The Pennsylvania State University
The Graduate School

ADDITIONAL CONTOUR PROPERTIES FOR THE HARD-CORE MODEL ON $\mathbb{Z}^2$

A Thesis in

Computer Science

by

Daniel Stebbins

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2024

The thesis of Daniel Stebbins was reviewed and approved by the following:

Antonio Blanca
Dorothy Quiggle Assistant Professor of Computer Science and Engineering
Thesis Advisor

Debarati Das
Assistant Professor of Computer Science and Engineering

Chitaranjan Das
Distinguished Professor of Computer Science and Engineering
Department Head of Computer Science and Engineering

ABSTRACT

In statistical physics, the hard-core model simulates the behavior of lattice gases with particles of non-negligible size. The size consideration is enforced by requiring particles not to be adjacent; once one vertex of the lattice is occupied, its neighbors must remain unoccupied. This rule strongly parallels the independent set problem of graph theory. The hard-core model features a parameter λ that determines the probability with which each possible configuration of occupied and unoccupied vertices arises. When λ is small, configurations with few occupied vertices are likely. These configurations are relatively chaotic and loosely packed. However, when λ is large, configurations with many occupied vertices become exponentially more likely. These configurations tend toward order, as conflicting placements waste space and reduce the number of occupied vertices.

One subject of interest in this area is finding the critical value λ_c such that for all $\lambda < \lambda_c$ the expected configuration is disordered and for all $\lambda > \lambda_c$ the expected configuration is ordered. The critical value depends on the dimensions of the lattice considered. Here we consider $\mathbb{Z}^2$, where it is conjectured that $\lambda_c \approx 3.796$. However, the rigorous bounds are still relatively loose. An upper bound on λ_c can be shown by bounding the number of possible contours separating disagreeing regions of a configuration, which requires formalizing the properties of said contours. It has previously been shown that $\lambda_c < 5.3485$ using this method. This thesis presents a new contour property, explores its effect on the self-avoiding walks used for computation, and concludes with an updated bound $\lambda_c < 5.1885$.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

The hard-core model on $\mathbb{Z}^2$ involves sets of "occupied" vertices that can never be adjacent. These are also called independent sets. The optimal density of occupied vertices is achieved by two independent sets - the set of all even vertices and the set of all odd vertices, where the parity of a vertex $(x,y) \in \mathbb{Z}^2$ is the parity of $x+y$. At the other extreme, an independent set could contain no vertices or any one vertex. The hard-core model takes a parameter λ which determines the probability of each independent set occurring. Specifically, an independent set I occurs with probability proportional to $\lambda^{|I|}$. As λ increases, large independent sets become dominant. The larger an independent set is, the more it resembles one of the two optimal-density sets. This is because switching between a region in the odd configuration and a neighboring region in the even configuration creates a gap of two unoccupied vertices. Increasing the size of the independent set requires reducing the number of such gaps, which in turn forces large swaths of the lattice to conform to a single parity.

The critical value λ_c is intuitively the value for which all $\lambda < \lambda_c$ expect disordered, small independent sets and all $\lambda > \lambda_c$ expect ordered, large independent sets. The Peierls argument creates an upper bound on λ_c by linking it to the number of possible contours separating regions with different parity. These contours inhabit the gaps that are forced to be unoccupied, so long contours become unlikely as λ increases. This description of λ_c is imprecise, especially for an infinite lattice with an infinite number of independent sets. The formalization of this problem involves a mathematical tool called Gibbs measures. The case where small independent sets are favored is represented by a singular Gibbs measure or "uniqueness of phase." The case where large independent sets are favored then admits multiple Gibbs measures or exhibits "phase coexistence." Blanca et al. applied the logic of the Peierls argument for the hard-core model on $\mathbb{Z}^2$, but their proofs involving Gibbs measures are beyond the scope of this thesis [1]. Instead, we focus on improving the upper bound for the number of contours of a given length while ensuring their proofs still apply.

Given an independent set on $\mathbb{Z}^2$, which exhibits one parity at the origin and the opposite parity everywhere far from the origin, it is possible to construct a contour that separates the two parity regions. This idea can be expanded to formalize the properties of such a contour. Blanca et al. discovered that these properties include never self-intersecting, never taking two consecutive turns, and following the directions of the Manhattan lattice. Chapter 2 briefly restates some of the definitions required to construct the contour. The remainder of the thesis presents a new contour property: no contour can contain a narrow pair of parallel edges. This observation simplifies the definition of the contour and reduces the number of associated self-avoiding walks. This number is further reduced by the addition of an unrelated rule, closability. Finally, Chapter 5 describes the transfer-matrix algorithm used to upper-bound the number of length 2500 walks and arrive at the stated bound $\lambda_c < 5.1885$.

Chapter 2

Preliminary Definitions

Definition 2.1. Given some $S \subseteq \mathbb{Z}^2$, G_S is an undirected graph with vertex set S and edges connecting any vertices with distance exactly 1.

Definition 2.2. For some $i \in \mathbb{Z}^+$, $U_i = \{-i, -(i-1), \ldots, i-1, i\}^2$ is a subset of $\mathbb{Z}^2$ forming a square centered on the origin.

Definition 2.3. An independent set is a subset of $\mathbb{Z}^2$ such that no two vertices in the set are adjacent in $G_{\mathbb{Z}^2}$.

Definition 2.4. I is an independent set with the following properties relating to positive integers m and n, where $m < n$. I includes every even vertex outside box U_n and, for every odd vertex inside box U_m, I includes none of its even neighbors.

Definition 2.5. I' is a subset of $\mathbb{Z}^2$ constructed from some I. I' is the union of (1) all odd vertices in $\mathbb{Z}^2$ with no neighbors in I, and (2) the neighbors of every vertex added in step (1). Note that $U_m \subset I' \subset U_n$.

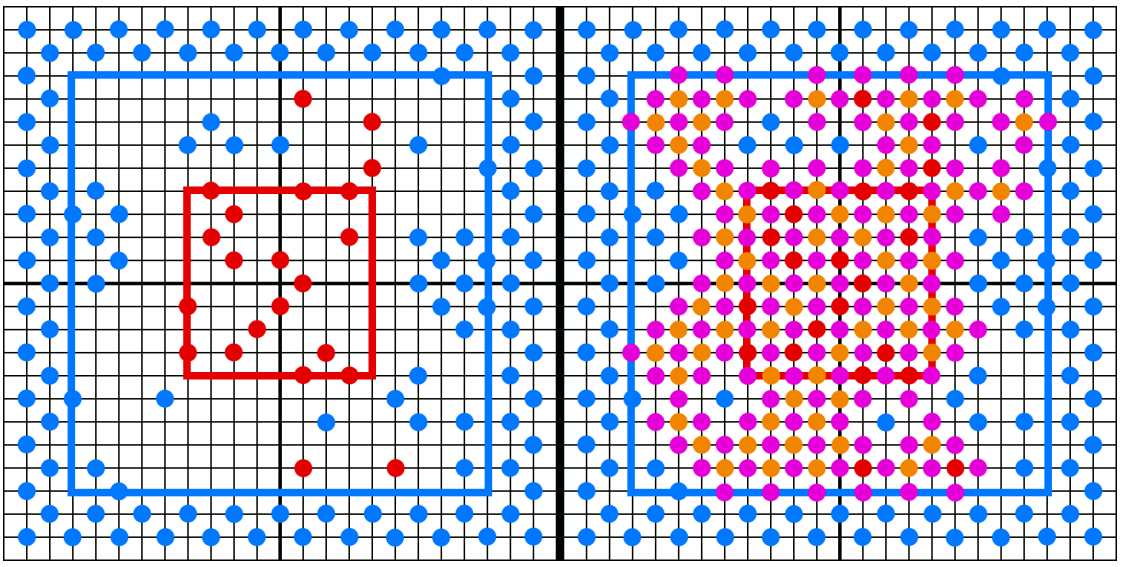

Figure 2.1: Left: Some I with $m = 4$ and $n = 9$. Right: I' for that I. Red: odd in I, blue: even in I, orange: odd added in I', pink: even added in I'.

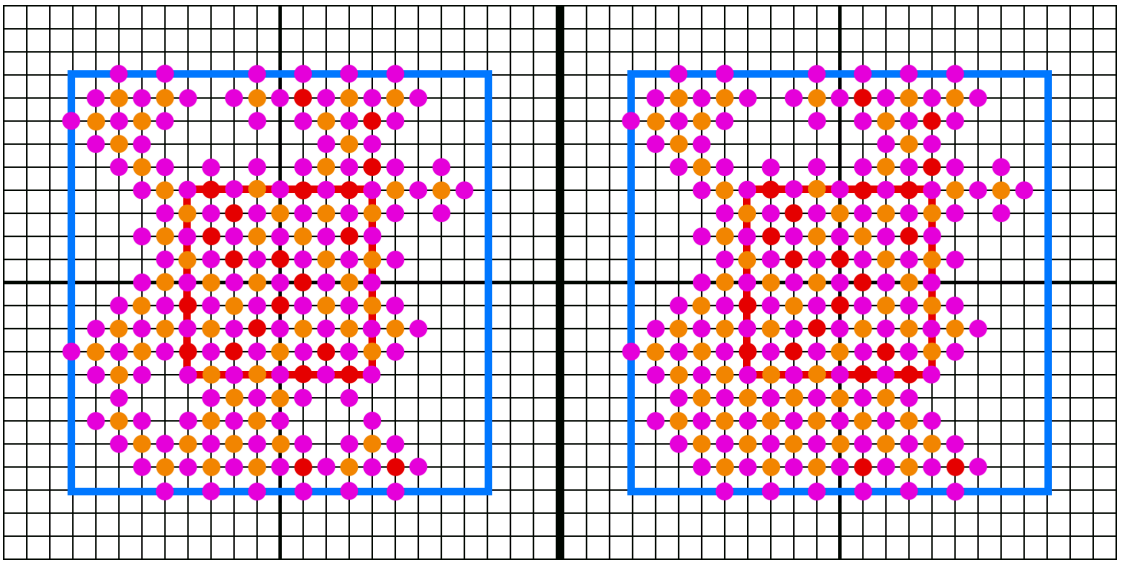

Figure 2.2: R (left) and R' (right) corresponding to Figure 2.1.

Definition 2.6. R is the vertex set of the connected component of $G_{I'}$ that contains box U_m.

Definition 2.7. R' is a superset of R that additionally contains every vertex $v \notin R$ for which there is no path of adjacent vertices $P \subset \mathbb{Z}^2 \setminus R$ connecting v to some vertex v' outside box U_n. Informally, R' is R plus any vertices from "holes" in R.

The small disconnected component in the top right of Figure 2.1 is excluded from R in Figure 2.2 because R is a single connected component. The unoccupied "holes" near the bottom of R are likewise ignored in R' by definition.

Definition 2.8. γ is the set of all edges in $G_{\mathbb{Z}^2}$ which connect a vertex $u \in R$ to a vertex $v \notin R'$. Note that no $u \in R' \setminus R$ can be adjacent to a $v \notin R'$. Any such u could escape outside box U_n through v and would not be in R'.

Definition 2.9. $G^{\diamond}_{\mathbb{Z}^2}$ is the undirected graph with vertices on the midpoints of edges in $G_{\mathbb{Z}^2}$ and edges connecting vertices if their associated edges in $G_{\mathbb{Z}^2}$ are incident and perpendicular. $G^{\diamond}_{\mathbb{Z}^2}$ is $G_{\mathbb{Z}^2}$ rotated $\frac{\pi}{4}$ radians in either direction, reduced by a factor of $\sqrt{2}$, then translated $\frac{1}{2}$ unit in any cardinal direction.

Chapter 3

New Contour Properties

Before completing the definition of the contour, I first provide a formal proof of a simple fact about γ stated by Blanca et al.

Lemma 3.1. Let s and t be adjacent vertices in $\mathbb{Z}^2$. If $\{s,t\} \in \gamma$, then the vertex in R is even and the other is odd.

Proof. Let s be the vertex in R and thus in I'. $t \notin R'$ is then implied by the definition of γ. Assume for the sake of contradiction that s is odd. By the construction of I', all neighbors of s are also in I'. Additionally, all neighbors of s are in the same connected component as s. Since s is in R, t must also be in R - a contradiction on the assumption that $t \notin R'$. Knowing that s is even, all its neighbors must be odd. □

Definition 3.1. A contour Γ is a 2-regular unicyclic subgraph of $G^{\Diamond}_{\mathbb{Z}^2}$ which encircles every vertex in R'. Blanca et al. provided steps to construct Γ from γ and, for their Lemma 3.2, proved that Γ has exactly one cycle including all its vertices [1]. Here, we show a simplified definition. Γ is the graph with vertex set γ and edges connecting all incident elements of γ.

Proof. Consider some $\{u,v\} \in \gamma$, along with two other vertices s and t in $\mathbb{Z}^2$ such that $stuv$ is a unit square. Let u be the endpoint in R and v be the endpoint outside R'. This implies u and s are even and v and t are odd by Lemma 3.1. The construction of Γ can be expressed as a case analysis over the inclusion or exclusion of s and t in R'. The first three cases are unchanged.

1. $s \notin R'$ and $t \in R$: By the construction of I', t odd and $t \in R$ imply $s \in R$. Therefore this case cannot occur.

2. $s \notin R'$ and $t \notin R'$: Both $\{u,v\}$ and $\{t,u\}$ are in γ; connect their midpoints with an undirected edge in Γ. $\{s,t\}$ and $\{v,s\}$ are not in γ.

3. $s \in R'$ and $t \in R'$: Both $\{u,v\}$ and $\{v,s\}$ are in γ; connect their midpoints with an undirected edge in Γ. $\{s,t\}$ and $\{t,u\}$ are not in γ.

4. $s \in R'$ and $t \notin R'$: This case is considered in the original construction, but can never occur. The assumption that $\{u,v\} \in \gamma$ implies $u,s \in R$ and $v,t \notin R'$. While not adjacent, u and s share a connected component, implying a path of connected vertices exists between u and s. Such a path must include or enclose either v or t. Here, to "enclose" means to surround with a set of vertices such that no series of steps out from the enclosed vertex can escape to infinity while avoiding all the vertices in the set. The enclosed vertex would then be included in R', contradicting the assumption that $v,t \notin R'$.

The relatively complicated original fourth case is therefore removed from consideration, leaving only the straightforward second and third cases. Every edge added in these cases has vertices in γ, so the vertex set of Γ is exactly γ. This analysis can be repeated for the other unit square containing $\{u,v\}$ and for every $\{u,v\} \in \gamma$, as the edge set of Γ disregards the repetition caused by considering

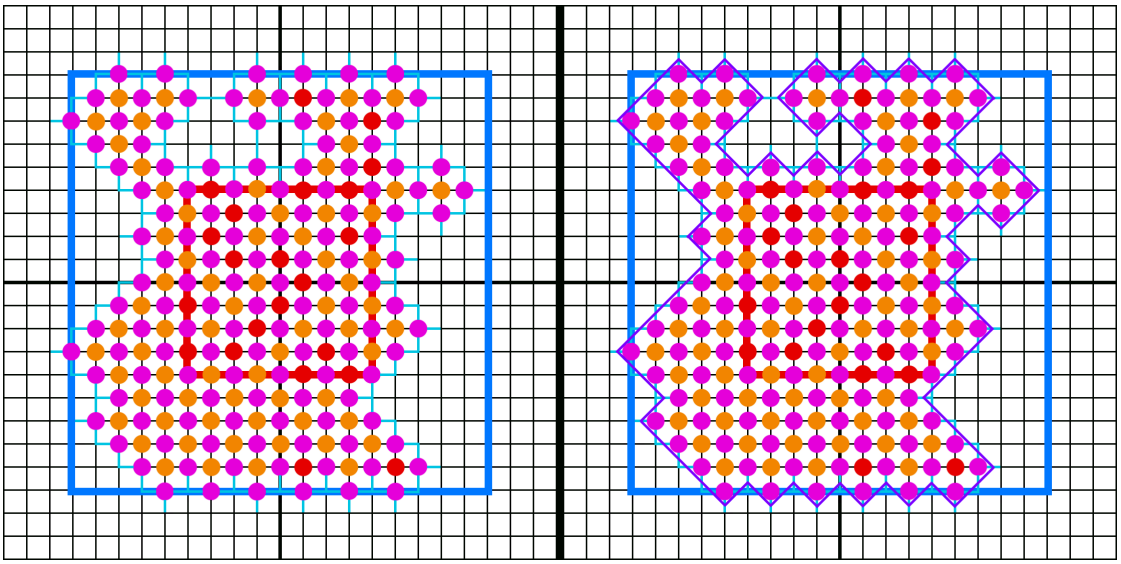

Figure 3.1: Left: Edges of γ in cyan. Right: The contour Γ in purple.

both endpoints of every edge in Γ. Edges added in the second case connect elements of γ incident on u, while edges added in the third case connect elements of γ incident on v. Therefore, every edge in Γ connects incident elements of γ. To see that Γ connects every incident pair of elements in γ, note that the case analysis above includes all possible configurations of incident elements of γ. Every unit square containing at least one element of γ contains exactly one incident pair, and every such pair receives an edge in Γ. □

The logic that eliminated case 4 in the proof above can be rephrased as a property of the edges of the Γ instead of the vertices of $\mathbb{Z}^2$. This simplifies later computation, which abstracts away $\mathbb{Z}^2$ and relies entirely on contour edge properties. The pattern of contour edges invalidated by this logic is called a "narrow."

Definition 3.2. A narrow N is a subset of the edges of $G^{\diamond}_{\mathbb{Z}^2}$. N contains two parallel edges $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_3, v_4\}$ such that $\{v_1, v_4\}$ and $\{v_2, v_3\}$ are also edges in $G^{\diamond}_{\mathbb{Z}^2}$. The four specified vertices form a unit square in $G^{\diamond}_{\mathbb{Z}^2}$, with e_1 and e_2 on opposing sides.

Theorem 3.2. For all contours $\Gamma = (\gamma, E)$, there exists no narrow N such that $N \subset E$.

Proof. We name four additional vertices in $\mathbb{Z}^2$. Let s be the vertex of $\mathbb{Z}^2$ closest to the midpoint of edge $\{v_4, v_1\}$. In the same way, let vertices t, u, and v be associated with edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, and $\{v_3, v_4\}$, respectively. The contour separates R' from the rest of $\mathbb{Z}^2$. In other words, when stepping between two vertices and crossing the contour, the first must be in R and the second outside R' or vice-versa. In the context of a narrow that is part of a contour, this means that s and u are either both in R or both outside R', with t and v taking on the opposite state. Whichever diagonal pair is in R shares a connected component despite not being adjacent. This implies the existence of a path of adjacent vertices connecting them. Such a path must include or enclose one of the vertices of the other pair. The enclosed vertex is then included in R', contradicting the assumption that the two diagonal vertices not in R are outside R'. □

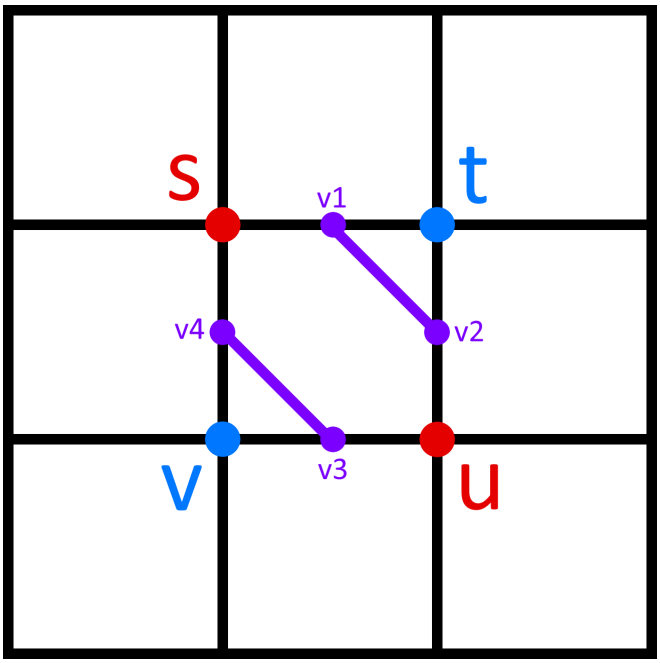

Figure 3.2: A labeled narrow N from the perspective of $G_{\mathbb{Z}^2}$, such that the contour edges are diagonal.

Definition 3.1 and Theorem 3.2 are the main theoretical contributions of this thesis. While a simplified contour definition is certainly an interesting result and may inspire future work, the "no narrows" rule is the form that immediately affects the numerical methods for bounding λ_c. This condition is relatively simple to implement while enumerating contours or the associated self-avoiding walks and causes a moderate reduction in their counts.

Chapter 4

Bounding Contours with Walks

Definition 4.1. A length-n self-avoiding walk (SAW) on a graph is a sequence of $n+1$ distinct vertices $v_0, v_1, \ldots, v_n$ such that an edge connects v_i to v_{i+1} for all i. Here, we consider only walks on a directed version of $G_{\mathbb{Z}^2}$ where v_0 is the origin.

Definition 4.2. A function $f(n)$ defined on positive integers is subexponential if, for all $\varepsilon > 0$, there exists some $n(\varepsilon)$ such that $f(n) < (1+\varepsilon)^n$ for all $n > n(\varepsilon)$.

Definition 4.3. The connective constant $\mu > 0$ of a class of SAWs is a constant that asymptotically determines the number of walks of length n. Specifically, the number of walks of length n is equal to $f(n)\mu^n$, where $f(n)$ is some subexponential function.

In their Theorem 1.1, Blanca et al. proved that the hard-core model on $\mathbb{Z}^2$ admits multiple Gibbs measures when its parameter λ is greater than 5.3506 [1]. This bound was later improved to 5.3485 with more involved computation [2]. The Peierls argument that establishes this bound involves $|C_\ell^m|$, the number of contours of length 4ℓ around a sufficiently large box U_m. $|C_\ell^m|$ is needed for every $\ell \geq \frac{\sqrt{2}m}{2}$, an infinite number of ℓ values, so computing the count directly is not possible. Instead, an upper bound on $|C_\ell^m|$ for every ℓ has previously been established using μ_{taxi}, the connective constant of a class of SAWs called taxi walks. Any class of SAWs could be used for this purpose if they satisfy two requirements. First, the series describing the number of SAWs of length n must be submultiplicative. Second, the class of SAWs must be contour bounding.

Definition 4.4. A series $\{s_n\}, n \geq 1$ is submultiplicative if, for all m and n, $s_{m+n} \leq s_m s_n$.

Definition 4.5. A class of SAWs with connective constant μ is contour bounding if $|C_\ell^m| \leq g(\ell)\mu^{4\ell}$ for some subexponential function $g(\ell)$. Intuitively, this means that the exponential growth rate of the SAWs is not slower than the exponential growth rate of the contours as ℓ increases.

In addition to being self-avoiding, all taxi walks adhere to the two following rules. First, taxi walks traverse a directed version of $G_{\mathbb{Z}^2}$ called the Manhattan lattice ($\overrightarrow{G_{\mathbb{Z}^2}}$). The horizontal edges of $\overrightarrow{G_{\mathbb{Z}^2}}$ are oriented in the positive x-direction if their y-coordinate is even and the negative x-direction if their y-coordinate is odd. Similarly, the vertical edges of $\overrightarrow{G_{\mathbb{Z}^2}}$ are oriented in the positive y-direction if their x-coordinate is even and the negative y-direction if their x-coordinate is odd. Second, taxi walks may not make two consecutive turns.

Previous binary string encodings of taxi walks used characters s and t to denote straight edges (edges parallel to the previous edge) and turns (edges perpendicular to the previous edge). Here, we instead use H to denote a horizontal step and V to denote a vertical step. Each walk of length n is then encoded by a binary string of length n. This encoding is valid as every vertex in $\overrightarrow{G_{\mathbb{Z}^2}}$ has exactly one outgoing vertical edge and one outgoing horizontal edge. Using this encoding, turns are represented by the substrings HV or VH, and two consecutive turns are represented by the substrings HVH and VHV.

Blanca et al. proved that taxi walks fulfill the two required properties to provide a useful upper bound on the number of contours [1]. The remainder of this section provides updated proofs for a new class of SAWs that follow a few additional rules. Each rule reduces the number of walks of

length n and the known λ value above which the hard-core model admits multiple Gibbs measures. Not every rule reduces μ, but small numerical improvements to the bounds on λ_c are nonetheless possible by reducing the subexponential function $f(n)$. For example, compared to the connective constant of SAWs on $G_{\mathbb{Z}^2}$, SAWs constrained to the Manhattan lattice have a smaller connective constant [3] [4]. In contrast, forward-extendable SAWs on $G_{\mathbb{Z}^2}$ have the same connective constant as general SAWs on $G_{\mathbb{Z}^2}$ [5].

4.1 New SAW Rules

We define a narrow $N = (e_1, e_2)$ similarly to Definition 3.2, but on $\overrightarrow{G_{\mathbb{Z}^2}}$ instead of $G^{\diamondsuit}_{\mathbb{Z}^2}$. Let $e_1 = v_i v_{i+1}$ and $e_2 = v_j v_{j+1}$ where $0 \leq i \leq n-13$ and $i+12 \leq j \leq n-1$. Assume that v_i has distance 1 to v_{j+1} and v_{i+1} has distance 1 to v_j.

We refer to the new class of SAWs as contour walks. Let c_n be the number of contour walks of length n and μ_c be the contour walk connective constant. Contour walks are a subset of taxi walks constrained by the following multipart rule:

Definition 4.6. A contour walk is a taxi walk for which there exists some non-empty sequence of steps that, when appended to the end of the walk, return the walk to the origin and create a rectilinear polygon with the following properties:

1. The polygon is self-avoiding. Starting from the origin, exactly one intersection is allowed for the step that returns to the origin.

2. The polygon follows the directions of the Manhattan lattice.

3. The polygon never takes two consecutive turns. Equivalently, the substrings HVH and VHV do not appear in the polygon's string representation.

4. The polygon has no narrows.

Three thoughts inspired the shift to constraining a polygon created from the walk instead of the walk itself. The first is a simple observation that the number of self-avoiding polygons (SAPs) is generally far lower than the number of self-avoiding walks. Counting SAPs would therefore give tighter bounds and be more intuitively linked to counting the closed contours of the hard-core model. Unfortunately, the counts of SAPs are also generally supermultiplicative, meaning any approximation to μ_c obtained by enumerating SAPs of some finite length would be a lower bound on the true value of μ_c instead of our desired upper bound.

Requiring a SAW to be closable into a SAP which is then validated maintains the submultiplicativity of the counts, as proven in Theorem 4.1. A similar idea is requiring the SAW to be extendable to infinity from one or both of its endpoints. This extendable requirement does not affect the connective constant, but it does reduce the number of SAWs of a given length and improve the resultant numerical bounds [5]. For this reason, the idea of forward-backward extensibility was recently used by Couronné as part of the computation for the best upper bound on the connective constant of the square lattice to date [3]. Here, we use closable SAWs instead of extendable SAWs for their similarity to the contours. As every forward-backward extendable SAW is closable but not all closable SAWs are forward-backward extendable, this change causes a small but negligible increase in the count.

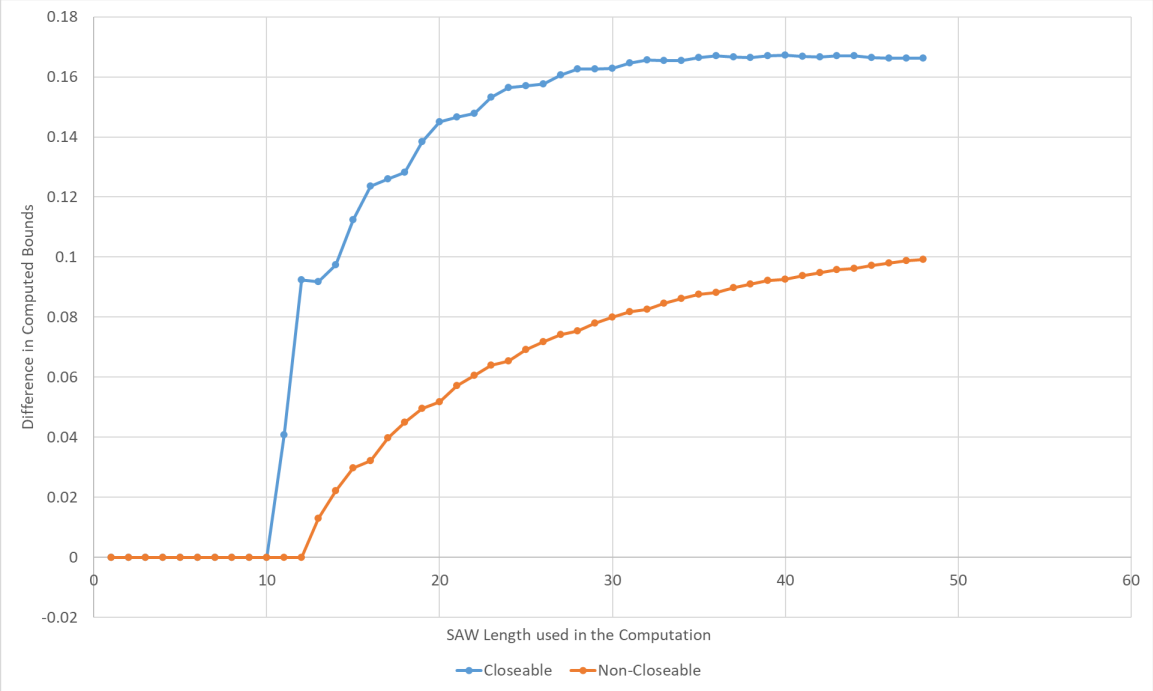

Figure 4.1: Improvement of λ_c by contour walks both closable and non-closable. Both are compared to the existing bounds from enumerating taxi walks.

Finally, enforcing the rules on the polygon created by closing a SAW strengthens the "no narrows" condition. For example, a SAW containing two parallel edges separated by a gap of two units would require a separate rule to be considered invalid. However, any such structure requires the closing path to travel between the two edges, which creates a narrow and leads to the SAW being rejected. It is unknown whether this effect causes a decrease in the connective constant over what is already achieved by the "no narrows" rule.

The benefit of validating the polygons of closable contour walks over validating the walks themselves is illustrated in Figure 4.1. Applying the new "no narrows" rule has a positive impact in both cases when compared to taxi walks. The non-closable counts found by validating the SAWs themselves give an increasing difference that suggests the "no narrows" rule reduces the connective constant. The closable contour walks described above create a greater difference at far lower values of n. However, the improvement over the non-closable case begins to decrease near $n = 40$. Whether the closable and non-closable bounds will eventually converge is unknown as it depends on whether closability strengthens the "no narrows" rule enough to alter the connective constant.

4.2 Contour Walk Validity Proofs

The following proofs closely mirror the referenced proofs by Blanca et al. [1]. In addition to the changes necessitated by the new rule imposed on the SAWs, minor changes have been made to simplify the relationship between contours and contour walks.

Lemma 4.1 (Blanca et al. Lemma 4.1)**.** The series $\{c_n\}, n \geq 1$ is submultiplicative.

Proof. To show that $\{c_n\}$ is submultiplicative, we split an arbitrary contour walk W into two pieces and show that both pieces must themselves be contour walks. Let W have length $m+n$ and be the concatenation of a length m piece called M and a length n piece called N. The number of possible concatenations of contour walks of lengths m and n is equal to $c_m c_n$. Not all concatenations will themselves be valid contour walks. For example, $HHHVVVHHH$ and VVV are both valid contour walks, but their concatenation $HHHVVVHHHVVV$ is invalid as it intersects itself before concatenating some non-empty series of steps to return it to the origin. Showing that every contour walk of length $m+n$ is included in the set of concatenations of valid length m and n contour walks proves the desired $c_{m+n} \leq c_m c_n$.

Consider M and N to be sets of edges such that M begins at the origin and N begins at M's endpoint. Because W is a valid contour walk, it must be closable by appending some series of steps such that the completed polygon is self-avoiding, follows the Manhattan lattice, takes no consecutive turns, and has no narrows. Let C be that series of steps. The concatenation of $N+C$ then closes M, while the concatenation of $C+M$ closes N. These concatenations result in identical polygons. M is therefore a contour walk. However, as N does not begin at the origin, it is not a contour walk.

To show that N is mapped to exactly one contour walk beginning at the origin, note that the Manhattan lattice is vertex-transitive. This means that for each vertex $(x,y) \in \mathbb{Z}^2$, there exists a transformation $f_{(x,y)} : \mathbb{Z}^2 \to \mathbb{Z}^2$ that sends (x,y) to the origin while preserving the relative orientation of every edge. $f_{(x,y)}$ first translates every vertex by $(-x,-y)$. The following transformations depend on the parities of x and y. If both are even, no further transformation is necessary because the outgoing edge directions of such an (x,y) match those of the origin. If x is odd, a reflection over the x-axis is required, and if y is odd, a reflection over the y-axis is required. Using this mapping, the outgoing edge directions of (x,y) may be made to match those of the origin while maintaining each vertex's relationship with its neighbors. By applying $f_{(x,y)}$ to the initial vertex of N, it is mapped to exactly one contour walk that maintains all the properties shown above.

Note that considering the binary representation of N (a substring of the binary representation of W) as a contour walk starting from the origin and following the prescribed edge directions of the Manhattan lattice implicitly performs the transformation $f_{(x,y)}$. □

Corollary 4.1.1. $c_n \leq f_c(n)\mu_c^n$ for some subexponential function $f_c(n)$.

Proof. Let $d_n = \log c_n$ for all $n \geq 1$. Because $\{c_n\}$ is submultiplicative, $c_{m+n} \leq c_m c_n$. Taking the logarithm of both sides gives $d_{m+n} \leq \log(c_m c_n) = d_m + d_n$, proving that $\{d_n\}$ is subadditive. Applying Fekete's Lemma to $\{d_n\}$ guarantees that $\lim_{n\to\infty} \frac{d_n}{n}$ exists; let its value be $\log \mu_c$. Fekete's Lemma also says that $\log \mu_c$ is equal to the infimum of $\frac{d_n}{n}$, the smallest real number less than or equal to every $\frac{d_n}{n}$.

This implies any c_n obtained for some finite n can be used to calculate an upper bound $\frac{d_n}{n} \geq \log \mu_c$, or equivalently $c_n^{1/n} \geq \mu_c$. Raising both sides to the power of n gives $c_n \geq \mu_c^n$. Thus, for every n, the number of contour walks is greater than the connective constant raised to the power of n. However, the connective constant μ_c is the limit of $c_n^{1/n}$, implying μ_c fully accounts for the exponential growth rate of c_n. Therefore, $c_n = f_c(n)\mu_c^n$ for some subexponential function $f_c(n)$. □

Lemma 4.2 (Blanca et al. Lemma 3.10)**.** Let m_x, m_y be integers such that $(m_x, m_y + \frac{1}{2})$ is the peak of a $\wedge$-shaped turn in a contour Γ. Then $(m_x - \frac{1}{2}, m_y)$ is adjacent to $(m_x, m_y + \frac{1}{2})$ in Γ, and

$(m_x, m_y + \frac{1}{2})$ is adjacent to $(m_x + \frac{1}{2}, m_y)$. The tuple (Γ, m_x, m_y) can be mapped to a single contour walk of length $4\ell - 1$ through a translation that sends $(m_x, m_y + 1/2)$ to the origin, a rotation counterclockwise through $\frac{\pi}{4}$ radians, an expansion by a factor of $\sqrt{2}$, and removing the edge which (after the previous transformations) connects the origin to $(0, -1)$.

Proof. The new rules enforced on contour walks eliminate only those walks that cannot be part of any valid contour. Contour walks are required to be closable, which is true for any walk created using the above transformations applied to a valid contour. Replacing the edge between the origin and $(0, -1)$ closes such a walk. Additionally, the new "no narrows" rule only eliminates walks that violate the contour property described in Theorem 3.2. □

Lemma 4.3 (Blanca et al. Lemma 3.4)**.** Contour walks are contour bounding.

Proof. By Lemma 4.2, each element $\Gamma \in C_\ell^m$ is fully described by specifying a vertex $(m_x, m_y) \in \mathbb{Z}^2$, followed by a contour walk of length $4\ell - 1$. Multiplying the number of possible vertices (m_x, m_y) by the number of contour walks of length $4\ell - 1$ therefore gives an upper bound on $|C_\ell^m|$. As Γ is a simple closed curve of length $O(\ell)$ that encloses the origin, there are $O(\ell^2)$ possible points (m_x, m_y) which could describe of a $\wedge$-shaped turn in some Γ. By Corollary 4.1.1, there are at most $f_c(4\ell - 1)\mu^{4\ell-1}$ choices for the contour walk. Because $f_c(4\ell - 1)$ is subexponential, $g(\ell) = O(\ell^2) f_c(4\ell - 1)$ is also subexponential, and $|C_\ell^m| \leq g(\ell)\mu_c^{4\ell}$. □

Theorem 4.4 (Blanca et al. Theorem 2.3)**.** The hard-core model on $\mathbb{Z}^2$ with activity λ admits multiple Gibbs measures for all $\lambda > \mu_c^4 - 1$.

Proof. Lemma 4.3 shows that $|C_\ell^m| \leq g(\ell)\mu_c^{4\ell}$. Therefore, μ_c is a suitable replacement for μ_{taxi} in the proof of phase coexistence. Assuming $\lambda > \mu_c^4 - 1$, we can select some μ such that $\mu_c^4 - 1 < \mu < \lambda$. We then set m sufficiently large that $g(\ell)\mu_c^{4\ell} < \mu^{4\ell}$ for all $\ell > \sqrt{2}m/2$ (Blanca et al. Equation 6). The remainder of the proof is unchanged. □

Chapter 5

Numerical Methods

By Theorem 4.4, $\lambda_c \leq \mu_c^4 - 1$. Our attention now shifts to showing an upper bound on μ_c. This usually involves enumerating c_n to high values of n. Corollary 4.1.1 discusses the most common upper bound, $\mu_c \leq c_n^{1/n}$. As n increases, $c_n^{1/n}$ converges to μ_c from above. However, an upper bound that converges slightly faster was discussed by Alm [6]. His Corollary 1 implies that $\mu_c \leq (\frac{c_n}{c_1})^{\frac{1}{n-1}} = (\frac{c_n}{2})^{\frac{1}{n-1}}$. Contour walks satisfy the condition $K_1 = 1$, meaning the two contour walks of length 1 can be mapped to each other through some combination of translations, rotations, and reflections which preserve the orientation of the Manhattan lattice. The requisite series of translations is in fact just one: a reflection over the line $y = x$. For the rest of this thesis, I use Alm's inequality to convert contour walk counts to upper bounds on μ_c.

5.1 Practical Closability

Enumerating contour walks requires determining which SAWs are closable. This procedure is complicated by the possibility that an exhaustive search for the origin from the endpoint may not terminate. For example, none of the infinite extensions from the endpoint of $HVVVHHHVVVV$ reach the origin in a way that follows the four rules. We detect walks like this one using two tests.

Let the bounding box of a SAW be the set $\{(x,y) \in \mathbb{Z}^2 \mid \min_x \leq x \leq \max_x \wedge \min_y \leq y \leq \max_y\}$, where $\min_x, \max_x, \min_y$, and $\max_y$ are the minimum and maximum x- and y-coordinates achieved by any point in the SAW. A SAW W is forward extendable if there exists some series of steps F such that the concatenation $W+F$ terminates at a vertex outside the bounding box of W. Here, we additionally require $W+F$ to satisfy the four contour walk polygon rules. Backward extensibility is defined similarly for some concatenation $B+W$, where the steps of B translate the bounding box of W so that it continues surrounding W. Due to the symmetry of the Manhattan lattice, a walk W is backward extendable if and only if its reversed walk W^R is forward extendable. For example, $W = HVV$ has $W^R = VVH$. Forward-backward extensibility is defined similarly for a simultaneous concatenation $B+W+F$, where we require both endpoints to be outside the translated bounding box of W.

A walk that is forward-backward extendable while following the four rules is trivially closable into a polygon that also follows the four rules. The path between the two endpoints can conservatively circumnavigate the bounding box of the walk with a large rectangle that approaches both endpoints parallel to the step that left the bounding box. This is the first test for closability. To simplify the computation, we approximate forward-backward extensibility as a check for forward extensibility, and then an independent check for backward extensibility. Both checks require a breadth-first search from one endpoint, terminating either when a path outside the bounding box is discovered or all possible paths are exhausted. Whether these checks admit any walks that are not forward-backward extendable in the context of the four contour walk rules is unproven. However, any forward-backward extendable walk must pass those two checks, and we accept any such walk as a contour walk.

The second test for closability follows closely from the first. Any walk that fails the first test must either be not forward extendable or not backward extendable. Either case is detected by the

breadth-first search exhausting all of its options, which is possible if the endpoint being checked is enclosed by the walk with no escape. Such a walk may still be closable if both endpoints are within the same loop. In such a case, one of the examined paths must connect the endpoints. This defines the second test. If some path during the breadth-first search connects the endpoints while following the four rules, the walk is a contour walk and there is no need to check the other extensibility direction (if not already checked). If all paths out of an endpoint are examined without discovering the other endpoint or a point outside the walk's bounding box, the walk is not closable and therefore is not a contour walk.

5.2 Transfer Matrix Algorithm

The exponential growth of SAW counts makes them difficult to enumerate using brute force. Taxi walks have only been enumerated exactly to a length of 70, which gives the bounds $\mu_{taxi} < 1.5973$ and $\lambda_c < \mu_{taxi}^4 - 1 < 5.5082$ [2]. This is far from the lowest previous bound $\lambda_c < 5.3485$ because requiring exact enumerations is computationally wasteful. More effective approaches, such as Alm's method and the transfer matrix method used here, approximate the relationship between walks to achieve a tighter final bound, as demonstrated by Couronné on the general square lattice [3].

The central observation of the transfer matrix method published by Pönitz and Tittmann is that, when considering loops of length at most k, the early steps of some SAW with length near k do not contribute any information about which steps appended to the SAW make the result invalid. [7]. For example, consider $k = 12$ and the contour walk $HHHH$. This walk ends at the point $(4,0)$, and on the Manhattan lattice would require 12 more steps to return to the origin. The minimum-length loop from the endpoint to the origin is $HHHHHVVVHHHHHVVV$, with a length of 16. This means that when considering only loops of length 12 or less, the initial step of this walk has no importance. Reducing $HHHH$ to HHH by removing the first step maintains every series of steps that can be appended to the SAW such that the result has a length at most 12 and is self-intersecting. We can use this idea to create a finite automaton representing the transitions between SAWs for loops of length at most k.

Many steps of building the automaton are similar to brute force enumeration. We first consider the empty contour walk, which sits at the origin. For each contour walk, we extend it in both the horizontal and vertical directions. That is, we append H and V to two separate copies of the current walk. We then check that each result is a valid contour walk. Every walk encoded in this way necessarily follows the Manhattan lattice, and confirming that a walk takes two consecutive turns is as simple as checking if the most recent three steps are of the form HVH or VHV. That the walk is self-avoiding is verified using two $O(n)$ scans through the string representation. The first determines the coordinates of the walk's endpoint, and the second compares those coordinates to every other coordinate reached during the walk. Finally, the "no narrows" condition is verified with another $O(n)$ scan searching for two adjacent vertices adjacent to the endpoint and the vertex preceding the endpoint. Despite the definition of contour walks implying these conditions are checked only after a polygon is closed, checking them after every individual step is appended speeds up the computation by eliminating invalid walks as early as possible. The closability check follows a similar procedure of repeatedly appending H and V and verifying that the results are valid. When a closed polygon is found, the length-three substrings where the endpoint meets the

origin are checked to ensure they are not two consecutive turns.

The step after validation is where the transfer matrix approach differs from the brute force approach. While the brute force algorithm would add every valid walk to a queue and repeat the process until it has generated every possible walk of length n, building the automaton instead adds only novel walks to the queue. Walks determined to have transitions equivalent to a shorter existing walk instead create a transition from their parent to the shorter walk. A walk has a shorter equivalent if it cannot intersect the origin or create a narrow including the origin within k total steps.

These conditions could be checked with a breadth-first search out from the endpoint of the walk and avoiding all vertices used in the walk, but this is too slow in practice. Other implementations of this algorithm use the Manhattan distance to the origin to approximate the origin's reachability. However, given the relative complexity of the rules constraining these walks, I opted for a more precise approach. I generated the minimum number of steps necessary to reach the origin, $(0,1)$, or $(0,-1)$ from every point in the box U_{100} while following the contour walk rules. The rule forbidding consecutive turns limited the valid first steps of the origin-seeking walk based on the final two steps of the origin-leaving walk. To account for this, I stored one minimum path length for each of the four possible sets of final two steps HH, HV, VH, and VV. The total number of path lengths generated is then 161604, though only those within k units of the origin are needed. These counts were generated by reversing the Manhattan lattice and generating walks outwards from the goal points. The only error from this approximation comes from the possibility that the shortest path to one of the goal points may be obstructed by the walk itself.

Walks that cannot intersect the origin or create a narrow including the origin in k total steps have transitions roughly equivalent to the walk created by removing their initial step. Note that the walk created by this removal may need to be reduced to an even shorter walk by repeating the same condition and procedure. The automaton is stored as an array of states, where each state contains the walk's information and the indices of its two children. Here, "children" are the states a given state transitions into. Usually, a state's children represent the walks created by appending H and V, respectively. However, the reduction procedure occasionally causes a state to transition to a much shorter state. Note that invalid child states result in a child index of zero, which is ignored. As novel walks are discovered, their states are appended to the automaton array. The number of novel states for loops up to length k is exponentially smaller than the number of contour walks of length k, allowing the automaton to be generated in a fraction of the time the brute force algorithm requires.

The automaton itself does not yield any information about the connective constant of the walks; it must be "run" for n iterations to generate an upper bound on c_n. Here, running the automaton refers to placing a count of 1 on the H state and allowing the value to propagate along the transitions. We begin with H instead of the empty state because of the symmetry of the Manhattan lattice. All walks beginning with V can be accounted for by multiplying the final sum by 2. Beginning with H replaces the first step, so the second step propagates the sum such that HH and HV both have a count of 1, and H has a count of 0. In general, the count on state s after iteration i is the sum of the counts after iteration $i-1$ on all states that transition to s. Note that the counts on the states may become quite large. I used a vector of unsigned 64-bit integers to store each count. The final upper bound on c_n is obtained by summing the counts of all states after n iterations. This process of obtaining a bound for c_n from the automaton can be viewed as the power method for estimating the largest eigenvalue of a matrix, specifically the transfer matrix of the automaton.

$c_1 = 2$	$c_{11} = 284$	$c_{21} = 28892$	$c_{31} = 2831570$	$c_{41} = 270185704$
$c_2 = 4$	$c_{12} = 444$	$c_{22} = 45890$	$c_{32} = 4460796$	$c_{42} = 426179458$
$c_3 = 6$	$c_{13} = 712$	$c_{23} = 72618$	$c_{33} = 7049436$	$c_{43} = 671731072$
$c_4 = 10$	$c_{14} = 1140$	$c_{24} = 114530$	$c_{34} = 11138318$	$c_{44} = 1057157282$
$c_5 = 16$	$c_{15} = 1814$	$c_{25} = 181460$	$c_{35} = 17576600$	$c_{45} = 1666923318$
$c_6 = 26$	$c_{16} = 2868$	$c_{26} = 287548$	$c_{36} = 27678476$	$c_{46} = 2627884090$
$c_7 = 42$	$c_{17} = 4568$	$c_{27} = 454518$	$c_{37} = 43700880$	$c_{47} = 4140392814$
$c_8 = 68$	$c_{18} = 7278$	$c_{28} = 716398$	$c_{38} = 68983362$	$c_{48} = 6514829530$
$c_9 = 110$	$c_{19} = 11534$	$c_{29} = 1133402$	$c_{39} = 108787714$	
$c_{10} = 178$	$c_{20} = 18200$	$c_{30} = 1793012$	$c_{40} = 171252602$	

Table 5.1: Counts of contour walks, enumerated using brute force.

The fact that iterating the automaton gives an upper bound on c_n is necessary to ensure the resulting μ_c and λ_c are both upper bounds. For SAWs not constrained to be closable, iterating the automaton to $n \leq k$ gives an exact enumeration. However, iterating to $n > k$ means approximating the true count with an automaton that considers only loops of length at most k. In this case, the reduction procedure was applied to walks based on the relatively small value of k, when some of those reduced walks would need to be considered novel to fully model loops of length n. Another source of over-approximation is the closability rule. A walk with children that are not closable could be reduced to a walk whose children are closable. For example, $HVVVHHHVVVV$ traps the origin; the endpoint can only reach it by creating a narrow. Therefore, its parent state $HVVVHHHVVV$ has one child which is not closable. If $HVVVHHHVVV$ were reduced, it would become $VVVHHHVVV$, a state with two closable children.

The reduction operation leads to upper bounds on c_n because every reduced state has fewer invalid descendants than the state it was reduced from. If some series of steps appended to the reduced state results in a walk that self-intersects, appending that same series of steps to the unreduced walk will also lead to an intersection. The same logic holds for descendants of the reduced state which take two consecutive turns, contain a narrow, or become unclosable. Visualizing the transitions between every valid contour walk as an infinite tree rooted at the empty walk, this observation states that redirecting edges backward, towards an ancestor node, cannot decrease the eigenvalue of the resulting graph. The automaton is one such graph constructed to approximate the infinite tree.

5.3 Results

Enumerating contour walks exactly to length $n = 48$ using the brute force approach required 10 hours of computation on a single core of an Intel i7-10750H. The results are shown in Table 5.1. These exact counts are not used to compute the final bound but are included for context.

For the main upper bound computation, I generated automatons for several loop length parameters k and iterated each to $n = 2500$, measuring the c_n bound for every n multiple of 100. The results are shown in Figure 5.1. To achieve the best possible upper bound for μ_c and λ_c, k and n must be set as high as the available computational resources allow. For a fixed k, increasing n gives

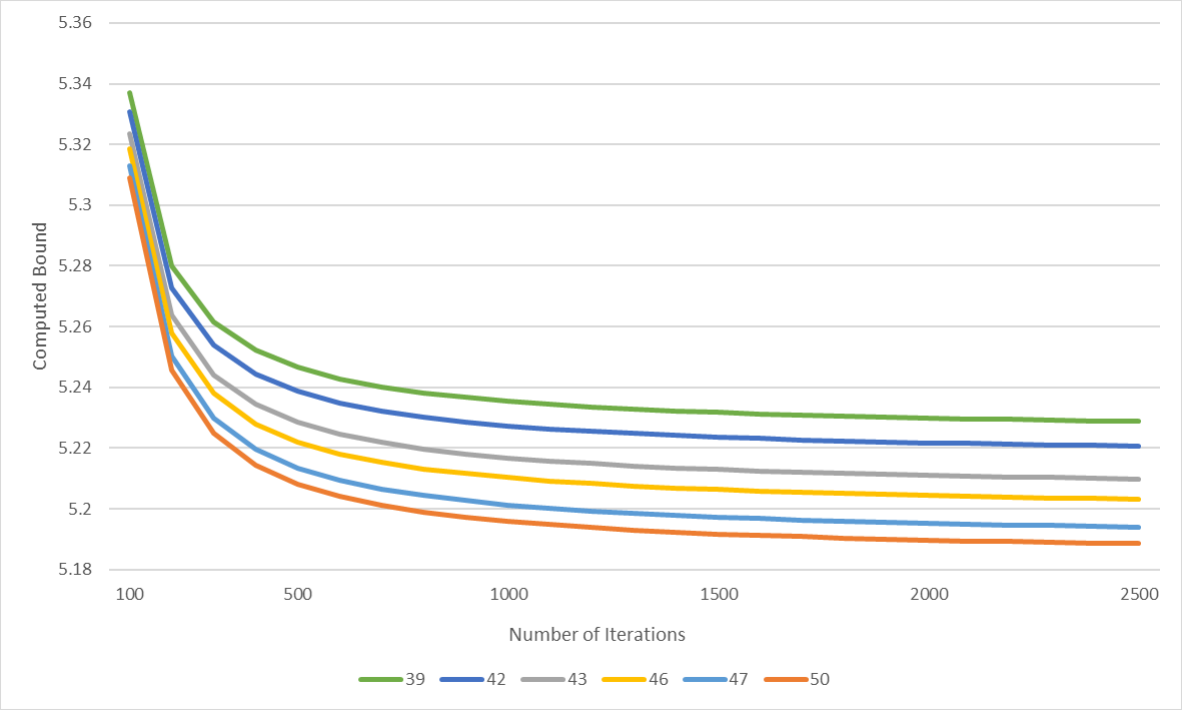

Figure 5.1: Computed bounds on λ_c by iteration count for several loop length parameters k.

diminishing returns. Increasing k also gives diminishing returns, but the available improvement to λ_c from increasing k past 50 is still likely on the order of 0.01 based on the trend in the graph.

$k = 50$ and $n = 2500$ are the maximum parameters used for this thesis. That computation took 3 hours (including 20 minutes to generate the automaton) on a single core of an Intel i7-10750H, but required 11GB of RAM to store the 58947413-state automaton and the counts for each state. I attempted $k = 51$, but it requires over 12GB of RAM, which is beyond the limits of my computer. The $k = 50$ automaton computed the upper bound $c_{2500} < 6.9408 \cdot 10^{494}$, the integer form of which is too large to write here in full. I used Python to handle the integer outputs, which are large enough to overflow the floating-point data type. For this reason, I used the logarithm form of the conversion from c_n to an upper bound on μ_c: $\mu_c \leq exp(\frac{1}{n-1}\log(\frac{c_n}{2}))$, which yielded $\mu_c \leq 1.5773$. Finally, the formula $\lambda_c \leq \mu_c^4 - 1$ yields the promised $\lambda_c \leq 5.1885$.

Chapter 6

Future Work

The gap between the new upper bound $\lambda_c < 5.1885$ and the conjectured value $\lambda_c \approx 3.796$ is still quite large, but I am certain the limits of the Peierls argument have not yet been reached. Simply utilizing more computational power might push the bound lower by a few hundredths. The previous best bound was computed using Alm's method [2]. The transfer matrix method used in this thesis gave a small improvement of 0.002 when enumerating taxi walks, but an implementation of Alm's method with the new "no narrows" and closability rules might still be beneficial. A new lower bound on μ_c including the rules presented in this thesis could be computed using irreducible bridges, as was done for taxi walks [1]. This would indicate how far the current approach can be stretched before more theoretical alterations are required.

Definition 3.1 shows a bijective mapping between a set γ and a contour Γ. The construction of γ implies two equivalent mappings. First, R' and Γ have a one-to-one mapping. Second, the even vertices in R' adjacent to vertices outside R' are sufficient to specify γ and thus Γ. Exactly one R' corresponds to any such set of even vertices because R' has no holes; it must include every vertex enclosed by the even vertices. These facts provide a way to check if an enumeration scheme admits rectilinear polygons that do not correspond to valid contours. Every polygon generated by some set of rules, such as the four rules governing closed contour walks, can be confirmed to have a corresponding hard-core configuration by placing a set of adjacent even vertices along the inside edge. If the scheme admits polygons that are not valid contours, there must be some additional rule that could be included. On the other hand, if every generated polygon is a valid contour, the set of rules is complete. I have not yet run such a validation procedure on the polygons associated with contour walks, but I believe it would be worthwhile regardless of the outcome. If a set of rules is proven to perfectly model the contours, future efforts can be safely focused on other approaches to reducing the upper bound.

A related possibility is enumerating walks along the adjacent even vertices immediately inside Γ instead of enumerating SAWs which mimic Γ. While showing a bijection between Γ and the even vertex set is easy, a relationship between the connective constant of walks on the even vertex set and μ_c remains to be shown. The exact properties of such walks would need to be formalized as the properties of Γ have been. I believe the difficulty and computational complexity of this approach would be similar to the current one, but it may still be worth looking into.

References

[1] A. Blanca, Y. Chen, D. Galvin, D. Randall, and P. Tetali, "Phase coexistence for the hard-core model on z2," *Combinatorics, Probability and Computing*, vol. 28, no. 1, pp. 1–22, 2019.

[2] L. Haoquan, "Phase coexistence for the hard-core model on z2: Improved bounds," 2021.

[3] O. Couronné, "New upper bound for the connective constant for square-lattice self-avoiding walks," *arXiv preprint arXiv:2211.16146*, 2022.

[4] A. J. Guttmann, "Bounds on self-avoiding walks on directed square lattices," *Journal of Physics A: Mathematical and General*, vol. 16, no. 1, pp. 3885–3984, 1983.

[5] G. Grimmett, A. Holroyd, and Y. Peres, "Extendable self-avoiding walks," *Annales de l'Institut Henri Poincaré D. Combinatorics, Physics and their Interactions*, vol. 1, 07 2013.

[6] S. E. Alm, "Upper bounds for the connective constant of self-avoiding walks," *Combinatorics, Probability and Computing*, vol. 2, no. 2, p. 115–136, 1993.

[7] A. Poenitz and P. Tittmann, "Improved upper bounds for self-avoiding walks in zd." *Electr. J. Comb.*, vol. 7, 04 2000.