Lecture given at DPG Physics School
Efficient Algorithms in Computational Physics
(September 10-14, 2012, Bad Honnef, Germany)

Phase Transitions and Clustering Properties of Optimization Problems

Alexander Hartmann
Institute of Physics
Carl von Ossietzky Universität Oldenburg
D-26111 Oldenburg
Germany

Abstract. One central subject in computer science are NP-complete problems. These are problems which are hard in the sense that no fast algorithms to solve them exist. Interestingly, many practical applications belong to this class. When studying problems on suitably parametrized ensembles, one finds phenomena strongly reminiscent of phase transitions as appearing for physical systems. One also observes regions in phase space where the problem can typically be quickly solved using the available algorithms, i.e., where the problem does not typically appear to be hard. One can better understand these phase transitions using concepts and methods from statistical physics. These developments are exemplified considering the Vertex-cover Problem. In particular, we investigate the relationship between the behavior of the ensemble, the typical running time of algorithms and the structure of the solution landscape. We use branch-and-bound type algorithms to obtain exact solutions of these problems for moderate system sizes as well as parallel tempering simulations ("MC^3"). Using two methods (direct neighborhood-based clustering and hierarchical clustering), we investigate the structure of the solution space for Erdős-Reny random graphs. For the vertex cover problem we observe a drastic change of the solution space from large single clusters (for small connecitivities $c < e \approx 2.71$) to multiple nested levels of clusters (for connectivities $c > e$). This is in agreement with statistical-mechanics calculations, where a transition a $c = e$ from a so-called replica symetric to a so-called

replic-symmetry broken phase was found.

Contents

1 Introduction to Graphs

1.1 The bridges of Königsberg and Eulerian graphs

The earliest use of graph theoretical methods probably goes back to the 18th century. At this time, there were seven bridges crossing the river Pregel in the town of Königsberg. The folks had long amused themselves with the following problem: Is it possible to walk through the town using every bridge just once, and returning home at the end? The problem was solved by Leonhardt Euler (1707–1783) in 1736 by mapping it to a graph problem and solving it for arbitrary graphs [1], i. e., for arbitrary towns, city maps, etc. In the case of Königsberg, he had to draw the slightly disappointing consequence that no such round-walk existed.

Fig. 1 shows the river Pregel, Königsberg was situated on both sides and both islands. The seven bridges are also shown. The mapping to a graph is given below. Every river side, island and bridge is assigned a *vertex*, drawn as a circle, two vertices

are connected by an *edge*, drawn as a line, if they are also physically connected. To give a more precise description, we have to introduce the basic graph-theoretical terminology which is summarized in the definitions below.

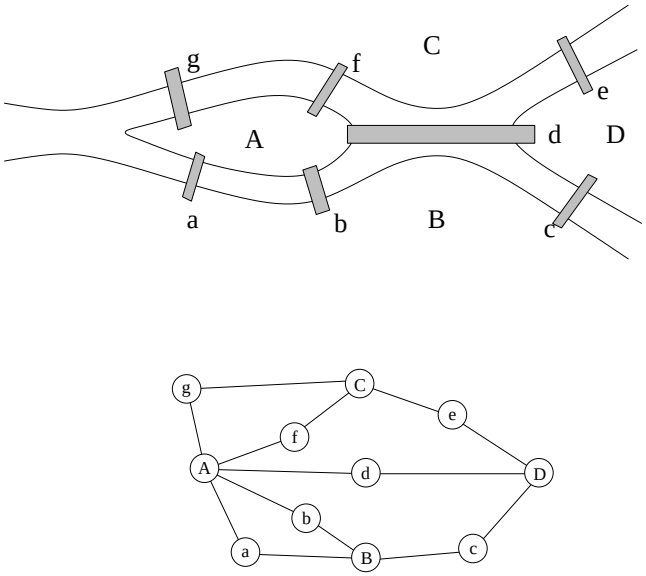

Figure 1: The bridges of Königsberg and its graph representation. Vertices are denoted by circles, edges by lines.

Basic definitions:

- An *(undirected) graph* $G = (V, E)$ is given by its *vertices* $i \in V$ and its undirected *edges* $\{i, j\} \in E \subset V^{(2)}$. Note that both $\{i, j\}$ and $\{j, i\}$ denote the same edge.
- The *order* $N = |V|$ counts the number of vertices.
- The *size* $M = |E|$ counts the number of edges.
- Two vertices $i, j \in V$ are *adjacent / neighboring* if $\{i, j\} \in E$.
- The edge $\{i, j\}$ is *incident* to its end vertices i and j.
- The *degree* $\deg(i)$ of vertex i equals the number of adjacent vertices. Vertices of zero degree are called *isolated*.
- A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subset V,\ E' \subset E$.
- A graph $G' = (V', E')$ is an *induced subgraph* of G if $V' \subset V$ and $E' = E \cap (V')^{(2)}$, i. e., E' contains all edges from E connecting two vertices in V'.

- A subgraph $G' = (V', E')$ is a *path* of G if it has the form $V' = \{i_0, i_1, \ldots, i_l\}$, $E' = \{\{i_0, i_1\}, \{i_1, i_2\}, \ldots, \{i_{l-1}, i_l\}\}$. The *length* of the path is $l = |E'|$. i_0 and i_l are called *end points*. The path goes from i_0 to i_l and vice versa. One says i_0 *and* i_l *are connected by the path*. Note that, within a path, each vertex (possibly except for the end points) is "visited" only once.
- A path with $i_0 = i_l$, i. e., a *closed path*, is called a *cycle*.
- A sequence of edges $\{i_0, i_1\}$, $\{i_1, i_2\}, \ldots, \{i_{l-1}, i_l\}$ is called a *walk*. Within a walk some vertices or edges may occur several times.
- A walk with pairwise distinct edges is called a *trail*. Hence a trail is also a subgraph of G.
- A *circuit* is trail with coinciding end points, i. e., a *closed* trail. (NB: cycles are circuits, but not vice versa, because a circuit may pass through several times through the same vertex).
- The graph G is *connected* if all pairs i, j of vertices are connected by paths.
- The graph $G' = (V', E')$ is a *connected component* of G if it is a connected, induced subgraph of G, and there are no edges in E connecting vertices of V' with those in $V \setminus V'$.
- The *complement graph* $G^C = (V, E^C)$ has edge set $E^C = V^{(2)} \setminus E = \{\{i, j\} \mid \{i, j\} \notin E\}$. It is thus obtained from G by connecting all vertex pairs by an edge, which are not adjacent in G and disconnecting all vertex pairs, which are adjacent in G.
- A *weighted graph* $G = (V, E, \omega)$ is a graph with edge weights $\omega : E \to \mathbb{R}$.

Example: Graphs

We consider the graph shown in Fig. 1. It can be written as $G = (V, E)$ with

$$\begin{aligned} V &= \{A, B, C, D, a, b, c, d, e, f, g\} \\ E &= \{\{A, a\}, \{A, f\}, \{A, d\}, \{A, f\}, \{A, g\}, \{B, a\}, \{B, b\}, \{B, c\}, \\ &\quad \{C, e\}, \{C, f\}, \{C, g\}, \{D, c\}, \{D, d\}, \{D, e\}, \}\,. \end{aligned}$$

Hence, the graphs has $|V| = 11$ vertices and $|E| = 14$ edges. Since $\{D, e\} \in E$, the vertices D and d are adjacent. Vertex d has degree $\deg(d) = 2$, while vertex A has degree 5.

For example, $G' = (V', E')$ with $V' = \{A, g, C, e, D\}$ and $E' = \{\{A, g\}, \{g, C\}, \{C, e\}, \{e, D\}, \}$ is a path from A to D. G is connected, because all vertices are connected by paths to all other vertices. The sequence

of edges $\{B, c\}, \{c, D\}, \{D, c\}$ is a walk, but it does not correspond to a path, because some vertices are visited twice. The sequence of edges $\{A, b\}$, $\{b, B\}$, $\{B, a\}$, $\{a, A\}$, $\{A, g\}$, $\{g, C\}$, $\{C, f\}$, $\{f, A\}$ is a trail, in particular it is a circuit.

Going back to the problem of the people from Königsberg, formulated in graph-theoretical language, they were confronted with the following question:
EULERIAN CIRCUIT: Given a graph, is there a circuit using every *edge* exactly once?

The amazing point about Euler's proof is the following: The existence of a Eulerian circuit – which obviously is a global object – can be decided looking to purely local properties, in this case to the vertex degrees.
Theorem: *A connected graph $G = (V, E)$ is Eulerian (has an Eulerian cycle) iff all vertex degrees are even.*
Proof:
($\rightarrow$) This direction is trivial. Following the Eulerian circuit, every vertex which is entered is also left, every time using previously unvisited edges. All degrees are consequently even.

($\leftarrow$) The other direction is a bit harder. We will proof it by induction on the graph size M, for arbitrary graphs having only even vertex degrees.

The theorem is obviously true for $M = 0$ (one isolated vertex) and $M = 3$ (triangle) which are the simplest graphs with even degrees.

Now we take any $M > 0$, and we assume the theorem to be true for all graphs of size smaller than M. We will show that the theorem is also true for a connected graph G of size M having only even degrees.

Because of $M > 0$, the graph G is non-trivial, because of the even degrees it contains vertices of degree at least 2. Then G contains a cycle, which can be seen in the following way: Start in any vertex and walk along edges. Whenever you enter a new vertex, there is at least one unused edge which you can use to exit the vertex again. At a certain moment you enter a vertex already seen (at most after M steps), the part of the walk starting there is a cycle.

Every cycle is also a circuit. Consider now a circuit $C = (G', E')$ of maximal size $|E'|$. If C is a Eulerian circuit in G, everything is OK. If not, we have $|E'| < |E|$, and the subgraph $H = (V, E \setminus E')$ has at least one non-trivial connected component H'. A circuit has even degrees, thus H and all its connected components have even degrees. Since H' has size $< M$, it has an Eulerian circuit which can be added to C, violating the maximality of C. Thus, C has to be an Eulerian circuit, and the proof is complete. QED

Going back to Königsberg, we see that there are vertices of odd degrees. No Eulerian circuit can exist, and the inhabitants have either to skip bridges or to cross some twice if they walk through their town.

Note that in the above definitions for undirected graphs, edges have no orientation. This is different for *directed* graphs (also called *digraphs*), where an edge is denoted by a pair (i, j) which is different from the pair (j, i). Since directed graphs are not

needed here, we do not go into details.

1.2 Edge covers and vertex covers

Now we introduce edge and vertex covers. Both problems are similar to each other, but they are fundamentally different with respect to how easily they can be solved algorithmically. This we have already seen for Eulerian circuits and Hamiltonian cycles. The vertex-cover problem will serve as the prototype example in this book. All concepts, methods and analytical techniques will be explained using the vertex-cover problem. We begin here with the basic definitions, but additional definitions, related to vertex-cover algorithms, are given in Sec. 3.

For a graph $G = (V, E)$, an *edge cover* is a subset $E' \subset E$ of edges such that each vertex is contained in at least one edge $e \in E'$. Each graph which has no isolated vertices has an edge cover, since in that case E itself is an edge cover. A *minimum edge cover* is an edge cover of minimum cardinality $|E^{'}|$. In Fig. 2 a graph and a minimum edge cover are shown. A fast algorithm which constructs a minimum edge cover can be found in Ref. [2].

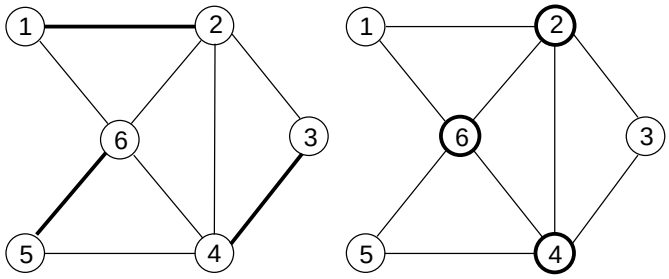

Figure 2: A graph and a minimum edge cover (left) and a minimum vertex cover (right). The edges/vertices belonging to the cover are shown in bold.

The definition of a *vertex cover* is very similar. It is a subset $V' \subset V$ of vertices such that each edge $e = \{i, j\} \in E$ contains at least one vertex out of V', i. e., $i \in V'$ or $j \in V'$. Note that V itself is always a vertex cover. A *minimum vertex cover* is an vertex cover of minimum cardinality $|V^{'}|$.

Vertex covers are closely related to *independent sets* and *cliques*. An independent set of a graph $G = (V, E)$ is a subset $I \subset V$ of vertices, such that for all elements $i, j \in I$, there is no edge $\{i, j\} \in E$. A clique is a subset $Q \subset V$ of vertices, such that for all elements $i, j \in Q$ there is an edge $\{i, j\} \in E$.

Theorem: *For a given graph $G = (V, E)$ and a subset $V' \subset V$ the following three statements are equivalent.*

(A) V' is a vertex cover of G.

(B) $V \setminus V'$ is an independent set of G.

(C) $V \setminus V'$ is a clique of the complement graph G^C (see definition on page 3).

Proof:
(A $\rightarrow$ B)
Let V' be a vertex cover, and $i, j \in V \setminus V'$. We assume that there is an edge $\{i, j\} \in E$. Since $i, j \notin V'$, this is an edge with both vertices not in V', and V' is not a vertex cover. This is a contradiction! Hence, there cannot be an edge $\{i, j\} \in E$, and $V \setminus V'$ is an independent set.

(B $\rightarrow$ C)
Let $V \setminus V'$ be an independent set, and $i, j \in V \setminus V'$. By definition, there is no edge $\{i, j\} \in E$, and so there is an edge $\{i, j\} \in E^C$. Therefore, $V \setminus V'$ is a clique of G^C.

(C $\rightarrow$ A)
Let $V \setminus V'$ be a clique of G^C, and $\{i, j\} \in E$. This means $\{i, j\} \notin E^C$ by definition of G^C. Thus, we have $i \notin V \setminus V'$ or $j \notin V \setminus V'$ because $V \setminus V'$ is a clique. Hence, $i \in V'$ or $j \in V'$. By definition of vertex cover, V' is a vertex cover. QED

The *minimum vertex cover* is a vertex cover of minimum cardinality. From the theorem above, for a minimum vertex cover V', $V \setminus V'$ is a maximum independent set of G and a maximum clique of G^C. In Fig. 2, a graph together with its minimum vertex cover is displayed. Related to the minimization problem is the following decision problem, for given integer K:

VERTEX COVER (VC): Does a given graph G have a vertex cover V' with $|V^{'}| \leq K$?

VC is a so-called NP-complete problem (see the contribution of Stephan Mertens), which means in particular that no fast algorithm for solving this problem is known – and not even expected to be able to be constructed. This proposition and its relation to statistical physics will be the main subject of this book.

2 Random graphs

We will not study VC on arbitrary graphs, but on a certain *ensemble* of randomly generated graphs, i.e., classes of graphs which have certain properties and appear in the ensemble with a certain probability. Here we restrict ourselves to very simple ensembles.

2.1 Two ensembles

The simplest idea goes back to a seminal paper published by Erdős and Rényi in 1960 [3]. They assumed all graphs of the same order N and size M to be equiprobable. There are mainly two slightly different ensembles used in the field:

The ensemble $\mathcal{G}(N, M)$ contains all graphs of N vertices and M edges. The measure is flat, i. e., every graph has the same probability, see, e. g., Fig. 3. Note that all graphs obtained from a given graph by permuting the vertices are different graphs, i. e., they are counted individually.

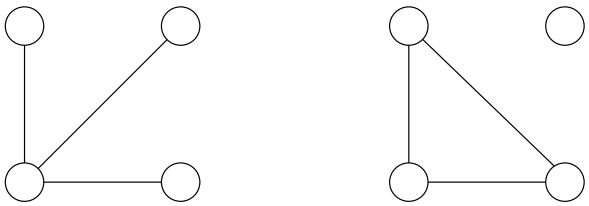

Figure 3: These two graphs are equiprobable in $\mathcal{G}(4,3)$, even if their structure is quite different. Whereas the left graph is a connected tree, the right one is not connected and contains a cycle.

The ensemble $\mathcal{G}(N,p)$, with $0 \leq p \leq 1$, contains graphs with N vertices. For every vertex pair i,j an edge $\{i,j\}$ is drawn independently with probability p. For $p=0$, the graph has no edges, for $p=1$, the graph is complete (K_N). On average, the number of edges for given p is

$$\overline{M} = p\binom{N}{2} = p\frac{N!}{(N-2)!2!} = p\frac{N(N-1)}{2} \tag{1}$$

where the over-bar denotes the average over $\mathcal{G}(N,p)$. This ensemble is analytically easier to handle, so we mainly work with $\mathcal{G}(N,p)$.

The specific case $\mathcal{G}(N,1/2)$ can also be considered as *the* random ensemble of graphs: All graphs of N vertices are equiprobable, independently of their edge numbers.

One important type of statement is that a $\mathcal{G}(N,p)$-graph fulfils *almost surely* (or *with probability one*) some condition C. This expression means that, for $N \to \infty$, the probability that a graph drawn from $\mathcal{G}(N,p)$ fulfils this condition C, converges to one.

2.2 Evolution of graphs

Sometimes, it is very instructive to imagine the ensemble $\mathcal{G}(N,p)$ via an evolutionary process of graphs of N vertices which, with time, get more and more edges. This can be realized in the following way. We take $V = \{1,\ldots,N\}$, and for all $i,j \in V$, $i<j$, we independently draw a random number x_{ij} being equally distributed in $(0,1)$. Initially the graph has no edges. Now, we start to grow $p(t)$. Whenever it exceeds a number x_{ij}, an edge between vertices i and j is added. Thus, at time t, the graph belongs to $\mathcal{G}(N,p(t))$. There are some interesting stages in this evolution:

- $p(t) \sim 1/N^2$: The first isolated edges appear.
- $p(t) \sim 1/N^{3/2}$: The first vertices have degree 2, i. e., the first edges have a common vertex.

- $p(t) \sim 1/N^\alpha$, any $\alpha > 1$: The graph is almost surely a forest.
- $p(t) \sim 1/N$: The average vertex degree stays finite for $N \to \infty$, first cycles appear, the first macroscopic (i. e., of order $\mathcal{O}(N)$) subgraph appears, macroscopic q-cores appear.
- $p(t) \simeq \ln(N)/N$: The graph becomes connected.
- $p(t) \simeq (\ln(N) + \ln(\ln(N))/N$: The graph becomes Hamiltonian.

For the proof see the book by Bollobás [4].

2.3 Finite-connectivity graphs: The case $p = c/N$

The most interesting case is given by $p = c/N$, where the medium size of the graph $\overline{M} = c(N-1)/2$ grows linearly with the graph order N. In this section, we first discuss the fluctuations of the total edge number M, and some local properties like degrees and the existence of small cycles. In the following two sections we finally switch to global properties. We discuss *random-graph percolation* which describes a phase transition from graphs having only small connected components, even for $N \to \infty$, to graphs also having one giant component which unifies a finite fraction of all vertices, i. e., whose order also grows linearly with the graph order N. The last subsection discusses the sudden emergence of a q-core.

2.3.1 The number of edges

For a graph from $\mathcal{G}(N, c/N)$, every pair of vertices becomes connected by an edge with probability c/N, and remains unlinked with probability $1 - c/N$. In contrast to the $\mathcal{G}(N, M)$-ensemble, the total number of edges is a random variable and fluctuates from sample to sample. Let us therefore calculate the probability P_M of it having exactly M edges. This is given by

$$P_M = \binom{N(N-1)/2}{M} \left[\frac{c}{N}\right]^M \left[1 - \frac{c}{N}\right]^{N(N-1)/2-M} . \tag{2}$$

The combinatorial prefactor describes the number of possible selections of the M edges out of the $N(N-1)/2$ distinct vertex pairs, the second factor gives the probability that they are in fact connected by edges, whereas the last factor guarantees that there are no further edges. In the limit of large N, and for $M \sim N$, where $M \ll N(N-1)/2$, we use

$$\left(\frac{N(N-1)}{2}\right)\left(\frac{N(N-1)}{2} - 1\right)\cdots\left(\frac{N(N-1)}{2} - M + 1\right) = \left(\frac{N(N-1)}{2}\right)^M + \mathcal{O}\left(N^{2(M-1)}\right) \tag{3}$$

and $(1 - c/N)^{zN} \approx \exp(-cZ)$, hence the above expression can be asymptotically approximated by

$$P_M \simeq \frac{(N(N-1)/2)^M}{M!} \left[\frac{c}{N}\right]^M \exp(-c(N-1)/2) \ . \tag{4}$$

Plugging in $\overline{M} = c(N-1)/2$, this asymptotically leads to a *Poissonian distribution* with mean $\overline{M}$,

$$P_M \simeq \exp\left(-\overline{M}\right) \frac{\overline{M}^M}{M!} \ . \tag{5}$$

The fluctuations of the edge number M can be estimated by the *standard deviation* of P_M,

$$\sigma(M) = \sqrt{\overline{(M-\overline{M})^2}} = \sqrt{\overline{M^2} - \overline{M}^2} \ . \tag{6}$$

We first calculate

$$\begin{aligned}
\overline{M^2} &= \overline{M} + \overline{M(M-1)} \\
&= \overline{M} + \sum_{M=0}^{\infty} M(M-1) P_M \\
&= \overline{M} + \exp\left(-\overline{M}\right) \sum_{M=2}^{\infty} \frac{\overline{M}^M}{(M-2)!} \\
&= \overline{M} + \overline{M}^2 \exp\left(-\overline{M}\right) \sum_{m=0}^{\infty} \frac{\overline{M}^m}{m!} \\
&= \overline{M} + \overline{M}^2 \ .
\end{aligned} \tag{7}$$

In the third line, we have eliminated the cases $M = 0, 1$ which vanish due to the factor $M(M-1)$, in the fourth line we have introduced $m = M + 2$ which runs from 0 to ∞. The sum can be performed and gives $\exp(\overline{M})$. Using Eq. (6) we thus find

$$\sigma(M) = \sqrt{\overline{M}} \ , \tag{8}$$

which is a special case of the *central limit theorem*. The relative fluctuations $\sigma(M)/\overline{M} = 1/\sqrt{\overline{M}}$ decay to zero for $\overline{M} = c(N-1)/2 \to \infty$, see Fig. 4. In this limit, the sample-to-sample fluctuations of the edge number become less and less important, and the ensemble $\mathcal{G}(N, c/N)$ can be identified with $\mathcal{G}(N, \overline{M})$ for most practical considerations.

2.3.2 The degree distribution

Let us now discuss the degrees of the graph. We are interested in the probability p_d that a randomly chosen vertex has exactly degree d. The set of all p_d is called the *degree distribution*. It can be easily calculated:

$$p_d = \binom{N-1}{d} \left[\frac{c}{N}\right]^d \left[1 - \frac{c}{N}\right]^{N-d-1} \ . \tag{9}$$

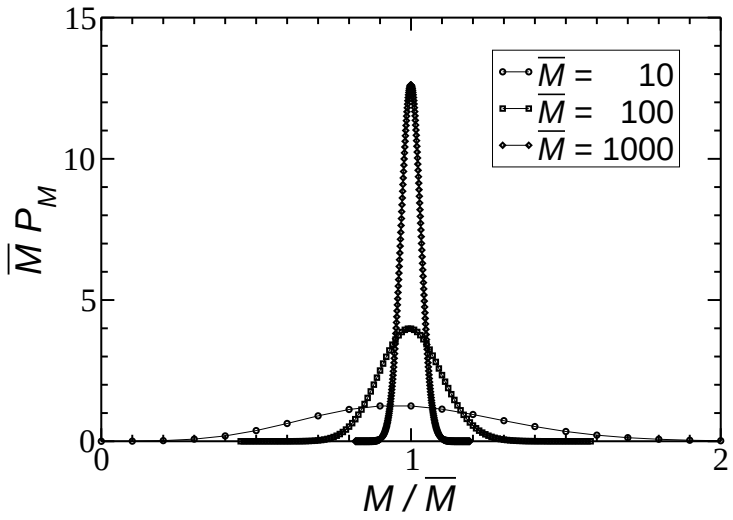

Figure 4: The distribution of edge numbers for $\overline{M} = 10, 100, 1000$, rescaled by $\overline{M}$. The distributions obviously sharpen for increasing $\overline{M}$.

The meaning of the terms on the right-hand side is the following: The factor $\binom{N-1}{d}$ enumerates all possible selections for d potential neighbors. Each of these vertices is connected to the central vertex with probability c/N, whereas the other $N-d-1$ vertices are not allowed to be adjacent to the central one, i. e., they contribute a factor $(1-c/N)$ each. In the large-N limit, where any fixed degree d is small compared with the graph order N, we can continue in an analogous way to the last subsection and find

$$\begin{aligned} p_d &= \lim_{N\to\infty} \frac{(N-1)(N-2)\cdots(N-d)}{N^d} \left[1-\frac{c}{N}\right]^{N-d-1} \frac{c^d}{d!} \\ &= e^{-c}\frac{c^d}{d!}\,, \end{aligned} \tag{10}$$

i. e., also the degrees are distributed according to a Poissonian distribution. It is obviously normalized, and the average degree is

$$\begin{aligned} \sum_{d=0}^{\infty} d\; p_d &= \sum_{d=1}^{\infty} e^{-c}\frac{c^d}{(d-1)!} \\ &= c\,. \end{aligned} \tag{11}$$

This was clear since the expected number of neighbors to any vertex is $p(N-1) \to c$. Note also that the fluctuation of this value are again given by the standard deviation

$\sigma(c) = \sqrt{c}$, which, however, remains comparable to c if $c = \mathcal{O}(1)$. The degree fluctuations between vertices thus also survive in the limit $N \to \infty$, there is, e. g., a fraction e^{-c} of all vertices which is completely isolated.

If we randomly select an edge and ask for the degree of one of its end-vertices, we obviously find a different probability distribution q_d, e. g., degree zero cannot be reached ($q_0 = 0$). This probability is obviously proportional to p_d as well as to d, by normalization we thus find

$$q_d = \frac{dp_d}{\sum_d dp_d} = \frac{dp_d}{c} = e^{-c}\frac{c^{d-1}}{(d-1)!} \tag{12}$$

for all $d > 0$. The average degree of vertices selected in this way equals $c + 1$, it includes the selected edge together with, on average, c additional *excess edges*. The number $d - 1$ of these additional edges will be denoted as the *excess degree* of the vertex under consideration.

2.3.3 Cycles and the locally tree-like structure

The degrees are the simplest local property. We may go slightly beyond this and ask for the average number of small subgraphs. Let us start with subtrees of k vertices which thus have $k - 1$ edges. We concentrate on labeled subtrees (i. e., the order in which the vertices appear is important) which are not necessarily induced (i. e., there may be additional edges joining vertices of the tree which are not counted). Their expected number is proportional to

$$N(N-1)\cdots(N-k+1)\left[\frac{c}{N}\right]^{k-1} \quad = \quad Nc^{k-1} + \mathcal{O}(1)\ , \tag{13}$$

combinatorial factors specifying $k - 1$ specific edges out of the $k(k-1)/2$ possible ones, are omitted. This number thus grows linearly with the graph order N. If, on the other hand, we look to cycles of finite length k, we have to find k edges. The above expression takes an additional factor c/N, the expected number of cycles is thus of $\mathcal{O}(1)$, i. e., the number of triangles, squares, etc. stays finite even if the graph becomes infinitely large! This becomes more drastic if one looks to cliques K_k with $k \geq 4$, these become more and more likely to be completely absent if N becomes large. Thus, if we look locally to induced subgraphs of any finite size k, these are almost surely trees or forests. This property is denoted as *locally tree-like*.

There are, however, loops, but these are of length $\mathcal{O}(\ln N)$ [4]. In the statistical physics approach, we will see that these loops are of fundamental importance, even if they are of diverging length.

Another side remark concerns the dimensionality of the graph, i. e., the possibility of "drawing" large graphs in a finite-dimensional space. If you look to any D-dimensional lattice, the number of neighbors up to distance k grows as k^D. On the other hand, in a tree of fixed average degree, this number is growing exponentially! In this sense, random graphs have to be considered to be infinite dimensional. This sounds strange at first, but finally explains the analytical tractability which will be observed later in this book.

2.4 The phase transition: Emergence of a giant component

From Sec. 2.2 we know that random graphs with $p = c/N$ are almost surely not connected. What can we say about the components?

Having in mind the growth process described above, we may imagine that for small c there are many small components, almost all of them being trees. If c increases, we add new edges and some of the previously disconnected components now become connected. The number of components decreases, the number of vertices (order) of the components grows. Concentrating on the largest component $L^{(0)}(G)$ of a graph G, the following theorem was demonstrated in 1960 by Erdős and Rényi [3]:

Theorem: *Let $c > 0$ and G_c drawn from $\mathcal{G}(N, c/N)$. Set $\alpha = c - 1 - \ln c$.*
(i) If $c < 1$ then we find almost surely

$$|L^{(0)}(G_c)| = \frac{1}{\alpha} \ln N + \mathcal{O}(\ln \ln N)$$

(ii) If $c > 1$ we almost surely have

$$|L^{(0)}(G_c)| = \gamma N + \mathcal{O}(N^{1/2})$$

with $0 < \gamma = \gamma(c) < 1$ being the unique solution of

$$1 - \gamma = e^{-c\gamma} \ .$$

All smaller components are of order $\mathcal{O}(\ln N)$.

This theorem makes a very powerful statement: As long as $c < 1$, all components have order up to $\mathcal{O}(\ln N)$. This changes markedly if $c > 1$. There appears one *giant component* connecting a finite fraction of all vertices, but all the other components are still small compared to the giant one, they have only $\mathcal{O}(\ln N)$ vertices. This means that at $c = 1$ the system undergoes a *phase transition*. In contrast to physical phase transitions, it is not induced by a change in temperature, pressure or other *external* control parameters, but by the change of the average vertex degree c, i. e., by a *structural* parameter of the graph. Due to the obvious analogy to percolation theory [5], this phase transition is also called *random-graph percolation* in the literature.

This theorem is one of the most fundamental results of random-graph theory, but we do not present a complete proof. There is, however, a simple argument that the component structure changes at $c = 1$. It is based on the locally tree-like structure of all components, and is presented in the limit $N \to \infty$.

If we select any vertex, on average it will have c neighbors. According to Eq. (12), each of these will have c additional neighbors, the first vertex thus has, on average, c^2 second neighbors. Repeating this argument, we conclude that the expected number of kth neighbors equals c^k (A vertex j is called a kth neighbor of a vertex i if the minimum path in the graph G connecting both contains exactly k edges).

Now we can see the origin of the difference for $c < 1$ and $c > 1$. In the first case, the prescribed process decreases exponentially, and it is likely to die out after a few steps. If, in contrast, we have $c > 1$, the expected number of kth neighbors grows exponentially. Still, there is a finite probability that the process dies out after a few

steps (e. g., e^{-c} if dying in the first step, i. e., if there are no neighbors at all), but there is also a finite probability of proceeding for ever.

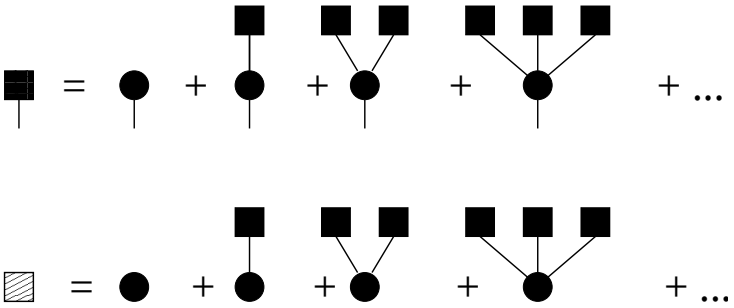

Figure 5: Schematic representation of the iterative solution for the probability that a vertex does not belong to the giant component. The first line shows the self-consistent equation for a vertex reached by a random edge: The black square with the half-edge represents the probability that the vertex reached is not connected to the giant component by one of its excess edges. This can happen because it has no further neighbors, or it is connected to one, two, etc., vertices not connected with the giant component. The second line shows the resulting equation for a randomly selected vertex, as represented by the shaded square.

This argument can be made more rigorous by considering the following iterative construction: Fist we calculate the probability π, that a randomly selected end-vertex of a randomly selected link is not connected via other edges with the giant component of the graph. In Fig. 5 this is represented by a black square connected to a half-edge. This vertex can either have degree one, i. e., it has no further incident edges which would be able to connect it to the giant component, or it is connected to other vertices which themselves are not connected to the giant component by their excess edges. Having in mind that almost all small connected components are trees, these neighbors are connected with each other only via the selected vertex, and the probability that d neighbors are not connected to the giant component equals simply π^d. Using the probability distribution q_d Eq. (12) of degrees of vertices reached by random edges, we thus find:

$$\begin{aligned} \pi &= q_1 + q_2\pi + q_3\pi^2 + q_4\pi^3 + \dots \\ &= \sum_{d=1}^{\infty} e^{-c}\frac{c^{d-1}}{(d-1)!}\ \pi^{d-1} = e^{-c}\sum_{d'=0}^{\infty}\frac{(c\pi)^{d'}}{(d')!} \\ &= e^{-c(1-\pi)}\ . \end{aligned} \tag{14}$$

This quantity can thus be determined self-consistently for every value of c, and allows us to calculate the probability $1-\gamma$ that a randomly selected vertex does not belong to the giant component. Applying similar arguments to before, this vertex can either

be isolated, or connected only to other vertices which, via their excess edges, are not connected to the giant component, see the second line of Fig. 5. We thus find

$$\begin{aligned} 1-\gamma &= p_0 + p_1\pi + p_2\pi^2 + p_3\pi^3 + \dots \\ &= \sum_{d=1}^{\infty} e^{-c}\frac{c^d}{d!}\,\pi^d \\ &= e^{-c(1-\pi)}\,. \end{aligned} \tag{15}$$

From these equations we see first that, for our random graph, $\pi = 1-\gamma$. Plugging this into the last line, we obtain the equation

$$1-\gamma = e^{-c\gamma} \tag{16}$$

as given in the theorem.

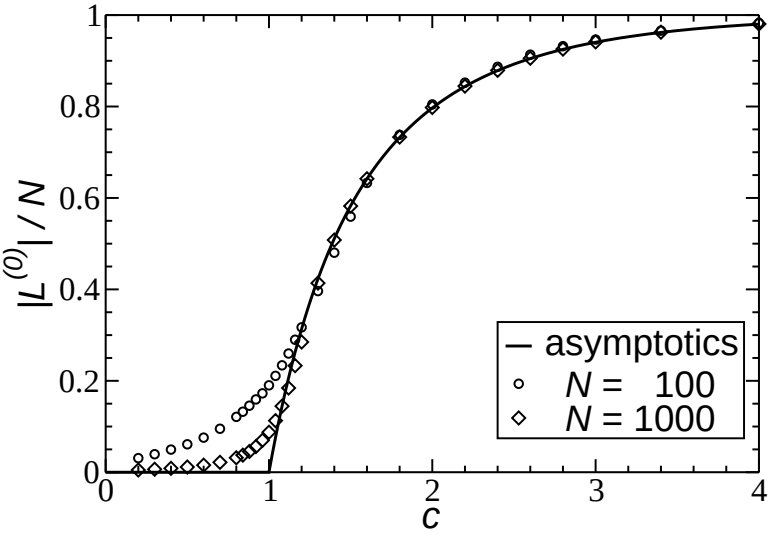

Figure 6: Fraction of vertices belonging to the largest component of a random graph, as a function of the average degree c. The symbols are numerical data for $N = 100, 1000$, averaged always over 100 randomly generated graphs. The full line gives the asymptotic analytical result: Below $c = 1$, the largest component is sub-extensive, above it is extensive.

Let us shortly discuss the shape of $\gamma(c)$. For $c = 1+\varepsilon$, with $0 < \varepsilon \ll 1$, we also expect γ to be small, and expand the above equation to second order in γ:

$$1-\gamma = 1-(1+\varepsilon)\gamma + \frac{1}{2}(1+\varepsilon)^2\gamma^2 + \mathcal{O}(\gamma^3)\,. \tag{17}$$

The term $1-\gamma$ cancels on both sides. Dividing by γ, and keeping only the first order in ε and γ, we thus find

$$\gamma = 2\varepsilon \ . \tag{18}$$

The relative order of the giant component thus starts to grow linearly in $c-1$ (we say the critical exponent for the giant component equals one), and converges exponentially fast to 1 for larger c. The full curve is given in Fig. 6, together with numerical data for finite random graphs.

3 Optimization Algorithms for Vertex Cover

The vertex-cover problem is an NP-hard problem in graph theory, as we have seen before in the complexity-theoretic chapter. As a reminder, we recall the definition and introduce some further concepts which are useful in the description of algorithms.

To do so, we take a dynamical view of the covering process, since any algorithm for determining covers exhibits some kind of dynamic. One can imagine that the algorithm places *covering marks* at the vertices of a graph $G = (V, E)$, one after the other. Let, at a certain time, V' be a set collecting these vertices. We will denote the members of V' *covered*, and all other vertices *uncovered*. Since it is the aim to place covering marks on at least one end-point of each edge, an edge $\{i, j\} \in E$ is analogously called *covered* iff at least one of its end-vertices is *covered*, $i \in V'$ or $j \in V'$. The edge is called uncovered iff both end-points are uncovered. According to the definition, V' thus constitutes a *vertex cover* iff *all* edges are covered. In this case, we call the graph covered as well.

Note that vertex covers always exist, e. g., the full vertex set V is always a vertex cover. They are not unique, because deleting, e. g., for each vertex $i \in V$, the set $V \setminus \{i\}$ is a vertex cover as well.

Example: Vertex cover

Consider the graph shown in the left half of Fig. 7. Vertices 1 and 2 are *covered* ($V' = \{1, 2\}$), while the other vertices $3, 4, 5$ and 6 are *uncovered*. Thus, edges $\{1, 3\}$, $\{1, 4\}$ and $\{2, 3\}$ are *covered* while edges $\{3, 4\}$, $\{3, 5\}$, $\{4, 6\}$ and $\{5, 6\}$ are *uncovered*. Hence, the graph is not *covered*.
In the right half of Fig. 7 vertices 4 and 5 are also *covered*. Thus, edges $\{3, 4\}$, $\{3, 5\}$, $\{4, 6\}$ and $\{5, 6\}$ are now *covered* as well. This means all edges are *covered*, i. e., the graph is *covered* by $V_{\text{vc}} = \{1, 2, 4, 5\}$, thus V_{vc} is a vertex cover.

The *vertex-cover decision problem* asks whether, for a given graph G, there are VCs V_{vc} of fixed given cardinality $X = |V_{\text{vc}}|$, we define $x = X/N$. In other words we are interested if it is possible to cover all edges of G by covering xN suitably chosen vertices, i. e., by distributing xN covering marks. To measure the extent a graph is

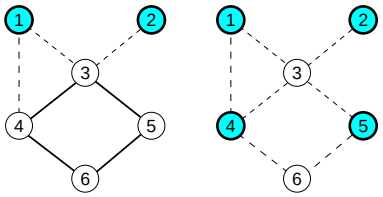

Figure 7: Graphs and covers. *Covered* vertices are shown in bold and dark, *covered* edges are indicated by dashed lines. Left: a partially covered graph. Vertex 1 and 2 are *covered*. Thus, edges $\{1,3\}$, $\{1,4\}$, and $\{2,3\}$ are *covered*. Right: If we also cover vertices 4 and 5, the graph becomes covered.

not covered, we introduce a *cost function* mapping arbitrary vertex subsets $V' \subset V$ to the number of uncovered edges,

$$H(V') = |\{\{i,j\} \in E \mid i,j \notin V'\}| \,, \tag{19}$$

and the corresponding constraint minimum

$$E(G,x) = Ne(G,x) = \min\{H(V') \mid V' \subset V,\ |V'| = xN\} \tag{20}$$

as the minimum realizable cost of putting exactly xN covering marks. Thus, a graph G is coverable by xN vertices if $e(G,x) = 0$. This means that you can answer the VC decision problem by first solving a *minimization problem*, and then testing whether or not the minimum $e(G,x)$ is zero.

For the preceding case, the energy was minimized with fixed $X = xN$. The decision problem can also be solved by solving another *optimization problem*. For a given graph G, we look for the *minimum vertex cover* , i. e., a vertex cover of minimum size

$$X_\mathrm{c}(G) = Nx_\mathrm{c}(G) = \min\{|V'| \mid H(V') = 0\} \,. \tag{21}$$

Thus, here the number of vertices in the subset is minimized, while the energy is kept at zero. The answer to the vertex cover decision problem is "yes" iff $X \geq X_\mathrm{c}$.

In the next two sections, numerical methods to solve the vertex-cover problem are presented. Note that we have to distinguish between the two presented ways to state the problem. Either we minimize the energy at given subset cardinality, or we directly construct a minimum VC. We always present the algorithms in a form in which they are suitable for the second kind of problem. Afterwards, we outline how the methods can be changed to treat problems of the first kind.

3.1 Heuristic algorithms

Let us start with two heuristics. We introduce a fast heuristic, which will be utilized also within the exact algorithm discussed in the next section. It can, however, be applied stand-alone as well. In this case only an approximation of the true minimum

vertex cover is calculated, which is, empirically, found to differ only by a few percent from the exact value. No exact bounds are available for this method, i. e., there is no rigorous control of its performance. For this reason, we present another approximation algorithm, which allows for a bound with respect to the true optimum. Unfortunately, the bound is not very good. All methods can easily be implemented in C/C++ via the help of the LEDA library [6] or the Boost library [7] which offer many useful data types and algorithms for graph problems.

We begin with a fast greedy heuristic. The basic idea is to cover as many edges as possible by using as few vertices as necessary. Thus, it is favorable to cover vertices with a high degree. This step can be iterated, while the degree of the vertices is adjusted dynamically by removing edges which are *covered*. This leads to the following greedy algorithm, which returns an approximation of the minimum vertex cover $V_{\rm vc}$, where the size $|V_{\rm vc}|$ is an upper bound of the true minimum vertex-cover size:

algorithm greedy-cover($G = (V, E)$
begin
  initialize $V_{\rm vc} = \emptyset$;
  while there are *uncovered* edges (i. e., $E \neq \emptyset$) **do**
  begin
    take one vertex i of highest current degree d_i;
    mark i as *covered*: $V_{\rm vc} = V_{\rm vc} \cup \{i\}$;
    remove from E all edges $\{i, j\}$ incident to i;
  end;
  return($V_{\rm vc}$);
end

Example: Greedy heuristic

To demonstrate how the heuristic operates, we consider the graph shown in Fig. 8. The vertices 3,7 and 8 have maximum degree 3. Let us assume that vertex 3 is first covered. Hence, the incident edges $\{1, 3\}$, $\{3, 4\}$, and $\{3, 6\}$ are removed. Still, vertices 7 and 8 have the highest degree 3. We assume that in the second iteration vertex 7 is covered, resulting in deleting edges $\{4, 7\}$, $\{6, 7\}$ and $\{7, 8\}$. In the two final iterations, e. g., vertices 5 and 8 may be covered. Then the algorithm stops, because all edges are *covered*. This cover has size 4, and is indeed a minimum-vertex cover.

In the preceding example we have seen that the heuristic is sometimes able to find a true minimum vertex cover. But this is not always the case. In Fig. 9 a simple counterexample is presented, where the heuristic fails to find the true minimum vertex cover. First, the algorithm covers the root vertex, because it has degree 3. Thus,

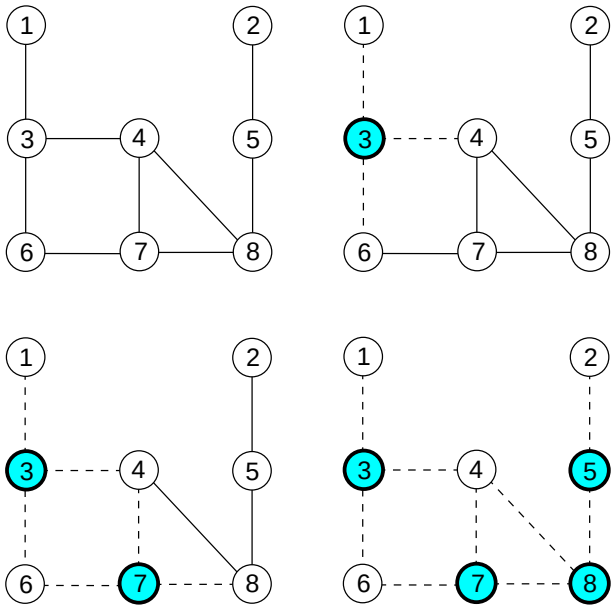

Figure 8: Example of the vertex-cover heuristic. Upper left: Initial graph. Upper right: Graph after the first iteration, vertex 3 has been covered (shown in bold) and the incident edges have been removed (shown with dashed line style). Bottom: Graph after second and fourth (final) iteration.

three additional vertices have to be subsequently covered, i. e., the heuristic covers four vertices. But the minimum vertex cover has only size 3, as indicated in Fig. 9.

The heuristic can easily be altered for the case in which the number X of *covered* vertices is fixed and we ask for a minimum number of uncovered edges. Then the iteration has to stop as well, when the size of the cover set has reached X. In case a vertex cover is found before X vertices are covered, arbitrary vertices can be added to the vertex-cover set V_{vc} until $|V_{\text{vc}}| = X$.

Unfortunately, it has not been possible so far to derive a rigorous bound on how the result of the heuristic compares to the true minimum vertex cover. Deriving such a bound is possible for the following algorithm [8], although the bound is not very good. The algorithm is based on the relationship between vertex covers and matchings. We recall from Sec. 1, that a matching is a subset of edges, such that each vertex is contained at most once in each matching. The following algorithm stores the vertex cover being built in V_{vc} and the matching in M.

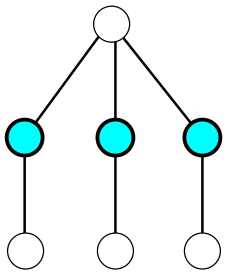

Figure 9: A small sample graph with minimum vertex cover of size 3. The vertices belonging to the minimum $V_{\rm vc}$ are indicated by dark/bold circles. For this graph, the greedy heuristic fails to find the true minimum cover, because it starts by covering the root vertex, which owns the highest degree 3.

algorithm 2-approximation($G = (V, E)$
begin
  initialize $V_{\rm vc} = \emptyset$;
  initialize $M = \emptyset$;
  while there are *uncovered* edges (i. e., $E \neq \emptyset$) **do**
  begin
    take one arbitrary edge $\{i, j\} \in E$;
    mark i and j as *covered*: $V_{\rm vc} = V_{\rm vc} \cup \{i, j\}$;
    add $\{i, j\}$ to the matching: $M = M \cup \{\{i, j\}\}$;
    remove from E all edges incident to i or j;
  end;
  return($V_{\rm vc}$);
end

Example: 2-Approximation heuristic

To demonstrate, how the 2-approximation heuristic operates, we consider again the graph from the previous example, see Fig. 10. We assume that the algorithm first takes edge $\{3, 4\}$, hence after the first iteration $V_{\rm vc} = \{3, 4\}$, $M = \{\{3, 4\}\}$ and the edges $\{1, 3\}$, $\{3, 4\}$, $\{3, 6\}$ and $\{4, 7\}$ are covered and removed from the graph. In the next iteration edge $\{7, 8\}$ may be chosen, hence we have now $V_{\rm vc} = \{3, 4, 7, 8\}$, $M = \{\{3, 4\}, \{7, 8\}\}$ and edges $\{4, 8\}$, $\{5, 8\}$, $\{6, 7\}$ and $\{7, 8\}$ are covered and removed from E. Now only edge $\{2, 5\}$ is left. Hence, after the final iteration, we have $V_{\rm vc} = \{2, 3, 4, 5, 7, 8\}$ and $M = \{\{3, 4\}, \{7, 8\}, \{2, 5\}\}$.

Note that, in the case when the algorithm "chose" the edges, e. g., in the order $\{1, 3\}$, $\{2, 5\}$, $\{6, 7\}$ and $\{4, 8\}$, then the vertex cover would contain

all eight vertices, i. e., twice the size of the minimum vertex cover. We will show below that the algorithm never achieves a worse result.

On the other hand, the 2-approximation algorithm will never be able to "find" the minimum cover for this graph, because in the minimum cover, e. g., $V_{\mathrm{vc}}^{\mathrm{min}} = \{3, 5, 7, 8\}$, there are always vertices which have no neighbor in $V_{\mathrm{vc}}^{\mathrm{min}}$. This is not possible within the 2-approximation algorithm, because pairs of neighbors are always added to V_{vc}.

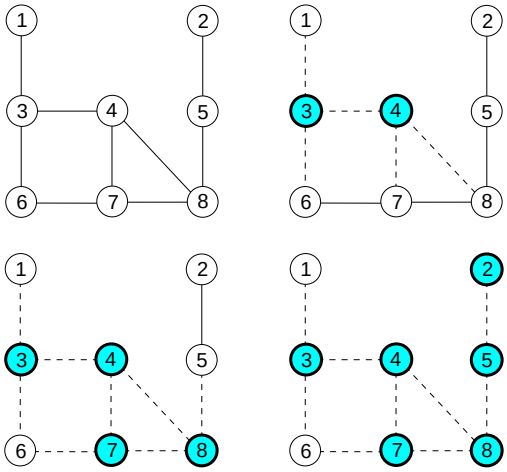

Figure 10: Example of the 2-approximation heuristic. Upper left: initial graph. Upper right: graph after the first iteration, vertices 3 and 4 have been covered (shown in bold) and the incident edges removed (shown with dashed line style). Bottom: graph after second and final iteration.

Each edge is touched exactly once by the 2-approximation algorithm, hence the running time is of order $\mathcal{O}(|E|)$, if one assumes a constant execution time for all fundamental operations. Furthermore, the algorithm removes within the loop only *covered* edges from E. Since the algorithm halts when E is empty, all edges are covered, hence V_{vc} is a vertex cover.

The size $|V_{\mathrm{vc}}|$ of the vertex cover is, at most, twice the size of the minimum vertex cover $V_{\mathrm{vc}}^{\mathrm{min}}$: $|V_{\mathrm{vc}}| \leq 2|V_{\mathrm{vc}}^{\mathrm{min}}|$.

Proof:

The algorithm also constructs a matching M. Since the algorithm adds two vertices to V_{vc} for each edge which is added to M, we have exactly

$$|V_{\mathrm{vc}}| = 2|M|\,. \tag{22}$$

Since no vertex appears twice in the edges of the matching by definition, i. e., the edges do not "touch" each other, one has to cover at least one vertex per edge of any

matching, hence also per edge of M. This means for the minimum vertex cover $V_{\rm vc}^{\rm min}$ we have

$$|V_{\rm vc}^{\rm min}| \geq |M| \,. \tag{23}$$

Combining Eqs (22) and (23) we get $|V_{\rm vc}| = 2|M| \leq 2|V_{\rm vc}^{\rm min}|$. QED

3.2 Branch-and-bound algorithm

So far we have presented two simple heuristics to find approximations of minimum vertex covers. Next, an exact algorithm is explained: *branch-and-bound*, which incorporates the first heuristic to gain high efficiency. Without the heuristic, the algorithm would still be exact, but it would run considerably slower.

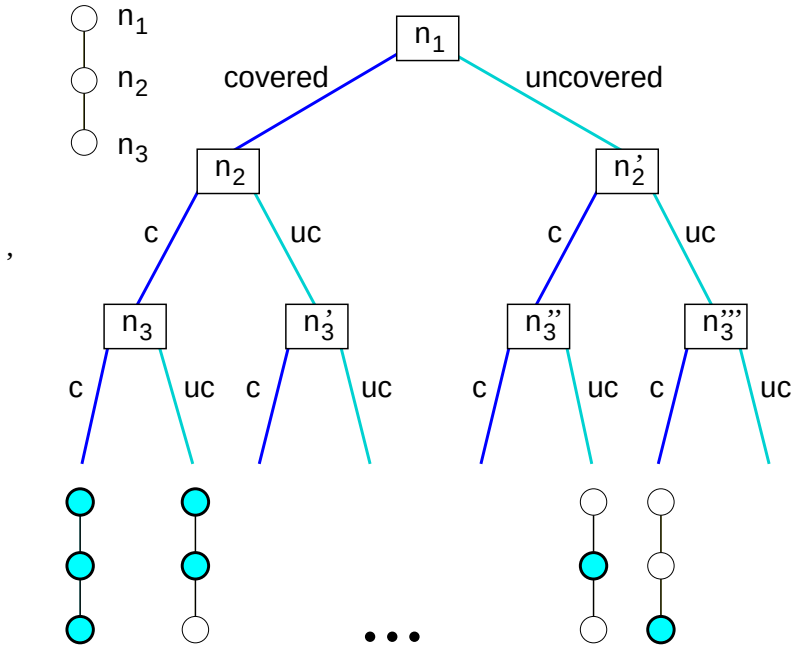

Figure 11: Binary configuration tree for the VC. Each node of the configuration tree corresponds to a vertex which is either *covered* ("left subtree") or *uncovered* ("right subtree").

The basic idea of the method is as follows. Again we are interested in a VC of minimum size. As each vertex becomes either *covered* or *uncovered*, there are 2^N possible configurations which can be arranged as leaves of a binary configuration tree, see Fig. 11. At each node, the two subtrees represent the subproblems where the corresponding vertex is either *covered* ("left subtree") or *uncovered* ("right subtree"). Vertices which have not been touched at a certain level of the tree, are said to be *free*. Note that for different nodes on the same level of the tree, the vertices corresponding

to the node do not have to be the same in different subtrees, e. g., n_2 may be different from n_2'. This depends on the heuristic which is used to determine the current vertex at each node of the configuration tree. The algorithm does not have to descend further into the tree when a cover has been found, i. e., when all edges are *covered*. Then the search continues in higher levels of the tree for a cover which has possibly a smaller size, i. e., *backtracking* occurs. Since the number of nodes in a tree grows exponentially with system size, algorithms which are based on configuration trees have a running time which may grow exponentially with the system size. This is not surprising, since the minimum-VC problem is NP-hard, so all known exact methods exhibit an exponential growing worst-case time complexity.

To decrease the running time, the algorithm presented below makes use of the fact that only proper vertex covers are to be obtained. Therefore, when a vertex i is marked as *uncovered*, all neighboring vertices can be *covered* immediately. For these vertices, only the left subtrees are present in the configuration tree, hence the size of the tree is already reduced.

A further substantial speedup can be obtained by applying the branch-and-bound approach [9,10]. The idea is that some subtrees of the configuration tree are omitted, i. e., they are not visited at all, by introducing a *bound*. This is achieved by storing three quantities, assuming that the algorithm is currently found at some node in the configuration tree:

- The *best* size of the smallest vertex cover found in subtrees visited so far (initially $best = N$).
- X denotes the number of vertices which have been covered so far (in higher levels of the tree).
- A table of *free* vertices indexed by the *current* degree d_i is always kept, i. e., for each vertex, the number of currently *uncovered* incident edges.

Thus, to achieve a better solution than *best*, at most $F = best - X$ vertices are allowed to additionally be covered in a subtree of the current node. The number of edges coverable in this way is obviously bounded from above by the sum $D = \sum_{l=1}^{F} d_l$ of the F highest current degrees. If this number is smaller than the total number of not yet covered edges, we know that the subtree cannot contain a smaller VC of the full graphs. It can be omitted for sure. Note that some edges may exist between the F vertices of the highest current degree, they are counted twice in D. Therefore the bound is not necessarily tight, a subtree may be entered, even if it contains no smaller VC. Other types of bounds are mentioned in Ref. [10], e. g., the size of a maximal matching is a lower bound for a minimum vertex cover.

The algorithm can be summarized as below. The size of the smallest cover is stored in *best*, which is passed by reference (i. e., the variable, not its value is passed). The current number of *covered* vertices is stored in variable X:

```
algorithm branch-and-bound(G, best, X)
begin
   if all edges are covered then
   begin
      if X < best then best := X
      return;
   end;
   calculate F = best − X; D = Σ_{l=1}^{F} d_l;
   if D < number of uncovered edges then
      return;          comment bound;
   take one free vertex i with the largest current degree d_i;
   mark i as covered; comment left subtree
   X := X + 1;
   remove from E all edges {i, j} incident to i;
   branch-and-bound(G, best, X);
   reinsert all edges {i, j} which have been removed;
   X := X − 1;
   if (F > number of current neighbors) then
   begin              comment right subtree;
      mark i as uncovered;
      for all neighbors j of i do
      begin
         mark j as covered; X := X + 1;
         remove from E all edges {j, k} incident to j;
      end;
      branch-and-bound(G, best, X);
      for all neighbors j of i do
         mark j as free; X := X − 1;
      reinsert all edges {j, k} which have been removed;
   end;
   mark i as free;
   return;
end
```

Example: Branch-and-bound algorithm

Here we consider again the graph from Fig. 8. During the first descent into the configuration tree, the branch-and-bound algorithm operates exactly as the heuristics. Iteratively vertices of highest current degree are taken, covered, and the incident edges removed. The recursion stops the first time the graph is covered. This situation is shown in Fig. 12, where the graph and the corresponding current configuration tree are displayed. Since $X = 4$ vertices have been covered, $best := 4$.

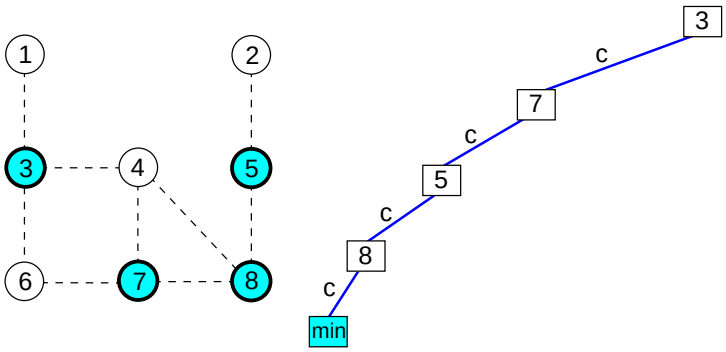

Figure 12: Example of the branch-and-bound algorithm. Result after the first full cover has been found. Left: graph. Right: configuration tree. In the graph, covered vertices are shown by bold and dark circles, covered edges indicated by dashed lines. The current node of the configuration tree is highlighted as well, c=*cover*, uc=*uncover*.

Then the algorithm returns to the preceding level of the configuration tree. Vertex 8 is now set *uncovered*. All its *uncovered* neighbours are *covered*, i. e., vertex 4 in this case. The resulting situation is shown in Fig. 13. At the next level of the configuration tree, it is detected that again a full vertex cover has been found, but not a smaller one. Hence, the algorithm backtracks to the previous level.

Both possibilities for vertex 8 have been considered, hence vertex 8 is set to *free* again and the algorithm backtracks another level. Now the second possibility for vertex 5 is considered, i. e., it is *uncovered*, while its neighbours, vertices 2 and 8 are *covered*, see Fig. 14. Again a vertex cover of size 4 has been found, no further descent into the tree beyond the next level is necessary.

This means that the treatment of vertex 5 is finished, it is *freed* again, and the algorithm backtracks one level, to continue considering vertex 7. It is *uncovered*, and its neighbours, vertices $4, 6$ and 8 are *covered*, see Fig. 15. In the subsequent recursive call, it is found that no cover has been found, because edge $\{2, 5\}$ is uncovered. Hence, the bound is evalated. Since the current number of *covered* vertices is $X = 4$, we obtain $F = best - X = 0$. This results trivially in $D = 0$, which is smaller than the number of *uncovered* edges. Therefore, the bound becomes active, the left and right subtrees of the current node are omitted, and the algorithm backtracks to the previous level of the configuration tree.

Thus, the treatment of vertex 7 is finished and the algorithm backtracks further to the top level and *uncovers* vertex 3. Thus, its neighbours, vertices 1,4 and 6 are *covered*, see Fig. 16. Again no cover has been

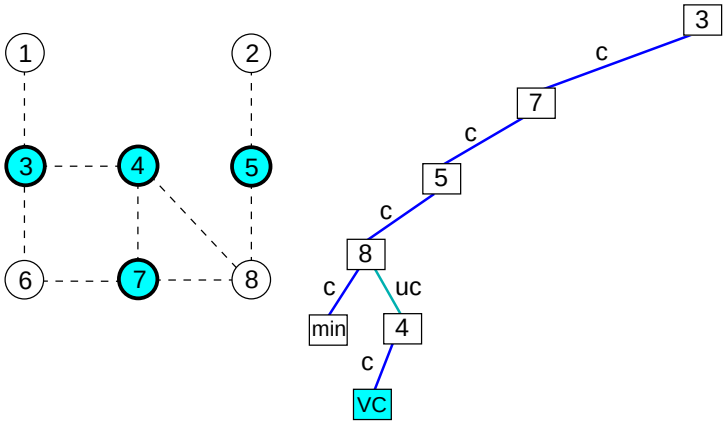

Figure 13: Example of the branch-and-bound algorithm: Situation after vertex 8 has been *uncovered* and its neighbour 4 has been *covered*. A new vertex cover has been found, but not a smaller than before, indicated by a "VC" in the current node.

obtained, hence the bound is evaluated during the next recursive call to the algorithm. Now we have $X = 3$ vertices *covered*, hence we can cover $F = best - X = 4 - 3 = 1$ additional vertices, i. e., one. Vertex 8 has the highest current degree $d_8 = 2$, hence $D = 2$. But the number of uncovered edges is 3. Thus, the bound becomes active in a non-trivial way, the algorithm returnes to the top level and terminates.

Hence, the minimum vertex cover has indeed size $X = 4$, as found by the heuristics. Note that the configuration tree contains only 18 nodes, compared to 511 nodes (with $2^8 = 256$ leaves) of the complete configuration tree.

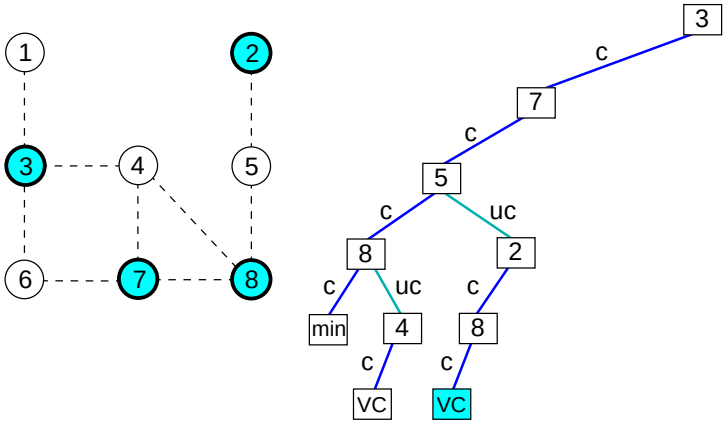

Figure 14: Example of the branch-and-bound algorithm: Situation after vertex 5 has been *uncovered* and its neighbours 2,8 have been *covered*.

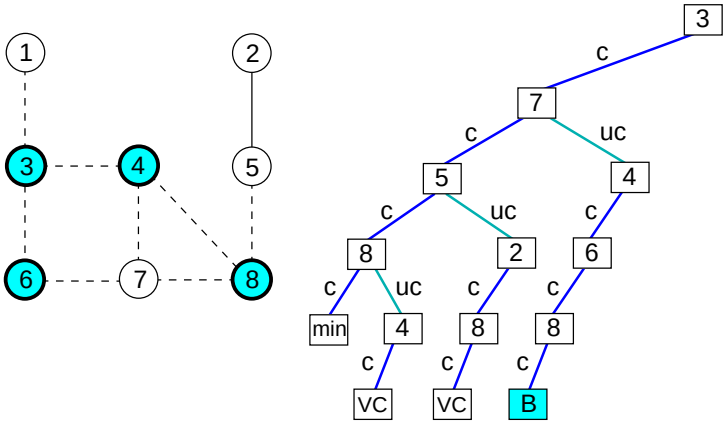

Figure 15: Example of the branch-and-bound algorithm: Situation after vertex 7 has been *uncovered* and its neighbours 4,6,8 have been *covered*. Here the bound is active, indicated by a "B" in the current node of the configuration tree.

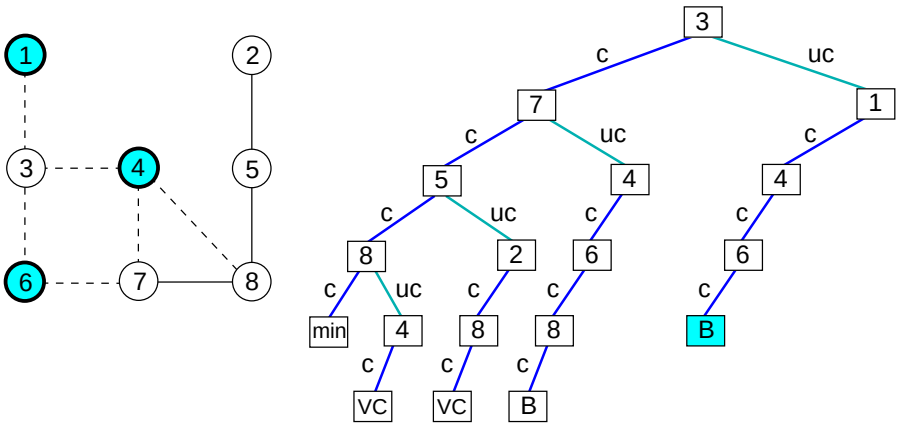

Figure 16: Example of the branch-and-bound algorithm. Siutation after vertex 3 has been *uncovered* and all its neighbours *covered*. Again the bound becomes active.

For every calculation of the bound, one has to access the F vertices of largest current degree. Therefore, it is favorable to implement the table of vertices as two arrays v_1, v_2 of sets of vertices. The arrays are indexed by the current degrees of the vertices. The sets in the first array v_1 contain the F *free* vertices of the largest current degree, while the other array contains all other *free* vertices. Every time a vertex changes its degree, it is moved to another set, and eventually even changes the array. Also, in case the mark (*free/covered/uncovered*) of a vertex changes it may be entered in or removed from an array and possibly the smallest degree vertex of v_1 is moved to v_2 or vice versa. Since we are treating graphs of finite average connectivity, this implementation ensures that the running time spent in each node of the graph is growing slower than linearly in the graph size[1]. For the sake of clarity, we have omitted the update operations for both arrays from the algorithmic presentation.

The algorithm, as it has been presented, is suitable for solving the optimization problem, that is, finding the smallest feasible size $X_\mathrm{c} = Nx_\mathrm{c}$ of a vertex cover, i. e., the minimum number of *covered* vertices needed to cover the graph fully. The algorithm can be easily modified to treat the problem, where the size $\tilde{X} = |V_\mathrm{vc}|$ is given and a configuration with minimum energy is to be found, i. e., the case where the graph may not be fully coverable. Then, in *best*, not the current smallest size of a vertex cover but the smallest number of uncovered edges (i. e., the energy) is stored. If X again denotes the current number of covered vertices at any node in the configuration tree, the algorithm can cover $F = \tilde{X} - X$ additional vertices. Again $D = \sum_{l=1}^{F} d_l$ is

[1] Efficient implementation of *sets* requires at most $\mathcal{O}(\log S)$ time for the operations, where S is the size of a set. In this case also a double indexed structure is possible allowing all operations to be performed in constant time. Double indexed structure means that sets are also implemented as arrays and, for each vertex, the current position in the corresponding array must be stored.

the sum of the F highest degrees. A subtree should not be entered, if the number of uncovered edges so far minus the maximum possible number of edges coverable within the subtree is larger than the current optimum. Hence, the bound becomes active, if $D + best$ is smaller than or equal to the current number of uncovered edges. Furthermore, when a vertex is *uncovered*, the step where all neighbors are *covered* cannot be applied, because the configuration of the lowest energy may not be a VC. On the other hand, if a VC has been found, i. e., all edges are covered, the algorithm can stop, because no better solution can definitely be obtained. But the algorithm can stop only for the case when *one* optimum has to be obtained. If all minima have to be enumerated, the algorithm has to proceed and the bound becomes active only when $D + best$ is strictly smaller (not equal) to the current number of *uncovered* edges.

4 Monte Carlo simulations

In this section, we will introduce another method, called *Monte Carlo* (MC) *simulation*, which is a general simulation approach used to study computationally models in statistical physics. Here we are able to give only a short introduction, good and complete text books on this subject were written, e. g., by Newmann and Barkema [11], or by Landau and Binder [12]. When applied to VC, MC simulations allow for the calculation of approximations of minimum vertex covers. For Erdős–Rényi random graphs, the method does usually very well and one is able to find true minimum vertex covers for *all* connectivities, at least for sizes where a comparison with exact results is possible.

Although being able to calculate approximations of vertex covers, the nature of MC simulations is substantially different from the heuristics presented in Sec. 3.1. The basic approach is to interpret the graph and its covers as a configurations of a physical system, a *hard-core lattice gas*, the exact definition is given below. Then the system is evolved by MC simulations under the rules of statistical mechanics in the *grand-canonical ensemble* characterized by a *chemical potential* μ. By increasing μ continuously, or performing simulations at different values of μ in parallel (i. e., using *parallel tempering*, see below), one can obtain minimum-size vertex covers. The fundamental difference from the heuristics presented before is that the simulation does not run in one linear sweep, but many iterations have to be performed. To obtain true minimum vertex covers, one has to tune some parameters, e. g., the number of iterations and the number of different values for the chemical potential, which determine the overall computer time. Now we will present the details steps by step. First we introduce the hard-core gas, then we explain *Markov chains*, which provide the theoretical background for Monte Carlo simulations, we then discuss the algorithm for the case of the hard-core gas, and at the end we explain how parallel tempering works.

4.1 The hard-core lattice gas

For the definition of the hard-core gas on a given graph $G = (V, E)$, we consider arbitrary covers V_{vc}, including those of larger magnitude than the minimum vertex cover. This means all edges have to be *covered*, i. e., at least at one end-point of any edge there is a covering mark. Now we define the *uncovered* vertices as *occupied* by *particles* of the gas on G. Since edges having both neighbors being *uncovered* are prohibited, it is not allowed by definition that both end-points of any edge are occupied by particles. This can be interpreted as the particles having a *chemical radius* of one, i. e., they exhibit a hard-core repulsion that prevents them from coming too close to each other. An example of a hard-core gas is depicted in Fig. 17. Note that the trivial cover, i. e., choosing $V_{\mathrm{vc}} = V$, corresponds to having an empty configuration without any particle. For a detailed discussion of this model see the analytical approach in the next chapter. There the equivalence between vertex covers and particle packings will be exploited to use statistical-mechanics methods in an analytic description of VCs over random graphs.

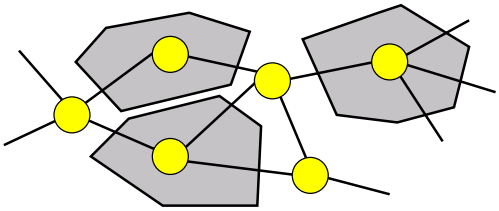

Figure 17: In an arbitrary vertex cover, every uncovered vertex can be seen as the position of a hard particle of chemical radius one. Due to the vertex cover constraint, no particles are allowed to overlap, i. e., they cannot occupy neighboring vertices.

4.2 Markov chains

The aim is to generate configurations of a given system, such that the generated configurations are distributed according to some given distribution. Here we want the hard-core gas to be sampled according to the grand-canonical distribution with chemical potential μ. Now we will describe the general formalisms of Markov Chains which can be used to perform this task.

The starting point is a system with a finite number of configurations $\{\underline{y}\}$ and given probabilities $P(\underline{y})$. Very often, as in the case of the hard-core gas, the configurations are vectors describing the states of a number N of vertices. Hence, the number of possible configurations is usually exponentially large in N. This means that typically $P(\underline{y}) \in \mathcal{O}(1)$ only for a few configurations, but the probability is exponentially small in N for most configurations. This will be an obstacle for a simple algorithm, as we will see.

The aim is the measurements of averages of observable quantities $A(\underline{y})$

$$\langle A\rangle := \sum_{\underline{y}} A(\underline{y})P(\underline{y})\,, \tag{24}$$

e. g., the average density of the hard-core gas. Since the number of configurations is assumed to be exponentially large in the number of degrees of freedom, a pure enumeration of all configurations is not feasible.

The most basic approach is to generate a certain number L of configurations $\{\underline{y}^i\}$ randomly, such that all configurations are equiprobable. Then we have:

$$\langle A\rangle \approx \overline{A}^{(a)} \equiv \sum_{\underline{y}^i} A(\underline{y}^i)P(\underline{y}^i)/\sum_{\underline{y}^i} P(\underline{y}^i)\,.$$

The equiprobable generation of the configurations is normally very easy. The problem is that usually the probability $P(\underline{y})$ is exponentially small, for almost all configurations, as already mentioned, hence the result $\overline{A}^{(a)}$ is very inaccurate. For the case of the hard-core gas, one can generate configurations independently and equiprobably by selecting the state *occupied/unoccupied* of each vertex independently, and then having $P(\underline{y}) = 0$ if the hard-core constraint is violated. This is obviously inefficient, since many configurations do not contribute at all.

It is better to generate the configurations in such a way that more important configurations, those with large probability, occur more often. This is called *importance sampling*. In the ideal case, this means that a number L of configurations $\{\underline{y}^i\}$ are generated independently such that they are immediately distributed according $P(\underline{y}^i)$. Then we can approximate the average directly by an arithmetic mean:

$$\langle A\rangle \approx \overline{A}^{(b)} \equiv \sum_{\underline{y}^i} A(\underline{y}^i)/L\,. \tag{25}$$

This direct way of generating random objects works in only few simple cases, e. g., when generating random numbers according to a Gaussian distribution. Unfortunately, such algorithms do not exists for most interesting models exhibiting many interacting degrees of freedom.

One way out of this problem is to use a *probabilistic dynamic* which generates a sequence (or chain) $\underline{y}(t)$ of configurations at discrete times $t = 0, 1, 2, \ldots$: $\underline{y}(0) \to \underline{y}(1) \to \underline{y}(2) \to \ldots$. We assume that configuration $\underline{y}(t+1)$ depends only on a (pseudo) random number and on the previous configuration $\underline{y}(t)$ in the chain. In this case $\{\underline{y}(t)|t = 0, 1, 2, \ldots\}$ is called a *Markov chain*. We describe the transitions $\underline{y}(t) \to \underline{y}(t+1)$ by *transition probabilities* $W_{\underline{y}\underline{z}} = W(\underline{y} \to \underline{z})$, i. e., the probability that the system moves from configuration $\underline{y}$ (at time t) to configuration $\underline{z}$ (at time $t+1$). For simplicity we assume furthermore that $W_{\underline{y}\underline{z}}$ does not depend on the time t explicitly. The transition probabilities have the following properties:

$$\begin{aligned} W_{\underline{y}\underline{z}} &\geq 0 \quad \forall \underline{y}, \underline{z} \quad (\textit{positivity}) \\ \sum_{\underline{z}} W_{\underline{y}\underline{z}} &= 1 \quad \forall y \quad (\textit{conservation})\,. \end{aligned} \tag{26}$$

The configuration space together with the transition probabilities is called a *Markov process*. Now we analyze the Markov process by introducing $P(\underline{y}, t)$, which is the the probability that the Markov process is at time t in configuration $\underline{y}(t) = \underline{y}$. We describe the change of the probability to be in state $\underline{y}$ when going from time t to $t+1$. For this purpose, we have to consider all transitions which go out of configuration $\underline{y}$, i. e., which decrease $P(\underline{y}, t+1)$, and the transitions which go into $\underline{y}$ from other configurations, i. e., which increase $P(\underline{y}, t+1)$, see Fig. 18.

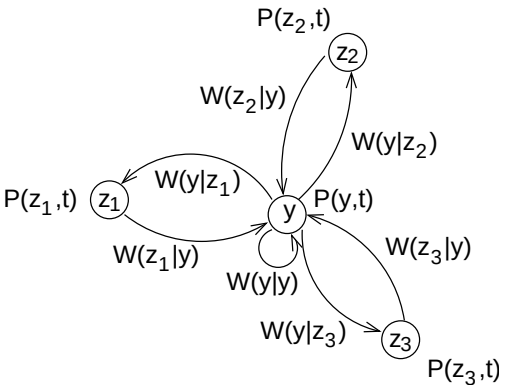

Figure 18: The change of the probability of being in configuration y in a Markov process is determined by the transitions into and out of the "neighboring" configurations and the probabilities of being in the "neighboring" configurations. An example with four configurations. Transitions having transition probability zero are not shown.

The change of probability for configuration $\underline{y}$ is then given by the *master equation*

$$\Delta P(\underline{y}, t) := P(\underline{y}, t+1) - P(\underline{y}, t) = \sum_{\underline{z}} W_{\underline{z}\underline{y}} P(\underline{z}, t) - \sum_{\underline{z}} W_{\underline{y}\underline{z}} P(\underline{y}, t) \quad \forall \underline{y}. \tag{27}$$

Under certain circumstances (e. g., if there is only one eigenvalue $\lambda = 1$ for the matrix $(W_{\underline{y}\underline{z}})$, see [13]), the probability distribution $P(\underline{y}, t)$ converges towards the *stationary* (time-independent) distribution

$$P_{ST}(\underline{y}) \equiv \lim_{t \to \infty} P(\underline{y}, t)\,.$$

It is independent of the starting configuration $\underline{y}(0)$. Such a system is called *ergodic*. Ergodicity means that there exists between any two configurations a path with all transitions along the path having non-zero transition probabilities. Hence, one can reach each configuration from all other configurations.

Now we want to set up the transition probabilities, such that the stationary distribution is the distribution P we want to have:

Target: Choose $W_{\underline{y}\underline{z}}$ such that $P_{ST} = P$

Since $P(.)$ is time-independent, it follows from Eq. (27):

$$0 = \Delta P(\underline{y}) = \sum_{\underline{z}} W_{\underline{z}\underline{y}} P(\underline{z}) - \sum_{\underline{z}} W_{\underline{y}\underline{z}} P(\underline{y}) \quad \forall \underline{y}\,.$$

This is a third type of condition for the transition probabilities. This condition means that the total change in time of the probability for each configuration is zero, i. e., the "flows" of probability in and out of the configurations balance out. There are many ways to fulfil this condition. One way is to request that the balance holds for all pairs of configurations, i. e.,

$$W_{\underline{z}\underline{y}} P(\underline{z}) - W_{\underline{y}\underline{z}} P(\underline{y}) = 0 \quad \forall \underline{y}, \underline{z}\,. \tag{28}$$

This condition is called *detailed balance*.

By obeying detailed balance, and having an ergodic system, it is guaranteed that the Markov process generates configurations, which are distributed according to $P(.)$ in the long-time limit. Hence one can take averages as in Eq. (25). Note that the stationary distribution is obtained formally only in the case $t \to \infty$. This means for practical situations that the configurations at the beginning of the Markov chain depend strongly on the initial configuration $\underline{y}(0)$. Therefore, one usually omits the first $t < t_{\mathrm{eq}}$ configurations from the calculation of averages. One says, the system has to be *equilibrated*. A suitable choice for t_{eq} usually has to be determined within the simulation of the model under investigation, e. g. one can measure correlations with $\underline{y}(0)$, or start two simulations with two atypical, strongly different initial configurations and wait till the measurable quantity one is interested in has converged in both cases to the same value, for details see [11, 12]. Furthermore $\underline{y}(t+1)$ is usually strongly correlated with $\underline{y}(t)$. Hence, only "distant" configurations $\underline{y}(t), \underline{y}(t+\Delta t), \underline{y}(t+2\Delta t), \ldots$ exhibit a small statistical correlation. Again Δt has to be determined empirically for each model, for each observable and for the current parameters.

4.3 Monte Carlo for hard-core gases

Now we present transition rules for the hard-core gas, such that the stationary distribution is the grand-canonical distribution. We describe a configuration by the vector $\underline{\nu} = \{\nu_i\}$ $(i \in V)$, with

$$\nu_i = \begin{cases} 1 & \text{for } i \text{ is occupied by a particle} \\ 0 & \text{else .} \end{cases} \tag{29}$$

To distinguish valid configurations from those where the hard-core constraint is violated, we use the following indicator function:

$$\chi(\underline{\nu}) = \prod_{\{i,j\} \in E} (1 - \nu_i \nu_j)\,. \tag{30}$$

The function χ takes value one whenever $\underline{\nu}$ corresponds to an allowed particle packing (or a VC) because all factors equal one. Whenever there are two neighboring vertices occupied by particles, i. e., $\nu_i = \nu_j = 1$, the value of χ equals zero.

Thus, the grand-canonical distribution is given by

$$P(\underline{\nu}) = \frac{1}{\Xi}\chi(\underline{\nu})e^{\mu \sum_i \nu_i} \tag{31}$$

where Ξ is the grand-canonical partition function, i. e., the normalization constant, μ the chemical potential and $\sum_i \nu_i$ evaluates the number of particles. In this distribution, packings of different numbers of particles are allowed, weighted according to these numbers. The system is said to be coupled to a particle bath. This non-constant particle number is important for our purpose, because we do not know the size X_c of a minimum VC before having constructed one.

For $\mu > 0$, configurations of higher particle number have higher statistical weight. This means that for $\mu \to \infty$, configurations with the highest density $\rho = \frac{1}{N}\sum_i \nu_i$ are obtained. The number of *unoccupied* vertices is minimized, hence a minimum vertex cover is obtained.

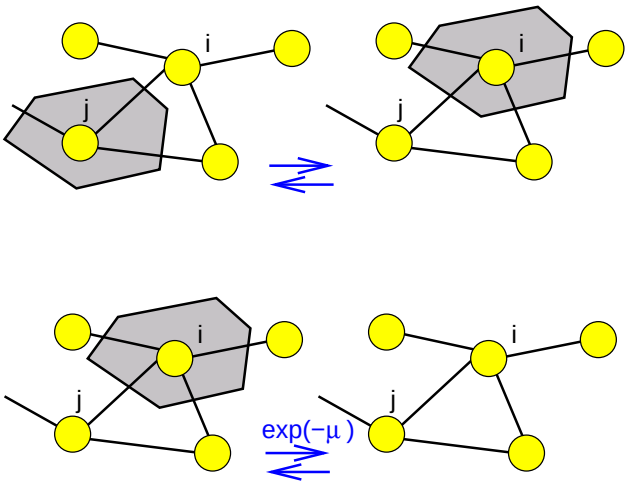

Figure 19: Two type of move are used for the hard-core lattice gas simulation. Top: particles are translated to neighboring vertices (move). Bottom: particles are inserted into/removed from the lattice (exchange).

The MC simulations consist of two types of transition: the "move" (M) transition and the "exchange" (E) transition. For each MC step, either the M or the E transition is performed, each with probability 1/2. For both types of step, a vertex i is selected randomly with probability $1/|V|$. The transitions proceed as follows, see also Fig. 19.

M If the vertex is not *occupied* by a particle ($\nu_i = 0$), *and* if exactly one neighboring vertex j is *occupied* ($\nu_j = 1$), then the particle at j is moved to vertex i ($\nu_i \leftarrow 1$,

$\nu_j \leftarrow 0$). In all other cases, nothing happens, i. e., the configuration remains unchanged.

E If vertex i is *unoccupied* and *all* neighbors are also *unoccupied*, then a particle is inserted on vertex i. If some neighbors are occupied, the configuration remains unchanged.

If vertex i is *occupied*, the particle is removed with probability $\exp(-\mu)$. This means a (pseudo) random number r is drawn which is uniformly distributed in the interval $[0, 1]$. If $r < \exp(-\mu)$, then the particle is removed, otherwise not.

Now, we want to prove that the algorithm fulfils detailed balance. Since the M and E transitions are independent of each other, it is sufficient to show that both types of transition fulfil detailed balance separately. The cases where the configuration does not change, obey detailed balance by definition, because for $\underline{\xi} = \underline{\zeta}$, Eq. (28) is fulfilled trivially. Therefore, we only have to consider the cases, where $\underline{\xi} \neq \underline{\zeta}$ and the M or the E transition is possible between the two configurations.

M Let $\underline{\xi}, \underline{\zeta}$ be two valid configurations (i. e., $\chi(\underline{\nu}) = \chi(\underline{\zeta}) = 1$), which are the same except $\xi_i = 0, \xi_j = 1$ and $\zeta_i = 1, \zeta_j = 0$. For the M transition, the move $\underline{\xi} \to \underline{\zeta}$ is performed if vertex i is selected, i. e., with probability $W_{\underline{\xi}\underline{\zeta}} = 1/|V|$. Similarly, the move $\underline{\zeta} \to \underline{\xi}$ is performed if vertex j is selected, i. e., $W_{\underline{\zeta}\underline{\xi}} = 1/|V| = W_{\underline{\zeta}\underline{\xi}}$. Since the number of particles does not change by this transition, we have trivially $P(\underline{\xi}) = P(\underline{\zeta})$. It follows $W_{\underline{\zeta}\underline{\xi}} P(\underline{\zeta}) - W_{\underline{\xi}\underline{\zeta}} P(\underline{\xi}) = 0$, i. e., detailed balance is fulfilled according to Eq. (28).

E Let $\underline{\xi}, \underline{\zeta}$ be two valid configurations, which are the same except $\xi_i = 0$ and $\zeta_i = 1$. For the E transition, $W_{\underline{\xi}\underline{\zeta}} = 1/|V|$ and $W_{\underline{\zeta}\underline{\xi}} = \exp(-\mu)/|V|$. Furthermore we have

$$\begin{aligned} P(\underline{\xi}) &= Z^{-1} \chi(\underline{\xi}) \exp\Big(\mu \sum_i \xi_i\Big) \\ &= Z^{-1} \chi(\underline{\xi}) \exp\Big(-\mu) \exp(\mu(1 + \sum_i \xi_i)\Big) \\ &= Z^{-1} \chi(\underline{\zeta}) \exp\Big(-\mu) \exp(\mu \sum_i \zeta_i\Big) \\ &= \exp(-\mu) P(\underline{\zeta})\,. \end{aligned}$$

Hence, we get $W_{\underline{\zeta}\underline{\xi}} P(\underline{\zeta}) - W_{\underline{\xi}\underline{\zeta}} P(\underline{\xi}) = \frac{1}{|V|} \exp(-\mu) P(\underline{\zeta}) - \frac{1}{|V|} \exp(-\mu) P(\underline{\zeta}) = 0$, i. e., detailed balance is fulfilled according to Eq. (28).

The transition rates fulfilling detailed balance are not uniquely determined. It is possible to invent other types of transition than the M and E transitions explained above. Note that the M transition is defined on the basis of the *unoccupied* vertices. When defining another "move" transition on the basis of the *occupied* vertices, the

possible number of *unoccupied* neighbors, to where the particle can move, might be different for two neighboring vertices i, j. Hence, to ensure detailed balance, one would have to include these numbers in the transition probabilities.

The simulation is performed in terms of *Monte Carlo sweeps*. One sweep for a graph of $N = |V|$ vertices consist of exactly N tried transitions, each time randomly selecting a vertex and then, with equal probabilities, trying either an M or an E transition.

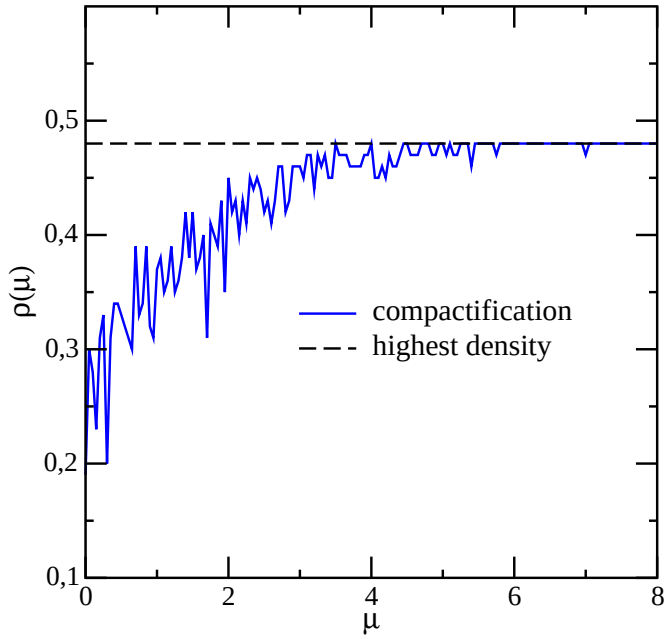

Figure 20: Compactification using MC simulations of a hard-core gas on a sample random graph of size $N = 100$ for average degree $c = 4.0$. Density ρ as a function of chemical potential μ for one single run. The highest possible density, corresponding to minimum vertex covers of this graph, are indicated by a horizontal line.

As an example of the application of the MC method, we show in Fig. 20 the *compactification* of a hard-core gas on a random graph of size $N = 100$ with connectivity $c = 4.0$. The simulation is started at $\mu_i = 0$, then μ is gradually increased by $\delta\mu = 0.05$ up to $\mu_f = 8$. For each value of μ, 10 MC sweeps are performed, and the density ρ after the 10th sweep is recorded. As expected, the density has a tendency to increase with the value of μ. The function is not smooth, because only the data for one single run is shown. When averaging over many independent runs, $\rho(\mu)$ would be a smooth monotonically increasing function. Interestingly, the MC simulation is able to find the configurations with the highest densities, corresponding to minimum vertex covers.

Note that the quality of the MC compactification depends on the ensemble of graphs. If one considers, e. g., random graphs made of randomly joined tetrahedrons, i. e., of cliques of size 4, then the corresponding system of hard-core particles behaves in a *glassy* way. This means the algorithm gets stuck in meta-stable configurations which are very different from the true ground states [14]. The reason for this behavior is that the configuration space has a very complicated and rugged structure. The meta-stable configurations have large but not maximum densities, and they are surrounded by configurations of lower densities. This means that, at high values of the chemical potential, the simulation will stay in the meta-stable configurations. In this case, an enhanced version of the MC method, the *parallel tempering*, may help. This approach is outlined in the following section.

4.4 Parallel tempering

The basic idea of parallel tempering [15, 16], is to simulate several copies of the same system, but with possibly different configurations, kept at different values $\mu_1 < \mu_2 < \ldots < \mu_n$ of the chemical potential, see Fig. 21. The M and E moves introduced above act locally on all configurations at the different values of the chemical potential. For parallel tempering, a *swap* (S) transition is additionally introduced, which tries to exchange the configurations having any two neighboring values of the chemical potential μ_k, μ_{k+1} ($k \in [1, n-1]$). This allows the configurations to be subjected to different values of μ during the simulation. Hence a configuration being stuck in a meta-stable state at a high value of μ, might be simulated later on with considerably smaller values of the chemical potential, i. e., at much lower expected density. When this configuration again visits high values of μ, it might explore a different region in configuration space, where it possibly can reach higher densities than before.

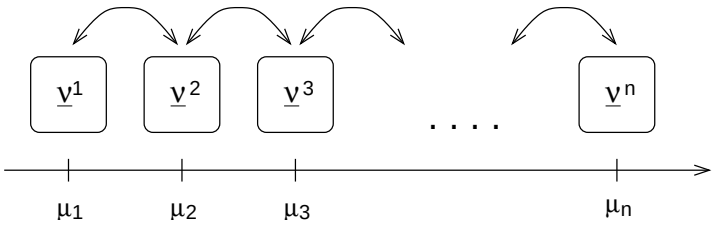

Figure 21: Parallel tempering with n different values of the chemical potential $\mu_1 < \mu_2 < \ldots < \mu_n$. At each value μ_i a system is simulated using conventional MC. From time to time, configurations are exchanged between neighboring values μ_k, μ_{k+1}, such that detailed balance is fulfilled for the combined systems.

The key point is that the S transition must be implemented in a way that detailed balance still holds, although two configurations being held at two different values of the chemical potential are treated. For an S transition, first one chooses a value $k \in \{1, 2, \ldots, n-1\}$ randomly. Let $\underline{\xi}, \underline{\zeta}$ be the configurations being simulated currently

at values μ_k,μ_{k+1}. The joint probability for both configurations in the grand-canonical ensemble, which is the crucial quantity to be taken into account here, is given by

$$P_{k,k+1}(\underline{\xi},\underline{\zeta}) = \frac{1}{\tilde{\Xi}_{k,k+1}} \chi(\underline{\xi}) \exp\left(\mu_k \sum_i \xi_i\right) \chi(\underline{\zeta}) \exp\left(\mu_{k+1} \sum_i \zeta_i\right), \tag{32}$$

with $\tilde{\Xi}_{k,k+1}$ being the corresponding partition function and the dependency on the chemical potential indicated by the index $k, k+1$. To define the transition probabilities, we evaluate the quantity

$$\Delta_{k,k+1}(\underline{\xi},\underline{\zeta}) \equiv (\mu_k - \mu_{k+1})\left(\sum_i \xi_i - \sum_i \zeta_i\right) \tag{33}$$

and choose

$$W_{k,k+1}([\underline{\xi},\underline{\zeta}] \rightarrow [\underline{\zeta},\underline{\xi}]) = \exp(-\max[\Delta_{k,k+1}(\underline{\xi},\underline{\zeta}), 0])\,. \tag{34}$$

The swap does not take place with the probability $1 - W_{k,k+1}([\underline{\xi},\underline{\zeta}] \rightarrow [\underline{\zeta},\underline{\xi}])$. This definition of the transition probability means that when at the larger value μ_{k+1} the configuration $\underline{\zeta}$ has a lower density than $\underline{\xi}$ (which is at the smaller value μ_k), i. e., when one encounters an atypical situation, we get $\Delta(\underline{\xi}|\mu_k, \underline{\zeta}|\mu_{k+1}) < 0$. This results in a transition rate 1, i. e., the swap will be performed always in this case, hence transitions to typical situations a favored. Note that $\Delta_{k,k+1}(\underline{\xi},\underline{\zeta}) = -\Delta_{k,k+1}(\underline{\zeta},\underline{\xi})$.

To prove detailed balance, we assume w.l.o.g. $\sum_i \xi_i - \sum_i \zeta_i \geq 0$, hence we obtain $\Delta_0 \equiv \Delta_{k,k+1}(\underline{\xi},\underline{\zeta}) < 0$ leading to $W_{k,k+1}([\underline{\xi},\underline{\zeta}] \rightarrow [\underline{\zeta},\underline{\xi}]) = 1$ for one direction of the transition and $W_{k,k+1}([\underline{\zeta},\underline{\xi}] \rightarrow [\underline{\xi},\underline{\zeta}]) = \exp(\Delta_0)$ for the inverse transition. This leads to (omitting $\chi(\underline{\xi}) = \chi(\underline{\zeta}) = 1$)

$$\begin{aligned}
& W_{k,k+1}([\underline{\xi},\underline{\zeta}] \rightarrow [\underline{\zeta},\underline{\xi}]) P_{k,k+1}(\underline{\xi},\underline{\zeta}) - W_{k,k+1}([\underline{\zeta},\underline{\xi}] \rightarrow [\underline{\xi},\underline{\zeta}]) P_{k,k+1}(\underline{\zeta},\underline{\xi}) \\
= \;& 1 P_{k,k+1}(\underline{\xi},\underline{\zeta}) - \exp(\Delta_0)\, P_{k,k+1}(\underline{\zeta},\underline{\xi}) \\
= \;& 1 \frac{1}{\tilde{Z}_{k,k+1}} \exp\left(\mu_k \sum_i \xi_i\right) \exp\left(\mu_{k+1} \sum_i \zeta_i\right) - \\
& \exp\left((\mu_k - \mu_{k+1})(\sum_i \xi_i - \sum_i \zeta_i)\right) \frac{1}{\tilde{Z}_{k,k+1}} \exp\left(\mu_k \sum_i \zeta_i\right) \exp\left(\mu_{k+1} \sum_i \xi_i\right) \\
= \;& 0\,.
\end{aligned} \tag{35}$$

Therefore, detailed balance is fulfilled according to Eq. (28).

Within a parallel tempering simulation one has to deal with several parameters: one has to determine the range of chemical potentials, its number n, and the precise values μ_k. Usually it is efficient to have more values in the region of high chemical potentials, and only few values for small chemical potentials. One possibility is to choose the values of the chemical potential automatically by an iterative procedure, such that the transition probability is 0.5 for all swaps [16]. Also one has to choose how often a "swap" step is performed in comparison with the local M and E transitions. One can, e. g., perform one MC sweep for each copy and then try $n - 1$ "swaps". Another parameter is the total number of iterations.

Indeed, using parallel tempering, for larger random graph of size, e. g., $N = 1000$, higher density states can be found compared with simple compactification [17]. Also higher density configurations for "glassy" ensembles can be obtained.

5 Numerical Results

5.1 Phase transitions

The Branch-and-Bound algorithm has obviously an exponential worst-case behavior since the configuration tree is *a priori* exponentially large. We apply the aforementioned algorithmic methods to randomized problem samples. A natural choice is provided by the Erdős–Rényi ensemble $\mathcal{G}(N, c/N)$ of random graphs of finite average degree, which was discussed in some detail in Sec. 2.

Although the branch-and-bound algorithm is very simple, we can treat random graphs up to size $N = 140$ if we restrict the average degrees to the region $c < 10$. It is, however, difficult to compare our algorithm with more elaborate ones existing in the literature [10,18], because they have usually been tested on a different graph ensemble in which edges appear with a fixed probability, independently of the graph size (high-connectivity regime). Nevertheless, in the computer-science literature, usually graphs with up to 200 vertices are treated, which is slightly larger than the systems considered here. On the other hand, the algorithm presented here has the advantage that it is easy to implement and its power is sufficient to study interesting problems.

First, we consider the problem of minimizing the energy (20) as a function of the relative VC size x. The average of the energy density over the random graph ensemble $\mathcal{G}(N, c/N)$ for fixed average degree c will be denoted by $e(c, x) = \overline{e(G, x)}$. Another interesting quantity in this context is the probability $P_{\mathrm{cov}}(c, x)$ that a $\mathcal{G}(N, c/N)$-graph is completely coverable with xN covering marks. Two limiting cases are obvious. For $x = 0$, we have no covering marks, the energy density becomes $e(c, 0) = c$, the graphs are almost surely uncoverable, $P_{\mathrm{cov}}(c, 0) \to 0$ (only the extremely improbable case of a graph without edges would be covered). For $x = 1$, we can cover all vertices and consequently all edges, $e(c, 1) = 0$ and $P_{\mathrm{cov}}(c, 0) = 1$. It is also clear that this energy density, seen as a function of x at fixed c is monotonously decreasing, because a higher number of covering marks allow us to cover a higher number of edges. Using the same argument, we conclude that the probability P_{cov} is monotonously increasing.

In Fig. 22 these two quantities are plotted for average degree $c = 2.0$, analogous results are found for other c-values. Curves are shown for different system sizes $N = 25, 50, 100$, and data are averages over 10^3 ($N = 100$) to 10^4 ($N = 25, 50$) randomly generated graphs. As expected, the value of $P_{\mathrm{cov}}(2, x)$ increases with the fraction of *covered* vertices. The astonishing result is, however, that the change from values close to zero to those close to one happens in a very restricted x-interval. With growing graph order, this change becomes even steeper. All different curves intersect almost at the same point. This suggests that in the limit $N \to \infty$, in which we are interested, a *sharp threshold* $x_{\mathrm{c}}(c = 2) \approx 0.39$ exists. Above $x_{\mathrm{c}}(c)$ a graph is almost surely coverable, below it is almost surely uncoverable. Thus, in the language of

physics, a *phase transition* from an uncoverable phase to a coverable phase occurs. Note that the value $x_c(c)$ of the critical threshold depends on the average connectivity c. The result for the phase boundary $x_c(c)$ as a function of c can be extracted from the simulations and is shown later on.

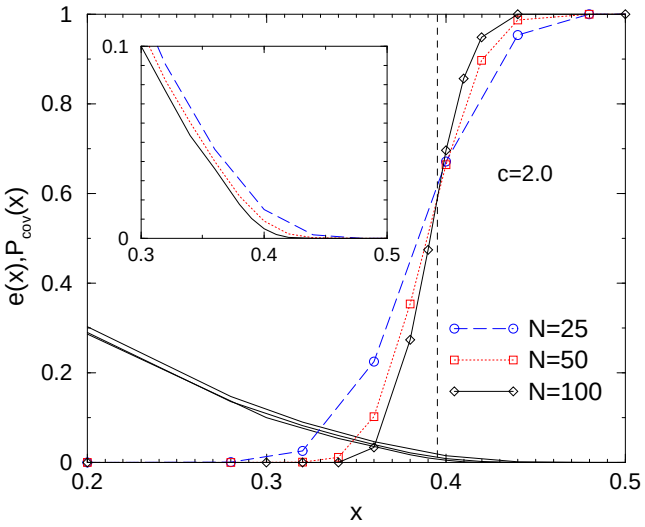

Figure 22: Probability $P_{\rm cov}(2,x)$ that a VC exists for a random graph ($c=2$) as a function of the fraction x of *covered* vertices. The result is shown for three different system sizes $N=25,50,100$ (averaged over 10^3-10^4 random graphs). Lines are guides to the eyes only. In the left part, where the $P_{\rm cov}$ is zero, the average energy density $e(2,x)$ (see text) is displayed. The inset enlarges the result for the energy in the region $0.3 \le x \le 0.5$.

In Fig. 23 the median (i. e., typical) running time of the branch-and-bound algorithm is shown as a function of the fraction x of covered vertices. The running time is measured in terms of the number of nodes which are visited in the configuration tree. Again graphs with $c=2.0$ were considered and an average over the same realizations has been performed. The most characteristic result is a sharp peak observed in the direct vicinity of the transition point x_c. This means that, near the phase transition, those instances are located, which are typically the hardest to solve. Note that for values $x < x_c$, the running time still increases exponentially, as can been seen from the inset of Fig. 23. A second important result is that, for values x considerably larger than the critical value x_c, the typical running time becomes *linear*. The reason is that the heuristic is already able to find a VC, i. e., the algorithm terminates after the first descent into the configuration tree[2]. So, even if the problem is NP-hard, there are parameter regions where the problem is *typically easy*.

[2] The algorithm terminates after a full vertex cover of the graph has been found.

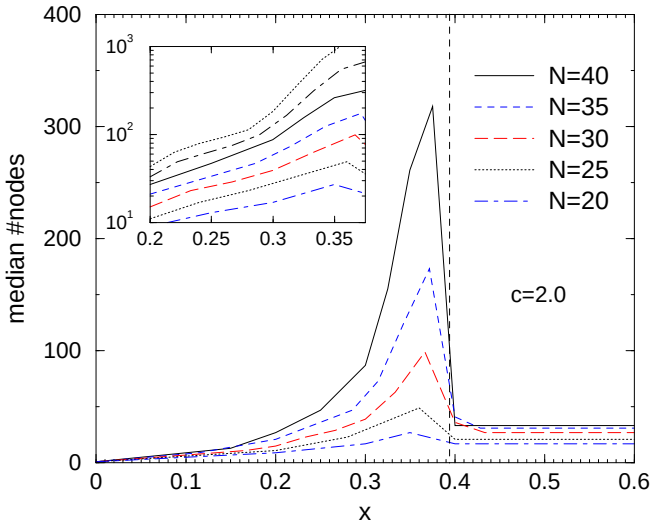

Figure 23: Time complexity of the vertex-cover algorithm. We display the median number of nodes visited in the configuration tree as a function of the fraction x of *covered* vertices for graph sizes $N = 20, 25, 30, 35, 40$, the average vertex degree is fixed to $c = 2.0$. The inset shows the region below the threshold on a logarithmic scale, including also data for $N = 45, 50$. The equidistant position of the curves in this representation illustrates that the time complexity grows exponentially with N.

Note that phase transitions in a physical system are usually indicated by a divergence of measurable quantities such as specific heat or magnetic susceptibilities. The peak appearing in the time complexity serves as a similar indicator, but is not really equivalent, because the time complexity diverges everywhere, only the rate of divergence is much stronger near the phase transition.

An indicator, which is more closely related to physical quantities, is the *correlation volume*. We define it in the following way. Usually, the minimum energy configuration with given size X is not unique, physically speaking the configuration is *degenerate*. If we force one vertex to change its state, other vertices have to change as well to obtain again a minimum energy configuration of size X. The correlation volume is the minimum number of vertices which have to be changed, averaged over the different vertices, which are not frozen to one state in all minimum-energy configurations. As shown in the inset of Fig. 24, close to the phase transition, a divergence of the correlation volume can be observed [19]. This is comparable to the divergence of the correlation length at second-order phase transitions. Unfortunately, the calculation of the correlation volume requires the enumeration of all ground states, hence only small system sizes can be treated.

Coming back to the running time, for small values of x in the uncoverable region,

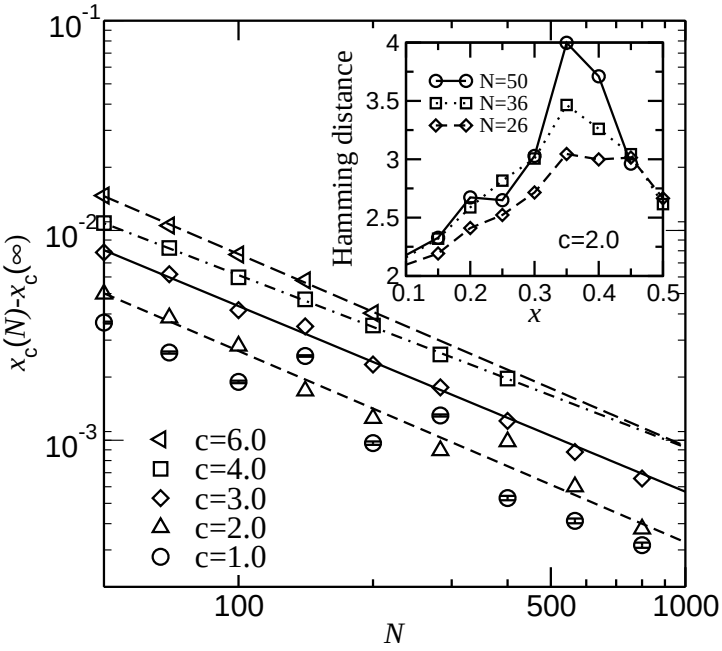

Figure 24: Finite-size scaling behavior of the critical cover size. The location of the transition point $x_c(N)$ as a function of graph size N for different average degree c. Inset: scaling of the correlation volume as a function of x for different sizes. Error bars are, at most, of the order of the symbol size.

the running time is also faster than that close to the phase transition, but still exponential. This is due to the fact that a configuration with a minimum number of *uncovered* edges has to be obtained. If only the question whether a VC exists or not is to be answered, the algorithm can be easily improved[3], such that for small values of x again a polynomial running time will be obtained.

Now we are going to numerically calculate the phase diagram, i. e., the position of the critical fraction x_c of covered vertices as a function of c. In this case it is sufficient to calculate for each graph the size $X_c = Nx_c$ of a minimum vertex cover, as done by the second version of the branch-and-bound algorithm presented in the last chapter. To compare with the analytical results which will be presented in the next chapter, one has to perform the limit $N \to \infty$ numerically. This can be achieved by calculating an average value $x_c(N)$ for different graph sizes N. Then one fits a function

$$x_c(N) = x_c + aN^{-b} \tag{36}$$

to the data. The form of the function is purely heuristic, no exact justification is

[3] Set *best* := 0 initially.

known so far for the case of VC. Analogous scaling forms are, however, well known in the *finite-size scaling* analysis of second-order phase transitions [20–22], where the location of the transition points in finite systems is governed by $b = 1/\nu$, ν being the critical exponent of the correlation length. As can be seen from Fig. 24, the fits match well, although for small average degrees a small but significant scattering of the data occurs. Interestingly, the exponent b describing the finite-size scaling term does not depend much on the connectivity c, for $c > 1$ one obtains $b(c = 2) = 0.91(9)$, $b(3) = 0.88(4)$, $b(4) = 0.82(4)$, and $b(6) = 0.92(11)$. Hence, within error bars, this exponent seems to be universal.

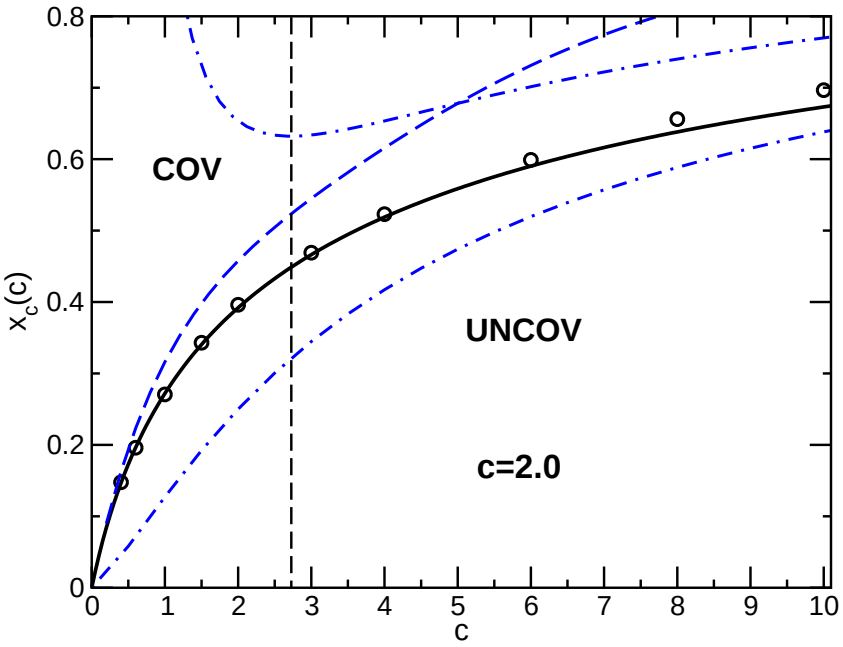

Figure 25: Phase diagram showing the fraction $x_c(c)$ of vertices in a minimum vertex cover as a function of the average degree c. For $x > x_c(c)$, almost all graphs have covers with xN vertices, while they have almost surely no cover for $x < x_c(c)$. The solid line shows the analytical result. The circles represent the results of the numerical simulations. Error bars are much smaller than symbol sizes. The upper bound of Harant is given by the dashed line, the bounds of Gazmuri by the dash-dotted lines. The vertical line is at $c = e \approx 2.718$.

This procedure has been performed for several values of c. The result is indicated in Fig. 25 by the symbols. Using probabilistic tools, rigorous lower and upper bounds for this threshold [23] and the asymptotic behavior for large connectivities [24] have been deduced, cf. also the next two sections. These bounds are indicated by dotted and dashed lines in in Fig. 25.

In the next chapter, we will show how the problem can be investigated using a statistical physics approach [25]. Let us for a moment anticipate the main result: Up to an average degree $c = e \approx 2.718$ the transition line

is given exactly by

$$x_c(c) = 1 - \frac{2W(c) + W(c)^2}{2c}\,, \tag{37}$$

where $W(c)$ is the Lambert-W function defined by $W(c)\exp(W(c)) = c$. The transition is shown in the phase diagram in Fig. 25 by a solid line. As already conjectured on the basis of numerical data, for $x > x_c(c)$, a random graph is almost surely coverable, for $x < x_c(c)$ the available covering marks are not sufficient. For higher connectivities no exact result for $x_c(c)$ is known. Note, however, that the region where the exact result has been obtained, extends fairly well into the percolating regime of random graphs, since the percolation threshold is $c_{\rm crit} = 1.0 < 2.718$, cf. Sec. 2.

As is to be expected, the data obtained from simulations followed by a finite-size scaling analysis, and the analytical result, agree very well in the region $c < e \approx 2.718$. For larger degrees, where the analytical result is not exact, slowly but systematically growing deviations show up. A stronger deviation between numerical and analytical results can be observed when studying the *backbone*, which are the nodes which have in all degenerate solutions the same value, i.e. which are always covered or always uncovered.

5.2 Clustering of minimum vertex covers

Apart from identifying phase transitions, one can analyze the organization of minimum vertex covers in more detail. One possible approach is to look for clusters of solutions [17].

Usually the minimum vertex covers are not equally distributed over the configuration space. They cluster in one or many groups that are separated by regions where no cover exists, or where the covers have larger size. Understanding this organization, it is possible to discover many interesting things about the nature of the model, and possibly also about its computational hardness.

Such clustering effects have already been observed in statistical physics for spin glasses [26, 27]. For the mean-field Ising spin glass, also called the Sherrington–Kirkpatrick (SK) model [28], which is defined on a complete, i. e., fully connected graph, Parisi has constructed [29] an analytical solution for the free energy. This solution exhibits so-called replica symmetry breaking (RSB), which means that the configuration space is organized in an infinitely nested hierarchy of clusters of configurations, characterized by ultrametricity [30]. Recently, this solution for the free energy was mathematically proven to be the exact one [31]. Also in numerical studies the clustering structure of the SK model has been observed, e. g., by calculating the distribution of overlaps [32–34], by studying the spectrum of spin–spin correlation matrices [35,36], or by directly applying clustering algorithms [50]. For finite-dimensional spin glasses, RSB seems not to be fully present [37, 38] (at least not in the same way as for the mean-field model), since clustering has been observed numerically, but it is

not non-ultrametric [50]. On the other hand, for models like Ising ferromagnets it is clear that they do not exhibit RSB and all solutions are organized in, at most, a few clusters which are related by the system's symmetries.

The use of the analytical tools from statistical mechanics enables physicists to contribute to the analysis of problems that originate in theoretical computer science. Often, one can only calculate the solution in the case of replica symmetry [39, 40], or in the case of one-step replica symmetry breaking (1-RSB) [41–43]. For this reason, the relation between the (partial) analytical solution and the clustering structure is not well established. It is far from being clear for most models how the clustering structure appears. However, most statistical physicists believe that the failure of replica symmetry (RS) leads indeed to clustering [44, 45]. On the other hand, it is unlikely that the clustering of models on dilute graphs, like the random graphs treated here, is exactly the same as is found for the mean-field SK spin glass. So, from the physicists point of view, it is quite interesting to study the organization of the phase space using numerical methods to understand better the meaning of "complex organization of phase space" for not fully connected models, such as combinatorial optimization problems. Here we will just present the result of one numerical study of the clustering properties of the vertex-cover problem.

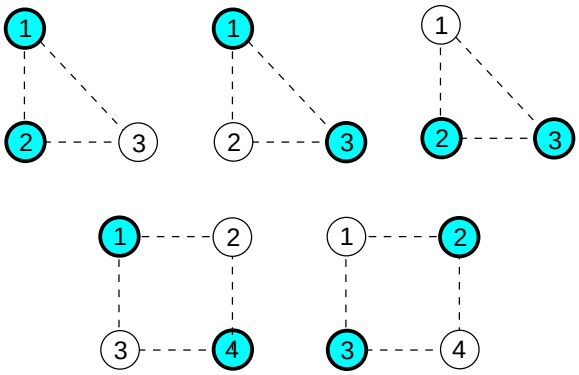

Figure 26: A triangle graph with 3 vertices (top) has 3 different minimum vertex covers, which are all neighbors and hence form one cluster. A square graph (bottom) exhibits two minimum clusters, which are not neighbors according to the definition used here.

Let us first define what we mean by the notation *cluster*. We call two minimum vertex cover configurations *neighbors*, if they differ by the configurations of exactly two vertices i, j. This means a covering mark has been moved from vertex i to vertex j. Since we are just interested in minimum covers, it follows immediately, that i and j are connected by an edge. A cluster is now the transitive closure of the neighbor relation. This means that two configurations belong to the same cluster, if one can move in configuration space from one to the other by just moving covering marks around, while always keeping the graph covered. This means that all three solutions

of a triangle graph belong to one cluster, while the two minimum vertex covers of a square graph, do *not* belong to the same cluster, see Fig. 26.

Note that one could define clusters in different ways, e. g., that one could consider configurations as neighbors if they can be reached by moving at most a finite fixed number $k_{\max}$ of covering marks. For $k_{\max} \geq 2$ the two minimum vertex covers of the square graph in Fig. 26 would also belong to the same cluster. Increasing $k_{\max}$ leads to the merging of previously separated clusters, $k_{\max} = 1$ gives the finest possible clustering. We study only the case $k_{\max} = 1$ here. Other ways of studying clustering exist, e. g., to study the spectrum of spin–spin correlation matrices [35,36] or so-called hierarchical clustering, see below. First, we consider only the definition of a cluster given above.

For small systems, one can enumerate all minimum vertex covers, and perform the clustering by just testing pairwise whether configurations are neighbors. Note that for the numerics, it helps to identify first the backbone vertices (see page 44). Then one can remove all backbone vertices and the graph usually splits into smaller components, which then can be treated independently, which sometimes greatly reduces the running time.

For larger systems, such an enumeration is no longer possible, because the number of minimum vertex covers grows, in principle, exponentially with the system size. Furthermore, even obtaining exact minimum vertex covers becomes impossible if the system size is too large. In this case, one can obtain multiple vertex covers by performing several independent runs of parallel tempering MC simulation, as explained in Sec. 4. Now, it does not make sense to look for direct neighbors, but one can apply a test, the *ballistic search* [46] which checks in a probabilistic manner whether two given configurations belong to the same cluster. Note that the minimum vertex covers of different connected components of a graph are independent of each other. We perform the clustering only for the solutions of the largest component.

Having performed the clustering, one interesting quantity to evaluate is the *number of clusters*. The reason is that for a simple structure of the configuration space, i. e., in the case of replica symmetry, one believes that only one cluster, or a small finite number of clusters exists in the limit $N \to \infty$. As it was shown by statistical mechanis methods [25], the solution of VC is replica symmetric for connectivities $c < e$. And indeed, as can be seen in Fig. 27, the cluster number stays finite and small for $c < e$. For larger connectivities, the number of clusters seems to grow logarithmically. This information is interesting, since it cannot be derived from analytical studies so far. More results on the clustering properties are presented in Ref. [17].

As an alternative method, we will use a clustering approach that organizes the configurations in a hierarchical structure. Such clustering methods [47] are widely used in general data analysis, sometimes also used in statistical mechanics, see e.g. references [48–50]. The methods all start by assuming that all configurations belong to separate clusters. Similarity between clusters (and configurations) is defined by a measure called *proximity matrix* $\tilde{d}_{\alpha,\beta}$. At each step two very similar clusters are joined and so a hierarchical tree of clusters is formed. As proximity measure for two initial clusters, each containing only a single configuration $\underline{\nu}^{\alpha}$, $\underline{\nu}^{\beta}$ (see Eq. (29)), respectively,

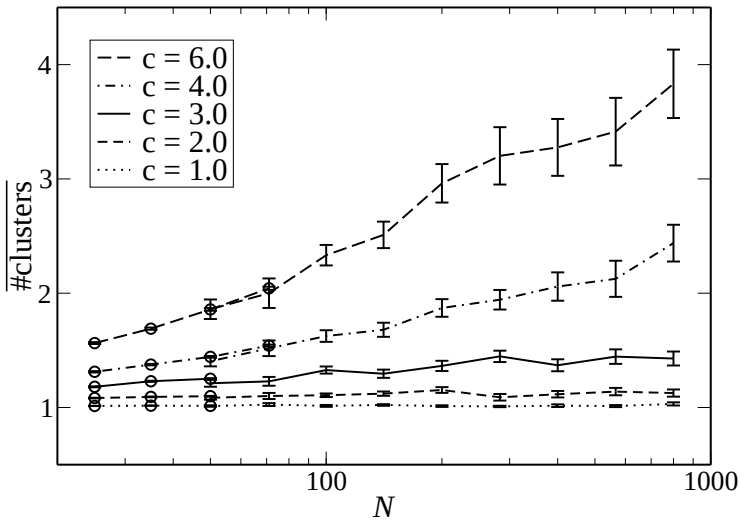

Figure 27: Average number of clusters in the solution space of the largest component as a function of system size. The circle symbols for small system sizes have been obtained by clustering complete sets of ground states. For large systems we sampled ground states with a parallel tempering algorithm at large but finite chemical potential μ.

we naturally choose the *Hamming distance* between these two configurations, divided by the number of vertices:

$$\tilde{d}_{\alpha,\beta} = \frac{1}{N} d_{\alpha\beta} \equiv \frac{1}{N} \sum_i |\nu_i^{(\alpha)} - \nu_i^{(\beta)}| \tag{38}$$

At each step the two clusters C_α and C_β with the minimal distance are merged to form a new cluster C_γ. Hence, in each step the number of clusters is reduced by one and the clusters will contain more and more configurations. This continues until just one cluster is left, containing all configurations.

During the iteration the proximity matrix is updated by deleting the distances involving C_α and C_β and adding the distances between C_γ and all other clusters C_δ in the system. So we need to extend the proximity measure to clusters with more than one configuration, based on some suitable update rule which is usually a function of the distances $\tilde{d}_{\alpha,\beta}$, $\tilde{d}_{\alpha,\delta}$ and $\tilde{d}_{\beta,\delta}$.

The choice of this function is a widely discussed field since it can have a great impact on the clustering obtained [47]. It should represent the natural organization present in the data and not some artificial structure induced from the choice of the update rule. Here we will use *Ward's method* (also called *minimum-variance method*) [51]. The distance between the merged cluster C_γ and some other cluster C_δ is given

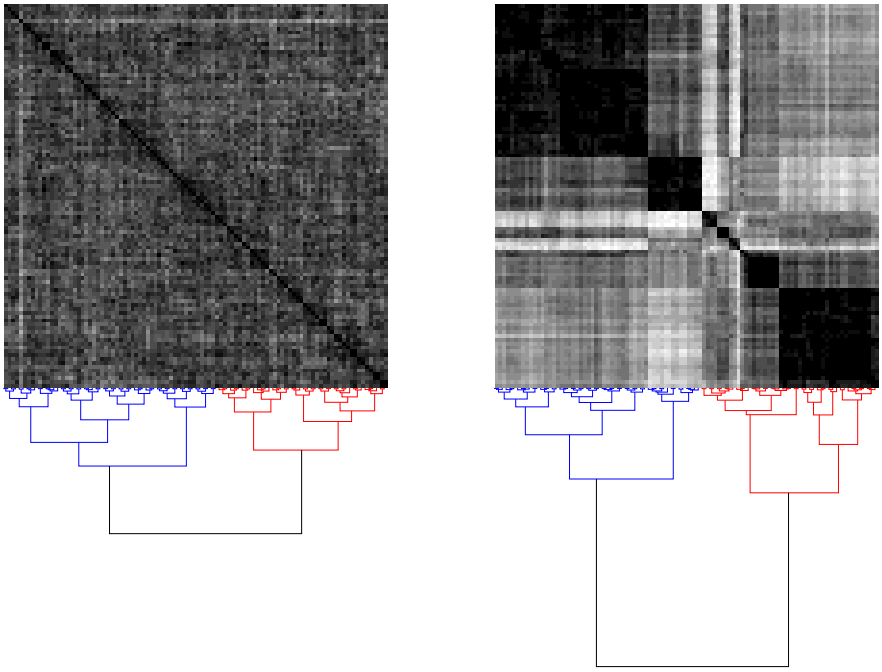

Figure 28: Sample dendrograms of 100 VC solutions for a graph with 400 vertices. Darker colors correspond to closer distances. The left one is at $c = 2$, i. e. in the RS phase. There is no structure present. For $c = 6$ the dendrogram provides a structure, where the solutions form clusters. The careful reader may recognize a second or third level of clustering in the right picture.

by

$$\tilde{d}_{\gamma,\delta} = \frac{(n_\alpha + n_\delta)\tilde{d}_{\alpha,\delta} + (n_\beta + n_\delta)\tilde{d}_{\beta,\delta} - n_\delta \tilde{d}_{\alpha,\beta}}{n_\alpha + n_\beta + n_\delta}\,, \tag{39}$$

where $n_\alpha, n_\beta, n_\delta$ are the number of elements in cluster $C_\alpha, C_\beta, C_\delta$, respectively. Heuristically Ward's method seems to outperform other update rules. The choice guarantees that at each step the two clusters to be merged are chosen in a way that the variance inside each cluster summed over all clusters increases by the minimal possible amount.

The output of the clustering algorithm can be represented as a *dendrogram*. This is a tree with the configurations as leaves and each node representing one of the clusters at different levels of hierarchy, see the bottom half of the examples in Fig. 28. Note that Ward's algorithm is able to cluster any data. Even if no structure were present, the data could always be displayed as a dendrogram. Hence, one has to perform additional checks. Here, we use a visual check, i.e. we plot the hamming distances as a matrix where the rows and columns are ordered according to the dendrogram.

The results of the hierarchical clustering for VC are shown in Fig. 28. The configurations were obtained using Monte Carlo simulations (see Sec. 4) for a large value of the chemical potential $\mu = 9$. Darker colors in the matrix correspond to smaller

distances. The figure shows two different realizations: For small values of $c < e$, the system is in the RS phase, only a single cluster is present. For larger values of c, the ordering of the configurations obtained by the clustering algorithm reveals an underlying structure which can be seen in the right part of the figure. One can see that the configurations form groups where the hamming distance between the members is small (dark colors) while the distance to other configurations is large. Thus, our results are compatible with clustering being present for realizations with $c > e$. If you look carefully you can see more structure inside the clusters. Multiple levels of clustering indicate higher levels of RSB which we expect to be present for these values of c [52, 53].

References

[1] L. Euler, Commentarii Academiae Scientiarum Imperialis Petropolitanae **8**, 128 (1736).

[2] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, (Holt, Rinehard and Winston, New York 1976).

[3] P. Erdős and A. Rényi, Magyar Tud. Akad. Mat. Kutat Int. Kőzl. **5**, 17 (1960).

[4] B. Bollobás, *Random Graphs*, (Academic Press, New York 1985).

[5] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, (Taylor & Francis, London 1994).

[6] K. Mehlhorn and St. Näher, *The LEDA Platform of Combinatorial and Geometric Computing* (Cambridge University Press, Cambridge 1999);
see also `http://www.mpi-sb.mpg.de/LEDA/leda.html`

[7] J. Siek, L.-Q. Lee, A. Lumsdainesee, *The Boost Graph Library*, (Addison-Wesley, Reading (MA) 2001);
see also `http://www.boost.org/libs/graph/doc/index.html`

[8] J. Hromkovic, *Algorithms for Hard Problems*, (Springer, Heidelberg, 2001).

[9] E. L. Lawler and D. E. Wood, Oper. Res. **14**, 699 (1966).

[10] R. Lüling and B. Monien, in: *Sixth International Parallel Processing Symposium*, (IEEE Comput. Soc. Press, Los Alamitos, USA 1992).

[11] M. E. J. Newman und G. T. Barkema, *Monte Carlo Methods in Statistical Physics* (Clarendon Press, Oxford, 1999).

[12] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, (Cambridge University Press, Cambridge 2000).

[13] L. E. Reichl, *A modern Course in Statistical Physics*, (Wiley, New York 1998).

[14] M. Weigt and A. K. Hartmann, Europhys. Lett. **62**, 533 (2003).

[15] E. Marinari and G. Parisi, Europhys. Lett. **19**, 451 (1992).

[16] K. Hukushima and K. Nemoto, J. Phys. Soc. Jpn. **65**, 1604 (1996).

[17] W. Barthel and A. K. Hartmann, to appear in Phys. Rev. B (2004), preprint cond-mat/0403193.

[18] M. Shindo and E. Tomita, Syst. Comp. Jpn. **21**, 1 (1990).

[19] A. K. Hartmann, W. Barthel and M. Weigt, submitted to Comp. Phys. Comm.

[20] M. N. Barber, in: C. Domb and J. L. Lebowitz, *Phase Transitions and Critical Phenomena* **8**, 146, (Academic Press, London 1983).

[21] V. Privman (ed.), *Finite Size Scaling and Numerical Simulation of Statistical Systems*, (World Scientific, Singapore, 1990).

[22] J. Cardy, *Scaling and Renormalization in Statistical Physics*, (Cambridge University Press, Cambridge 1996).

[23] P. G. Gazmuri, Networks **14**, 367 (1984).

[24] A. M. Frieze, Discr. Math. **81**, 171 (1990).

[25] M. Weigt and A. K. Hartmann, *Phys. Rev. Lett.* **84**, 6118 (2000).

[26] Reviews on spin glasses can be found in: K. Binder and A. P. Young, Rev. Mod. Phys. **58**, 801 (1986); K. H. Fisher and J. A. Hertz, *Spin Glasses*, (Cambridge University Press, Cambridge 1991); A. P. Young (ed.), *Spin glasses and Random Fields*, (World Scientific, Singapore 1998).

[27] M. Mézard, G. Parisi, M. A. Virasoro, *Spin Glass Theory and Beyond*, (World Scientific, Singapore 1987).

[28] D. Sherrington and S. Kirkpatrick, Phys. Rev. Lett. **35**, 1792 (1975).

[29] G. Parisi, Phys. Rev. Lett. **43**, 1754 (1979); J. Phys. A **13**, 1101 (1980); **13**, 1887 (1980); **13**, L115 (1980); Phys. Rev. Lett. **50**, 1946 (1983).

[30] R. Rammal, G. Toulouse, and M. A. Virasoro, Rev. Mod. Phys. **58**, 765 (1986).

[31] M. Talagrand, C.R.A.S. **337**, 111 (2003).

[32] A. P. Young, Phys. Rev. Lett. **51**, 13 (1983).

[33] G. Parisi, F. Ritort and F. Slanina, J. Phys. A **26**, 3775 (1993).

[34] A. Billoire, S. Franz, and E. Marinari, J. Phys. A **36** (2003).

[35] J. Sinova, G. Canright, and A. H. MacDonald, Phys. Rev. Lett. **85**, 2609 (2000).

[36] J. Sinova, G. Canright, H. E. Castillo, and A. H. MacDonald, Phys. Rev. B **63**, 104427 (2001).

[37] F. Krzakala and O. C. Martin, Phys. Rev. Lett. **85**, 3013 (2000).

[38] M. Palassini and A. P. Young, Phys Rev. Lett. **85**, 3017 (2000).

[39] R. Monasson and R. Zecchina, Phys. Rev. Lett. **76**, 3881 (1996); Phys. Rev. E **56**, 1357 (1997).

[40] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, Nature **400**, 133 (1999).

[41] G. Biroli, R. Monasson, and M. Weigt, Eur. Phys. J. B **14**, 551 (2000).

[42] M. Mézard, G. Parisi, and R. Zecchina, Science **297**, 812 (2002).

[43] M. Mézard and R. Zecchina, Phys. Rev. E **66**, 056126 (2002).

[44] M. Weigt, in: A. K. Hartmann and H. Rieger (eds.), *New Optimization Algorithms in Physics*, 121, (Wiley-VCH, Weinheim 2004).

[45] R. Zecchina, in: A. K. Hartmann and H. Rieger (eds.), *New Optimization Algorithms in Physics*, 183, (Wiley-VCH, Weinheim 2004).

[46] A. K. Hartmann, J. Phys. A **33**, 657 (2000).

[47] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data* (Englewood Cliffs, USA: Prentice-Hall) (1988).

[48] G. Hed, A.K. Hartmann, D. Stauffer and E. Domany, *Phys. Rev. Lett.* **86**, 3148 (2001).

[49] S. Ciliberti and E. Marinari *J. Stat. Phys.* **115**, 557 (2004).

[50] G. Hed, A. P. Young, and E. Domany, Phys. Rev. Lett. **92**, 157201 (2004).

[51] J. Ward, *J. of the Am. Stat. Association* **58**, 236 (1963)

[52] M. Weigt and A. K. Hartmann, Phys. Rev. E **63**, 056127 (2001).

[53] H. Zhou, *Eur. Phys. J. B* **32**, 265 (2003)