the minimum partition cost as a function of the mean degree in the vicinity of the bipartition threshold, as well as the exponent of finite size scaling. To the best of our knowledge, this is the first work to tackle this issue.

We observed that all small-world networks have the same exponents, different from those of the square lattice, regardless of the number of modified edges. The values for the exponents of the random network are in accordance with previous results.

Contents

List of Figures

List of Tables

List of Acronyms

EO	Extremal Optimization.
GBP	Graph Bipartitioning Problem.
GCC	Giant Connected Component.
MC	Monte Carlo.
MCMC	Markov Chain Monte Carlo.
MCS	Monte Carlo steps.
MF	Multilevel Framework.
MLK	Multilevel K-way Partitioning.
OLS	Ordinary Least Squares.
PT	Parallel Tempering.
SA	Simulated Annealing.
SOC	Self-Organized Criticality.
WSM	Watts-Strogatz Model.

List of Symbols

a Exponent of partition cost dependence on $p_{d,p} - p_d$.

B Partition cost.

B_i Number of edges connecting vertex i to vertices in different subset the partition.

c Exponent of partition cost size dependence.

G_i Number of edges connecting vertex i to vertices in same subset of the partition.

H Ising model energy.

H_0 Ground state of the Ising Hamiltonian.

$H_{0,i}$ Contribution of vertex i to ground state of the Ising Hamiltonian.

k_i Degree of vertex i.

$\overline{k}$ Mean degree of network.

$\overline{k}_c$ Percolation threshold mean degree.

$\overline{k}_p$ Partition threshold mean degree.

n Number of vertices of network.

n_∞ Fraction of vertices in GCC.

m Number of edges of network.

p Edge occupation probability.

p_c Percolation threshold edge occupation probability.

p_d Edge deletion probability.

$p_{d,c}$ Percolation threshold edge deletion probability.

$p_{d,p}$ Partition threshold edge deletion probability.

p_r Edge rewiring probability.

Chapter 1

Introduction

1.1 Motivation

The Graph Bipartitioning Problem (GBP) is easy to formulate, but hard to solve. Take n vertices, where n is an even number, and where a portion of the vertices pairs are connected by an edge. Then divide the vertices into two sets of equal number $n/2$ such that the number of edges connecting both sets, the partition cost B, is minimized. The global constraint that the division of vertices be equal makes the GBP one of the hardest problems in combinatorial optimization, since determining the exact solution would require a computational effort that grows faster than any power of n [1].

The GBP has been studied extensively, due to its impact in circuit design [2], computer vision [3], social network analysis [4] and load scheduling in parallel processing units [5].

In this work, we focus on the GBP in bond percolating small-world networks. In particular, we are interested in obtaining the exponents of the minimum partition cost at the graph bipartition threshold of these networks. To our knowledge, this is the first work to approach this issue.

The subject of bond percolation is also well trodden, due to its impact in network security [6] and disease spread, such as in the SI/R (susceptible-infected-recovered) models [7].

The small-world model was introduced by Watts and Strogatz [8] as a simple model of real world networks, such as social networks or networks of physical contact through which a disease could spread.

1.2 Outline

We give in Chapter 2 a brief introduction of network theory concepts, including the small-world model and percolation, and four methods for solving the GBP: Parallel Tempering, Simulated Annealing, Extremal Optimization and Multilevel K-way Partitioning. In Chapter 3 we present the methods we used and explain our choices. In Chapter 4 we present results that validate our approach as well as novel results for the exponents of the partition cost at the percolation threshold.

1.3 Related work

The subject of partitioning percolating networks has been approached from various perspectives.

The works we take most inspiration from are the works by Boettcher *et al* [9] and Percus [10].

Both works use Extremal Optimization to determine the partition cost for geometric and random networks.

The paper by Boettcher [9] compares Simulated Annealing and Extremal Optimization according to the quality of the partitions they create, their runtime and the scaling exponent, for random and geometric graphs and dilute ferromagnets with varying mean connectivity.

The paper by Percus [10] uses the concept of expander graphs to determine that random graph bisection is replica symmetric up to and beyond the partition threshold.

We evaluate the quality of our results against what is shown in those papers.

Chapter 2

Theoretical Concepts

2.1 Network theory concepts

A network G can be described as the collection of a set of vertices V and a set of edges E and is usually denoted by $G = (V, E)$ [11]. Edges can be directed or undirected. Directed edges *point* from one vertex to another, while undirected edges have no such property. We denote the cardinality of sets V and E, $|V|$ and $|E|$, by n and m, respectively.

The degree or connectivity k of a vertex of a network is the number of edges that are incident to the vertex.

An important quantity in the characterization of networks is the clustering coefficient C, given by the fraction of paths of length two in the network that are closed, i.e.,

$$C = \frac{3 \times \#\text{triangles}}{\#\text{connected triples}}. \tag{2.1}$$

One must also take care in the definition of distance in the context of network theory. We take the distance between vertices to be the "geodesic distance", given by the number of edges in the shortest path between the vertices.

Using this definition, the characteristic path length of the network is given by the median of the average path length

$$L = \text{med}\left\{\frac{\sum\limits_{j(\neq i)} d_{ij}}{n-1}\right\}, \tag{2.2}$$

where d_{ij} is the distance between vertices i and j.

We take the dimensionality of a network to be given by the Hausdorff dimension

$$d_H = \lim_{r\to\infty} \frac{\partial \log N(\leq r)}{\partial \log r}, \tag{2.3}$$

where $N(\leq r)$ is the cumulative distribution of the number of vertices within a distance r of another vertex, i.e.,

$$N(\leq r) = \sum_i N(\leq r, i), \tag{2.4}$$

where $N(\leq r, i)$ is the number of vertices within distance r of vertex i.

A matching M in a graph G is a set of edges in which no two edges share a common vertex. A maximal matching is a matching of G with the property that if any edge not in M is added to M it is no longer a matching. This concept is the cornerstone of one our chosen partition algorithms, Multilevel K-way Partitioning.

2.2 Random networks

A random network is, in general, a network where some specific set of parameters take fixed values, but the network is random in others. One of the simplest example of this type of graph is where we fix only the number of vertices n and the number of edges m. This model is known as $G(n, m)$ [11].

Another straightforward and well studied model for the creation of such a network is the $G(n, p)$ model, also known as the Erdős-Rényi model. In $G(n, p)$ we fix not the number but the probability of edges. We again have n vertices, but we place an edge between each distinct pair with probability $0 < p < 1$. To be more specific, $G(n, p)$ is an ensemble of networks with n vertices in which each simple graph G (a graph with no loops or multiple edges) appears with probability

$$P(G) = p^m (1-p)^{\binom{n}{2}-m}, \tag{2.5}$$

where m is the number of edges in the graph and $\binom{n}{2}$ is the number of all possible assignments of n vertices into pairs.

One of the most important quantities in this model, one which we will use at length in the following work, is the mean degree $\overline{k}$. The total probability of drawing a graph with m edges from the ensemble given by Eq. (2.5) is given by

$$P(m) = \binom{\binom{n}{2}}{m} p^m (1-p)^{\binom{n}{2}-m}. \tag{2.6}$$

.

The mean value of m is then given by

$$\langle m \rangle = \sum_{m=0}^{\binom{n}{2}} mP(m) = \binom{n}{2} p. \tag{2.7}$$

It follows that $\overline{k}$ is given by

$$\overline{k} = \frac{2\langle m \rangle}{n} = \frac{2}{n}\binom{n}{2} p = (n-1)p. \tag{2.8}$$

This relation shows that to produce a graph with average degree $\overline{k}$, independent of n, we should consider a probability p that decreases with n according to

$$p = \frac{\overline{k}}{n-1}. \tag{2.9}$$

2.3 Small-world networks

Real world networks present something known as *small-world effect*, the observation that the largest distance in the network is small. Specifically, a small-world network is defined as a network where the characteristic path length L grows proportionally to the logarithm of the number of vertices n in the network, i.e.,

$$L \propto \log(n). \tag{2.10}$$

Watts-Strogatz model

The Watts-Strogatz Model (WSM) is an attempt to model real world networks, which simultaneously display high clustering coefficients and short average path lengths [8].

The original model constructs a simple small-world network by starting with an underlying network, going through each of its edges in turn and, with some rewiring probability p_r, connecting one of its end vertices with a randomly chosen vertex of the network.

The parameter p_r in this model interpolates between the underlying network and the random graph. When $p_r = 0$, no edges are rewired and we retain the original graph. When $p_r = 1$, all edges are rewired to random positions and we have a random graph. For intermediate values, we get networks that lie in between. Notice that for $p_r = 0$, this model shows clustering but no small-world effect, while for $p_r = 1$ it shows the reverse. As a result, there is a substantial range of intermediate values for which the model shows both effects simultaneously, as expected.

In our modification of this model, besides connecting one of the end vertices of the chosen edge to a randomly chosen one, the rewiring step also deletes the chosen

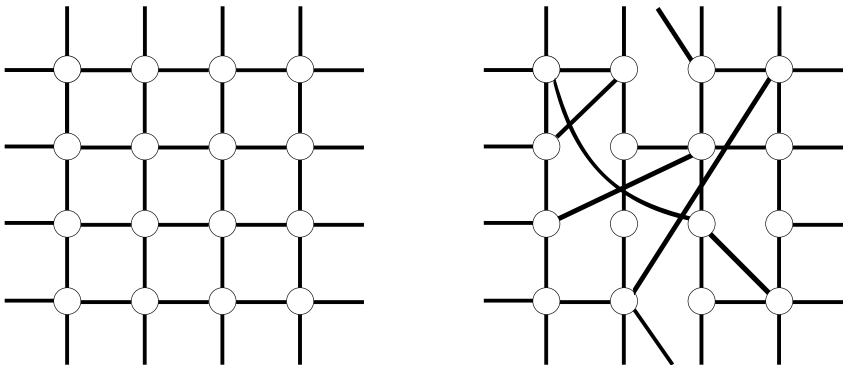

(a) $p_r = 0$ (b) $p_r = 0.3$

Figure 2.1: Random rewiring procedure for interpolating between a square lattice and a random network. We start with a lattice of 16 vertices, each connected to its 4 closest neighbours with periodic boundary conditions.

edge, keeping m constant. We also introduce a second parameter, the edge deletion probability p_d, which controls the mean connectivity of the network. The relationship between these two is given by

$$\overline{k} = \overline{k}_0(1 - p_d), \tag{2.11}$$

where $\overline{k}_0$ is the mean connectivity of the underlying lattice.

In this work we start with a square lattice of n vertices, connecting nearest neighbours vertices on the lattice, going through each of the edges in turn and, with some probability p_r, rewiring that edge by replacing one of the randomly chosen end vertices with a randomly chosen vertex and deleting the chosen edge. After this step, we again go through each of the edges and, with some probability p_d, delete it.

In this context, we have

$$\overline{k} = 4(1 - p_d). \tag{2.12}$$

2.4 Percolation and the Giant Connected Component

Percolation is the process by which networks are damaged by the deletion of vertices or the deletion of edges while keeping the number of vertices constant [11].

This process results in the creation of components, subgraphs in which any two vertices are connected to each other, either directly or indirectly through other vertices belonging to the component, and which is connected to no additional vertices in the network. A network component whose size grows with n is called a *giant component*, or, in the context of percolating networks, a giant connected component (GCC).

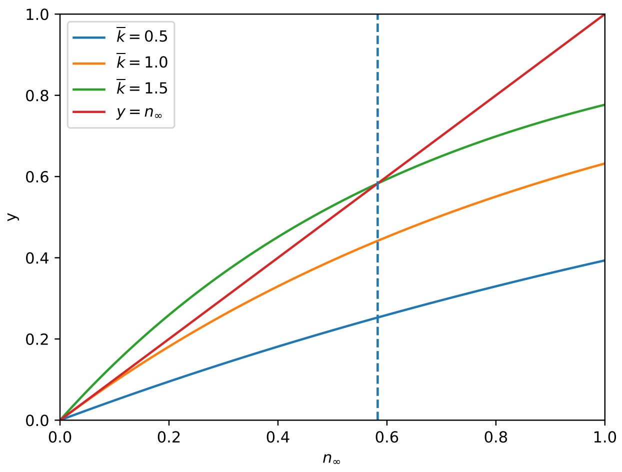


Figure 2.2: Graphical solution for the size of the GCC in random networks, with y given by the right hand side of Eq. (2.13). For the bottom curve there is only one intersection, at $n_\infty = 0$, and there is no GCC. For the top curve there is a solution at $n_\infty = 0.583$. The middle curve is the threshold between those two regimes, which occurs at $\overline{k} = 1$.

Random graphs

In a random graph with mean degree $\overline{k}$ the fraction of vertices belonging to the GCC, denoted by n_∞, is given asymptotically almost surely (a.a.s.) by

$$n_\infty = 1 - e^{-\overline{k}n_\infty}, \tag{2.13}$$

as seen in Ref. [11].

We can find the critical value of $\overline{k}$ by plotting the right hand size of Eq. (2.13) against n_∞ as seen in Fig. 2.2. The transition between regimes falls at the point where the gradient of the curves and the gradient of the dashed line match at $n_\infty = 0$. That is, the transition takes place when

$$\frac{\mathrm{d}}{\mathrm{d}n_\infty}(1 - e^{-\overline{k}n_\infty}) = 1, \tag{2.14}$$

which gives us

$$\overline{k}e^{-\overline{k}n_\infty} = 1. \tag{2.15}$$

Setting $n_\infty = 0$, we get $\overline{k}_c = 1$ as the percolation threshold for a random network. In the context of bond percolation, it is usual to express this threshold as a critical occupation probability p_c, obtained from Eq. (2.9)

$$p_c = \frac{\overline{k}_c}{n-1} = \frac{1}{n-1}. \tag{2.16}$$

This corresponds to values $p_r = 1$ and $p_d = p_{d,c} = 0.75$ in our modified WSM, where $p_{d,c}$ was obtained using Eq. (2.12).

Only clusters of finite size exist above the percolation threshold (since p_d is a deletion probability), with a characteristic length $s_{\max}$. The probability per vertex that a randomly chosen vertex belongs to a cluster of size s, denoted by n_s, obeys the forms

$$n_s \propto s^{-\tau} f(s/s_{\max}) \tag{2.17}$$

and

$$s_{\max} \propto |p - p_c|^{-1/\sigma}. \tag{2.18}$$

Note that n_s is also the average number of finite clusters of size s that exist per vertex in the network.

The values for the critical exponents in the random network are $\tau = 5/2$ and $\sigma = 1/2$ [12].

Square Lattice

The critical probability of bond percolation on the square lattice is $p_c = 0.5$ [13], which results in a mean connectivity $\overline{k}_c = 2$. This corresponds to values $p_r = 0$ and $p_d = p_{d,c} = 0.5$ in our modified WSM.

The values for the critical exponents in the square lattice are $\tau = 187/91$ and $\sigma = 36/91$ [12].

Small-world networks

Small-world networks have two competing length scales, the percolation correlation length ξ and the crossover length ζ [14, 15], given by

$$\xi \propto |p - p_c|^{-\nu} \tag{2.19}$$

and

$$\zeta \propto \left(\frac{1}{2p_r pkd}\right)^{1/d}, \tag{2.20}$$

where k is the number of connected neighbours in each direction, d is the dimension of the underlying network and p is the probability for a given edge to be present.

The crossover length is the linear dimension on the volume on the underlying lattice which contains the end of one shortcut on average.

In Ref. [14] the small-world model with long-range shortcuts was studied and it was found that, in the limit of infinite size, any observable must satisfy a scaling relation of the form

$$Q \approx \xi^{d\alpha} \zeta^{-d\beta} f(\xi/\zeta), \tag{2.21}$$

where α and β are scaling exponents and $f(x)$ is an universal scaling function.

Making use of equation (2.20) and the fact the ξ^d is the number of sites $\langle s_0 \rangle$ in a region of length ξ on the underlying network [12], we obtain

$$Q \approx \langle s_0 \rangle^{\alpha} (p_r pkd)^{\beta} F(2p_r pkd \langle s_0 \rangle), \tag{2.22}$$

where $F(x)$ is another universal scaling function.

Taking $Q = \langle s \rangle$, being $\langle s \rangle$ the average size of a cluster to which a randomly chosen site belongs, which must take the square lattice value $\langle s_0 \rangle$ for $p_r = 0$, we immediately obtain $\alpha = 1$ and $\beta = 0$, and

$$\frac{\langle s \rangle}{\langle s_0 \rangle} = F(2p_r pkd \langle s_0 \rangle). \tag{2.23}$$

It was shown in [14] that the scaling function takes the form

$$F(x) = \frac{1}{1 - x}. \tag{2.24}$$

Thus, the percolation transition occurs at $x = 1$, when the two length scales are equal $\xi = \zeta$ and the average finite cluster size diverges.

A possible interpretation of this result is that the effect of the shortcuts is to connect clusters of average size $\langle s_0 \rangle$ in the starting underlying network. When there is at least one shortcut on average connecting one of these clusters to another one, i.e., when $\xi = \zeta$, there occurs a percolation transition.

Moore and Newman obtained the values $\tau = 5/2$ and $\sigma = 1/2$ for the one-dimensional small-world network in [16], using arguments that could be easily extrapolated to the two-dimensional model that we use. These values are the same as in the mean field case, which seems reasonable since the dimensionality of both models in the limit of large systems is infinite.

In light of this, we expect random networks and small-world networks to exhibit the same critical percolation behaviour.

2.5 Graph Bipartitioning

The GBP has an interesting phase structure. The minimum partition cost, defined as

$$B_{\min} = \sum_i \frac{B_i}{2}, \tag{2.25}$$

where B_i is the number of edges connecting vertex i to vertices of the other set in an optimal partitioning, becomes null for values of $\overline{k} < \overline{k}_p$.

Taking this into account, it is evident that $\overline{k}_p$ must satisfy the condition

$$n_\infty \left(\overline{k}_p\right) = 1/2, \tag{2.26}$$

by considering that at this point there are two groups of size $n/2$, the GCC and the finite clusters outside of it, that have no connecting edges. For values of n_∞ less than 1/2, one can always group the GCC with isolated finite clusters, creating a partition with no edges between the two sets of vertices, resulting in a null partition cost.

The minimum partition cost undergoes a continuous transition at this point, i.e., it obeys a power law of the form

$$B_{\min} = C(n)(p_{d,p} - p_d)^a, \tag{2.27}$$

where $C(n)$ is the finite size scaling coefficient, p_d refers to the edge deletion probability of the modified WSM and $p_{d,p}$ can be obtained by rearranging Eq. (2.12)

$$p_{d,p} = 1 - \frac{\overline{k}_p}{4}, \tag{2.28}$$

.

Random graphs

Inputting $n_\infty = 1/2$ in Eq. (2.13), we obtain

$$\frac{1}{2} = 1 - e^{-\overline{k}_p/2}, \tag{2.29}$$

which gives us $\overline{k}_p = 2\log 2$ for a random network. This corresponds to the values $p_r = 1$ and $p_d = p_{d,p} = 1 - \frac{2\log 2}{4} = 0.6534$ in our modified WSM model.

For a given $\overline{k}_{\mathrm{RSB}} > \overline{k}_p$ the solution space undergoes a structural transition that corresponds to replica symmetry breaking [17]. For $\overline{k}_{\mathrm{RSB}} > \overline{k} > \overline{k}_p$, any two partitionings of the graph are connected by a path of adjacent solutions. The solution space can be though of as a single cluster. But when $\overline{k} > \overline{k}_{\mathrm{RSB}}$, the cluster fragments into multiple non-adjacent clusters [10], and finding optimal solutions by local optimization becomes computationally much harder.

Square lattice

There is no analogous equation to (2.13) in the context of the square lattice, and as such we cannot obtain an exact value of $p_{d,p}$.

We can, however, approximate it, given that we are sufficiently close to the percolation threshold, making use of the equation

$$n_\infty \propto |p_d - p_{d,c}|^\beta, \tag{2.30}$$

where β has the value 5/36 [12].

Inputting $n_\infty = 1/2$, we get

$$|p_{d,p} - p_{d,c}| \approx \left(\frac{1}{2}\right)^{1/\beta}. \tag{2.31}$$

.

The right hand side term is of the order 10^{-3}, which means that the value of $p_{d,p}$ is expected to be close to that of $p_{d,c}$.

2.6 Bipartitioning Algorithms

The GBP is a NP-hard problem [1], and as such, there exists no exact method to solve it which runs in useful time. To see why this is, consider that there are $\binom{n}{n/2}$ ways of choosing a subset of size $n/2$. For a graph with size $n = 100$ this value is of the order of 10^{29}, and as such, an exhaustive search procedure is completely out of the question. We must then search for approximate solutions, which can be found much more quickly.

2.6.1 Monte Carlo methods

We first make use of the Ising Model, which has the Hamiltonian

$$H = -\sum_{i,j} J_{ij} s_i s_j, \tag{2.32}$$

where J_{ij} is a coupling factor, which, for our purposes, can be

$$J_{ij} = \begin{cases} 1 & \text{if there is an edge between } i \text{ and } j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.33}$$

and s_i is the Ising spin of vertex i, which can take two values $s_i = \pm 1$. Notice that H can be written as

$$H = \sum_i H_i, \tag{2.34}$$

where H_i is given by

$$H_i = -s_i \sum_j J_{ij} s_j. \tag{2.35}$$

An important quantity of this model is the total magnetization M, given by

$$M = \sum_i s_i. \tag{2.36}$$

In this context, a solution for the GBP is a state of minimum energy of the model with null magnetization, where we can associate the spin $+1$ to a vertex belonging to one of the sets of the partition and -1 belonging to the other set. The minimum partition cost is then given by

$$B_{\text{min}} = \sum_i \frac{B_i}{2} = \sum_i \frac{k_i + H_{0,i}}{4} = \frac{m + H_0}{2}, \tag{2.37}$$

where k_i is the connectivity of vertex i and $H_{0,i}$ is the contribution of vertex i to the state of minimum energy. We make use of the equalities $B_i = k_i - G_i$ and $B_i = H_{0,i} + G_i$, where G_i is the number of unique "good edges", connecting i to vertices of the same set. The optimal partition should thus correspond to states of minimum energy in the system.

Monte Carlo (MC) schemes almost always rely on Markov processes as the generating engine for the set of used states [18]. A Markov process is a stochastic process which, given a system in one state μ, generates a new state of that system ν. The probability of generating state μ from ν is called the *transition probability*, denoted by $P(\mu \to \nu)$.

Spin exchange

As mentioned in the previous section, the GBP amounts to a system of null magnetization. As such, we must make use of a magnetization conserving spin dynamics, such as the Kawasaki spin exchange [19]. In this method, each update simultaneously reverses the state of neighbouring antiparallel spins, i.e., *exchanges* their value. To decide whether to exchange a randomly chosen pair of spins, we calculate the change in energy and accept or reject the move based on the Metropolis transition probability (also known as Metropolis acceptance probability)

$$P(\mu \to \nu) = \begin{cases} e^{-\beta \Delta H} & \text{if } \Delta H > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{2.38}$$

where $\Delta H = H(\nu) - H(\mu)$.

More succinctly,

$$P(\mu \to \nu) = \min\{1, e^{-\beta \Delta H}\}. \tag{2.39}$$

This algorithm is ergodic, in that it can reach any configuration of the solution space in a finite number of steps, and satisfies the condition of detailed balance [18].

However, since we are not interested in the way in which we reach equilibrium but only in the final result, we have no need to limit ourselves to spin exchanges between neighbouring sites. We can reach equilibrium much more quickly by using non local MC moves. A simple non local move is to exchange a random pair of antiparallel spins, based on the Metropolis acceptance probability. One can show, by the same line of arguments used for the Kawasaki spin exchange, that these non local dynamics are ergodic and satisfy the detailed balance condition.

Parallel Tempering

Parallel Tempering (or replica simulation) is a simulation method that tackles the problem of most Markov Chain Monte Carlo (MCMC) simulations, by which they are trapped in local minima and are not able to reach a global minimum.

This method uses several replicas of the system at different temperatures, usually increasing from the target temperature of the test case [20]. Each replica is simulated simultaneously and independently for a few Monte Carlo steps (MCS). Each replica is in contact with its own heat bath, at different temperatures, resulting in the partition function for the extended ensemble

$$\mathcal{Z} = \mathrm{Tr}_{\{X\}} \exp\left(-\sum_{i=1}^{R} \beta_i \mathcal{H}(X_i)\right) = \prod_{i=1}^{R} Z(\beta_i), \tag{2.40}$$

where $\{X\} = \{X_1, X_2, \dots X_R\}$ is a state of the extended ensemble, β_i is the temperature of a given replica and $Z(\beta_i)$ is the partition function of the same replica.

Replicas at higher temperatures can reach a larger portion of the configuration space. The reachable spaces by neighbour temperatures overlap, and as such, we can create a new update for the system composed of the two replicas at β_i and β_j.

In order to ensure that the system remains in equilibrium when two replicas are exchanged, we must impose the detailed balance condition

$$\begin{aligned} &P(\dots; X, \beta_i; \dots; X', \beta_j; \dots) W(X', \beta_i; X, \beta_j | X, \beta_i; X', \beta_j) \\ &= P(\dots; X', \beta_i; \dots; X, \beta_j; \dots) W(X, \beta_i; X', \beta_j | X', \beta_i; X, \beta_j), \end{aligned} \tag{2.41}$$

where $P(\{X, \beta\})$ is given by

$$P(\{X, \beta\}) = \prod_{i}^{R} Z^{-1}(\beta_i) \exp(-\beta_i \mathcal{H}(X_i)) \tag{2.42}$$

and $W(X', \beta_i; X, \beta_j | X, \beta_i; X', \beta_j)$ is the probability of exchanging replicas i and j.

From (2.42) we obtain

$$\frac{W(X', \beta_i; X, \beta_j | X, \beta_i; X', \beta_j)}{W(X, \beta_i; X', \beta_j | X', \beta_i; X, \beta_j)} = e^{(\beta_j - \beta_i)(\mathcal{H}(X') - \mathcal{H}(X))}, \tag{2.43}$$

and the replica-exchange probability can then be expressed as

$$W(X', \beta_i; X, \beta_j | X, \beta_i; X', \beta_j) = \min\left(1, e^{(\beta_j - \beta_i)(\mathcal{H}(X') - \mathcal{H}(X))}\right). \tag{2.44}$$

It is usual to set $j = i + 1$ such that we limit the exchange to neighbouring replicas, since the probability of exchange decreases with the distance between replicas.

Simulated Annealing

Simulated Annealing (SA) is based on the heuristics of the real world physics of annealing (slow cooling) of solids [21]. It is known that a hot material, such as a molten metal, will, if cooled sufficiently slowly to a low enough temperature, eventually find its ground state.

The SA algorithm consists of a discrete-time inhomogeneous MCMC with the Metropolis acceptance probability seen in Eq. (2.39), but with a time varying $\beta(t)$, corresponding to the system temperature at time t. SA makes use of a decreasing temperature cooling schedule.

As $T(t)$ decreases, the probability distribution is concentrated on the set of global minima [22]. To see why this is, consider a Markov chain at temperature T. Assuming that the Markov chain is irreducible and aperiodic, it is then also reversible, and its probability distribution is given by

$$\mathcal{F}(\mu) = \frac{1}{Z_T} \exp\left(\frac{-H(\mu)}{T}\right), \tag{2.45}$$

where μ is a given state of the system and Z_T is a normalization constant.

It is evident that as T approaches 0, values of minimum energy will be preferred.

A key feature of the SA algorithm is that it is guaranteed to find an optimal solution asymptotically [22]. To be more specific, SA converges if

$$\lim_{t\to\infty} T(t) = 0 \tag{2.46}$$

and

$$\sum_{t=1}^{\infty} e^{-d^* T(t)} = \infty, \tag{2.47}$$

where d^* is a measure of the difficulty of escaping from local minima and go from a suboptimal state to a state in the set of global minima. This behaviour must be taken into account when choosing a cooling schedule. In practical application, the computational effort to find the optimal solution may be very large.

A schematic view of the algorithm can be seen in Alg. 1.

Algorithm 1 Generic Simulated Annealing algorithm

```
Initialize configuration of S at will.
Define a temperature schedule.
while $|\Delta H_0| > \varepsilon \vee$ total number of iterations $< N_{\text{iter}}$ do
    Update temperature.
    for $i < n$ do
        Choose random pair of antiparallel spins.
        Exchange them according to Metropolis acceptance rule.
        If exchange is accepted, $H \leftarrow H + \Delta H$.
    end for
    if $H < H_0$ then
        $H_0 \leftarrow H$
    end if
end while
return $H_0$
```

Algorithm 2 Generic Extremal Optimization algorithm

```
Initialize configuration S at will.
$S_{\text{best}} \leftarrow S$
while total number of iterations $< N_{\text{iter}}$ do
    Evaluate $\lambda_i$ for each variable $s_i$ of the current solution $S$.
    Sort the variables $s_i$ into ranking $\pi$ based on their fitness $\lambda_i$.
    Choose the rank $k$ according to a suitable distribution probability
      so that the variable $s_j$ with $j = \pi(k)$ is selected.
    Choose $S'$ such that $s_k = {s'}_k, \forall k \neq j$ and $s_j \neq {s'}_j$.
    Accept $S \leftarrow S'$ unconditionally.
    if $\lambda(S) < \lambda(S_{\text{best}})$ then
        $S_{\text{best}} \leftarrow S$
    end if
end while
return $S_{\text{best}}$ and $\lambda(S_{\text{best}})$
```

2.6.2 Extremal Optimization and the Multilevel Framework

Extremal Optimization

Extremal Optimization (EO) is a simulation method based on the phenomenon of Self-Organized Criticality (SOC) first introduced by Per Bak and Kim Sneppen in their seminal paper [23]. Species in the Bak-Sneppen model are located on sites of a lattice, and each one has a fitness λ represented by a value between 0 and 1. At each step, one of the smallest values (representing one of the most poorly adapted species) is discarded and replaced by a new value drawn randomly from a flat probability distribution defined between $[0, 1]$. Without any interactions, all the fitnesses in the system would eventually approach 1. But interdependencies between species provide constraints for balancing the system's overall condition with that of its members: the change of fitness of one species impacts the fitness of an interrelated species. Therefore, at each update step, the Bak-Sneppen model replaces the fitness values on the sites neighbouring the smallest value with new random numbers as well (see Alg. 2).

No explicit definition is provided for the mechanism by which these neighbouring species are related. Yet, after a certain number of updates, the systems organizes itself into a highly correlated state. This state is what is known as SOC. In that state, almost all species have reached a fitness above a certain threshold. But these species only possess what is called self-punctuated equilibrium: since only one's weakened neighbour can undermine one's own fitness, the systems undergoes long periods of *stasis*, with a fitness above the threshold, interrupted by bouts of activity. This co-evolutionary activity cascades in a chain reaction, known as *avalanche*, through the system.

In this model, the high degree of adaptation of most species is obtained by the elimination of poorly adapted ones rather that by an "engineering" of better ones. While such dynamics may not result in an optimal solution as could be engineered under specific circumstances, it provides near-optimal results with a high degree of latency for a rapid adaptation response to changes in the resources that drive the system.

In this work, after computing the initial fitness λ for all vertices (playing the role of species) of the system, we then rank them in order of least to most fitness. In each step we swap two vertices according to the probability distribution $k^{-\tau}$, where k is the index after ranking, recalculate the fitness, and rank every vertex once again. After a sufficient number of iterations, the system should reach a state of equilibrium.

One of the great advantages of this algorithm over an algorithm such as SA is that EO can arguably escape local minima with less computational effort and without the need for fine-tuning algorithmic SA parameters such as the cooling schedule.

Multilevel Framework

One of the most common approaches to solve the GBP is the Multilevel Framework (MF), where a graph is approximated by a sequence of smaller graphs [24]. The main idea of this framework is to reduce the size of the graph, find a partition for the coarsest graph and project it back to the original one. The most evident advantage of the MF is that size of the graph can be reduced without losing its topological properties and thus the problem can be solved much faster.

In this work, we used Multilevel K-way Partitioning (MLK). This algorithm is divided in three phases: coarsening, initial partitioning and uncoarsening.

Coarsening phase During the coarsening phase, a sequence of smaller graphs $G_i = (V_i, E_i)$ is constructed from the original graph $G_0 = (V_0, E_0)$, such that $n_i > n_{i+1}$ [25]. Graph G_{i+1} is obtained from G_i by finding a maximal matching $M_i \subseteq E_i$ of G_i and collapsing the vertices that are incident on the edge of the matching. When vertices $u, v \in V_i$ are collapsed to form vertex $w \in V_{i+1}$, the weight of vertex w is set to the sum of the weights of vertices v and u, and the edges incident on w are set to the union of edges incident on u and v minus the edge (v, u). Thus, during successive coarsening levels, the weight of both vertices and edges increases. If no initial weights are assigned, we assume all weights equal to 1.

Vertices which are not incident on each edge of the matching are copied to G_{i+1}. Since the goal of this collapse of edges is to decrease the size of the graph G_i, the matching must be maximal.

Maximal matchings can be computed in a number of ways. MLK uses a modified Heavy Edge Matching, as seen in Ref. [24].

Partitioning phase The second phase of the MLK is to compute a k-partitioning of the coarsest graph G_m such that each partition contains roughly n_0/k vertex weight of the original graph. Any sufficiently accurate partitioning method can be used at this step, since the graph is small.

Uncoarsening phase Finally, during the uncoarsening phase, the partitioning of the coarsest graph is projected back to the original graph by going through the graphs $G_m, G_{m-1}, \ldots, G_1$. Since each vertex $u \in V_{i+1}$ contains a distinct subset U of vertices of V_i, the projection of the partitioning from G_{i+1} to G_i is achieved by simply assigning the vertices in U to the same partition in G_i to which vertex u belongs in G_{i+1}. After the partitioning is projected, a local refinement algorithm must be used, due to the now larger number of degrees of freedom, which allows for further minedge minimization. The most commonly used algorithm is the Kernighan-Lin heuristic.

Kernighan-Lin heuristic The Kernighan-Lin heuristic seeks to compute partitions by minimizing a cost function T [26]. In essence, we start with any arbitrary partition A, B of a set S of n points, with an associated cost matrix $C = (c_{ij})$ and cost function $T = \sum_{A \otimes B} c_{ab}$. By a series of interchanges of subsets of A and B, we seek to decrease the initial cost, until no further improvement is possible. At this point, the partition A', B' is locally minimum.

Of course, we cannot consider all possible choices of subsets of A and B, and we must make an approximation. To do this, we define for each $a \in A$ an external cost

$$E_a = \sum_{y \in B} c_{ay} \tag{2.48}$$

and an internal cost

$$I_a = \sum_{x \in A} c_{ax}. \tag{2.49}$$

In similar fashion, we define E_b and I_b for each $b \in B$. It can be shown that the reduction in the cost from interchanging a and b is given by

$$\Delta T = D_a + D_b - 2c_{ab}, \tag{2.50}$$

where $D_x = E_x - I_x$.

The algorithm begins by computing D values for all elements of S. We then choose $a, b \in A, B$ such that

$$g_i = D_{a_i} + D_{b_i} - 2c_{a_i,b_i} \tag{2.51}$$

is maximum. We then recalculate D for the elements of $A - \{a_i\}$ and $B - \{b_i\}$

$$D_{x'} = D_x + 2c_{xa_i} - 2c_{xb_i}, x \in A - \{a_i\} \tag{2.52}$$

$$D_{y'} = D_y + 2c_{yb_i} - 2c_{ya_i}, y \in B - \{b_i\} \tag{2.53}$$

and choose another pair of vertices to interchange such that the cost of the move is maximum. We continue in such a fashion until all vertices have been exhausted, identifying the pairs $(a_i, b_i) \ldots (a_n, b_n)$ and the corresponding maximum gains $g_i \ldots g_n$.

We now choose a cutoff l that maximizes the partial sum $G_{\text{sum}} = \sum_{i=1}^{l} g_i$. If $G_{\text{sum}} > 0$, a reduction in cost can be made by interchanging $\{a_1 \ldots a_l\}$ and $\{b_1 \ldots a_l\}$. After this is done, the resulting partition is treated as the initial partition, and the process is repeated. If $G_{\text{sum}} = 0$, we have reached an optimal partition.

This method is quite exhaustive, and, as such, not suitable for the partition of large graphs. But, for the purposes of local refinement, it results in optimal partitions with a great degree of certainty.

Chapter 3

Methods

3.1 General considerations

We were interested in obtaining a description of the behaviour of the minimum partition cost in the vicinity of the bipartition threshold point for a wide range of rewiring probabilities.

The runtime and memory usage of our implementation of PT were too large for the number of networks we wanted to analyse. So, even though its results should be satisfactory, we chose other bipartition algorithms, using PT instead to establish baseline values for the random network, i.e, $p_r = 1$, in conjunction with the results obtained by Percus *et al.*

We then tested SA. We discovered that SA is quite susceptible to the initialization scheme, and that the results if offers in the vicinity of the bipartition threshold are quite distant from the expected theoretical values for $p_r = 0$ and $p_r = 1$. This makes it a unsatisfactory candidate for the task at hand.

We then tested EO and MLK, expecting a more satisfactory performance at the threshold point.

We compared initialization schemes, according to quality of produced partitions and runtime, settling on the scheme described in Section 3.4.

3.2 Network generation

We used a modified WSM, with the addition of a probability of edge deletion p_d to the existing rewiring probability, which we denote by p_r.

We start with a square lattice with periodic boundary conditions. In this work we use square lattices with $\sqrt{n}$ values of 16, 32, 64 and 128. We also use square lattices with $\sqrt{n}$ equal to 256 and 512 to validate some of our assumptions regarding the prevalence of finite size effects in the graph bipartitioning threshold.

The network generating algorithm is divided into a rewiring and a deletion phase. The rewiring phase begins by choosing $n_r = p_r \times m$ random unique edges to be unconditionally rewired. From each of these edges we choose a random end vertex which will then be connected to a random vertex of the network, followed by the deletion of the chosen edge. This completes the rewiring step. The deletion step goes in a similar fashion, choosing $n_d = p_d \times m$ random unique edges which are then unconditionally deleted.

At any point of the process we only allow any given vertex to be connected to a maximum $k_{\max}$ of 20 other vertices, in order to lessen the memory requirements of the algorithm.

3.3 Threshold value determination

In our model of small-world networks, a high value of p_d results in clusters of small size, while a low value of p_d results in the appearance of a GCC. As the clusters of small size are absorbed into the GCC, the mean cluster size of finite clusters should increase and then decrease again. We can conclude then that the percolation threshold occurs at the maximum value of the mean cluster size, given that we ignore the contribution of the GCC.

The probability that a randomly chosen vertex not belonging to the GCC belongs to a cluster of size s is given by

$$\omega_s = \frac{s n_s}{\sum_s s n_s}, \tag{3.1}$$

where n_s is the average number of finite clusters of size s per vertex.

It follows that the mean finite cluster size is given by

$$\chi = \sum_s s w_s = \frac{\sum_s s^2 n_s}{\sum_s s n_s}. \tag{3.2}$$

This quantity is the susceptibility for the percolation problem that diverges at the percolation threshold.

The percolation threshold of a finite system of size n can be estimated by

$$p_{d,c}(n) = \operatorname*{argmax}_{p_d} \chi. \tag{3.3}$$

The partitioning threshold is given by the probability $p_{d,p}$ such that

$$n_\infty(p_{d,p}) = 1/2. \tag{3.4}$$

We have simply to find the value for which this condition is met, and we obtain $p_{d,p}$.

In this work, we calculated these values using 1001 values of p_d for each of 101 values of p_r, for a combined total of 101101 networks for each size.

3.4 Network initialization

As discussed in Section 3.1, the minimum partition cost for values of $n_\infty < 1/2$ should be null. Taking this into account, we employ two initialization schemes according to whether n_∞ is larger or smaller than 1/2.

Take wihout loss of generality a partition given by two sets $\uparrow$ and $\downarrow$. Then, our initialization schemes can be summarized as follows:

$\mathbf{n_\infty > 1/2}$

Place all finite clusters in the $\uparrow$ set and the GCC in the $\downarrow$ set. Select the least connected $\downarrow$ vertexs within the GCC and place them in the $\uparrow$ set. Update the connectivity between vertexs of the $\uparrow$ and $\downarrow$ sets within the GCC. Iterate until null magnetization is reached.

If the GCC takes up more than half of the vertices in the network, edges connecting the two set of the optimal partition must occur inside the GCC. This scheme focuses on interfaces within the GCC, resulting in a better final result, with shorter runtimes.

$\mathbf{n_\infty < 1/2}$

Place finite clusters in the $\uparrow$ set and the GCC in the $\downarrow$ set. Choose finite cluster of size N_c such that $n/2 - |\downarrow| - N_c$ is minimized and place it in the $\downarrow$ set. Iterate until null magnetization is reached.

3.5 Partitioning algorithms

For the determination of the behaviour of the minimum partition cost in the vicinity of the bipartition threshold point, we generated 441 networks (21 values of p_d for each of 21 values of p_r) in the case of the MLK algorithm and 111 networks (11 values of p_d for each of 11 values of p_r) in the case of the EO algorithm. This difference in the number of simulated networks, on account of the longer runtime of EO, should not greatly affect the quality of our results. We chose values of p_d such that $0.6 \times p_{d,c} \geq p_d \geq 0.4 \times p_{d,c}$ for each rewiring probability, as seen in Ref. [10]. This gives us a good coverage of the overall solution space, without sacrificing performance.

3.5.1 Simulated Annealing

We used a temperature schedule of $n_{\text{steps}} = 10^7$ from $\beta_{\text{high}} = 0.01$ to $\beta_{\text{low}} = 1000$, given by

$$\beta_i = \beta_{\text{high}} \exp\left(\frac{i-1}{\gamma}\right), \tag{3.5}$$

with γ given by

$$\gamma = \frac{n_{\text{steps}} - 1}{\log\left(\beta_{\text{low}}/\beta_{\text{high}}\right)}. \tag{3.6}$$

We used the non local version of the Kawasaki method.

In order to lessen the runtime of the algorithm, we keep updated lists of which sites belong to which set. We then randomly draw a site from each set, which ensures that our selected pair is always antiparallel.

3.5.2 Extremal Optimization

We considered each vertex of a graph as an individual species with its own fitness. We assigned to each vertex i the fitness

$$\lambda_i = \begin{cases} \dfrac{G_i}{G_i + B_i} & \text{if } k_i > 0, \\ 1 & \text{otherwise,} \end{cases} \tag{3.7}$$

as seen in Ref. [9]. Notice that λ can only take values between 0 and 1.

The first case of the previous equation can be written as

$$\lambda_i = \frac{1}{2} + \frac{s_i \sum\limits_{j=1}^{k_i} s_j}{2k_i}, \tag{3.8}$$

by making use of the equalities $B_i + G_i = k_i$ and $G_i - B_i = s_i \sum_{j=1}^{k_i} s_j$.

The partition cost can then be given by

$$b = \sum_i \frac{k_i(1-\lambda_i)}{2n}. \tag{3.9}$$

In each update step, only two spins i and j and are exchanged. This gives us

$$\Delta b = -\frac{s_i \sum\limits_{l=1}^{k_i} s_l + s_j \sum\limits_{l=1}^{k_j} s_l}{2n}. \tag{3.10}$$

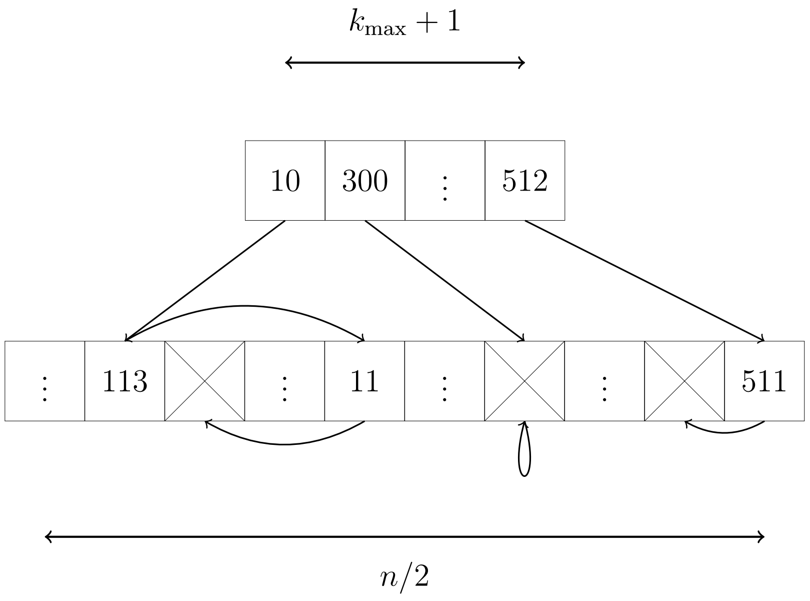


Figure 3.1: Schematic representation of our doubly linked list.

When the chosen spins are neighbours, the previous expression must be corrected, resulting in

$$\Delta b_{\text{neigh}} = \Delta b - \frac{s_i s_j}{2}. \tag{3.11}$$

This means that we need to explicitly calculate only the initial fitness. In every update step, the variation of the partition cost depends only on the value of the spin of the vertices which will be exchanged and the spins of their neighbours. The result is an algorithm of reduced computational effort and more efficient memory usage.

Notice that the fitness can only take a certain number of discrete values, and as such it can be stored in a doubly linked list (Fig. 3.1). This data structure allows for insertion and deletion in $\mathcal{O}(1)$, greatly increasing the algorithm's performance.

We choose a bucket size $n_b = \frac{1}{k_{\max}}$. This bucket size does not allow for a perfect ordering of the fitnesses (some of the values will be of the form $1/p$, p being some prime, which are not divisible), but this algorithm will nonetheless converge to a suitable value given enough iterations. A constant bucket size allows for a much shorter runtime, making up for the necessity of more iterations, and makes for simpler code.

As in our SA implementation, we keep a tally of which vertices belong to which set. In this case, this means keeping two bucket lists, one for each set.

We draw a bucket $1 \leq c \leq k_{\max}+1$ from each list, with probability

$$\mathcal{F}(c) = \mathcal{F}_{\text{sum}}\left(\sum_{l\,=\,1}^{c} n_l\right) - \mathcal{F}_{\text{sum}}\left(\sum_{l\,=\,1}^{c-1} n_l\right), \tag{3.12}$$

where n_l is the number of vertices in bucket l and $\mathcal{F}_{\text{sum}}(k)$ is given by

$$\mathcal{F}_{\text{sum}}(k) = \sum_{j=1}^{k} \mathcal{F}(j), \tag{3.13}$$

with $\mathcal{F}(j)$ given by

$$\mathcal{F}(j) = \frac{j^{-\tau}}{\sum\limits_{j=1}^{n/2} j^{-\tau}}. \tag{3.14}$$

We then randomly draw a vertex from each bucket. These vertices will be unconditionally swapped. This process is repeated in every step.

We ran the algorithm for $10n^2$ steps, which we found to be sufficient for convergence in our test cases. We used $\tau = 1.45$, as seen in Refs. [9, 10].

3.5.3 Multilevel K-way partitioning

We used the Python and Fortran implementation of the METIS software suite [24] with default parameters. The default partitioning algorithm for the MLK implementation in METIS is the multilevel bisection algorithm described in Ref. [24].

3.6 Statistics

The data generated with these algorithms was randomly chosen from the ensemble of possible realizations of the network. This choice was made by using different seeds for the random number generators in Fortran (Linear feedback shift register [27]) and Python (Mersenne Twister [28]), comprising a total of 100 realizations (samples) for each network. Each realization was given equal weight in the ensemble average.

Chapter 4

Results

4.1 Threshold values

In Fig. 4.1 we plot the phase diagrams of networks with $\sqrt{n}$ values of 16, 32, 64, 128, 256 and 512. There is a clear influence of the size in the quality of the results of the percolation threshold, but this effect is almost non existent in the bipartition threshold. This is in accordance to our expectations, since the correlation length does not diverge in the bipartition transition [10]. As discussed in sections 2.4 and 2.5 we expected that, for a random graph, the percolation threshold value should be $\overline{k}_c = 1$ and the partition threshold value should be $\overline{k}_p = 2\log 2$, while for a square lattice they should be $\overline{k}_p \approx \overline{k}_c = 2$ [13]. These values are plotted as straight lines, and it is clear that, as the size of the network increases, the experimental data comes closer to theoretically expected values, reaching a 1% margin for $\sqrt{n} = 512$.

4.2 Validation

In this work, we made the assumption that the minimum partition cost should have the same exponents for networks of all sizes, since the finite size effect should not play a great role in the bipartition transition. If this assumption is true, then the bipartition threshold values should not vary much with the size of network. This can be seen in Fig. 4.1, but in order to be more precise, we compare the bipartition threshold values of networks with $\sqrt{n}$ values of 16, 32, 64, 128 and 256 directly, as seen in Fig. 4.2. Although the data for $\sqrt{n} = 16$ is quite noisy, its deviation from the values for $\sqrt{n} = 256$ is small enough to justify our approach.

We also verified that the finite cluster size distribution obeys a power law at the percolation threshold of the form $n_s \propto s^{-\tau}$, where s is the size of a given cluster, normalized by the total number of vertices in the network. We plot n_s against s in

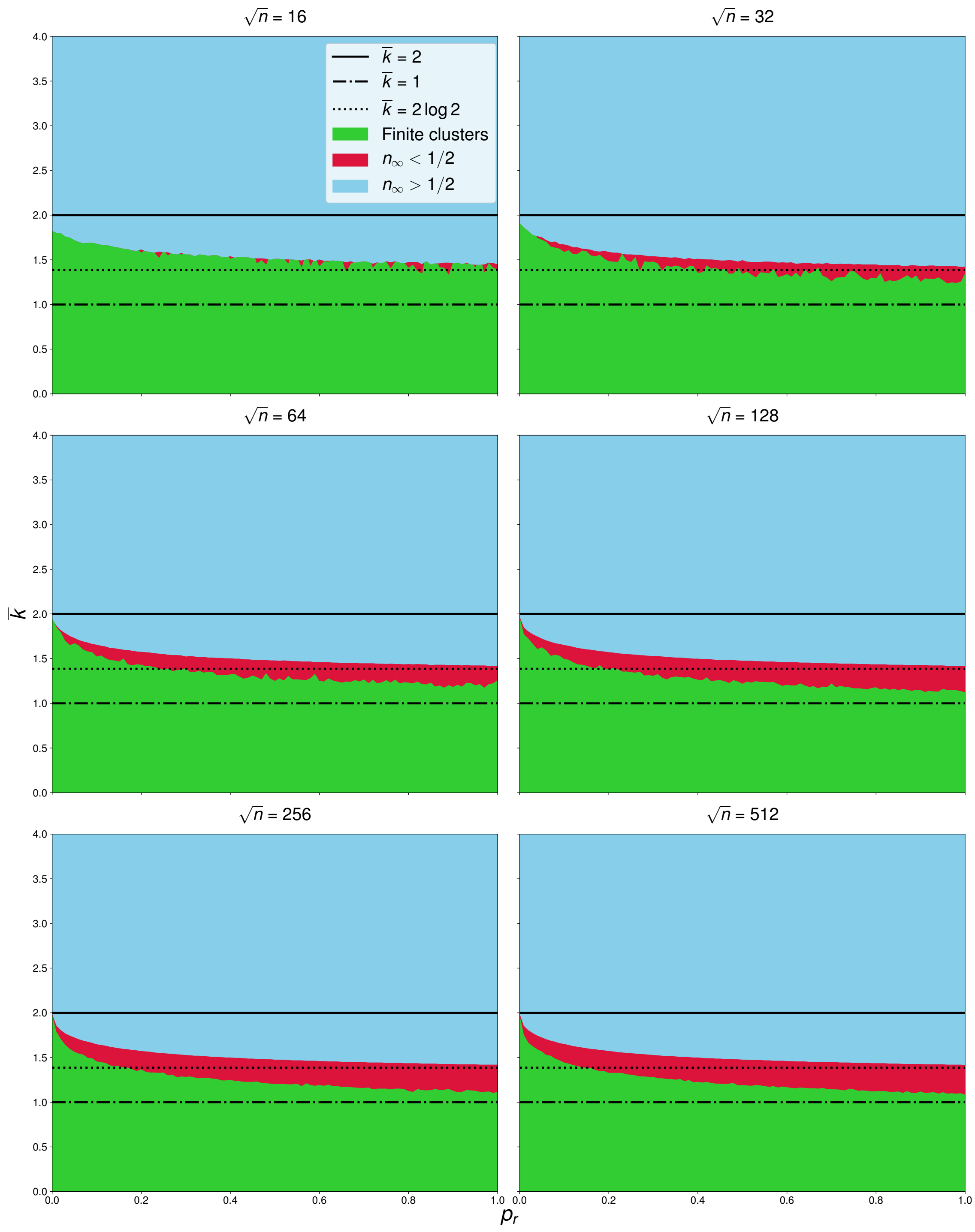


Figure 4.1: Phase diagram for networks with $\sqrt{n}$ values of 16, 32, 64, 128, 256 and 512, from left to right and top to bottom. Also shown are the theoretical values for the percolation and partition thresholds, $k_c \approx k_p = 2$ for $p_r = 0.0$ and $k_c = 1$ and $k_p = 2\log 2$ for $p_r = 1.0$.

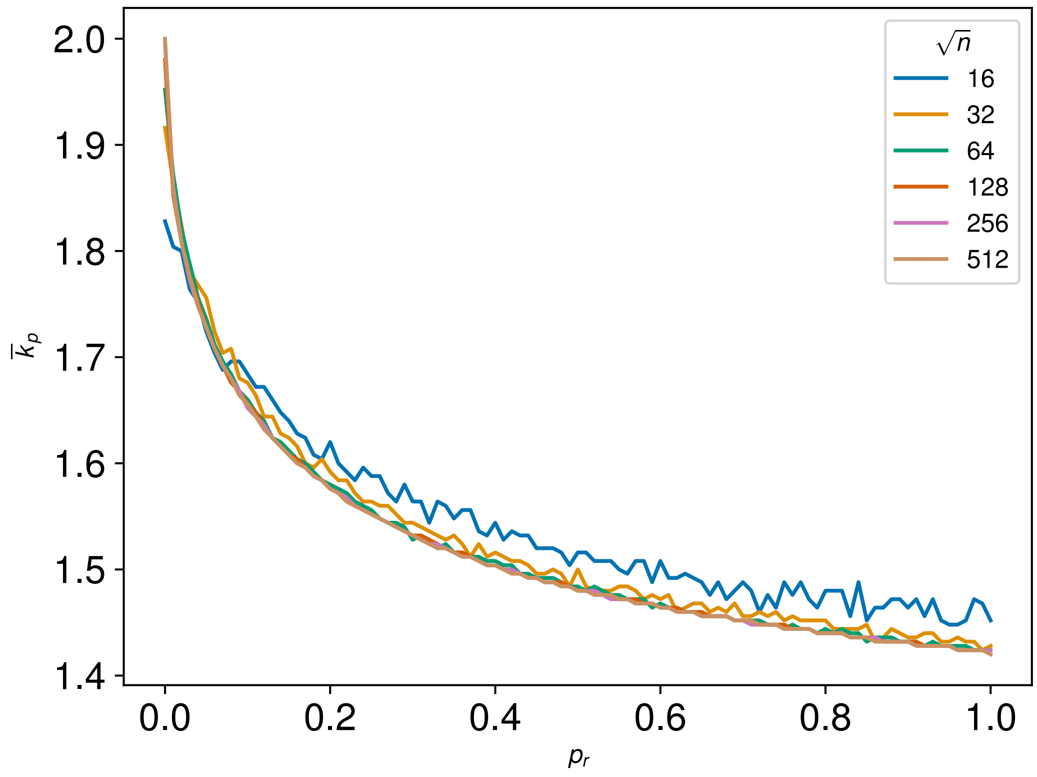


Figure 4.2: Partition threshold values for networks of several sizes along the entire rewiring spectrum.

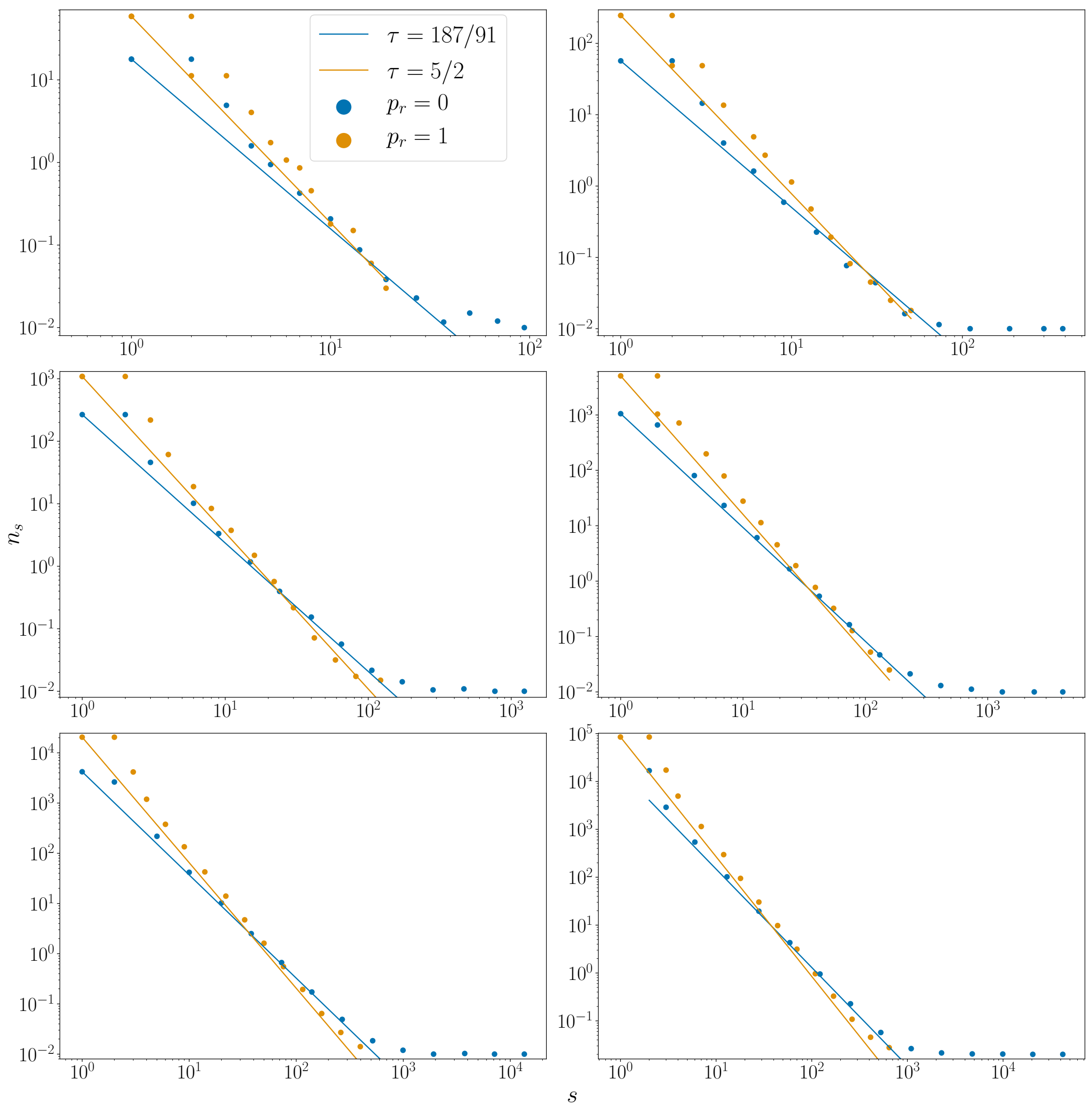


Figure 4.3: Number of clusters of size s vs s for square lattice and random networks with $\sqrt{n}$ values of 16, 32, 64, 128, 256 and 512, from left to right and top to bottom, with theoretical Fisher exponents.

Fig. 4.3 for networks with $\sqrt{n}$ values of 16, 32, 64, 128 and 256, as well as theoretically expected values. The data is quite noisy for large values of s, so we took a logarithmic sampling and averaging of s with 16 windows. In the smallest networks the curves deviate from theoretically expected curves but this deviation visibly diminishes as the size increases.

4.3 Algorithm comparison

We compare three partitioning algorithms using the initialization scheme laid out in Section 3.4 for a network of size 1024 and four values of p_r along the spectrum, so as to give a wide sample of each algorithm's behaviour. We choose deletion probabilities that lie in the middle region of the sampling space, as explained in Sec. 3.5. The results can be seen in Fig. 4.4. We also show the results obtained by EO with random initialization.

EO with our initialization scheme produces the best results in all cases. The results obtained with MLK improve as we approach the bipartition threshold point, while the results obtained with SA, although close to MLK for values of p_d distant from the bipartition threshold, are poor in all cases, especially for $p_r = 0$. We hypothesize that this stark difference between the algorithms is due to the varying susceptibility of each method to the initialization of the system, MLK being the most impervious to this effect. In conjunction with this, the poor quality of the results of SA for $p_r = 0$ can be attributed to the highly ordered structure of the network, which results in the algorithm becoming stuck in local minima. The results of EO with random initialization show how susceptible the algorithm is to the initial configuration of the system, resulting in poor values for the minimum partition cost.

4.4 Minimum partition cost

The minimum partition cost in the vicinity of the bipartition threshold point can be seen in Figs. 4.5 and 4.6. As of now, there is no precise description of what its behaviour should be in the full parameter space, so we can only base our analysis on the fact that it becomes null above the partition threshold, as expected, that it behaves differently for $p_r = 0$ and $p_r \neq 0$, and that it obeys a power law as a function of $p_{d,p} - p_d$, all of which can be seen in the data.

We obtained the parameters of this power law with an ordinary least squares (OLS) regression of the experimental data to the linearisation of Eq. (2.27)

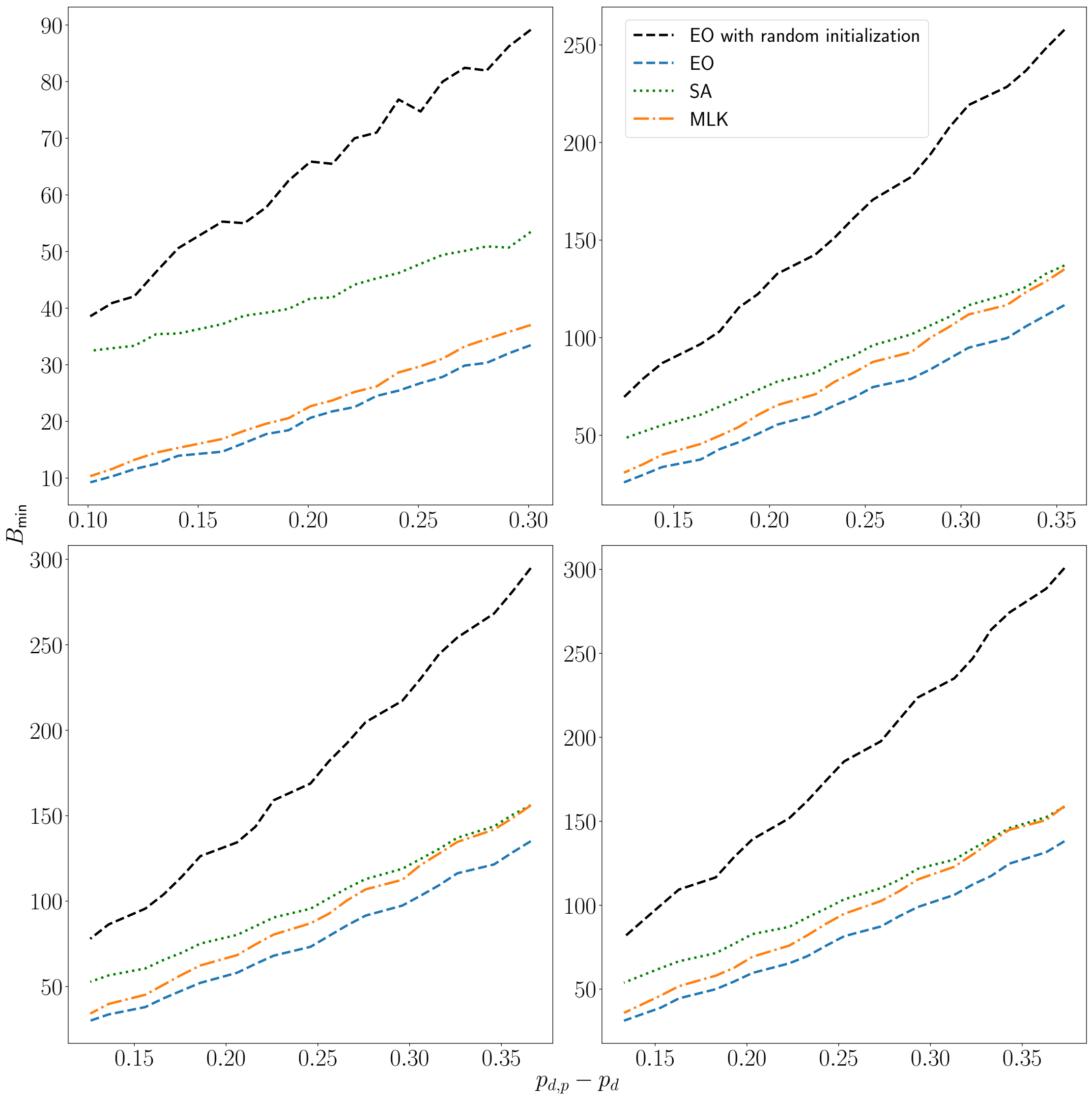


Figure 4.4: Comparison between algorithms for a network of size 1024 with $p_r = 0, 0.3, 0.7$ and 1, from left to right and top to bottom, In green we have the values obtained with SA, in orange the values obtained with MLK and in blue the values obtained with EO. The same initialization scheme was used for all three algorithms. We also show in black the values obtained with EO with random initialization.

$$\log B = \log C(n) + a \log{(p_{d,p} - p_d)}. \tag{4.1}$$

The complete results of this fit can be seen in Table 4.1 for EO and in Table 4.2 for MLK. We also overlay the fitted curves in figures 4.5 and 4.6. As we approach the limit of the random network, the fitted curves grow closer and their slope increases, until it converges on $a \approx 1.4$. For $p_r = 0$, we obtain values closer to 1. Both sets of results follow this trend, and the results shown for $p_r = 1$ are in agreement with those obtained by Boettcher and Percus in Refs. [9, 10]. As discussed in Section 2.4, small-world networks have the same critical percolation behaviour as random networks, so it is reasonable that the partition exponents are also the same.

We also investigate how the coefficient of finite size scaling $C(n)$ in Eq. (2.27) behaves across the rewiring spectrum. It is known that this quantity obeys a power law at the partition threshold of the form

$$\log C(n) = c \log n + C_0. \tag{4.2}$$

For random graphs, it is known that c must be 1, since any global property of these graphs is extensive. The same can be said of small-world networks, since the introduction of long-range shortcuts introduces extensivity. Thus, for all values of $p_r \neq 0.0$, this value should be 1. For random geometric graphs, the work in Ref. [9] obtained a value of 0.6. This value should be the same for the square lattice, since the critical percolation behaviour of these two types of graphs is the same [29]. Most likely, the value of c is 0.5, since the optimal partition of a square lattice should create an interface between the sets of length proportional to $\sqrt{n}$, i.e., the minimum partition cost should scale as $n^{1/2}$.

In order to obtain this value, we fitted the values of the intercept, which we denote by $C(n)$, obtained from the fit of Eq. (4.1) to experimental data, to the equation with OLS. The results of this fit can be seen in Fig. 4.7, and in more detail in Tables 4.3 and 4.4. We obtain a value of $c \approx 1$ for most values of $p_r \neq 0$, with the exception of $p_r = 0.05$ with MLK, which comes closer to 0.8. This might be due to the crossover between the square lattice and small-world network, most visible in smaller networks. That the values for $p_r = 0.1$ are in agreement for both methods, and much closer to the theoretical value of 1, lends more credence to this hypothesis. With EO, we obtain a value of $c = 0.5$ for the square lattice, which is in accordance with our expectations. With MLK we obtain a value of $c \approx 0.55$, closer to what is seen in Ref. [9].

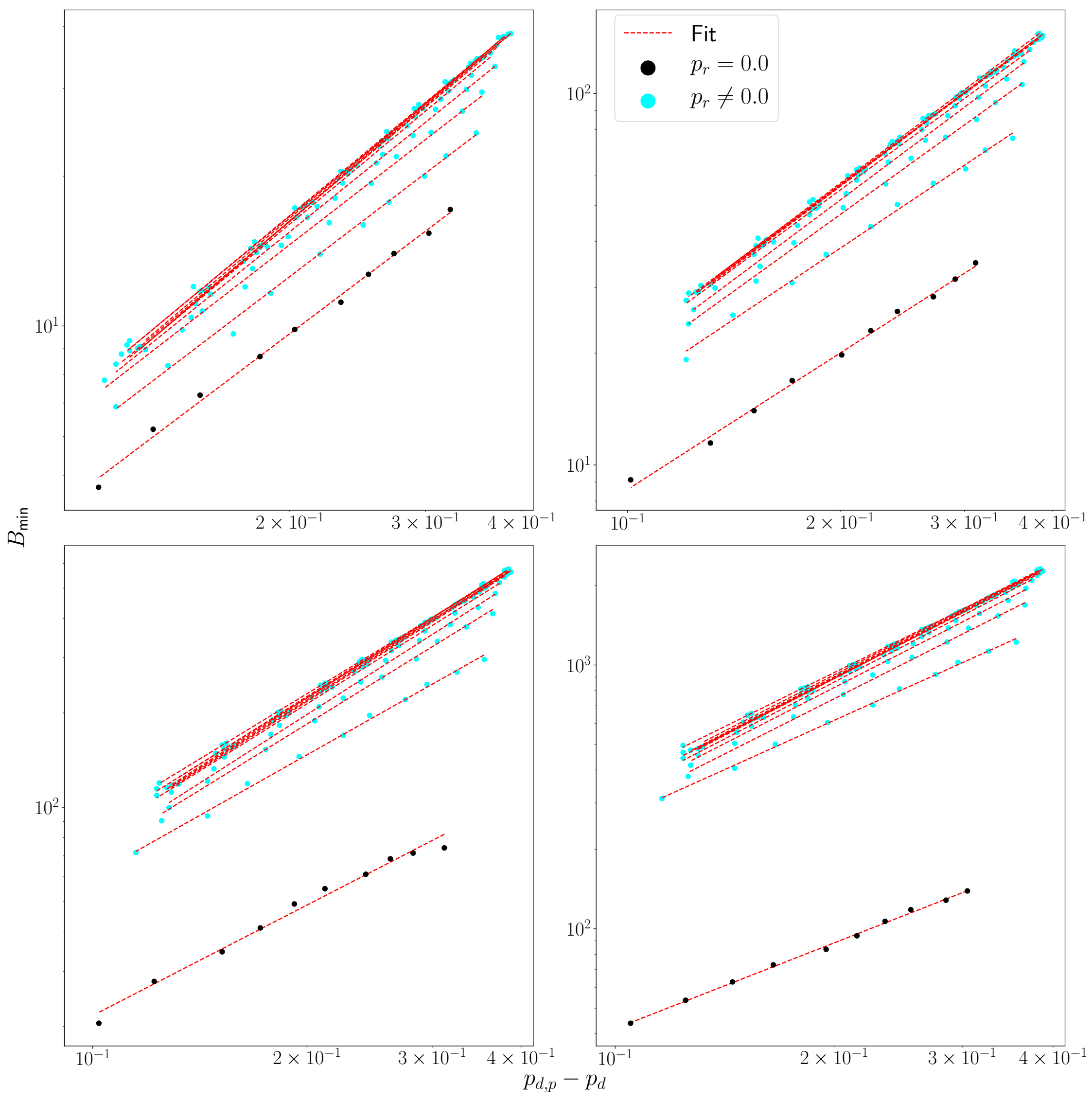


Figure 4.5: Minimum partition cost obtained with EO vs distance to bipartition threshold point for networks with $\sqrt{n}$ 16, 32, 64 and 128, from left to right and top to bottom. Points in black correspond to $p_r = 0.0$ while points in cyan correspond to all other values. Also shown in red are the curves of the fit of Eq. (4.1) to the experimental data.

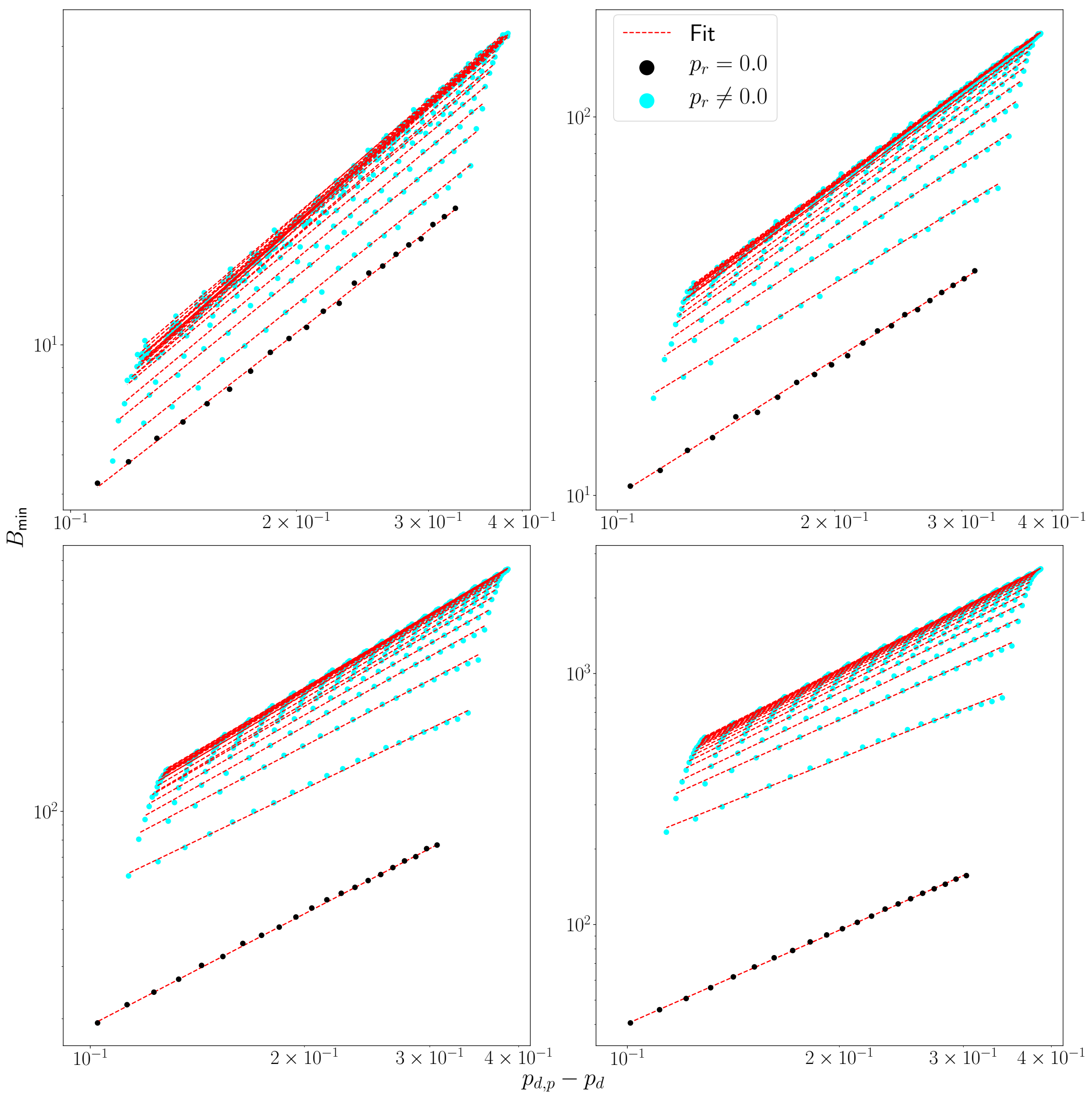
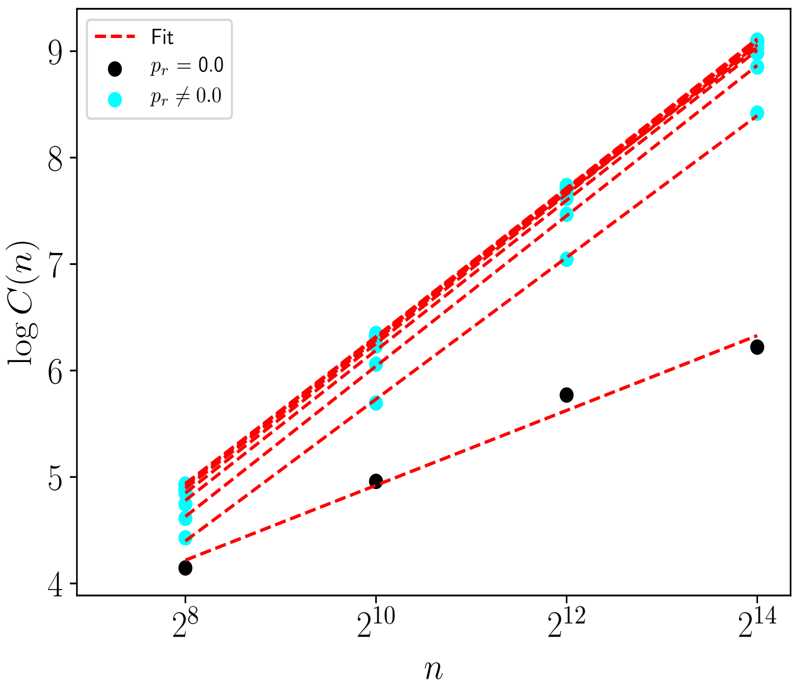
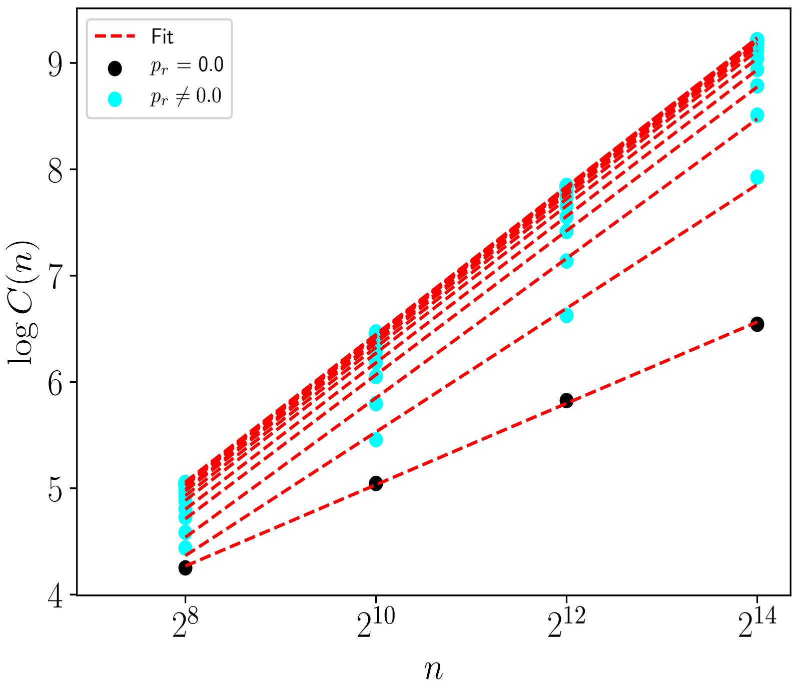


Figure 4.6: Minimum partition cost obtained with MLK vs distance to bipartition threshold point for networks with $\sqrt{n}$ 16, 32, 64 and 128, from left to right and top to bottom, overlaid with curves of the fit.



(a)



(b)

Figure 4.7: Log-log plot of the intercept $C(n)$, obtained from fit of Eq. (4.1) to experimental data obtained with (a) EO and (b) MLK, vs size of network. Points in black correspond to $p_r = 0.0$ while points in cyan correspond to all other values. Also shown in red are the curves of the fit of Eq. (4.2).

Table 4.1: Results of the fit of $\log B = \log C(n) + a \log(p_{d,p} - p_d)$ as a function of $p_{d,p} - p_d$ to experimental data obtained with EO.

$\sqrt{n}$		16			32			64			128	
p_r	intercept	slope	r^2	intercept	slope	r^2	intercept	slope	r^2	intercept	slope	r^2
0.0	4.15	1.17	0.996376	4.96	1.22	0.996224	5.77	1.17	0.986013	6.22	1.08	0.998147
0.1	4.43	1.18	0.997001	5.69	1.27	0.995998	7.05	1.28	0.997107	8.42	1.24	0.998189
0.2	4.61	1.20	0.996609	6.06	1.37	0.997796	7.47	1.40	0.997414	8.85	1.39	0.997311
0.3	4.75	1.25	0.997894	6.23	1.42	0.997877	7.62	1.45	0.998025	8.98	1.41	0.997903
0.4	4.85	1.29	0.996544	6.26	1.40	0.999193	7.65	1.42	0.998803	9.06	1.43	0.998747
0.5	4.88	1.30	0.997404	6.32	1.42	0.998935	7.70	1.44	0.998145	9.09	1.43	0.997992
0.6	4.89	1.29	0.998625	6.29	1.40	0.998932	7.68	1.41	0.998947	9.06	1.41	0.998980
0.7	4.93	1.33	0.998656	6.32	1.42	0.999010	7.71	1.44	0.998913	9.09	1.42	0.999062
0.8	4.91	1.31	0.996700	6.31	1.41	0.999290	7.74	1.45	0.998793	9.10	1.43	0.997980
0.9	4.87	1.28	0.998273	6.33	1.43	0.997883	7.72	1.43	0.997893	9.09	1.41	0.998212
1.0	4.91	1.32	0.998067	6.35	1.43	0.996706	7.66	1.38	0.999129	9.03	1.36	0.999053

Table 4.2: Results of the fit of $\log B = \log C(n) + a \log(p_{d,p} - p_d)$ as a function of $p_{d,p} - p_d$ to experimental data obtained with MLK.

$\sqrt{n}$		16			32			64			128	
p_r	intercept	slope	r^2	intercept	slope	r^2	intercept	slope	r^2	intercept	slope	r^2
0.00	4.250	1.175	0.999093	5.050	1.193	0.998873	5.820	1.249	0.999594	6.543	1.236	0.999764
0.05	4.440	1.210	0.998346	5.460	1.160	0.998576	6.630	1.150	0.998240	7.930	1.120	0.996780
0.10	4.580	1.218	0.999068	5.800	1.230	0.998941	7.140	1.260	0.998274	8.510	1.260	0.998301
0.15	4.730	1.260	0.998881	6.050	1.310	0.998956	7.420	1.340	0.998804	8.780	1.340	0.998926
0.20	4.800	1.260	0.999050	6.184	1.339	0.999688	7.550	1.366	0.999325	8.940	1.373	0.999417
0.25	4.880	1.289	0.999478	6.282	1.370	0.999748	7.657	1.390	0.999652	9.040	1.397	0.999695
0.30	4.920	1.310	0.998957	6.330	1.376	0.999539	7.735	1.424	0.999681	9.104	1.411	0.999751
0.35	4.970	1.323	0.999085	6.372	1.379	0.999820	7.773	1.424	0.999793	9.147	1.419	0.999837
0.40	4.970	1.303	0.999266	6.411	1.395	0.999793	7.793	1.417	0.999861	9.174	1.421	0.999869
0.45	5.020	1.343	0.999375	6.426	1.404	0.999844	7.820	1.431	0.999766	9.194	1.425	0.999899
0.50	5.050	1.354	0.999341	6.436	1.390	0.999707	7.826	1.425	0.999876	9.209	1.430	0.999899
0.55	5.040	1.352	0.999323	6.444	1.398	0.999923	7.842	1.429	0.999827	9.214	1.427	0.999901
0.60	5.040	1.328	0.999297	6.452	1.404	0.999737	7.837	1.422	0.999821	9.218	1.426	0.999930
0.65	5.050	1.358	0.999567	6.451	1.397	0.999816	7.842	1.429	0.999905	9.218	1.420	0.999925
0.70	5.040	1.336	0.999453	6.469	1.418	0.999622	7.846	1.428	0.999875	9.216	1.418	0.999923
0.75	5.000	1.300	0.998759	6.456	1.404	0.999826	7.848	1.429	0.999895	9.216	1.415	0.999952
0.80	5.030	1.333	0.999239	6.455	1.395	0.999781	7.841	1.417	0.999903	9.217	1.417	0.999927
0.85	5.048	1.369	0.999775	6.447	1.388	0.999695	7.837	1.420	0.999915	9.215	1.413	0.999944
0.90	5.030	1.335	0.999390	6.456	1.402	0.999833	7.834	1.413	0.999918	9.214	1.411	0.999933
0.95	5.050	1.361	0.999451	6.455	1.401	0.999804	7.838	1.414	0.999894	9.212	1.408	0.999945
1.00	5.020	1.338	0.999630	6.445	1.392	0.999749	7.832	1.409	0.999882	9.211	1.406	0.999919

Table 4.3: Results of the fit of $\log C(n) = c \log n + C_0$ to experimental data obtained with EO.

p_r	intercept	slope	r^2
0.0	1.40	0.510	0.984466
0.1	-0.90	0.960	0.999614
0.2	-1.02	1.019	0.999877
0.3	-0.80	1.020	0.999619
0.4	-0.76	1.011	0.999992
0.5	-0.71	1.011	0.999911
0.6	-0.66	1.003	0.999990
0.7	-0.62	1.001	0.999997
0.8	-0.69	1.010	0.999908
0.9	-0.72	1.010	0.999782
1.0	-0.50	0.990	0.999676

Table 4.4: Results of the fit of $\log C(n) = c \log n + C_0$ to experimental data obtained with MLK.

p_r	intercept	slope	r^2
0.00	1.210	0.5520	0.999489
0.05	-0.300	0.8400	0.997112
0.10	-0.700	0.9500	0.999301
0.15	-0.690	0.9750	0.999937
0.20	-0.710	0.9940	0.999990
0.25	-0.660	0.9990	0.999984
0.30	-0.650	1.0070	0.999952
0.35	-0.600	1.0050	0.999976
0.40	-0.610	1.0090	0.999891
0.45	-0.540	1.0040	0.999973
0.50	-0.497	1.0003	0.999999
0.55	-0.520	1.0040	0.999972
0.60	-0.520	1.0040	0.999972
0.65	-0.500	1.0020	0.999984
0.70	-0.510	1.0030	0.999900
0.75	-0.590	1.0100	0.999795
0.80	-0.540	1.0060	0.999934
0.85	-0.500	1.0020	0.999989
0.90	-0.530	1.0050	0.999933
0.95	-0.490	1.0000	0.999974
1.00	-0.550	1.0070	0.999941

Chapter 5

Conclusions and future work

In this work, we analysed the behaviour of the Graph Bipartitioning Problem at the partition threshold for small-world networks along the spectrum between the square lattice and a random network. We used a modified Watts-Strogatz to interpolate between these two extrema.

We obtained the partition and percolation threshold values for several networks of varying size along the spectrum, and found that the partition threshold value has very little dependence on the size of the network.

We tested several bipartition algorithms and concluded that Extremal Optimization and Multilevel K-way Partitioning had superior performance. We found that a proper initialization scheme is vital for quality results in the case of EO.

We obtained the exponents of the minimum partition cost in the vicinity of the partition threshold for networks of several sizes, and found that, regardless of size, the introduction of any number of shortcuts in the square lattice changes the behaviour of the network. We replicated the expected theoretical values for the exponents in the square lattice and the random network. The values of both methods for the square lattice are approximately 1, and values for networks in the spectrum start to converge to 1.4 when we approach the random network.

The finite size scaling exponents obtained with our methods are $c = 0.5$ for the square lattice and $c = 1$ for all other networks in the spectrum, in agreement with previous work and theoretically expected values.

For future work, we could study the GBP with site percolation and give a more thorough analysis of the interplay between the two length scales of small-world networks. It would also be interesting to use an exact method such as Parallel Tempering to give a more robust description of the problem and study the effect of initialization of EO using the MLK algorithm to produce initial configurations. For $p_r = 0$ it was found that for larger systems and close to $p_{d,p}$ the EO algorithm may have problems finding the optimal solution.

Bibliography

[1] R. M. Karp. “On the computational complexity of combinatorial problems”. In: *Networks* 5.1 (1975), pp. 45–68.

[2] A. B. Kahng et al. *VLSI Physical Design: From graph partitioning to timing closure*. Springer Netherlands, 2011.

[3] V. Kolmogorov and R. Zabih. “What energy functions can be minimized via graph cuts?” In: *European conference on computer vision*. Springer. 2002, pp. 65–81.

[4] D. Achlioptas and F. D. McSherry. *Partitioning social networks*. US Patent 7,668,957. Feb. 2010.

[5] U. V. Catalyurek et al. “Hypergraph-based dynamic load balancing for adaptive scientific computations”. In: *2007 IEEE International Parallel and Distributed Processing Symposium*. IEEE. 2007, pp. 1–11.

[6] P. Holme et al. “Attack vulnerability of complex networks”. In: *Physical review E* 65.5 (2002), p. 056109.

[7] C. Nowzari, V. M. Preciado, and G. J. Pappas. “Analysis and control of epidemics: A survey of spreading processes on complex networks”. In: *IEEE Control Systems Magazine* 36.1 (2016), pp. 26–46.

[8] D. J. Watts and S. H. Strogatz. “Collective dynamics of “small-world” networks”. In: *Nature* 393.6684 (1998), p. 440.

[9] S. Boettcher. “Extremal optimization of graph partitioning at the percolation threshold”. In: *Journal of Physics A: Mathematical and General* 32.28 (Jan. 1999), pp. 5201–5211.

[10] A. G. Percus et al. “The peculiar phase structure of random graph bisection”. In: *Journal of Mathematical Physics* 49.12 (2008), p. 125219.

[11] M. Newman. *Networks: An Introduction*. OUP Oxford, 2010.

[12] D. Stauffer and A. Aharony. *Introduction To Percolation Theory*. Taylor & Francis, 1994.

[13] H. Kesten. “The critical probability of bond percolation on the square lattice equals 1/2”. In: *Communications in mathematical physics* 74.1 (1980), pp. 41–59.

[14] M. Ozana. “Incipient spanning cluster on small-world networks”. In: *EPL (Europhysics Letters)* 55.6 (2001), p. 762.

[15] M. Newman, I. Jensen, and R. M. Ziff. “Percolation and epidemics in a two-dimensional small world”. In: *Physical Review E* 65.2 (2002), p. 021904.

[16] C. Moore and M. Newman. “Exact solution of site and bond percolation on small-world networks”. In: *Physical Review E* 62.5 (2000), p. 7059.

[17] G. Parisi. “The physical meaning of replica symmetry breaking”. In: *arXiv preprint cond-mat/0205387* (2002).

[18] M. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Clarendon Press, 1999.

[19] K. Kawasaki. “Diffusion constants near the critical point for time-dependent Ising models. I”. In: *Phys. Rev.* 145 (1 May 1966), pp. 224–230.

[20] K. Hukushima and K. Nemoto. “Exchange Monte Carlo method and application to spin glass simulations”. In: *Journal of the Physical Society of Japan* 65.6 (1996), pp. 1604–1608.

[21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing”. In: *Science* 220.4598 (1983), pp. 671–680.

[22] D. Bertsimas, J. Tsitsiklis, et al. “Simulated annealing”. In: *Statistical science* 8.1 (1993), pp. 10–15.

[23] P. Bak and K. Sneppen. “Punctuated equilibrium and criticality in a simple model of evolution”. In: *Physical Review Letters* 71.24 (1993), p. 4083.

[24] G. Karypis and V. Kumar. “A fast and high quality multilevel scheme for partitioning irregular graphs”. In: *SIAM Journal on Scientific Computing* 20.1 (1998), pp. 359–392.

[25] G. Karypis and V. Kumar. “Parallel multilevel series k-way partitioning scheme for irregular graphs”. In: *SIAM Review* 41.2 (1999), pp. 278–300.

[26] B. W. Kernighan and S. Lin. “An efficient heuristic procedure for partitioning graphs”. In: *The Bell System Technical Journal* 49.2 (Feb. 1970), pp. 291–307.

[27] R. C. Tausworthe. “Random numbers generated by linear recurrence modulo two”. In: *Mathematics of Computation* 19.90 (May 1965), pp. 201–201.

[28] M. Matsumoto and T. Nishimura. “Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator”. In: *ACM Trans. Model. Comput. Simul.* 8.1 (Jan. 1998), pp. 3–30.

[29] E. T. Gawlinski and H. E. Stanley. “Continuum percolation in two dimensions: Monte Carlo tests of scaling and universality for non-interacting discs”. In: *Journal of Physics A: Mathematical and General* 14.8 (1981), p. L291.