UNIVERSITY OF LIVERPOOL

Algorithmic and Combinatorial Problems in Crystal Structure Prediction

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Duncan Adamson

November 2021

Abstract

There are many challenges facing the global community that require the development of new materials requiring specific properties. These range from the need to develop strong, light weight alloys, to new insulating structures and flexible conductive materials. One fundamental form of matter is the *Crystal*. A crystal is defined by a unit cell that is periodically repeated infinitely in all dimensions. Each unit cell is a parallelepiped box containing a set of ions at fixed positions. This periodicity forms the primary advantage of crystals over other forms of matter, allowing the properties of a crystaline material to be determined from the comparatively small structure of the unit cell.

The problem of determining the structure of a crystal is known as Crystal Structure Prediction (CSP). The goal is to take a set of ion species and determine the unit cell by minimising the *energy* within the unit cell. The energy can be approximated as the sum of pairwise interactions between the ions in the unit cell and every other ion in the crystal. The pairwise interaction is determined by an energy function. The most commonly used of these is the *Buckingham-Coulomb* function, which determines the energy by a function of the distance and the *species* of the ions. The species of an ion can be thought of as the class to which an ion belongs, determining the interaction between each ion.

This thesis looks at CSP from a combinatorial and optimisation perspective. The first key results are on the computational complexity of CSP. Two models of CSP are considered in this direction; an abstraction based on constructing a unit cell of arbitrary size on a discrete grid so as to minimise pairwise energy, and an abstraction based on removing ions from an existing unit cell to minimise the pairwise energy. In both models, the energy functions are restricted to general classes of functions containing the Buckingham-Coulomb potential. The first model is shown to be undecidable in the general case, where the size of the unit cell is unbounded, Further, the first model is shown to be NP-hard to solve, as well as being NP-hard to approximate within any positive factor. The second model is also shown to be both NP-hard to solve and NP-hard to approximate within a factor of $n^{1-\epsilon}$ where n is the number of ions in the initial unit cell for any $\epsilon > 0$. These results form the first formally proven hardness results for crystal structure prediction, providing a foundational argument for the computational complexity of general techniques of CSP.

Despite the limitations placed on the general models of CSP by the hardness results, this thesis also introduces a novel approach to CSP in several settings. More specifically, we seek to construct a diverse *sample* of potential crystal structures from some implicit description of the crystals. We do so by developing a combinatorial representation for crystal structures. These structures seek to cover several symmetries inherent to the space occupied by crystal structures. To this end, we provide the first relation of crystals to *necklaces* and *bracelets*, classes of cyclic words. Beyond this connection, we formalise several new classes of cyclic words, most notably *multidimensional necklaces* that are of independent interest beyond the use by CSP.

These classes are used as the basis of the sampling process. The sampling process is formulated in the form of the k-centre problem for these combinatorial structures. Despite the general hardness of both the k-centre problem and CSP, we provide two strong polynomial time approximation algorithms for several variations on crystal structures. The first algorithm operates on the set of necklaces with no constraints placed on them, running in linear time. The second algorithm works in several settings including both fixed-content and multidimensional necklace running in polynomial time.

In order to use our second k-centre algorithm, a set of new results for *ranking* classes of cyclic word are derived. The ranking problem asks for the number of members of some ordered set smaller than a given member of the set. This thesis provides new algorithms for ranking bracelets, fixed-content necklaces, necklaces with forbidden subwords, multidimensional necklaces, and fixed content multidimensional necklaces. In addition to these algorithms, we provide new efficient algorithms to *count*, *generate* and *unrank* multidimensional necklaces.

Overall this thesis opens new links between crystal structure prediction and combinatorial and computational complexity questions in computer science. These new results provide new insights for both the fields of material science and computer science. Notably, our formal models of crystal structure prediction along with the corresponding results on the computational complexity of these models provide a strong indication on the complexity of more general crystal structure prediction models. Additionally, in trying to determine new ways of representing crystals as combinatorial structures we have constructed several new results regarding cyclic words. These include providing approximation algorithms for the k-centre problem on several classes of cyclic words, running in polynomial time relative to the description of the graph, conditional on being able to rank and unrank each class. The motivation provided by these approximation algorithms has in turn led to techniques to rank several classes of cyclic words, including bracelets, necklaces with forbidden subwords, necklaces constrained by linear equations, and multidimensional necklaces, the latter two of which we introduce.

Acknowledgements

First and foremost, I would like to thank my primary advisor Prof. Igor Potapov for his support throughout this project. Not only has he been invaluable in guiding my research, but also in helping with presenting my results and generally improve as a researcher. It has been a pleasure to work with Igor, and I look forward to having many opportunities in the future to continue to do so. I would additionally like to thank Dr. Vladimir V Gusev and Dr. Argyrios Deligkas, who have collaborated with me throughout all my research at Liverpool. As with Igor I am looking forward to continuing to collaborate with both Argyrios and Vladimir beyond simply publishing our current results. I would further like to thank my examiners Prof. Prudence Wong and Prof. Florin Manea for their helpful comments and suggestions both for my thesis, and for further work. Finally, I would like to thank my Friends, Family and Partner who have supported me throughout this endeavour. This thesis and the research involved in it has been the most demanding task in my life so far, and would not have been possible without their support.

Contents

List of Figures

Chapter 1

Introduction

Crystal Structure Prediction (CSP) is a highly studied and very complex problem fundamental to the fields of both material science and chemistry. The most general versions CSP asks for the chemical structure for a given set of ions - electromagnetically charged atoms - based on some model of interaction. At the same time, the discretised representation of a crystal forms a natural class of combinatorial objects corresponding to cyclic words in multiple dimensions. This leads to many natural algorithmic and combinatorial questions such as how to count or generate the number of potential crystals for some given set of elements.

Being able to solve CSP efficiently would lead to an explosion in the development of new materials. Despite this motivations, there has been a lack of work on CSP as a theoretical computer science problem. To the best of our knowledge the first formalisation of any method of CSP as a computer science problem was only published within the last year [2, 7]. Additionally, we believe that this thesis is the first work to connect cyclic words to crystals. This connection has lead to the identification of multidimensional necklaces as a new class of combinatorial objects.

Our results can be split into three themes. First are the results concerning the hardness of our formalisation of CSP. We show that the problem is undecidable in the most general case. Further we show that for several realistic restrictions CSP remains NP-hard to solve as well as NP-hard to approximate within any positive factor in polynomial time.

With optimally solving the problem being seemingly intractable, the second set of results look at the idea of *sampling* a diverse selection of potential crystal structures. We use the k-centre problem as a basis for the sampling approach. Here we use the set of potential crystal structures as the vertices for a graph, with edges weighted by a similarity metric. These graphs are defined in terms of the set of ions, and size, resulting in an exponential size of graph relative to the input. Despite this challenge, we provide a set of approximation algorithms running in polynomial time relative to the size of the input.

The third set of results we provide are a set of combinatorial and algorithmic results on various classes of cyclic words. More precisely, we look at the classes of Necklace, Bracelets, Necklace weighted by solutions to Diophantine equations, and multidimensional necklaces, the latter two being introduced in this work. Our primary results in all cases are algorithms to *rank* and *unrank* members of these sets. Informally, the ranking problem asks for the number of elements in a set smaller than some given element while the unranking problem asks for the element with some number of elements smaller than it. Beyond general interest, these results are utilised by our approximation algorithms.

The remainder of this section is structured as follows. In Section 1.2 we provide an overview of the current state of CSP, focusing on the current approaches to the problem. Section 1.3 provides a brief overview of the k-centre problem. Section 1.4 looks at current results on cyclic words that are applicable to the crystal setting. Section 1.5 provides an overview of the main results from this thesis. Finally Section 1.6 provides an overview of where the results from this thesis have been presented externally.

1.1 Key background

This section serves two key functions. First, we provide some key background results for the main themes of this thesis. This serves to allow the reader to fully understand our further results, without requiring further reading into related results. Secondly, we highlight the connections between these results and our results. This section is divided into three subsections. Subsection 1.1.1 provides results regarding crystal structure prediction. Subsection 1.1.2 provides key results and definitions for the k-centre problem. Finally, Subsection 1.1.3 provides the key results for classes of cyclic words.

1.1.1 Crystal Structure Prediction

While the problem of crystal structure prediction (CSP) has been highly studied from the prospective of both chemistry and materials science, there has been very little work on CSP from a computer science point of view. At a high level, CSP takes as input some set of ions, and asks for the crystal structure corresponding to the given set of ions. A crystal structure is defined by a period called the *unit cell.* Each ion belongs to a class called a *species*, determining the properties of the ion. In this way, the unit cell is a periodic mapping from some set of ion species to the space $\mathbb{R}^3$. In the discrete setting, the unit cell instead maps to some grid, generally the integer grid $\mathbb{Z}^3$. In the most general formulations of CSP the size and shape of the unit cell is left open, however bounding the size is a common restriction for many cases of CSP.

Crystal structure prediction can be thought of as the problem of finding the "best" configuration of ions within a three-dimensional box. The quality of a configuration is determined by the pairwise interaction energy between every pair of ions. The pairwise interaction is determined by an energy function, with a negative energy corresponding to the ions attracting each other, while a positive energy corresponding to a repulsion between the ions.

A comprehensive overview of the current state of CSP from the materials science perspective is provided by Oganov [75]. To the best of our knowledge, there has been only two publications formalising CSP as a computer science problem. First has been our own work [2], representing CSP as a weighted graph problem where the goal is to select the set of vertices forming a minimum weight clique. A similar formulation of CSP was used by Antypov et al. [7] who provided heuristic results for CSP.

1.1.2 The k-Centre Problem

The k-centre problem is a classic computer science problem. The k-centre problem takes as input a graph $G = \{V, E\}$ and integer k, and asks for the set $\mathbf{s} \subseteq V$ of k vertices from G minimising the objective function $\max_{v \in V} \left(\min_{u \in \mathbf{s}} Distance(u, v)\right)$, where $Distance(u, v)$ is some function determining the distance between two vertices. As a classical problem there have been many results in understanding the k-centre problem.

In the general case the problem is known to not be in APX [47]. When the distance satisfies the triangle inequality the problem becomes significantly easier, admitting a polynomial time (relative to the size of the graph) approximation algorithm with a factor of 2 [41, 48]. Further, it is known no polynomial time approximation algorithm can achieve a factor better than 2 unless $P = NP$ [50, 85]. Additionally the k-centre problem is unlikely to be fixed-parameter tractable (FPT) in a context of the most natural parameter k [29].

1.1.3 Cyclic words

In this thesis we consider cyclic words to be equivalence classes of words under the *cyclic shift* operation. For example, the words equivalent to $aaab$ are $aaba, abaa$ and $baaa$. In general, we consider necklaces of length n over a given alphabet Σ. The most fundamental result for necklaces are the equations for counting the number of necklaces due to folklore [42]. These equations have been followed by algorithms to generate the set of all necklaces of length n over Σ in lexicographic order in constant amortised time [90]. More recently, the dual problems of *ranking* and *unranking* necklaces have been solved in polynomial time relative to the length of the necklaces [92]. The ranking problem asks for the number of cyclic words in some set $\mathcal{N}$ that are smaller than some given word $\bar{w}$. The unranking problem asks the the member of some set $\mathcal{N}$ that has a rank i.

We generalise several of these results to more complex settings. These include generalising the ranking and unranking algorithms to the set of *bracelets* (equivalence classes under the cyclic shift and reflection operations), fixed content necklaces (necklaces sharing a fixed Parikh vector), and necklaces with forbidden subwords. We further generalise the methods of counting, generating, ranking and unranking necklaces to the *multidimensional* setting, where necklaces are defined over multidimensional words.

1.2 Crystal Structure Prediction

A crystal is a structure defined by an repeating period called a unit cell. For our purposes the unit cell can be thought of as a three dimensional box containing *ions*, see Figures 1.1 and 1.2. Each ion belongs to a class called a *species*, determining the properties of the ion. In this way, the unit cell is a periodic mapping from some set of ion species to the space $\mathbb{R}^3$. In the discrete setting, the unit cell instead maps to some grid, generally the integer grid $\mathbb{Z}^3$. In the most general formulations of CSP the size and shape of the unit cell is left open, however bounding the size is a common restriction for many cases of CSP.

Crystal Structure Prediction can be thought of as the problem of finding the “best” configuration of ions within a three-dimensional box. The quality of a configuration is determined by the average pairwise interaction between each ion in the structure. The pairwise interaction in turn is determined by an energy function, taking as input the distance between ions and a set of parameters based on the ion species. A positive interaction between ions corresponds to a force pushing the ions apart, while a negative interaction

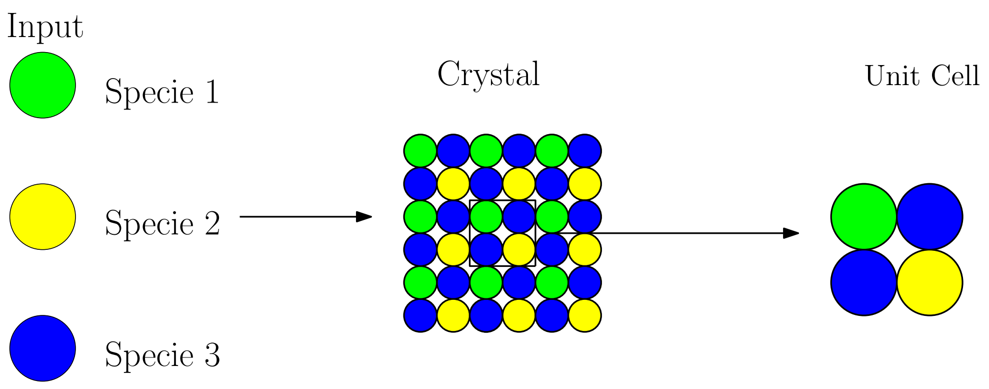

Figure 1.1: A high level example of the crystal structure prediction process. Here the input is a set of 3 species of ions (green, yellow, and blue), with each pair having some interaction determined by some function based on the distance and species. This set of ion species is transformed into a crystal structure (middle) defined by a repeating unit cell (right). Note that there are 4 equivalent unit cells, determined by where we take the "snapshot" defining the unit cell.

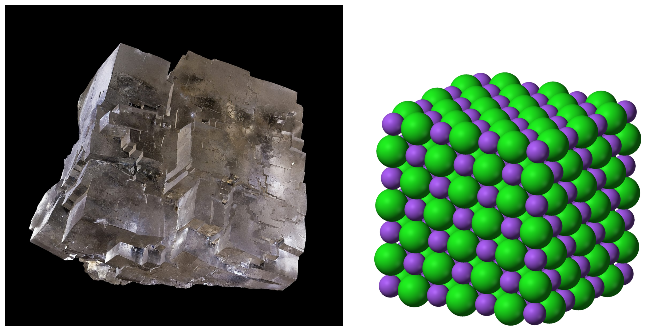

Figure 1.2: Example of a Sodium Chloride crystal (left) with the ionic structure (right).

indicates a force of attraction. The goal of CSP is to find a structure that corresponds to a stable structure, indicated by having the minimal average pairwise interaction [67].

Crystal Structure Prediction has been a noted open problem in materials science for over 30 years [70]. Despite this CSP has remained open due to the complexity of solving it optimally [16] and the combinatorial explosion following a brute-force approach. There have been many different heuristic approaches to CSP. An overview of several heuristic approaches is given in Section 1.2.1. One defining problem with all of these methods is a lack of any guarantees on optimality.

In addition to approaches to solve CSP, the problem has also been claimed to be NP-hard [77] without any formal proof of hardness. Two frequently cited results closely related to the NP-hardness of energy minimisation are those of Barahona [9] and those of Wille and Vennik and [102]. Barahona shows in [9] NP-hardness within the context of the Ising model, an energy model based on placing ± 1 charged vertices on a graph taking into account only interactions between connected vertices. The reduction works on a grid, where each vertex has degree at most 6, making the interaction very local. While this is close to CSP, the limitation to only very local interactions within the context of the electrostatic charge reduces the relevance of this result to CSP. Wille and Vennik show in [102] that the problem of placing ions for some given positions is in NP. However, the reduction goes only one way, and thus shows only containment in NP, and does not imply the NP-hardness of the problem. While both of these results are closely related, neither of them provides a satisfying answer to the CSP problem.

Our work strengthens these claims by showing that not only are many abstractions of CSP NP-hard, but also that several cases are in fact undecidable. This is an important result for two main reasons. Firstly, it strengthens the argument that CSP is intractable. Secondly, by analysing several verities of CSP we develop a deeper understanding of where the complexity of the problem originates, providing a basis for further work.

More precisely, we show that the natural formulation of the general CSP problem for discrete structures is undecidable. In this formulation, the only input is the species of ions and the interaction function, leaving the size of the unit cell, and the precise composition to be determined. This corresponds to the model used in some so-called *tiling* approaches to CSP [7, 22, 23]. We further show that when the size of the unit cell is constrained the problem remains NP-hard to approximate within any factor greater then 0. Additionally, we show that the related problem of determining the best set of ions to remove from a unit cell is NP-hard both to solve and approximate. Finally, we show that for the one dimensional problem, as used for example by MC-EMMA [23], may be solved by a parameterised algorithm.

1.2.1 Heuristic Methods of Crystal Structure Prediction

There have been many heuristic approaches to CSP. In this section we give an overview of several common approaches that are used in CSP. Before explaining the processes themselves, it is important to understand the notation of *relaxation* with regards to CSP. Informally, the relaxation process can be thought of as process of working out where the ions in a given unit cell would "settle". This corresponds to the process of letting the ions move from the starting positions according to the interaction model. In general, relaxation is computed by iteratively shifting each ion according to the energy model until the structure stabilises. While relaxation tools form a very powerful technique, they are computationally costly and often fall into local minimums, making them a tool best used sparingly.

Quasi-random sampling. One of the first approaches to CSP was what amounts to random sampling followed by the relaxation process. The main benefit of this approach is that the initial random sampling can be done very efficiently, moving the complexity to the relaxation process. Some examples of this style of approach are given by Freeman et al. [33], Pickard and Needs [81, 82], and Schmidt and Englert [95]. It should be noted that many of the more sophisticated approaches use random sampling to some degree. This widespread utilisation has motivated our k-centre based approach as a means of constructing a set of samples with some guarantee on the difference between samples.

Basin Hoping. A more sophisticated approach is that of basin hoping. This approach can be thought of as an extension of the relaxation process, with the goal of moving past the minimum energy's reached during the relaxation process. While this approach allows a more complete exploration of the state space, it only works when there is a large global minimum with relatively low barriers. Some examples of this kind of approach are shown in the work of Dyer et al. [27], Goedecker [39], and by Wales and Doye [100]. One related approach to the Basin hoping method is that of Antypov et al. [7], who looked at CSP through a combinatorial lens. They provide a local search algorithm for moving between basins by way of swapping ions within the structure.

Tiling. One more novel approach to CSP is the so-called tiling approach. These approaches use precomputed blocks as a building block in place of working on an ion by ion approach. Some notable examples of this is the work done by Collins et al. [22, 23] and by Mellot et

al. [73]. Note that these methods often act as a novel addition to the sampling or basin hoping approaches, using the tiles in the same way as the ions are used in those methods.
Computer Science based heuristics. Finally, there have been a variety of attempts to solve CSP through application of traditional computer science heuristic algorithms. Some notable examples of this are simulated annealing [79, 96], swarm optimisation [15, 101], and genetic algorithms [26, 66, 76]. While these have all worked to varying degrees, they lack any guarantees on optimality.

This thesis looks at CSP in three directions. First, in terms of understanding the complexity of the problem we look at the tiling models, both for blocks corresponding to a single ion and more general structures. Secondly, we look at an improved method for sampling potential structures. Finally we look at ways of uniquely representing crystals within a discrete space. We focus on the sampling problem as it is both a method of CSP and for its use in conjunction with other techniques.

1.3 The k-Centre Problem

The k-centre problem forms the basis for our sampling approach to CSP. We use the representation of crystals as described in Section 1.4 as a basis. This section provides some background on the k-centre problem. The k-centre problem is a classical graph problem. The objective in k-center problem is to find a set of k vertices for which the largest distance of any vertex of the graph and its closest vertex in this k-set is minimised. The numerous applications of the problem in various areas of computer science have lead to different definitions of connectivity and distance between the vertices depending on the setting at hand.

The k-center problem is a classical NP-hard problem, as such a great deal of research has been direct to trying to solve it. In the general case the problem is known to not be in APX [47]. When the distance satisfies the triangle inequality the problem becomes significantly easier, admitting a polynomial time (relative to the size of the graph) approximation algorithm with a factor of 2 [41, 48]. Further, it is known no polynomial time approximation algorithm can achieve a factor better than 2 unless $P = NP$ [50, 85]. Additionally the k-centre problem is unlikely to be fixed-parameter tractable (FPT) in a context of the most natural parameter k [29].

A different form of the k-center problem appears in stringology and it was linked with important applications in computational biology; for example to find the approximate gene clusters for a set of words over the DNA alphabet [65]. This problem is also NP-hard [30, 63]. Despite the hardness of the problem, there are fixed-parameter algorithms [43, 68] allowing some guarantee of optimality for solving the problem. The Closest String problem aims to find a new string within a distance d to each input of n strings and such that d is minimised. The natural generalisation of k-Closest String problem is of finding k-center strings of a given length minimising the distance from every string to closest center [37, 55]. This problem has been mainly studied for the popular Hamming distance. The major application of this distance is in the coding theory, but it also has been intensively used in biological applications aiming to discover a region of similarity or to design both probes and primers [62].

In this thesis we present and study a new variant of the *k-center problem* on the class of combinatorial *necklaces*. *Necklaces* are fundamental combinatorial structures [42], which are defined as a set of n-symbol strings over an alphabet of size k, that are equivalent under the cyclic shift operation. Necklaces are discussed in more details in Section 1.4.

In order to construct a graph from the set of necklaces for the k-centre problem, it is necessary to determine a similarity metric for necklaces. For the purpose of choosing a distinct set of crystals, we use subwords as a notion of similarity. The idea behind this approach is that local interactions between ions have much higher energy than long range interactions, and therefore are much more impactful on the overall structure of a crystal. Therefore, by choosing a set of necklaces with a diverse set of subwords, the corresponding unit cells should provide a good sample from which to find the optimal solution. To this end, we turn to the *the overlap coefficient* [21, 84, 86]. Informally, the overlap coefficient is a measure of the number of common subwords between necklaces, normalised by the total number of subwords in each necklace.

The graph in this setting corresponds to the set of all necklaces of some given length n over an alphabet Σ of size q. This setting has some unique properties. While the graph can be completely represented, it is of exponential size relative to the description in terms of n and q. Despite this, the graph has a highly symmetric structures due to the nature of necklaces. We show that verifying a solution to the k-centre problem for necklaces can not be done in polynomial time relative to n and q unless $P = NP$, indicating that the k-centre problem itself is likely to be at least NP-hard.

The idiosyncrasies of this setting motivates study for new algorithms to solve the problem in polynomial time relative to length of the necklaces and size of the alphabet. This corresponds to a logarithmic time algorithm relative to the size of the graph. We provide approximation algorithms running in such time for the various restrictions and generalisations of necklaces as described in Section 1.4.

1.4 Cyclic Words

The third theme of this thesis is that of the combinatorial representation of crystals in a discrete space. One natural formulation of such a structure would may be to use a word of dimensions corresponding to those of the unit cell. In this formulation, an alphabet Σ is constructed with a symbol corresponding to each ion species, along with an additional symbol to represent empty space. While this formulation allows many aspects of the crystal to be represented, it does not account for the symmetry inherent to crystal structures.

The main class of symmetry we consider is that of *translational symmetry*. Informally, translational symmetry can be thought of as the equivalence of two crystal under translation in space. This intuitively make sense in the context of real structures, where two different "snapshots" of a unit cell both represent the same global structure. Figure 1.3 provides an example of this equivalence. In order to represent crystals in this manner we turn to necklaces - an equivalence class of cyclic words - as a natural means to capture a periodic discrete structure. At a high level, the idea is to construct an alphabet Σ containing a symbol representing each ion in the crystal along with an additional symbol to represent empty space.

In the one dimensional case, each symbol represents a "layer", a precomputed 3D structure. These layers can be encoded as symbols from a finite alphabet and the discrete representations of materials (periodic crystals) can be seen as combinatorial necklaces (cyclic words) due to their invariance under the cyclic shift operation, e.g. see $SrTiO_3$ representation as two layers in Figure 1.4 (right). Alternatively, crystals can be represented by 2D slices as two dimensional necklaces or even on purely atomic scale by three dimensional necklaces over a finite alphabet of ions from a periodic table of elements, see three dimensional necklace representation of $SrTiO_3$ on Figure 1.4 (middle). To account for symmetry, we represent one dimensional crystal structures as either combinatorial necklaces or bracelets. Informally a necklace is a cyclic word of length n [42], while bracelets are necklaces under the reflection operation.

Both necklaces and bracelets are heavily studied objects. The most fundamental result is that of counting the number of necklaces. Graham et al. provide equations for counting both the number of necklaces and aperiodic necklaces, known as Lyndon words [42]. The

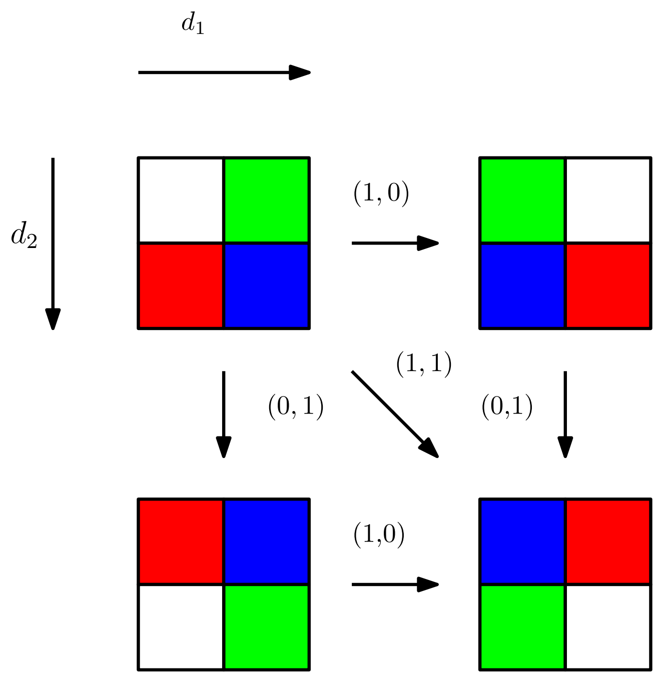

Figure 1.3: An illustration of translational symmetry for a 2×2 word. Note that all four words can be reached from one another through cyclic shifts, denote (g_1, g_2).

same result for both bracelets and aperiodic bracelets, known as Lyndon bracelets was given by Gilbert and Riordan [38].

Building on the results of counting the number of necklaces, Lyndon words, bracelets and Lyndon bracelets has been a set of algorithms to generate each set for a given length and alphabet in lexicographic order. The first algorithms for generating necklaces were provided by Fredricksen and Kessler [32], and Fredricksen and Maiorana [31], which were later proven to run in constant amortised time (CAT) by Ruskey et al. [88]. Cattell et al. provided a further CAT algorithm for the generation of necklaces and Lyndon words [17]. Sawada provided a similar CAT algorithm for bracelets and Lyndon bracelets [93].

One set of results that has been of particular interest to us is the work on *ranking* and *unranking*. Informally, the ranking problem, also known as the indexing problem, asks for the number of member of some given ordered set smaller than some element. Unranking is the reverse process, asking for the element of some ordered set with a given rank. Ranking has been studied for various objects including partitions [103], permutations [72, 74], combinations [97], etc. Unranking has similarly been studied for objects such as permutations [74] and trees [44, 78]. The first class of cyclic words to be ranked were *Lyndon words* by Kociumaka et al. [60] who provided an $O(n^3)$ time algorithm, where n is the length of the word. An algorithm for ranking necklaces was given by Kopparty et al. [61], without tight bounds on the complexity. A $O(n^2)$ time algorithm for ranking necklaces was provided by Sawada et al. [92]. More recently, we have provided an $O(q^2 \cdot n^4)$ time algorithm for ranking bracelets [4].

Beyond classical necklaces and bracelets, we consider the center problem on several generalisation of classical necklaces. The first of these is the idea of multidimensional necklaces which generalises classical necklaces and two dimensional necklaces. While a relatively new object, there have been some preliminary results in the two dimensional case motivated by coding theory [6, 71].

Further, motivated by the problem of determining sets of layers corresponding to a given chemical formula, we consider *weighted necklaces*. In the context of CSP a necklace is weighted by the chemical composition, represented by a vector. The weight of a symbol in this case corresponds to a vector representing the chemical composition of the corresponding layer. In this setting, the goal can be thought of as choosing a necklace with the Parikh vector corresponding to a positive integer solution to a set of linear Diophantine equations.

These can be seen as a generalisation on *fixed-content necklaces*. A set necklaces has fixed content if every necklace in the set has the same Parikh vector. As with general necklaces, there have been results for counting [38], and generating [57, 89] both fixed content necklaces and bracelets. Further, Hartman and Sawada provided a polynomial time algorithm to rank and unrank *fixed density* necklaces, fixed content necklaces restricted to a binary alphabet [45].

An additional constraint on necklaces is that of forbidden subwords. Informally, the set of necklaces with forbidden subwords is the set of necklaces such that no necklace contains any word from some given set as a subword. While this class has not been studied as intensely as the general or fixed content case, there are still some results of interest. Ruskey and Sawada [90] provide methods for counting and generating necklaces with a single forbidden subword.

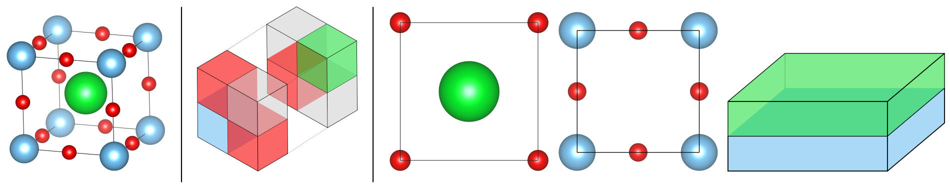
Figure 1.4: The crystal of $SrTiO_3$ (left) and its 3D (middle) and 1D (right) necklace representations.

These classes of cyclic words are utilised in the sampling algorithms we outline in Chapter 5, Section 1.3 provides the outline to this sampling problem looking by looking at the k-centre problem. At a high level, the goal is to use the k-centre problem as a means to generate samples via a weighted graph where each vertex corresponds to some member of the set.

One further class that is of interest to this thesis is that of *de Bruijn sequences*. Informally a de Bruijn sequence is a cyclic word defined over some alphabet Σ of length q such that every word of length n over Σ appears exactly once. The question of the existence of such sequences was solved twice, first by Sainte-Marie [24, 91] and more famously by de Bruijn [25]. Several key results for both generating [31] and ranking [60] necklaces have been motivated by solving these problems for de Bruijn Sequences.

1.5 Overview of Results

This thesis is broadly split into three themes regarding the results. First and foremost are results on the computational difficulty of Crystal Structure prediction. Chapters 3 and 4 provide results on the undecidability in the general case and NP-hardness in several specific cases of CSP respectively. Chapter 5 responds to this difficulty by providing a new technique for *sampling* possible crystal structures inspired by the k-centre problem. Finally, Chapters 6, 7 and 8 provide new combinatorial tools for describing crystal structures. The tools provided in Chapters 6, 7 and 8 may be used in conjunction with the sampling methods given in Chapter 5 to better sample the set of crystal structures.

1.5.1 Hardness of CSP

The first two chapters are dedicated to the showing the difficulty of solving abstract forms of CSP. Chapter 3 provides a proof of undecidablity for the most general formulation of CSP in Theorem 1. This undecidability result is derived by reduction from the *tiling problem*.

The tiling problem is a classic undecidable problem that asks if a given set of Wang tiles can completely tile the plane [11]. More relevant to us is the *periodic* tiling problem. In the periodic tiling problem, the goal becomes to find a periodic tiling, i.e. a fixed rectangle made of tiles that can be repeated infinitely to tile the plane properly. This problem is a natural fit for CSP as it looks at very local interactions being used to build an infinite, periodic, structure.

Theorem 1 show that the undecidability results hold for several chemically motivated interaction functions. Notably, we look at the Buckingham-Coulomb potential [14] and a generalisation of the Ising model [54, 98]. Both of these functions have been heavily utilised by CSP techniques, making any hardness results based on these interaction functions of great interest.

Corollary 3 strengthens Theorem 1 by showing that the problem remains undecidable when restricted to the chemically motivated Buckingham-Coulomb potential. This is of particular note as it is the model used by several "tiling" approaches such as those by Collins et al. [22]. This setting has the added complexity of balancing the *charge* of the unit cell, adding an integer charge to every ion species with the requirement that the sum of these charges is 0.

The first half of Chapter 4 provides hardness results in the setting where the size of unit cell is constrained. This corresponds more directly to many heuristic techniques for CSP, where the size of the unit cell is fixed due to computational constraints. Theorem 2 shows that in the one dimensional setting our main form of CSP is hard to approximate within any factor greater than 0. Theorem 3 compliments Theorem 2 by providing a parameterised algorithm for the one dimensional setting. This matches the one dimensional tiling model of Collins et al. [23].

The second half of Chapter 4 looks at a variation on CSP, where a unit cell is given as input with the goal of removing some subset of ions so as to minimise the average energy of the remaining ions. This is similar to the methods proposed by Antypov et al. [7] and Dyer et al. [27]. While those methods relied on swapping ions, this approach is based on the process of removing ions. Theorems 4 and 6 show that this model is both NP-hard to solve and to approximate within any factor $n^{1-\epsilon}$ where $\epsilon > 0$.

Theorem 8 strengthens Theorem 4 by showing that the problem remains NP-hard to solve even when the number of elements in the unit cell is restricted to 2 for the Buckingham-Coulomb potential. This is particularly important as 2 is the smallest feasible number of elements to construct a crystal with charged ions. As such, it can be seen as a particularly relevant result for understanding the complexity of CSP.

Finally Theorem 9 shows that the problem remains hard when restricted to the electrostatic potential. This is the simplest model of interaction between charged ions, based solely on the charges and distances between ions. This result helps to solidify the hardness of CSP even for highly simplified interactions.

1.5.2 The k-Centre Problem for classes of Cyclic Words

The second problem this thesis considers is the problem of sampling different crystal structures. Chapter 5 provides the underlying algorithms for the k-centre problem. Theorem 10 shows that the problem of verifying a solution to the sampling problem is NP-hard. While this does not necessarily show that the it is NP-hard to solve the problem, it does give a strong indication of complexity.

Theorem 12 provides our primary approximation algorithm for solving the k-centre problem. The exact approximation ratio varies as a function of the number of samples and size of necklaces, however it is in the worst case logarithmic relative to these product of the number of samples and size of the necklaces. Further, this algorithm runs in linear time relative to the size of the output. Theorem 12 relies on the generation of de Bruijn sequences. As such it can not be easily extended to the multidimensional case, nor to settings where the Parikh vectors is constrained.

In order to provide approximation algorithms for these settings, Theorem 13 provides a further more general approximation algorithm. While Theorem 13 provides a slightly worse approximation ratio, being only logarithmic in the number of samples, the ability to utilise this algorithm in more settings makes it of independent interest. In order to utilise Theorem 13, it is necessary to develop a deeper understanding of various classes of cyclic words, particularly in regards to ranking.

1.5.3 Cyclic Words

The final theme of this thesis is results on cyclic words. The main goal of this theme is to provide tools to rank various classes of cyclic words, however we also provide several tangential results that are of independent interest.

Chapter 6 provides a set of combinatorial results regarding bracelets. More precisely we look the problem of *ranking* bracelets, the problem of determining the number of bracelets smaller than some given bracelet. Theorem 14 shows that the rank of a bracelet can be computed in $O(q^2 \cdot n^4)$ time where n is the length of the bracelet and q the size of the alphabet. This not only of interest in the problem of sampling, but also provides new combinatorial insight into the class of bracelets.

Theorem 14 is proven through ranking two related classes of cyclic words, namely *palindromic* and *enclosing* bracelets. A palindromic bracelet is a bracelet corresponding to a necklace class such that the reflection of every word in the necklace class belongs to the same necklace class. An enclosing bracelet is defined relative to a word $\bar{w}$ as an apalindromic bracelet corresponding to a pair of necklaces one of which is smaller than $\bar{w}$ and the other greater than $\bar{w}$. Theorems 16 and 17 provide algorithms to rank palindromic and enclosing bracelets respectively. While neither of these classes are of particular interest from the perspective of CSP, they are of independent interest as novel classes of cyclic words.

Chapter 7 provides the tools to rank necklaces under three broad sets of constraints; Parikh vectors satisfying linear Diophantine equations, necklaces with Parikh vectors satisfying linear Diophantine equations, and necklaces with forbidden subwords. Section 7.1 looks at the problems related to determining the number of positive integer solutions to Diophantine equations with the constraint the sum of the variables sums to some given value n. Lemma 29 shows how to count the number of such solutions. Theorem 19 looks at the problem of ranking these solutions. Theorem 20 solves the problem of unranking these solutions. Lemma 29, and Theorems 19, and 20 run in polynomial time for a fixed number of equations. These algorithms are not only of interest regarding the sampling approach to CSP, but also as a means to solve linear Diophantine equations. One such motivation would be in the context of tiling approaches to CSP, where the number of combinations of tiles corresponding to a given chemical formula can be computed.

Section 7.2 extends the results from Section 7.1 to the set of necklaces with Parikh vectors corresponding to solutions to linear Diophantine equations. These results are of particular interest in the context of sampling starting stuctures for the tiling approach. Theorem 21 shows how to count the number of such necklaces. Theorem 22 provides an algorithm to rank these necklaces. Finally Theorem 23 provides an algorithm for unranking. As was the case in Section 7.1, these algorithms run in polynomial time for a fixed number of linear equations.

Section 7.3 looks at necklaces that do not contain any forbidden subword from some given set. Theorem 24 provides an algorithm to rank necklaces with a set of forbidden subwords. Theorem 25 compliments Theorem 24 with an algorithm to unrank necklaces with a set of forbidden subwords. In both cases, the algorithms run in polynomial time relative to the length of the necklaces, and size of the set of forbidden words.

Finally, Chapter 8 provides a suite of new algorithms and results for multidimensional necklaces. We look at both the general case and fixed content necklaces. In both cases we provide an expression to count the number of necklaces of a given set of dimensions and alphabet that may be evaluated in polynomial time relative to the size of the necklaces.

We follow this with an algorithm for ranking multidimensional necklaces, stated in Theorem 26. The algorithm operates in a recursive manner, taking advantage of each d-dimensional necklace of being composed of a set of $d-1$ dimensional necklaces. This algorithm runs in polynomial time relative to the size of the necklaces. Theorem 26 is further strengthened by Theorem 27 showing how to rank *fixed content* multidimensional necklaces, multidimensional necklaces where the Parikh vector is fixed. This algorithm uses the same techniques as in Theorem 26 for the unconstrained case, running in polynomial time for a fixed size alphabet.

Theorem 28 shows how to generate the set of multidimensional necklaces in linear time per necklace relative to the size of the necklaces. Notably, this corresponds to the lower bound required to output every necklace in order. Theorem 29 compliments both Theorems 26 and 28 by showing how to unrank multidimensional necklaces in polynomial time. Both Theorems 28 and 29 require the technical Lemma 54, showing that given any word $\bar{w}$, the necklace following $\bar{w}$ can be computed in polynomial time.

1.6 Publications and Presentations from this Thesis

This section provides an overview on the papers and external presentations regarding the results from this thesis. We first discus the results that have been presented externally.

The results presented in Section 4.2 regarding the hardness of CSP by ion removal have been published at the 46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM) [2]. Further, they have been presented at the British Colloquium for Theoretical Computer Science (BCTCS) 2019 hosted at the University of Durham [56]. Finally, these results are under review in journal format for Fundamenta Informaticae. The results in Chapter 5 and 8 are currently under review, and presently available as a paper at Arxiv [3]. Additionally, the results in Chapter 8 were presented at the British Colloquium for Theoretical Computer Science (BCTCS) 2020 hosted at the University of Swansea [12]. The results in Chapter 6 have been published at the 31st Annual Symposium on Combinatorial Pattern Matching [4]. These results were additionally presented at the British Colloquium for Theoretical Computer Science (BCTCS) 2021 hosted at the University of Liverpool [99]. Further, these results have been submitted for review in journal format to Algorithmica.

In addition to the above mentioned results, two further manuscripts are under preparation following the work done in this thesis. The first of these is on the hardness of CSP in the general setting, containing the results from Chapter 3 and Section 4.1. This manuscript serves to strengthen the results presented in [2] by showing that a more general model of CSP is not only NP-hard but is in fact undecidable in the most general case. Additionally, the results presented in Chapter 7 are under preparation as a manuscript. This manuscript strengthens the results from [3] by extending our approximation algorithms for the k-centre problem into new settings.

Chapter 2

Preliminaries

This chapter introduces the main definitions and notation that is used in the remainder of this thesis. This is primarily spilt into two Sections. Section 2.1 covers the definitions and notation for Crystal Structure Prediction. Section 2.2 covers definitions regarding cyclic words which are used as the foundation for the representation of discrete crystals.

2.1 Energy Minimisation for Combinatorial Crystal Structure Prediction

This work considers two models of energy minimisation based on the idea of either generating an optimal structure from scratch, or taking an existing structure and removing ions to improve it. While both models have some slight differences, there are many similarities between both in terms of the models and what is determined to be a “good” or “bad” solution. The remainder of this section is laid out as follows. Section 2.1.1 covers the general model for both cases, providing the underlying definitions that are used for the remainder of this work. Section 2.1.2 looks at the main functions used to determine the value of a given structure. Section 2.1.3 provides the basis for this work regarding the more general problem of determining the optimal structure from scratch. Finally Section 2.1.4 provides the basis for the deletion operation based model of Crystal Structure Prediction.

2.1.1 The General Model

This section provides the definitions that are used as a basis for both the colouring based model presented in Section 2.1.3 and the deletion based model presented in Section 2.1.4. In both cases, the goal of our abstractions of CSP is to determine a way of periodically colouring an infinite discrete space using a set of colours based on the input set of ions and the empty space. In this way, the problem can be thought of as determining the period of a colouring of the vertices on the integer grid $\mathbb{Z}^d$ for some number of dimensions d. For notation each vertex $v \in \mathbb{Z}^d$ may be referred to by the vector corresponding to the position of v on the integer grid. The period of a colouring is called the *unit cell*, which may equivalently be thought of as a mapping from the set of colours to the grid.

Definition 1. *A* unit cell *U of dimensions $\overline{\mathbf{n}} = (n_1, n_2, \dots, n_d)$ is a mapping from some set of colours $\mathcal{C}$ to the integer grid $\mathbb{Z}^d$. A unit cell can be thought of as a fully coloured finite grid $\{(x_1, x_2, \dots, x_d) : x_i \in [0, n_i - 1]\}$ such that every vertex in $\{(x_1, x_2, \dots, x_d) : x_i \in [0, n_i - 1]\}$ is assigned a colour from $\mathcal{C}$. Given a unit cell U, and vector $\overline{\mathbf{y}} \in \{(x_1, x_2, \dots, x_d) : x_i \in [0, n_i - 1]\}$, $U(\overline{\mathbf{y}})$ returns the colour at position $\overline{\mathbf{y}}$ of the grid. A unit cell U of dimensions $\overline{\mathbf{n}} \in \mathbb{Z}^d$ can be mapped to $\mathbb{Z}^d$ by colouring vertex $\overline{\mathbf{x}} \in \mathbb{Z}^d$ with $U((x_1 \bmod n_1, x_2 \bmod n_2, \dots, x_d \bmod n_d))$.*

For notation the number of vertices in a unit cell U of dimensions $\overline{\mathbf{n}}$ is denoted by $|U|$, i.e. $|U| = n_1 \cdot n_2 \cdot \ldots \cdot n_d$. Similarly $\overline{\mathbf{x}} \in U$ is used to denote that $\overline{\mathbf{x}}$ is a position in the finite grid defining U. Where it is clear from context given any vector $\overline{\mathbf{x}} \in \mathbb{Z}^d$ the colour of the vertex at position $\overline{\mathbf{x}}$ in the grid $\mathbb{Z}^d$ coloured by U is denoted $U(\overline{\mathbf{x}})$, giving $U(\overline{\mathbf{x}}) = U((x_1 \bmod n_1, x_2 \bmod n_2, \dots, x_d \bmod n_d))$.

One additional constraint that is sometimes applied in the context of crystal structure prediction is that of *charge neutrality*. In this setting, every colour is associated with an integer charge. In general, the charge for any colour not representing empty space is a non-zero. Given a unit cell U, the charge of $\overline{\mathbf{x}} \in U$ is denoted $Q(U(\overline{\mathbf{x}}))$. Note that the charge of any two points assigned the same colour are equal, i.e. if $U(\overline{\mathbf{x}}) = U(\overline{\mathbf{y}})$ then $Q(U(\overline{\mathbf{x}})) = Q(U(\overline{\mathbf{y}}))$ for any pair of vectors $\overline{\mathbf{x}}, \overline{\mathbf{y}} \in U$. Informally, a unit cell is charge neutral is the sum of the charge of every position in the unit cell is 0.

Definition 2. *A unit cell U is* charge neutral *if and only if $\sum_{\overline{\mathbf{x}} \in U} Q(U(\overline{\mathbf{x}})) = 0$.*

The goal of these colourings is to minimise the *average pairwise energy per vertex* of the coloured grid. Conceptually, the energy between two vertices can be thought of as the force between them, with a negative energy attracting the two vertices and a positive energy repelling them. As such, the goal can be thought of as maximising the negative energy per vertex, in an attempt to build a stronger structure. The pairwise energy between a pair of vertices in the grid is determined by a *pairwise energy function.*

This work considers parametric pairwise energy functions of the form $f(\overline{\theta}(c_1, c_2), t)$ where $c_1, c_2 \in \mathcal{C}$ are a pair of colours, $r \in \mathbb{R}$ is a euclidean distance and $\overline{\theta}(c_1, c_2) \in \mathbb{R}^p$ is a set of p parameters determined by the colours c_1 and c_2. Further, this work assumes that the set of parameters $\overline{\theta}(c_1, c_2)$ are predefined for every $c_1, c_2 \in \mathcal{C}$. Each such function returns a scalar real value, i.e. $f : \left(r \in \mathbb{R}, \overline{\theta}(c_1, c_2) \in \mathbb{R}^p\right) \mapsto \mathbb{R}$. As the parameters are predefined, the function $f(\overline{\theta}(c_1, c_2), r)$ may be rewritten as $f(c_1, c_2, r)$ when it is clear from context. Importantly, this work assumes that the energy for any two pairs of points at the same distance and sharing the same colours is the same. Formally, given two pairs of vectors $\overline{\mathbf{x_1}}, \overline{\mathbf{x_2}} \in \mathbb{Z}^d$ and $\overline{\mathbf{y_1}}, \overline{\mathbf{y_2}} \in \mathbb{Z}^d$, if $U(\overline{\mathbf{x_1}}) = U(\overline{\mathbf{y_1}}), U(\overline{\mathbf{x_2}}) = U(\overline{\mathbf{y_2}})$ and $D(\overline{\mathbf{x_1}}, \overline{\mathbf{x_2}}) = D(\overline{\mathbf{y_1}}, \overline{\mathbf{y_2}})$ then $f(\overline{\theta}(U(\overline{\mathbf{x_1}}), U(\overline{\mathbf{x_2}})), D(\overline{\mathbf{x_1}}, \overline{\mathbf{x_2}})) = f(\overline{\theta}(U(\overline{\mathbf{y_1}}), U(\overline{\mathbf{y_2}})), D(\overline{\mathbf{y_1}}, \overline{\mathbf{y_2}}))$. For a given unit cell U and pairwise energy function f, the average pairwise energy per vertex is defined as follows:

Definition 3. *Given a unit cell U of dimensions $\overline{\mathbf{n}}$ colouring the grid $\mathbb{Z}^d$, the* average pairwise energy per vertex *is given by* $\sum\limits_{\overline{\mathbf{x}} \in U} \sum\limits_{\overline{\mathbf{y}} \in \mathbb{Z}^d} \frac{f(\overline{\theta}(U(\overline{\mathbf{x}}), U(\overline{\mathbf{y}})), D(\overline{\mathbf{x}}, \overline{\mathbf{y}}))}{|U|}$ *where $f(\overline{\theta}(c_1, c_2), d)$ is the pairwise energy function, $D(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ denotes the euclidean distance between $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$, and $\overline{\theta}(c_1, c_2) \in \mathbb{R}^p$ is a set of p parameters.*

Alongside these definitions, some additional notation is used to simplify further discussion relating to the 2D integer grid. Given two vertices v_i at position at (x_1, y_1) and v_j at position (x_2, y_2), v_i is said to be *directly adjacent* to v_j if either $x_1 = x_2 \pm 1$ and $y_1 = y_2$ or $x_1 = x_2$ and $y_1 = y_2 \pm 1$. Similarly v_i is *diagonally adjacent* to v_j if $x_1 = x_2 \pm 1$ and $y_1 = y_2 \pm 1$, and v_i is *peripherally adjacent* if either $x_1 = x_2 \pm 2$ and $y_1 = y_2$ or $x_1 = x_2$ and $y_1 = y_2 \pm 2$. An overview is provided in Figure 2.1.

2.1.2 Pairwise Energy Functions

Following the discussion the previous section, this section covers the pairwise energy functions that are used in this thesis. This work focuses primarily on two classes of energy functions, the *n-distance common minimal value class* and the *controllable class.* Additionally two energy functions based on popular models used in chemistry are studied, the

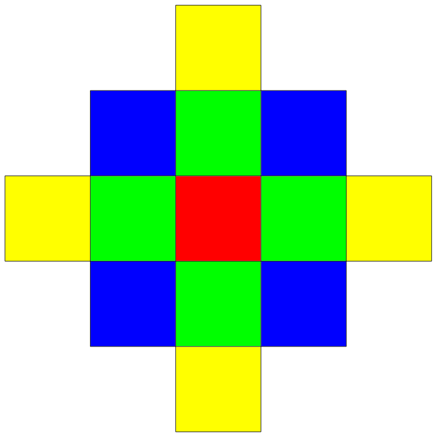

Figure 2.1: An overview of the adjacency's relative to the central (red) tile. The tiles in green are directly adjacent, the tiles in blue are diagonally adjacent and the tiles in yellow are peripherally adjacent.

Buckingham-Coulomb potential and the Ising model.

The n-Distance Common Minimal Value Class The first class considered in this section is the **n-distance common minimal value class** denoted $\mathcal{CMV}(n)$. This class of functions focuses on the interactions between vertices within a distance of r of each other, for some distance $r \in \mathbb{R}$. In order to simplify reasoning on the set $\mathcal{CMV}(r)$, it is assumed that given any distance $d > r$ the value of $f(\overline{\theta}, d) = 0$ for any set of values $\overline{\theta} \in \mathbb{R}^p$. In this way, the set $\mathcal{CMV}(r)$ can be thought of as determining the interaction between every vertex within a circle of radius r of some central vertex. For notation, let $D(r) = \{(i,j) \in \mathbb{Z}^2 | \sqrt{i^2+j^2} \leq r, (i,j) \neq (0,0)\}$ be the set of vertices on the integer grid $\mathbb{Z}^2$ within a distance of at most r of the central point $(0,0)$. Further let $d(r) = \{\sqrt{i^2+j^2} | (i,j) \in D(r)\}$ be the set of possible distances between the central vertex and any vertex in $D(r)$. As an example, $D(2) = \{(2,0),(1,1),(1,0),(1,-1),(0,2),(0,1),(0,-1),(0,-2),(-1,1),(-1,0),$ $(-1,-1),(-2,0)\}$ and $d(r) = \{1, \sqrt{2}, 2\}$. The goal of this class is to be able to "fix" the optimal distance between any two colours as either being some distance in $d(r)$, or as being outside of the $D(r)$ - in effect penalising two colours at a distance of r or less. To this end this work uses the idea of a *common minimal value*. Informally, the common minimal value can be thought of as some negative value M such that the smallest possible interaction between any pair of vertices is M. Further, the functions in this work restrict M to appear at most once in the set of possible distance between each colour, meaning that given some pair of colours c_1 and c_2, there exists at most one distance $d \in d(r)$ such that $f(\theta(c_1, c_2), d) = M$. The following definition formalises the *r-distance common minimal value class.*

Definition 4. *A parametric pairwise energy functions $f(\overline{\theta}, d) : (\overline{\theta} \in \mathbb{R}^p, d \in \mathbb{R}) \mapsto \mathbb{R}$ belongs to $\mathcal{CMV}(r)$ if there exists $M \in \mathbb{Q}$ such that the following conditions are met:*

1. *For any set of parameters $\overline{\theta} \in \mathbb{R}^p$ and distance $d > r$, the value of $f(\overline{\theta}, d) = 0$.*
2. *There exists some set of parameters $\overline{\theta} \in \mathbb{R}^p$ such that for every distance $d \in d(r)$ the energy $f(\overline{\theta}, d) > M$.*
3. *For every distance $d \in d(r)$, there exists some set of parameters $\overline{\theta}_d \in \mathbb{R}^p$ such that $f(\overline{\theta}_d, d) = M$, and for every other distance distance $d' \in d(r)$ where $d' \neq d$ the value of the energy function $f(\overline{\theta}_d, d') > M$.*

These conditions are used to help encode tiling problems into the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$. Condition 1 ensures that there is no interaction between vertices over a certain cut off distance, allowing these interactions to be safely ignored. Condition 2 ensures that there exists a set of parameters such that the corresponding colours must be placed further than r apart, or suffer a small energy penalty by having an interaction greater than M. Finally Condition 3 ensures that there exists a set of parameters such that the interaction of the corresponding colours is minimised at M at exactly one distance. The goal of these conditions is to be able to force a structure on the colouring based on the relative distances between colours. This allows the structure of the tiling problem to be utilised in the setting of the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$. In addition to these conditions, we assume that every function in $\mathcal{CMV}(r)$ can be computed in polynomial time. To show that these conditions have some grounding in the physical sciences, Sections 2.1.2.1 and 2.1.2.2 show that two models of interaction based interaction functions used in chemistry satisfy the $\mathcal{CMV}(2)$ condition.

The Controllable Class The second class introduced in this section is the **controllable** class, denoted $\mathcal{CF}$. This work assumes that any function in $\mathcal{CF}$ can be computed in polynomial time for any input. Intuitively, for every $f \in \mathcal{CF}$ there exists a set of parameters that counteract the effect of the distance parameter r.

Definition 5. *A function $f : \mathbb{R}^p \mapsto \mathbb{R}$ belongs to $\mathcal{CF}$ if and only if for any given $a \in \mathbb{R}$ and any fixed $r \in \mathbb{R}^+$ there exists a set of parameters $\overline{\theta}(c_1, c_2) \in \mathbb{R}^{p-1}$ for every pair of colours $c_1, c_2 \in \mathcal{C}$ such that $f(\overline{\theta}(c_1, c_2), r\} = a$.*

2.1.2.1 The Buckingham-Coulomb Interaction Function

As this work is motivated by the problem of determining the complexity of crystal structure prediction (CSP), it is important to show that chemical energy functions belong to both $\mathcal{CMV}(n)$ and $\mathcal{CF}$. Before looking at the these energy functions, it is useful to look at the CSP problem in terms of the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$. Using the same terminology as before, CSP takes a set of "colours", corresponding to a set of ion species plus a colour to represent empty space. Each of these colours is given an integer charge. The charge of the unit cell is the sum of the charge of each vertex. The goal is to determine the charge neutral unit cell with minimum energy per vertex.

One of the most popular pairwise energy functions is the *Buckingham-Coulomb potential*, denoted $BC(i, j, r_{i,j})$ for a pair of ions i and j at a distance of $r_{i,j}$. Let $S(i)$ return the species of ion i. The Buckingham-Coulomb potential is the sum of two functions; the close range **Buckingham** potential, and the long range **Coulomb** potential both determined by the species of the ions. The Buckingham potential, $U^B(i, j)$, for a pair of ions i, j is defined by four parameters: the distance between the ions, $r_{i,j}$, and the three parameters, known as the force field parameters, $A_{S(i),S(j)}$, $B_{S(i),S(j)}$, $C_{S(i),S(j)} \in \mathbb{R}$ defined by the species of the ions. Note that all three parameters have positive values. The energy is calculated as:

$$B(S(i), S(j)) = \frac{A_{S(i),S(j)}}{e^{B_{S(i),S(j)} \cdot r_{i,j}}} - \frac{C_{S(i),S(j)}}{r_{i,j}^6}$$

The Coulomb potential for a pair of ions i, j is defined as:

$$C(i, j, r_{i,j}) = \frac{q_{S(i)} q_{S(j)}}{r_{ij}}$$

Where r_{ij} is the Euclidean distance between the ions and $q_{S(i)}$ is the charge of ion species $S(i)$. The **Buckingham-Coulomb potential** is:

$$U^{BC}(i, j) = U^B(i, j) + U^C(i, j) = \frac{A_{S(i),S(j)}}{e^{B_{S(i),S(j)} \cdot r_{ij}}} - \frac{C_{S(i),S(j)}}{r_{ij}^6} + \frac{q_{S(i)} q_{S(j)}}{r_{ij}}.$$

To avoid computing the sum of the infinite series, a *cut-off* distance $d_{i,j}$ is introduced. The energy between any two ions at a distance greater than $d_{i,j}$ is set to 0 for simplicity. Note that as with the other parameters, $d_{i,j}$ is determined by the species of ions i and j. With a cut-off distance $d_{i,j}$, the Buckingham-Coulomb potential is defined as:

$$BC(i,j,r_{i,j}) = \begin{cases} 0 & r > d_{i,j} \\ \frac{A_{i,j}}{e^{r_{i,j}B_{i,j}}} - \frac{C_{i,j}}{r^6} + \frac{q_i \cdot q_j}{r_{i,j}} & r \leq d_{i,j} \end{cases}$$

The next problem is to show that $BC(i,j,r_{i,j}$ belongs to $\mathcal{CMV}(n)$. In order to fit $BC(i,j,r_{i,j})$ in to $\mathcal{CMV}(n)$, the equation is slightly modified by multiplying the distance by a factor of 10, giving:

$$BC(i,j,r_{i,j}) = \begin{cases} 0 & r > d_{i,j} \\ \frac{A_{i,j}}{e^{10 \cdot r_{i,j}B_{i,j}}} - \frac{C_{i,j}}{(10 \cdot r)^6} + \frac{q_i \cdot q_j}{10 \cdot r_{i,j}} & r \leq d \end{cases}$$

Note that this equation corresponds to $BC(i,j,r_{i,j})$ used on the grid $\mathcal{G} = \{(10 \cdot x_1, 10 \cdot x_2, \ldots, 10 \cdot x_d) | x_1, x_2, \ldots, x_d \in \mathbb{Z}\}$.

Proposition 1. *The Buckingham-Coulomb potential belongs to the* 2*-distance common minimal value class* $\mathcal{CMV}(2)$*.*

Proof. In order to show that The Buckingham-Coulomb potential belongs to the class $\mathcal{CMV}(2)$, it is necessary to show that there exists a set of parameters for each distance $d \in [10, \sqrt{200}, 20]$ such that (1) $BC(i,j,d) = M$, (2) $BC(i,j,d') > M$ for every $d' \in [10, \sqrt{200}, 20]$ where $d' \neq d$ and (3) there exists a set of parameters such that $BC(i,j,d') > M$ for every $d' \in [10, \sqrt{200}, 20]$. Here $M = -1$ and the cutoff distance is set to 2. This leaves 4 cases to consider.

Case 1, $d = 1$: In this case the energy between i and j at a distance of 1 is set to -1. Let $q_i \cdot q_j = -1$. The equation $\frac{A_{ij}}{e^{B_{ij}10}} - \frac{C}{10^6} - \frac{1}{10}$ must equal -1, which may be achieved by setting $A = e^{B_{ij}} \left(\frac{C}{10^6} - \frac{9}{10}\right)$. For any distance greater than 1, the energy must be greater than -1. Let P be the penalty energy for having j at a distance of $\sqrt{200}$. For simplicity let $B_{ij} = 0$. In this case at a distance of $\sqrt{2}$ the energy equals $\frac{C}{10^6} - \frac{9}{10} - \frac{C}{200^3} - \frac{1}{\sqrt{200}} = P$ which can be rearranged to give $C = \frac{P + \frac{9}{10} + \frac{1}{\sqrt{200}}}{\frac{1}{10^6} - \frac{1}{200^3}}$. Note that under this setting, the energy at a distance of 2 is strictly greater than P as both negative terms are decreased while the positive term remains constant.

Case 2, $d = \sqrt{2}$: In this case the energy between i and j at a distance of $\sqrt{2}$ is -1, and greater than -1 at any other distance. Let $q_i \cdot q_j = 1$. At a distance of $\sqrt{2}$, the equation $\frac{A_{ij}}{e^{B_{ij} \cdot \sqrt{200}}} - \frac{C_{ij}}{\sqrt{200}^6} + \frac{1}{\sqrt{200}} = -1$ must be satisfied. This can be done by setting $C_{ij} = \sqrt{200}^6 \left(\frac{A_{ij}}{e^{B_{ij} \cdot \sqrt{200}}} - 1 \right)$. This gives the equation at a distance of 1 as $\frac{A_{ij}}{e^{B_{ij} \cdot 10}} - \frac{\sqrt{200}^6}{10^6} \left(\frac{A_{ij}}{e^{B_{ij} \cdot \sqrt{200}}} - 1 \right) + \frac{1}{10} > -1$, which can be rearranged to give $A_{ij} > \frac{69}{10(\frac{1}{e^{B_{ij} \cdot 10}} - \frac{8}{e^{B_{ij} \cdot \sqrt{200}}}}$. Simultaneously, at a distance of 2 the equation $\frac{A_{ij}}{e^{B_{ij} \cdot 20}} - \frac{\sqrt{200}^6}{20^6} \left(\frac{A_{ij}}{e^{B_{ij} \cdot \sqrt{200}}} - 1 \right) + \frac{1}{20} > -1$, requiring $A_{ij} > \frac{\frac{-\sqrt{200}^6}{20^6} - \frac{21}{20}}{\left(\frac{1}{e^{B_{ij} \cdot 20}} - \frac{\sqrt{200}^6}{20^6 \cdot e^{B_{ij} \cdot \sqrt{20}}} \right)}$. Let $B_{ij} = 0$. Then at a distance of 1 the value of $A_{ij} > \frac{69}{10(\frac{1}{10} - 8\sqrt{200})}$, which can be satisfied for any positive A_{ij}. Similarly at a distance of 2 the value of $A_{ij} > \frac{64 - \frac{21}{20}}{7}$, which may be satisfied for positive values of A_{ij}.

Case 3, $d = 2$: In this case the energy between i and j at a distance of 2 must be -1, and the energy must be greater than this at every distance less than 2. Let $q_i \cdot q_j = 1$. To set the energy equal to -1 at a distance of 2 let $\frac{A_{ij}}{e^{B_{ij} 20}} - \frac{C_{ij}}{20^6} + \frac{1}{20} = -1$. This can be rearranged to give $C_{ij} = \frac{A_{ij} 20^6}{e^{B_{ij} 20}} + 20^6 + 20^5$. Let $B_{ij} = \frac{6 \ln(10)}{10}$, giving $C_{ij} = \frac{A_{ij} \cdot 20^6}{10^{12}} + 20^6 + 20^5$. For any distance smaller than 2, the interaction must be greater than -1. This gives the equations $\frac{A_{ij}}{10^3 \sqrt{2}} - \frac{\frac{A_{ij} 20^6}{10^{12}} + 20^6 + 20^5}{200^3} + \frac{1}{\sqrt{200}} > -1$ and $\frac{A_{ij}}{10^3} - \frac{\frac{A_{ij} 20^6}{10^{12}} + 20^6 + 20^5}{10^6} + \frac{1}{10} > -1$. These can be rearranged to give $A_{ij} \left(\frac{1}{10^3 \sqrt{2}} - \frac{1}{10^6 \cdot 50^3} \right) - \frac{20^5 \cdot 21}{200^3} + \frac{1}{\sqrt{200}} > -1$ and $A_{ij} \left(\frac{1}{10^3} - \frac{1}{10^6 \cdot 5^6} \right) - \frac{20^5 \cdot 21}{10^5} + \frac{1}{10} > -1$. These can be further rearranged to give $A_{ij} > \frac{\frac{20^5 \cdot 21}{200^3} - \frac{1}{\sqrt{200}}}{\left(\frac{1}{10^2 \sqrt{3}} - \frac{1}{10^{12} \cdot 2^6 \cdot \sqrt{50}} \right)}$ and $A_{ij} > \frac{\frac{20^5 \cdot 21}{10^5} - \frac{1}{10}}{\left(\frac{1}{10^3} - \frac{1}{10^6 \cdot 5^6} \right)}$. As neither case provides an upper bound on value of A_{ij}, there exists some positive value of A_{ij} satisfying these equations.

Case 4, greater than M at every distance: The final case is where the energy between i and j is greater than M at every distance. This can be done by setting $A_{ij} = 1$, $B_{i,j} = 0$ and $C_{i,j} = 0$. With these parameters, the energy equation becomes $1 + \frac{q_i \cdot q_j}{d'}$, which is greater than -1 for any distance $d' \in [10, \sqrt{200}, 20]$ and pair of unit charges $q_i = \pm 1, q_j = \pm 1$. $\square$

Proposition 2. *The Buckingham-Coulomb potential belongs to the controllable class $\mathcal{CF}$.*

Proof. To show that U^{BC} belongs to $\mathcal{F}$, it is sufficient to provide a constructive method to determine the force field parameters such that for any value $a \in \mathbb{R}$ and a pair of ions i, j the energy $U^{BC}(i, j) = a$. Let i and j be at a distance of r_{ij} with arbitrary charges q_i and q_j, the parameters may be set so that the potential at a distance of r_{ij} is a. The value of $B_{i,j}$ is set to 0 and the values of $A_{i,j}$ and $C_{i,j}$ as follows depending on whether a

is positive or not. If $a > 0$, then:

$$A_{i,j} = \begin{cases} a & \text{if } q_iq_j > 0; \\ a + \frac{|q_iq_j|}{r_{ij}} & \text{otherwise.} \end{cases} \quad \text{and} \quad C_{i,j} = \begin{cases} q_iq_jr_{ij}^5 & \text{if } q_iq_j > 0; \\ 0 & \text{otherwise.} \end{cases}$$

If $a \leq 0$, then:

$$A_{i,j} = \begin{cases} 0 & \text{if } q_iq_j > 0; \\ \frac{|q_iq_j|}{r_{ij}} & \text{otherwise.} \end{cases} \quad \text{and} \quad C_{i,j} = \begin{cases} |a|r_{ij}^6 + q_iq_jr_{ij}^5 & \text{if } q_iq_j > 0; \\ |a|r_{ij}^6 & \text{otherwise.} \end{cases}$$

In this case of the Buckingham-Coulomb potential, the equation becomes

$$U^{BC}(i,j) = A_{i,j} - \frac{C_{i,j}}{r_{i,j}^6} + \frac{q_iq_j}{r_{ij}}.$$

The Coulomb potential, $\frac{q_iq_j}{r_{ij}}$, is cancelled either by adding $q_iq_jr_{ij}^5$ to $C_{i,j}$, if $q_iq_j > 0$, or $\frac{|q_iq_j|}{r_{ij}}$ to $A_{i,j}$ in the case $q_iq_j \leq 0$. In the first case the energy added by the Coulomb potential is $\frac{|q_iq_j|}{r_{ij}}$, which is cancelled by the addition of $q_iq_jr_{ij}^5$ when multiplied by the $\frac{-1}{r_{ij}^6}$ term applied to $C_{i,j}$. Otherwise the Coulomb energy is $\frac{-|q_iq_j|}{r_{ij}}$, which is cancelled out by the relevant addition from $A_{i,j}$. With the Coulomb energy removed, a is either left as part of the A_{ij} term, if $a > 0$, or part of the C_{ij} term otherwise. □

2.1.2.2 The Generalised Ising Model

One further chemically motivated energy function studied in this thesis is the *generalised Ising* model. The basic Ising model corresponds to the problem of colouring an edge-weighted grid $G = (V, E)$ from a set of *spins*. At its most basic, the spins corresponds to values of ± 1. The goal of the Ising model problem is to minimise the value of the energy function $\sum_{(i,j)\in E} W_{ij} \cdot C_i \cdot C_j$ where $W_{i,j}$ is the weight of edge $E_{i,j}$ and C_i is the colour of vertex i. In the classic Ising model the interaction is restricted to the just the immediate neighbourhood, however there have been many generalisations to allow for interactions at longer distances.

The Ising model has been heavily studied from a computational perspective, with results on the hardness of the model in three dimensions [9], as well as the polynomial time tractability in the planar case [20]. Despite the challenges, there are approximation techniques through the formulation of the Ising model as a MAX-CUT problem [10]. This allows approximation algorithms such as the well known semi-definite linear programming approach due to Goemans and Williamson [40] to be applied for the Ising model.

This work focuses on a generalisation of the *n-vector model*, itself a generalisation of the Ising model introduced by Stanley [98] to better represent the set of possible *spins states* of quantum particles. A spin sate may be thought of in an abstract sense as colours determining the interactions. Each spin state is represented by a unit length vector, with $\overline{C_i}$ denoting the vector corresponding to the colour of vertex i. In the n-vector model, each state is represented by a vector of length n, with the energy function given as $\sum\limits_{(i,j)\in E} W(E_{i,j})\cdot\overline{C_i}\cdot\overline{C_j}$ where $\overline{C_i}\cdot\overline{C_j}$ denotes the standard Euclidean inner product of the vectors corresponding to C_i and C_j.

This work looks at the n-vector Ising model with radius r generalisation of the n-vector model to account for changes to the interaction between as distances increases. At a high level, the idea is to create a set of vectors corresponding to each combination of colour and $d \in d(r)$ where r is the maximum interaction distance. In this way, each colour can be thought of as corresponding to a set of $|d(r)|$ vectors, each specifying the interaction at a difference distance. The interactions between two vertices c_1 and c_2 at a distance of $d \in d(r)$ is determined by the inner product of the d^{th} vector in each set.

Formally, the n-vector model with radius contains a set $\mathcal{C}$ of colours each corresponding to a vector of $|d(r)|$ vectors. Let $c[d]$ denote the vector corresponding to distance d. The interaction between two vertices coloured c_1 and c_2 at a distance of d is the inner product of $c_1[d]$ and $c_2[d]$, denoted $c_1[d]\cdot c_2[d]$. Using this interaction, the energy of a grid coloured using some set of vertex sets is given by:

$$\sum_{d\in d(r)}\sum_{v_i,v_j\in V}\begin{cases} C_i[d]\cdot C_j[d] & d = d(v_i,v_j)\\ 0 & d \neq d(v_i,v_j)\end{cases}$$

Proposition 3. *The 2-radius n-vector Ising model belongs to the* 2*-distance common minimal value class* $\mathcal{CMV}(2)$.

Proof. In order to show that the 2-radius n-vector Ising model belongs to the class $\mathcal{CMV}(2)$, it is necessary to show that there exists a set of parameters for each distance $d \in [1,\sqrt{2},2]$ such that (1) $C_i[d]\cdot C_j[d] = m$ (2) $C_i[d']\cdot C_j[d'] > m$ for every $d' \in [1,\sqrt{2},2]$ where $d' \neq d$ and (3) there exists a set of parameters such that for ever distance $C_i[d']\cdot C_j[d'] > m$ for every $d' \in [1, \sqrt{2}, 2]$. Here $m = \frac{-2}{\sqrt{2q-2}}$ where q is the number of colours. Further, each vector has length q^2. The j^{th} entry of $C_i[d]$, denoted $C_i[d,j]$ is set to $\pm\frac{1}{\sqrt{2q-2}}$ if $j = t \cdot q + i$ or $j = i \cdot q + t$ for some $t \in [q]$. Otherwise, the value of $C_i[d,j]$ is set to 0. Note that given two vectors $C_i[d] \cdot C_j[d] = C_i[d, q \cdot i + j] \cdot C_i[d, q \cdot i + j] + C_i[d, q \cdot j + i] \cdot C_j[d, q \cdot j + i]$. Then to have $C_i[d] \cdot C_j[d] = m$, let $C_i[d, q \cdot i + j]$ and $C_j[d, q \cdot j + i]$ be set to $\frac{-1}{\sqrt{2q-2}}$ and both $C_j[d, q \cdot i + j]$ and $C_i[d, q \cdot j + i]$ be set to $\frac{1}{\sqrt{2q-2}}$. On the other hand, to set $C_i[d] \cdot C_j[d] = -m$ let $C_i[d, q \cdot i + j]$, $C_i[d, q \cdot j + i]$, $C_j[d, q \cdot i + j]$ and $C_j[d, q \cdot j + i]$ be set to $\frac{1}{\sqrt{2q-2}}$. By constructing 3 sets of vectors, the resulting model belongs to $\mathcal{CMV}(2)$. □

2.1.3 Pairwise Energy Minimisation for Periodic Grid Colouring

This section introduces the first set of abstract versions of CSP. More precisely, this section considers the generalisation of the discrete crystal structure problem to the problem of minimising average pairwise energy in the periodic grid colouring problem. There are two versions of this problem that are studied here. Informally, these can be thought of as the general case, where the dimensions of the unit cell is undefined, and the fixed size case, where the dimensions of the unit cell are given. Formally, these problems are stated as:

Problem 1. *The pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$*

Input: A goal energy $g \in \mathbb{Q}$, a set of colours $\mathcal{C}$, a number of dimensions $d \in \mathbb{Z}$ an energy function $f(\overline{\theta}(c_1, c_2), r) : \big(c_1, c_1 \in \mathcal{C}, r \in \mathbb{R}, \overline{\theta}(c_1, c_2) \in \mathbb{R}^p\big) \mapsto \mathbb{R}$ and a set of $|\mathcal{C}|^2$ parameters $\overline{\theta}(c_1, c_2) \in \mathbb{R}^p$.

Question: Does there exist a unit cell U where $\sum\limits_{\vec{x} \in U} \sum\limits_{\vec{y} \in \mathbb{Z}^d} \frac{f(D(\vec{x},\vec{y}), \theta(U(\vec{x}), U(\vec{y})))}{|V(U)|} \leq g$.

Problem 2. *The fixed period pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$*

Input: A goal energy $g \in \mathbb{Q}$, a dimension vector $(n_1, n_2, \ldots, n_d)$ a set of colours $\mathcal{C}$, an energy function, $f(\overline{\theta}(c_1, c_2), r) : \big(c_1, c_1 \in \mathcal{C}, r \in \mathbb{R}, \overline{\theta} \in \mathbb{R}^p\big) \mapsto \mathbb{R}$ and a set of $|\mathcal{C}|^2$ parameters $\overline{\theta}(c_1, c_2) \in \mathbb{R}^p$.

Question: Does there exist a unit cell U of dimensions $n_1 \times n_2 \times \ldots \times n_d$ where $\sum\limits_{\vec{x} \in U} \sum\limits_{\vec{y} \in \mathbb{Z}^d} \frac{f(\theta(U(\vec{x}), U(\vec{y})), D(\vec{x},\vec{y}))}{|V(U)|} \leq g$.

These abstractions of CSP serve two purposes. First, they generalise the problem from the highly specific chemical motivation to a more general computational problem. Secondly, the restriction to the discrete setting simplifies the reasoning, particularly by restricting the number of possible distances. In addition to the pairwise energy minimisation problem over

$\mathcal{C}$ on $\mathbb{Z}^d$ and fixed period pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$ problems, this work considers one further generalisation, the *charge neutral pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$* (and the *charge neutral fixed period pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$*), defined as:

Problem 3. *The Charge Neutral Pairwise Energy Minimisation Problem on $\mathbb{Z}^d$*

Input: A goal energy $g \in \mathbb{Q}$, a set of colours $\mathcal{C}$, a number of dimensions $d \in \mathbb{Z}$ an energy function $f(c_1, c_2, r, \overline{\theta}(c_1, c_2)) : \big(c_1, c_1 \in \mathcal{C}, r \in \mathbb{R}, \overline{\theta}(c_1, c_2) \in \mathbb{R}^p\big) \mapsto \mathbb{R}$ and a set of $|\mathcal{C}|^2$ parameters $\overline{\theta}(c_1, c_2) \in \mathbb{R}^p$.

Question: Does there exist a charge neutral unit cell U where $\sum\limits_{\overline{\mathbf{x}} \in U} \sum\limits_{\overline{\mathbf{y}} \in \mathbb{Z}^d} \frac{f(U(\overline{\mathbf{x}}), U(\overline{\mathbf{y}}), D(\overline{\mathbf{x}}, \overline{\mathbf{y}}), \theta(U(\overline{\mathbf{x}}), U(\overline{\mathbf{y}})))}{|V(U)|} \leq g$.

Problem 4. *The Charge Neutral Fixed Period Pairwise Energy Minimisation Problem on $\mathbb{Z}^d$*

Input: A goal energy $g \in \mathbb{Q}$, a vector of dimensions $(n_1, n_2, \ldots, n_d)$, a set of colours $\mathcal{C}$, an energy function, $f(c_1, c_2, r, \overline{\theta}(c_1, c_2)) : \big(c_1, c_1 \in \mathcal{C}, r \in \mathbb{R}, \overline{\theta} \in \mathbb{R}^p\big) \mapsto \mathbb{R}$ and a set of $|\mathcal{C}|^2$ parameters $\overline{\theta}(c_1, c_2) \in \mathbb{R}^p$.

Question: Does there exist a charge neutral unit cell U of dimensions $n_1 \times n_2 \times \ldots \times n_d$ where $\sum\limits_{\overline{\mathbf{x}} \in U} \sum\limits_{\overline{\mathbf{y}} \in \mathbb{Z}^d} \frac{f(U(\overline{\mathbf{x}}), U(\overline{\mathbf{y}}), D(\overline{\mathbf{x}}, \overline{\mathbf{y}}), \theta(U(\overline{\mathbf{x}}), U(\overline{\mathbf{y}})))}{|V(U)|} \leq g$.

2.1.4 Pairwise Energy Minimisation under the Deletion Operation

In all problems considered in this section, the input is assumed to be a unit cell U as described in Section 2.1.1. The objective is to select a set of vertices in U such that by replacing them with the colour representing empty space, the average energy decreases. As the non-empty positions are fixed in the initial input, it is easier to treat the input instead as a weighted geometric graph.

Crystals as weighted geometric graphs. The unit cell of a crystal may be thought of as a geometric graph $G = (V, E)$ in $\mathbb{R}^3$. Given a unit cell U, let $W \subseteq U$ correspond to the set of positions in U that are coloured with any colour other than the one representing empty space. For each position $\overline{\mathbf{x}} \in W$, a vertex $v_{\overline{\mathbf{x}}}$ is constructed. The vertex is given a weight equal to the charge of the corresponding ion. Given two vertices $v_{\overline{\mathbf{x}}}, v_{\overline{\mathbf{y}}} \in V$, corresponding to positions $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ respectively, the edge $(v_{\overline{\mathbf{x}}}, v_{\overline{\mathbf{y}}})$ is given a weight equal to $f(\overline{\theta}(U(\overline{\mathbf{x}}), U(\overline{\mathbf{y}})), D(v_{\overline{\mathbf{x}}}, v_{\overline{\mathbf{y}}}))$. For notation $\mathrm{w}(v)$ is used to denote the weight of a vertex and $\mathrm{w}((u,v))$ the weight of the edge between vertices u and v. A graph constructed in this way is referred to as a *Crystal Graph.* For notation, given a crystal graph $G = (V, E)$ weighted by the energy function f, $f(G)$ is used to denote the total pairwise energy of G, i.e. $f(G) = \sum_{(u,v)\in E} \mathrm{w}((u,v))$.

In the context of the deletion operation, crystals are referred to in terms of the corresponding crystal graph. Using the terminology of the crystal graph, the concept of neutrality for a subset of vertices may be redefined as:

Definition 6. *A set of vertices $R \subset V$ is* ***neutral*** *if* $\sum_{v_i\in R} \mathrm{w}(v_i) = 0$.

The *k-Charge Removal* Problem The k-Charge Removal problem, henceforth k-CHARGE REMOVAL, takes as input a crystal graph G corresponding to a "dense" initial arrangement of ions, with the goal of removing some vertices in order to minimise the energy of the new subgraph $G' \subset G$. It is assumed that the initial graph is *neutral* by Definition 6. As G' must also be neutral, any set of vertices which is removed must therefore be neutral. A natural number k of charges to remove is chosen, as defined in Definitions 8 and 9. In practical applications, the value of k may be chosen either using intuition from chemistry, or by exhaustively checking each value of k.

Definition 7. *For any set $S \subseteq V$, S^+ denotes the set of positively charged vertices in S, and S^- the set of negatively charged vertices.*

Definition 8. *A* ***set of k-charges*** *R in a crystal graph (V, E) is a neutral set of vertices, where $R \subseteq V$ and* $\left|\sum_{v\in R^+} \mathrm{w}(v)\right| = \left|\sum_{v\in R^-} \mathrm{w}(v)\right| = k$.

Informally, a set of k-charges is a set of vertices with a net charge of 0, where the magnitudes of the sums of all positively charged vertices is k, as is the sum of all negatively charged vertices.

Definition 9. *The* **removal** *operation of a set of vertices R from a graph $G = (V, E)$ returns the graph $G' = (V', E')$, where $V' = V \setminus R$ and E' is the set of edges in E with no endpoint in R.*

In other words, a removal of R from G returns the graph G' that is the complement of the graph induced by R.

Problem 5. *k-Charge Removal (k-CHARGE REMOVAL)*

Input: A crystal graph G, with edges weighted by a given common energy function U, a natural number k and a goal energy $g \in \mathbb{R}$.

Question: Does there exist a set of k-Charges $R \subset V$ such that removing R from G returns a graph G', where $U(G') \leq g$?

The $k_{\geq}$-*Charge Removal* Problem One variation of k-CHARGE REMOVAL is the $k_{\geq}$-*Charge Removal problem*, denoted $k_{\geq}$-CHARGE REMOVAL. This problem takes the same input as in k-CHARGE REMOVAL, however rather than looking to remove a set of exactly k-charges, it is instead sufficient to remove a neutral set of at least k positive and negative charges. In this generalisation, vertices of total weight more than k may be removed, provided the cell remains neutral. Note that any removal of exactly k is also acceptable for this generalisation.

Definition 10. *A set of $k_{\geq}$-**charges** R from a crystal graph (V, E) is a neutral subset, where $R \subseteq V$ and $\sum\limits_{v_i \in R^+} \mathrm{w}(v_i) \geq k$.*

Problem 6. *$k_{\geq}$-Charge Removal ($k_{\geq}$-CHARGE REMOVAL)*

Input: A crystal graph G, with edges weighted by a given common energy function U, a natural number k, and a goal energy $g \in \mathbb{R}$.

Question: Does there exist a set of $k_{\geq}$-Charges $R \subset V$ such that removing R from G returns a graph G', where $U(G') \leq g$?

Proposition 4. *A solution to k-CHARGE REMOVAL or $k_{\geq}$-CHARGE REMOVAL can be verified in polynomial time.*

Proof. A solution to k-CHARGE REMOVAL contains the set of charges R that are removed. This can be verified as a set of $k_{\geq}$-charges by simply summing up the positive and negative weights, checking that the set is neutral and that $\left|\sum\limits_{v_i \in R^+} \mathrm{w}(v_i)\right| = k$ for k-CHARGE REMOVAL or $\left|\sum\limits_{v_i \in R^+} \mathrm{w}(v_i)\right| \geq k$ for $k_{\geq}$-CHARGE REMOVAL. The time complexity is of the order of $O(|R|)$. Similarly the sum of the edges in the original graph G that do not have an endpoint in R can be checked against the goal value g. This is done in $O(|V|^2)$ time, as the graph is complete. As no step takes more than $O(|V|^2)$ time, a solution to either k-CHARGE REMOVAL or $k_{\geq}$-CHARGE REMOVAL can be verified in polynomial time. Hence k-CHARGE REMOVAL and $k_{\geq}$-CHARGE REMOVAL fall into the class of NP problems. □

The *Minimal-$k_{\geq}$-Charge Removal* Problem An alternative variation of $k_{\geq}$-CHARGE REMOVAL is the *minimal-$k_{\geq}$-Charge Removal* problem, denoted MINIMAL-$k_{\geq}$-CHARGE REMOVAL. This also serves as a generalisation of k-CHARGE REMOVAL, where the goal is to get close to a set of k-charges, accepting that it may not be possible to reach the exact value. In this problem a *minimal* set of $k_{\geq}$-charges is removed.

Definition 11. *A set R of $k_{\geq}$-charges is **minimal** if there exists no subset $R' \subset R$ such that $\left|\sum\limits_{v_i \in R'^{+}} \mathrm{w}(v_i)\right| \geq k$ and $\left|\sum\limits_{v_i \in R'^{+}} \mathrm{w}(v_i)\right| = \left|\sum\limits_{v_j \in R'^{-}} \mathrm{w}(v_j)\right|$.*

Informally, Definition 11 means that there is no way of getting closer to a set of k-charges from the set, without having fewer than k charges. It follows that for a given crystal graph, there may be multiple minimal $k_{\geq}$-charge sets for a given k. A removal of $k_{\geq}$-charges is minimal if the set of $k_{\geq}$-charges is minimal. It may be noted that a set of k-charges is always a minimal set of $k_{\geq}$-charges.

Problem 7. *Minimal-$k_{\geq}$-Charge Removal (*MINIMAL-$k_{\geq}$-CHARGE REMOVAL*)*

Input:	A crystal graph G, with edges weighted by a given common energy function U, a natural number k, and a goal energy $g \in \mathbb{R}$.
Question:	Does there exist a minimal set of $k_{\geq}$-charges $R \subset V$ such that removing R from G returns a graph G', where $U(G') \leq g$?

Proposition 5. *It is NP-hard to verify if a set of $k_{\geq}$-charges is minimal when no bounds are given on the charges of the vertices.*

Proof. This is shown by a reduction from the *subset-sum* problem. In the subset-sum problem there is a set of values S, and a goal k. The task is to choose some subset $S' \subseteq S$ such that $\sum\limits_{i \in S'} i = k$. Note that this problem remains NP-complete in the case the input is only positive integers.

Let $I = (S, k)$ be an instance of the subset sum problem for the set S with a goal value of k. Given I, a crystal graph is created as follows. For each integer $i \in S$ a new vertex with a charge of i is created, note these correspond to the set V^{+}. Two further ions are created, the first having a charge of $-k$ and the second having a charge of $-\left(\left(\sum\limits_{v_i \in V^{+}} \mathrm{w}(v_i)\right) - k\right)$, these correspond to V^{-}. The value k' is chosen as the greater of k and $\left(\sum\limits_{v_i \in V^{+}} \mathrm{w}(v_i)\right) - k$.

Given I, we claim that the only minimal k'-Charge Removal, R, from $S = (V, E)$ is $R = V$ if and only if there is no solution to I. To disprove that R is minimal there must be some subset $R' \subset R$ that is also a set of $k_{\geq}$-charges. As k' charges must be removed, any such V'^{-} must only contain the vertex in V^{-} with a charge of $-k'$. Therefore if this claim is false, there must be a set $R'^{+} \subseteq R^{+}$ such that $\sum\limits_{v_i \in R'^{+}} \mathrm{w}(v_i) = k'$. If there is such a R' then there must also be a solution to the subset sum instance as either R'^{+} or $R^{+} \setminus R'^{+}$. This is shown as if $k' = k$, the values in R'^{+} must sum to k, satisfying I. Conversely, if there is k'-Charge removal $R \subset V$ then following the above arguments, there must be a solution to I.

In the other direction, if there is a solution to I then trivially there must be exist such a R'^{+} that would make R' non-minimal. Similarly if there is no valid solution to I then the only minimal set of k-charges is the complete set of ions. Therefore it may not be determined if the a solution is minimal in polynomial time. Subsequently as a minimal set of charges for $k_{\geq}$-CHARGE REMOVAL is required, a solution can not be verified in polynomial time unless $P = NP$, therefore it is not in NP in the general case. □

Corollary 1. *It is NP-hard to determine if an instance of k-*CHARGE REMOVAL *has a valid solution in the case there are no bounds on the charges of the vertices.*

Proof. It follows from the arguments of Proposition 5 that an instance of k-CHARGE REMOVAL may be constructed for a subset sum instance $I = (S, k)$ such that it is only satisfiable if the subset sum instance is. □

Lemma 1. *A set of k-Charges may be verified as minimal in polynomial time for charges bounded by a polynomial size.*

Proof. In the case when the charges of the vertices are bounded, a solution to the subset sum may be found in polynomial time, for example, relative to either the upper limit on the weights due to Pisinger [83], or the number of distinct weights and the goal values due to Axiotis and Tzamos [8]. Using these algorithms a set of k-Charges R can be verified as minimal. This is done by, for every value $1 \leq t \leq \sum\limits_{v_i \in R^{+}} \mathrm{w}(v_i) - k$ checking if there a subset of charges $R'^{+} \subseteq R^{+}$ and $R'^{-} \subseteq R^{-}$ such that $t = \sum\limits_{v_i \in R'^{+}} \mathrm{w}(v_i) = \left| \sum\limits_{v_i \in R'^{-}} \mathrm{w}(v_i) \right|$. If there exists such a solution for any t then R is not minimal.

The claimed energy may also be verified by checking the sum of pairwise interactions relative to U, with may trivially be done in Polynomial time by the definition of U. Therefore under these restrictions $k_{\geq}$-CHARGE REMOVAL is in NP. $\square$

2.2 Words and Cyclic Words

In order to represent crystals in a discrete manner, this section introduces the concept of *combinatorial crystals.* Informally, a combinatorial crystal can be thought of as a description of crystal in discrete space. Here, a combinatorial crystal is represented explicitly by constructing an alphabet contain a symbol for both the empty space and for each ion in the crystal. In order to represent a crystal in this manner we turn to cyclic words.

At a high level, a cyclic word can be thought of as an infinite word resented by a period. This fits naturally with the unit cell as a representation of the crystal. Due to the infinite nature of cyclic words, they are generally considered in terms of equivalence classes. This thesis looks at several such equivalence classes, including the classes of necklaces, bracelets, necklaces with constrained Parikh vectors, necklaces with forbidden subwords, and multidimensional necklaces. This section is the first look at the most basic of these classes, those being the sets of necklaces and bracelets. Section 2.2.1 looks at necklaces where some constraints are placed on the corresponding Parikh vector. Section 2.2.2 looks at necklaces with a set of forbidden subwords. Section 2.2.3 looks at multidimensional words and necklaces. Finally Section 2.2.4 provides a brief discussion on how to represent a crystal as a multidimensional necklace.

In order to define necklaces and bracelets, the following notions are introduced regarding words. Let Σ be a finite alphabet. For the remainder of this thesis, let $q = |\Sigma|$. In general, this work assumes Σ to be made of symbols corresponding to the set $\{1, 2, 3, \dots, q\}$, ordered such that $1 < 2 < 3 < \dots < q$. We denote by Σ^* the set of all words over Σ and by Σ^n the set of all words of length n. The notation $\bar{w}$ is used to clearly denote that the variable w is a word. The length of a word $\bar{u} \in \Sigma^*$ is denoted $|\bar{u}|$. We use $\bar{u}_i$, for any $i \in 1 \dots |\bar{u}|$ to denote the i^{th} symbol of $\bar{u}$. Given two words $\bar{w}, \bar{u} \in \Sigma^*$, the *concatenation operation* is denoted $\bar{w} : \bar{u}$ and returns the word of length $|\bar{w}| + |\bar{u}|$ where $(\bar{w} : \bar{u})_i$ equals either $\bar{w}_i$, if $i \leq |\bar{w}|$ or $\bar{u}_{i-|\bar{w}|}$ if $i > |\bar{w}|$. The *reversal* operation on a word $\bar{w} = \bar{w}_1\bar{w}_2 \dots \bar{w}_n$, denoted by $\bar{w}^R$, returns the word $\bar{w}_n \dots \bar{w}_2\bar{w}_1$.

Let $[n]$ return the ordered set of integers from 1 to n inclusive. More generally, let $[i, j]$ return the ordered set of integers from i to j inclusive. Given 2 words $\bar{u}, \bar{v} \in \Sigma^*$ where $|\bar{u}| \leq$

$|\bar{v}|$, $\bar{u} = \bar{v}$ if and only if $|\bar{u}| = |\bar{v}|$ and $\bar{u}_i = \bar{v}_i$ for every $i \in [|\bar{u}|]$. A word $\bar{u}$ is *lexicographically smaller* than $\bar{v}$ if there exists an $i \in [|\bar{u}|]$ such that $\bar{u}_1\bar{u}_2 \dots \bar{u}_{i-1} = \bar{v}_1\bar{v}_2 \dots \bar{v}_{i-1}$ and $\bar{u}_i < \bar{v}_i$. For example, given the alphabet $\Sigma = \{a, b\}$ where $a < b$, the word *aaaba* is smaller than *aabaa* as the first 2 symbols are the same and a is smaller than b. For a given set of words $\mathbf{S}$, the rank of $\bar{v}$ with respect to $\mathbf{S}$ is the number of words in $\mathbf{S}$ that are smaller than $\bar{v}$.

The *translation* of a word $\bar{w} = \bar{w}_1\bar{w}_2 \dots \bar{w}_n$ by $r \in [n-1]$ returns the word $\bar{w}_{r+1} \dots \bar{w}_n$ $\bar{w}_1 \dots \bar{w}_r$, and is denoted by $\langle \bar{w} \rangle_r$, i.e. $\langle \bar{w}_1\bar{w}_2 \dots \bar{w}_n \rangle_r = \bar{w}_{r+1} \dots \bar{w}_n\bar{w}_1 \dots \bar{w}_r$. Under the translation operation, $\bar{u}$ is equivalent to $\bar{v}$ if $\bar{v} = \langle \bar{w} \rangle_r$ for some r. The t^{th} power of a word $\bar{w} = \bar{w}_1 \dots \bar{w}_n$, denoted $\bar{w}^t$, is equal to $\bar{w}$ repeated t times. For example $(aab)^3 =$ *aabaabaab*. A word $\bar{w}$ is *periodic* if there is some word $\bar{u}$ and integer $t \geq 2$ such that $\bar{u}^t = \bar{w}$. Equivalently, word $\bar{w}$ is *periodic* if there exists some translation $0 < r < |\bar{w}|$ where $\bar{w} = \langle \bar{w} \rangle_r$. A word is *aperiodic* if it is not periodic. The *period* of a word $\bar{w}$ is the length of the smallest word $\bar{u}$ for which there exists some value t for which $\bar{w} = \bar{u}^t$.

A *cyclic word*, also called a *necklace*, is the equivalence class of words under the translation operation. For notation, a word $\bar{w}$ is written as $\tilde{\mathbf{w}}$ when treated as a necklace. Given a necklace $\tilde{\mathbf{w}}$, the *necklace representative* is the lexicographically smallest element of the set of words in the equivalence class $\tilde{\mathbf{w}}$. The necklace representative of $\tilde{\mathbf{w}}$ is denoted $\langle \tilde{\mathbf{w}} \rangle$, and the r^{th} shift of the necklace representative is denoted $\langle \tilde{\mathbf{w}} \rangle_r$. The reversal operation on a necklace $\tilde{\mathbf{w}}$ returns the necklace $\tilde{\mathbf{w}}^R$ containing the reversal of every word $\bar{u} \in \tilde{\mathbf{w}}$, i.e. $\tilde{\mathbf{w}}^R = \{\bar{u}^R : \bar{u} \in \tilde{\mathbf{w}}\}$. Given a word $\bar{w}$, $\langle \bar{w} \rangle$ denotes the necklace representative of the necklace containing $\bar{w}$, i.e. the representative of $\tilde{\mathbf{u}}$ where $\bar{w} \in \tilde{\mathbf{u}}$. By convention the set of necklaces of length n over an alphabet of size q is denoted $\mathcal{N}_q^n$, the size of which is given by $|\mathcal{N}_q^n|$. An aperiodic necklace, known as a *Lyndon word*, is a necklace representing the equivalence class of some aperiodic word. Note that the number of unique words represented by a Lyndon word of length n is n. The set of Lyndon words of length n over an alphabet of size q is denoted $\mathcal{L}_q^n$.

A *subword* of the cyclic word $\bar{w}$, denoted $\bar{w}_{[i,j]}$ is the word $\bar{u}$ of length $|\bar{w}| + j - i - 1 \bmod |\bar{w}||)$ such that $\bar{u}_a = \bar{w}_{i-1+a \bmod |\bar{w}|}$. For notation $\bar{u} \sqsubseteq \bar{w}$ denotes that $\bar{u}$ is a subword of $\bar{w}$. Further, $\bar{u} \sqsubseteq_i \bar{w}$ denotes that $\bar{u}$ is a subword of $\bar{w}$ of length i. If $\bar{w} = \bar{u}\bar{v}$, then $\bar{u}$ is a prefix and $\bar{v}$ is a suffix. A prefix or suffix of a word $\bar{u}$ is *proper* if its length is smaller than $|\bar{u}|$. For notation, the set $\mathbf{S}(\bar{v}, \ell)$ is defined as the set of all subwords of $\bar{v}$ of length ℓ. Formally let $\mathbf{S}(\bar{v}, \ell) = \{\bar{s} \sqsubseteq \bar{v} : |\bar{s}| = \ell\}$. Further, $\mathbf{S}(\bar{v}, \ell)$ is assumed to be in lexicographic order, i.e. $\mathbf{S}(\bar{v}, \ell)_1 \geq \mathbf{S}(\bar{v}, \ell)_2 \geq \dots \mathbf{S}(\bar{v}, \ell)_{|\bar{v}|}$.

As both necklaces and Lyndon words are classical objects, there are many fundamental results regarding each objects. The first results for these objects were equations determining the number of necklaces or Lyndon words of a given length. The number of necklaces is given by the equation $|\mathcal{N}_q^n| = \frac{1}{n}\sum\limits_{d|n} \phi\left(\frac{n}{d}\right) q^d$ where $\phi(n)$ is Euler's totient function. Similarly the number of Lyndon words is given with the equation $|\mathcal{L}_q^n| = \sum\limits_{d|n} \mu\left(\frac{n}{d}\right) |\mathcal{N}_q^d|$, where $\mu(x)$ is the Möbius function. Graham et. al. provide a proof of these equations [42]. The problem of *generating* the set of all necklaces in lexicographic order was solved first by Fredricksen and Maiorana [31]. This algorithm was shown to run in constant amortised time (CAT) by Ruskey et. al. [88]. A more direct CAT algorithm for generating the set of all necklaces was introduced by Cattell et. al. [17].

Recently the dual problems of *ranking* and *unranking* necklaces have been studied. The *rank* of a word $\bar{w}$ in the set of necklaces $\mathcal{N}_q^n$ is in this work defined as the number of necklaces with a canonical representative smaller than $\bar{w}$. The *unranking* process is effectively the reverse of this. Given an integer $i \in [|\mathcal{N}_q^n|]$, the goal of the unranking process for i is to determine the necklace $\mathcal{N}_q^n$ with a rank of i. Lyndon words were first ranked by Kociumaka et. al. [60] without tight complexity bounds. The first algorithm to rank necklaces was given by Kopparty et al. [61], also without tight bounds on the complexity. A quadratic time algorithm for ranking both Lyndon necklaces was provided by Sawada et al. [92], who also provided a cubic time unranking algorithm.

A *bracelet* is the equivalence class of words under the combination of the translation and the reversal operations. In this way a bracelet can be thought of as the union of two necklace classes $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{w}}^R$, hence $\hat{\mathbf{w}} = \tilde{\mathbf{w}} \cup \tilde{\mathbf{w}}^R$. Given a bracelet $\hat{\mathbf{w}}$, the *bracelet representative* of $\hat{\mathbf{w}}$, denoted by $[\hat{\mathbf{w}}]$, is the lexicographically smallest word $\bar{u} \in \hat{\mathbf{w}}$. In a similar manner to necklaces, the set of bracelets of length n over an alphabet of size q is denoted $\mathcal{B}_q^n$, with $|\mathcal{B}_q^n|$ denoting the size of this set.

The number of bracelets of length n over an alphabet of size q is given by the equation $|\mathcal{B}_q^n| = \begin{cases} \frac{1}{2}|\mathcal{N}_q^n| + \frac{1}{4}(q+1)q^{n/2} & n \text{ is even.} \\ \frac{1}{2}|\mathcal{N}_q^n| + \frac{1}{2}(q+1)q^{(n+1)/2} & n \text{ is odd.} \end{cases}$, a proof of which is given by Gilbert and Riordan [38]. As with necklaces, a CAT algorithm for the generation of bracelets has been derived [93].

In order to rank bracelets, some useful subclasses of necklaces and bracelets must be introduced. A necklace $\tilde{\mathbf{w}}$ is *palindromic* if $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}^R$. This means that the reflection of every word in $\tilde{\mathbf{w}}$ is also in $\tilde{\mathbf{w}}$, i.e. given $\bar{u} \in \tilde{\mathbf{w}}, \bar{u}^R \in \tilde{\mathbf{w}}$. Note that for any word $\bar{w} \in \tilde{\mathbf{a}}$, where $\tilde{\mathbf{a}}$ is a palindromic necklace, either $\bar{w} = \bar{w}^R$, or there exists some translation i for which $\langle \bar{w} \rangle_i = \bar{w}^R$.

Let $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ be a pair of necklaces belonging to the same bracelet class. For simplicity assume that $\langle \tilde{\mathbf{u}} \rangle < \langle \tilde{\mathbf{v}} \rangle$. The bracelet $\hat{\mathbf{u}}$ *encloses* a word $\bar{w}$ if $\langle \tilde{\mathbf{u}} \rangle < \bar{w} < \langle \tilde{\mathbf{v}} \rangle$. An example of this is the bracelet $\hat{\mathbf{u}} = aabc$ which encloses the word $\bar{w} = aaca$ as $aabc < aaca < aacb$. The set of all bracelets which enclose $\bar{w}$ are referred to as the set of bracelets *enclosing* $\bar{w}$.

2.2.1 Necklaces with Constrained Parikh Vectors

One of the most studied restrictions on the set of necklaces is the set of *fixed-content* necklaces. The content of a word $\bar{w}$ is defined as the number of times each symbol in the alphabet Σ appears in $\bar{w}$. The content of a word is given in the form of a *Parikh vector* $\overline{\mathbf{p}}$ where $\overline{\mathbf{p}}_i$ is the number of times the i^{th} symbol of Σ appears in the word. For notation $\mathrm{P}(\bar{w})$ returns the Parikh vector of the word $\bar{w}$. For example, the word $aabb$ over the alphabet $\{a, b, c\}$ has the Parikh vector $\mathrm{P}(aabb) = (2, 2, 0)$.

The set of fixed content necklaces is the set of necklaces sharing a given Parikh vector. For notation, let $N_q^n(\overline{\mathbf{p}})$ denote the set of necklaces sharing the Parikh vector $\overline{\mathbf{p}}$, i.e. $\mathcal{N}_q^n(\overline{\mathbf{p}}) = \{\tilde{\mathbf{w}} \in \mathcal{N}_q^n \,|\, \mathrm{P}(\tilde{\mathbf{w}}) = \overline{\mathbf{p}}\}$. For a binary alphabet, fixed-content necklaces are sometimes referred to as *fixed-density* necklaces. As in the unconstrained case, the set of fixed-content necklaces has been heavily studied. The number of fixed-content necklaces was given by Gilbert and Riordan [38] as:

$$|\mathcal{N}_q^n(\overline{\mathbf{p}})| = \frac{1}{n} \sum_{d|\,\gcd(p_1,p_2,\ldots,p_q)} \phi(d) \frac{\left(\frac{n}{d}\right)!}{\left(\frac{p_1}{d}\right)! \ldots \left(\frac{p_q}{d}\right)!}$$

As in the unconstrained case, constant amortised time algorithms have been developed to generate fixed-density necklaces [89], fixed content necklaces [94] and fixed content bracelets [58]. More recently, ranking and unranking algorithms have been presented for fixed density necklaces [45] running in $O(n^3)$ and $O(n^4)$ time respectively.

We focus on a generalisation of this constraint, instead looking at necklaces corresponding to Parikh vectors that solve some given set of linear equations. At a high level, the idea is to represent each variable in the equation with a symbol. More precisely, given a system of m linear Diophantine equations for q variables determined by the $m \times q$ size matrix $A \in \mathbb{N}^{m,q}$ and m-length vector $\overline{\mathbf{C}} \in \mathbb{N}^m$, the alphabet $\Sigma(A) = \{1, 2, \ldots, q\}$ is constructed. Let $\overline{\mathbf{S}}$ be a positive integer solution to the system of Diophantine equations, i.e. a vector such that $A \cdot \overline{\mathbf{S}} = \overline{\mathbf{C}}$. Given a symbol $x \in \Sigma(A)$, $\mathrm{w}(x)$ is used to denote the vector corresponding to the x^{th} column of A, informally the vector corresponding to the impact of adding 1 to the x^{th} variable in the system of linear equations. The solution $\overline{\mathbf{S}}$ is represented using $\Sigma(A)$ by the word $1^{\overline{\mathbf{S}}_1} : 2^{\overline{\mathbf{S}}_2} : \ldots : q^{\overline{\mathbf{S}}_q}$, i.e. the word containing $\overline{\mathbf{S}}_i$ copies of symbol i. In the other direction, given the word $\bar{w} \in \Sigma(A)^*$ the Parikh vector $\mathrm{P}(\bar{w})$ may be used as a potential solution to the system of equations. Given a system of equations $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$ and length n, the Parikh vector of the word $\bar{w} \in \Sigma(A)^n$ solves $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{x}}$ if $A \cdot \mathrm{P}(\bar{w}) = \overline{\mathbf{C}}$. This thesis makes the following assumptions about every system of linear equations defined by $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$:

- The number of equations is m.
- The number of variables is q
- Every entry of A, $A[i,j]$, is at least 0, i.e. $A[i,j] \geq 0, \forall i \in [m], j \in [q]$.
- Every value of $\overline{\mathbf{C}}$ is at least 0, i.e. $\overline{\mathbf{C}}[i] \geq 0, \forall i \in [m]$.

This model is used in two distinct approaches. First, this work introduces *n-weight Parikh vectors* that are solutions to a given system of linear equations. A Parikh vector $\overline{\mathbf{P}}$ has weight n if the sum of every entry equals n, i.e. $P_1 + P_2 + \ldots + P_q = n$. In order to represent Parikh vectors uniquely, the canonical representation for a given Parikh vector $\overline{\mathbf{P}}$ is derived in the same way as the word representing a vector solution to a system of linear equations described above. Formally, given a Parikh vector $\overline{\mathbf{P}} = (P_1, P_2, \ldots, P_q)$ for the alphabet Σ, the canonical form of $\overline{\mathbf{P}}$, denoted $\langle\overline{\mathbf{P}}\rangle$, is the word $1^{P_1} : 2^{P_2} : \ldots : q^{P_q}$. For simplicity, we refer to the canonical form of a Parikh vector as a *Parikh word*. Note that every subword of a Parikh word is itself a Parikh word for the Parikh vector of the subword. The set of all n-weight Parikh vectors solving a given set of Diophantine equations is denoted $\mathbf{P}(n, A, \overline{\mathbf{C}})$. The set $\mathbf{P}(n, A, \overline{\mathbf{C}})$ is assumed to be ordered lexicographically with respect to the canonical representation of each Parikh vector, i.e. given $\overline{\mathbf{P}}, \overline{\mathbf{Q}} \in \mathbf{P}(n, A, \overline{\mathbf{C}})$, $\overline{\mathbf{P}} < \overline{\mathbf{Q}}$ if and only if $\langle\overline{\mathbf{P}}\rangle < \langle\overline{\mathbf{Q}}\rangle$.

The second approach is the more general set of *n-length necklaces* with Parikh vectors solving a given system of linear equations. A word $\bar{w} \in \Sigma(A)^n$ corresponds to a Parikh vector solving the linear equation defined by the matrix $A \in \mathbb{N}^{m,q}$ and vector $\overline{\mathbf{C}} \in \mathbb{N}^m$ if $A \cdot \mathrm{P}(\bar{w}) = \overline{\mathbf{C}}$. Note than any Parikh word corresponding to an n-weight Parikh vector that solves $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$ is also the canonical representation of a n-length necklace solving $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$. For notation, the set of necklaces with Parikh vectors corresponding to solutions of the set of linear equations defined by the matrix $A \in \mathbb{N}^{m,q}$ and vector $\overline{\mathbf{C}} \in \mathbb{N}^m$ is denoted $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$.

2.2.2 Necklaces with Forbidden Subwords

The final class of one dimensional cyclic words this work looks at is the set of necklaces with *forbidden subwords*. Informally, this is the set of necklaces for a given length and alphabet such that no necklace contains any subword from some set $\mathcal{F}$. The set of necklaces of length n over an alphabet of size q with no subword in the set $\mathcal{F}$ is denoted $\mathcal{N}_q^n(\mathcal{F})$. Similarly the set of words with no subword in $\mathcal{F}$ is denoted $\mathbf{F}_q^n(\mathcal{F})$.

Existing results due to Ruskey and Sawada have focused on the case of a single subword, providing a method to count and generate the set of necklaces in this setting [90]. The counting equation may be generalised to the case where $\mathcal{F}$ is of arbitrary size following the same arguments laid out by Ruskey and Sawada in [90] giving:

$$|\mathcal{N}_q^n(\mathcal{F})| = \frac{1}{n} \sum_{d|n} \phi\left(\frac{n}{d}\right) |\mathbf{F}_q^n(\mathcal{F})|$$

2.2.3 Multidimensional Necklaces

In order to establish multidimensional necklaces, notation for multidimensional words must first be introduced. A *d-dimensional word* over Σ is an array of dimensions $\overline{\mathbf{n}} = (n_1, n_2, \dots, n_d)$ of elements from Σ. In this work we tacitly assume that $n_1 \leq n_2 \leq \dots \leq n_d$. Let $|\bar{w}|$ be the dimensions of $\bar{w}$. Given a vector of dimensions $\overline{\mathbf{n}} = (n_1, n_2, \dots, n_d)$, $\Sigma^{\overline{\mathbf{n}}}$ is used to denote the set of all words of dimensions $\overline{\mathbf{n}}$ over Σ. Let $N = n_1 \cdot n_2 \cdot \dots \cdot n_d$ for a dimension vector $\overline{\mathbf{n}}$. For notation, given a vector $\overline{\mathbf{n}} = (n_1, n_2, \dots, n_d)$ where every $n_i \geq 0$, $[\overline{\mathbf{n}}]$ is used to denote the set $\{(x_1, x_2, \dots, x_d) \in \mathbb{N}^d | \forall i \in [d], x_i \leq n_i\}$. Similarly $[\overline{\mathbf{m}}, \overline{\mathbf{n}}]$ is used to denote the set $\{(x_1, x_2, \dots, x_d) \in \mathbb{N}^d | \forall i \in [d], m_i \leq x_i \leq n_i\}$.

For a d-dimensional word $\bar{w}$, the notation $\bar{w}_{(p_1, p_2, \dots, p_d)}$ is used to refer to the symbol at position $(p_1, p_2, \dots, p_d)$ in the array. Given 2 d-dimensional words $\bar{w}, \bar{u}$ such that $|\bar{w}| = (n_1, n_2, \dots, n_{d-1}, a)$ and $|\bar{u}| = (n_1, n_2, \dots, n_{d-1}, b)$, the concatenation $\bar{w} : \bar{u}$ is performed along the last coordinate, returning the word $\bar{v}$ of dimensions $(n_1, n_2, \dots, n_{d-1}, a + b)$ such that $\bar{v}_{\overline{\mathbf{p}}} = \bar{w}_{\overline{\mathbf{p}}}$ if $p_d \leq a$ and $\bar{v}_{\overline{\mathbf{p}}} = \bar{u}_{(p_1, p_2, \dots, p_{d-1}, p_d - a)}$ if $p_d > a$. See Figure 2.2.

A *multidimensional cyclic subword* of $\bar{w}$ of dimensions $\overline{\mathbf{m}}$ is denoted $\bar{v} \sqsubseteq_{\overline{\mathbf{m}}} \bar{w}$. As in the one-dimensional case, a subword is defined by a starting position in the original

$$\bar{w} = \begin{bmatrix} a & a & a & b \\ a & a & b & a \\ b & a & a & a \\ b & a & a & a \end{bmatrix}, \bar{u} = \begin{bmatrix} b & b & b & b \\ b & b & b & b \end{bmatrix}, \bar{w} : \bar{u} = \begin{bmatrix} a & a & a & b \\ a & a & b & a \\ b & a & a & a \\ b & a & a & a \\ b & b & b & b \\ b & b & b & b \end{bmatrix}$$

Figure 2.2: An example of the connotation between the 2D words $\bar{w}$ and $\bar{u}$.

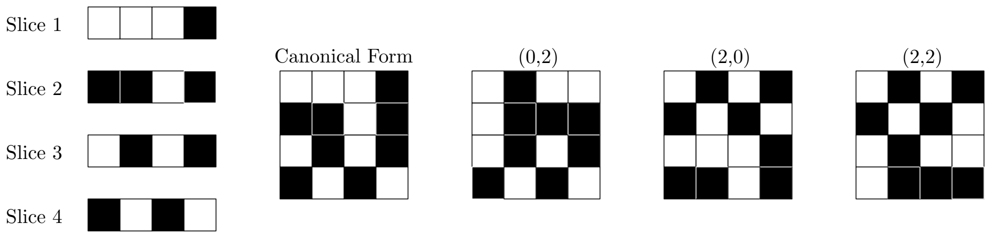

Figure 2.3: Example of a 2-dimensional word $\bar{w}$ of size $(4,4)$ over a binary alphabet: the 4 slices of $\bar{w}$; the canonical form of $\bar{w}$; and three translations of $\bar{w}$.

word and set of dimensions defining the size of the subword. The subword $\bar{v} \sqsubseteq \bar{w}$ starting at position $\overline{\mathbf{p}}$ with dimensions $\overline{\mathbf{m}}$ is the word $\bar{v}$ such that $\bar{v}_{\bar{\mathbf{i}}} = \bar{w}_{\bar{\mathbf{j}}}$ for all $\bar{\mathbf{j}}$ of the form $(p_1 + i_1 \bmod n_1, p_2 + i_2 \bmod n_2, \ldots, p_d + i_d \bmod n_d)$. Such a subword $\bar{v}$ is denoted by $\bar{w}_{\overline{\mathbf{p}},\overline{\mathbf{m}}}$. One important class of subwords are *slices*, an example of which is given in Figure 2.3. The i^{th} slice of $\bar{w}$, denoted by $\bar{w}_i$, is the subword of dimensions $(n_1, n_2, \ldots, n_{d-1}, 1)$ starting at position $(i, 1, \ldots, 1, 1)$ of $\bar{w}$. In the 2D case, the i^{th} slice corresponds to the i^{th} row of a word. This work uses $\bar{w}_{[i,j]}$ to denote $\bar{w}_i : \bar{w}_{i+1} : \ldots : \bar{w}_j$. A *prefix* of length l for a multidimensional word $\bar{w}$ is the first l slices of $\bar{w}$ in order. A *suffix* of length l for a multidimensional word $\bar{w}$ is the last l slices of $\bar{w}$ in order. In the two-dimensional case, prefix and suffix of length l corresponds to the first and last l rows respectively.

A d-dimensional translation r is defined by a vector $(r_1, r_2, \ldots, r_d)$. The translation of the word $\bar{w}$ of dimensions $\overline{\mathbf{n}}$ by r, denoted $\langle \bar{w} \rangle_r$ returns the word $\bar{v}$ such that $|\bar{v}| = \overline{\mathbf{n}}$ and $\bar{v}_{\overline{\mathbf{p}}} = \bar{w}_{\bar{\mathbf{j}}}$ for all $\bar{\mathbf{j}}$ of the form $(p_1 + r_1 \bmod n_1, p_2 + r_2 \bmod n_2, \ldots, p_d + r_d \bmod n_d)$. It is assumed that $r_i \in [0, n_i - 1]$, so the set of translations is equivalent to the direct product of the cyclic groups $Z_{n_1} \times Z_{n_2} \times \ldots \times Z_{n_d}$. Given two translations $r = (r_1, r_2, \ldots, r_d)$ and $t = (t_1, t_2, \ldots, t_d)$ in $Z_{\overline{\mathbf{n}}}$, $t + r$ is used to denote the translation $(r_1 + t_1 \bmod n_1, r_2 + t_2 \bmod n_2, \ldots, r_d + t_d \bmod n_d)$.

Definition 12. *A* ***multidimensional necklace*** *(multidimensional cyclic word)* $\tilde{\mathbf{w}}$ *is an equivalence class of all multidimensional words under the translation operation.*

Informally, given a necklace $\tilde{\mathbf{w}}$ containing the word $\bar{v}$, $\tilde{\mathbf{w}}$ contains every word $\bar{u}$ where there exists some translation r such that $\langle \bar{v} \rangle_r = \bar{u}$. Let $\mathcal{N}_q^{\overline{\mathbf{n}}}$ denote the set of necklaces of dimensions $\overline{\mathbf{n}}$ over an alphabet of size q. As in the 1D case, a *canonical representation* of a multidimensional necklace is defined as the smallest element in the equivalence class, denoted $\langle \tilde{\mathbf{w}} \rangle$. Similarly, given a word $\bar{v} \in \tilde{\mathbf{w}}$, $\langle \bar{v} \rangle$ denotes the canonical representation of the necklace $\tilde{\mathbf{w}}$, i.e. $\langle \bar{v} \rangle = \langle \tilde{\mathbf{w}} \rangle$. To determine the smallest element in the equivalence class, an ordering needs to be defined. First, we introduce an ordering over translations.

Definition 13. *Let* $Z_{\overline{\mathbf{n}}}$ *be the direct product of the cyclic groups* $Z_{n_1} \times Z_{n_2} \times \ldots \times Z_{n_d}$*, i.e. the set of all translations of words of dimensions* $\overline{\mathbf{n}}$*. The translation* $g \in Z_{\overline{\mathbf{n}}}$ *is indexed by the injective function* $\text{index}(g) \rightarrow \sum_{i=1}^{d} \left(g_i \cdot \prod_{j=1}^{i-1} n_j \right)$.

The translation $g \in Z_{\overline{\mathbf{n}}}$ is smaller than $t \in Z_{\overline{\mathbf{n}}}$ if $\text{index}(g) < \text{index}(t)$. Note that $(0, 0, \ldots, 0)$ is the smallest translation and $(n_1 - 1, n_2 - 1, \ldots, n_d - 1)$ is the largest. Using this definition an ordering on multidimensional words is defined recursively. The key idea is to compare each slice based on the canonical representations. For notation, given two words $\bar{u}, \bar{s} \in \tilde{\mathbf{w}}$, let $G(\bar{u}, \bar{s})$ return the smallest translation g where $\langle \bar{u} \rangle_g = \bar{s}$. Note that G can be computed in $O(N^2)$ time by simply checking each translation in $Z_{|\bar{u}|}$.

Definition 14. *Let* $\bar{w}, \bar{u} \in \Sigma^{\overline{\mathbf{n}}}$ *and let* i *be the smallest integer such that* $\bar{w}_i \neq \bar{u}_i$*. Then* $\bar{w} < \bar{u}$ *if either* $\langle \bar{w}_i \rangle < \langle \bar{u}_i \rangle$*, or* $\langle \bar{w}_i \rangle = \langle \bar{u}_i \rangle$ *and* $\text{index}(G(\bar{w}_i, \langle \bar{w}_i \rangle)) < \text{index}(G(\bar{u}_i, \langle \bar{u}_i \rangle))$*. Further, given necklaces* $\tilde{\mathbf{w}}$ *and* $\tilde{\mathbf{u}}$*,* $\tilde{\mathbf{w}} < \tilde{\mathbf{u}}$ *if and only if* $\langle \tilde{\mathbf{w}} \rangle < \langle \tilde{\mathbf{u}} \rangle$*.*

An example of the ordering is given in Figure 2.4. In what follows, $\mathcal{N}_q^{\overline{\mathbf{n}}}$ is assumed to be ordered as in Definition 14. The ***rank*** of a necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ is defined as the number of necklaces smaller than $\tilde{\mathbf{w}}$ in $\mathcal{N}_q^{\overline{\mathbf{n}}}$. In the other direction, the i^{th} necklace in $\mathcal{N}_q^{\overline{\mathbf{n}}}$ is the necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ with the rank i, i.e. the necklace $\tilde{\mathbf{w}}$ for which there are i smaller necklaces.

In order to answer some of the key questions regarding multidimensional necklaces, there are two further concepts that need to be defined for multidimensional necklaces. The first is the *period* of a word. Informally the period of $\bar{w}$ of dimensions $\overline{\mathbf{n}}$ can be thought of as the smallest subword that can tile d-dimensional space equivalently to $\bar{w}$. In order to define the period of a word, it is easiest to first define the concept of *aperiodicity*.

Definition 15. *A word $\bar{w}$ of dimensions $\overline{\mathbf{n}}$ is **aperiodic** if there exists no subword $\bar{v} \sqsubseteq \bar{w}$ of dimensions $\overline{\mathbf{m}} \neq \overline{\mathbf{n}}$ such that $m_i \leq n_i$ for every $i \in [1, d]$, and $\bar{w}_{\bar{\mathbf{j}}} = \bar{v}_{\bar{\mathbf{j}}'}$ where $\bar{\mathbf{j}}' = (j_1 \bmod m_1, j_2 \bmod m_2, \ldots, j_d \bmod m_d)$ for every position $\bar{\mathbf{j}} \in n_1 \times n_2 \times \ldots \times n_d$ in $\bar{w}$.*

Definition 16. *The **period** of a word $\bar{a}$ of dimensions $\overline{\mathbf{n}}$, denoted $\mathrm{Period}(\bar{a})$, is the aperiodic subword $\bar{b} \sqsubseteq \bar{a}$ of dimensions $\overline{\mathbf{m}}$ such that $\bar{a}_{\bar{\mathbf{i}}} = \bar{b}_{\bar{\mathbf{i}}'}$ for every position $\bar{\mathbf{i}} \in n_1 \times n_2 \times \ldots \times n_d$ and $\bar{\mathbf{i}}' = (i_1 \bmod m_1, i_2 \bmod m_2, \ldots, i_d \bmod m_d)$.*

By Definition 16 every word, including aperiodic ones, has a unique period [34]. In the case of an aperiodic word $\bar{w}$, the period is simply $\bar{w}$. A multidimensional necklace $\tilde{\mathbf{w}}$ is aperiodic if every word $\bar{v} \in \tilde{\mathbf{w}}$ is aperiodic. Note that if some word in $\tilde{\mathbf{w}}$ is aperiodic, then every word is. An aperiodic necklace is called a *Lyndon word.* A related but distinct concept is that of *atranslational* words. A word $\bar{w}$ is *atranslational* if there exists no translation $g \neq (n_1, n_2, \ldots, n_d)$ such that $\bar{w} = \langle \bar{w} \rangle_g$.

Definition 17. *A necklace $\tilde{\mathbf{w}}$ if dimensions $\overline{\mathbf{n}}$ is **atranslational** if there exists no pair of translations $g, h \in Z_{\overline{\mathbf{n}}}$ where $g \neq h$ and $\langle \tilde{\mathbf{w}} \rangle_g = \langle w \rangle_h$.*

In one dimension every aperiodic necklace is atranslational, while in any higher dimension every atranslational word is aperiodic, although not every aperiodic word is atranslational. For example $\begin{bmatrix} a & b \\ b & a \end{bmatrix}$ is aperiodic but not atranslational, as there are only two unique representations of the cyclic word. On the other hand $\begin{bmatrix} a & a \\ a & b \end{bmatrix}$ is both atranslational and aperiodic. For notation, $TR(\bar{w})$ is used to denote the index of the smallest translation $g \in Z_{\overline{\mathbf{n}}}$ where $\langle \bar{w} \rangle_g = \bar{w}$. The *translational period* of a word $\bar{w}$ is the subword of dimensions $\overline{\mathbf{g}}$ where $\overline{\mathbf{g}}$ is the smallest translations such that $\langle \bar{w} \rangle_{\overline{\mathbf{g}}} = \bar{w}$.

$$\bar{w} = \begin{bmatrix} \bar{w}_1 \\ \bar{w}_2 \\ \bar{w}_3 \\ \bar{w}_4 \end{bmatrix} = \begin{bmatrix} a & a & a & b \\ a & a & b & a \\ b & a & a & a \\ b & a & a & a \end{bmatrix}, \bar{u} = \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \bar{u}_4 \end{bmatrix} = \begin{bmatrix} a & a & a & b \\ a & a & b & a \\ a & b & a & a \\ b & a & a & a \end{bmatrix}, \bar{v} = \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \\ \bar{v}_4 \end{bmatrix} = \begin{bmatrix} a & a & a & b \\ a & a & b & a \\ a & a & b & b \\ b & a & a & a \end{bmatrix}$$

Figure 2.4: An example of three words, $\bar{w}, \bar{u}$, and $\bar{v}$, ordered as follows $\bar{w} < \bar{u} < \bar{v}$. Note that $\bar{w}_1 : \bar{w}_2 = \bar{v}_1 : \bar{v}_2 = \bar{u}_1 : \bar{u}_2$. However, $\langle \bar{w}_3 \rangle = \langle \bar{u}_3 \rangle = aaab$, which is smaller than $\langle \bar{v}_3 \rangle = aabb$. Further, $\bar{w}_3 < \bar{u}_3$ as $G(\bar{w}_3, \langle \bar{w}_3 \rangle) = 1$ and $G(\bar{u}_3, \langle \bar{u}_3 \rangle) = 2$, which is larger than 1.

As in the one dimensional case, constrained cases of these necklaces can be considered. This work considers only at fixed-content multidimensional necklaces. Given a Parikh vector $\overline{\mathbf{p}}$, the set of multidimensional necklaces of dimensions $\overline{\mathbf{n}}$ with the Parikh vector $\overline{\mathbf{p}}$ is denoted $\mathcal{N}_{\overline{\mathbf{p}}}^{\overline{\mathbf{n}}}$.

2.2.4 Combinatorial Crystals

The final concept presented in this chapter is how to represent combinatorial crystals as cyclic words. Intuitively, a cyclic word provides a natural basis for representing the unit cell of a crystal in discrete space. As with the unit cell of a crystal, a cyclic word provides a finite representation for an infinite period structure. For simplicity it is assumed the size of each "cell", the smallest unit within the discrete setting, is $1 \times 1 \times 1 \ldots \times 1$.

Let U be the unit cell of dimensions $\overline{\mathbf{n}}$ representing some crystal made with ions from the set $\mathbf{S}$. The alphabet $\Sigma(\mathbf{S})$ is constructed by making a symbol for each ion in $\mathbf{S}$, with an additional symbol added for the empty space. The word $\bar{U}$ is constructed from $\Sigma(\mathbf{S})$ as a multidimensional word of dimensions $\overline{\mathbf{n}}$ where the symbol at position $\overline{\mathbf{x}}$ of $\bar{U}$ is the symbol from $\Sigma(\mathbf{S})$ corresponding to the element or space at position $\overline{\mathbf{x}}$ of U. See Figure 1.4 as an example.

Chapter 3

Undecidability of the Pairwise Energy Minimisation Problem

The first results we provide are for the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$. The main claim of this Chapter is that the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$ is undecidable for any function in $\mathcal{CMV}(2)$. This Chapter is split into three parts. In Section 3.1, we provide background on the tiling problems from which the undecidablity result is derived. In Section 3.2, we present the all-distinct r-discretised rhombus periodic complete assignment problem as an auxiliary problem derived from the k-unique tiling problem to act as an intermediary step in proving the undecidablity of the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$. Finally in Section 3.3 we prove the undecidablity of the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$.

3.1 The Tiling Problem

In this section we provide some background on the *Tiling Problems* that are used as the basis for later reductions. In a tiling problem we are given a set of *tiles*, square plates with a fixed orientation where each edge is coloured from some set of colours $\mathcal{C}$. The goal of a tiling problem is to completely cover the plane with tiles such that every pair of adjacent tiles is coloured the same along the shared edge. In this section we introduce the further constraint that no two copies of the same tile may be within a distance of k of each other. This is used as an auxiliary problem in order to reduce the classical tiling problem to the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$.

Before discussing the new variations of the tiling problem, let us first present some notation. The edges of the tiles are labelled *East, West, North* and *South* such that the East edge is opposite the West edge, and the South edge is opposite the North edge. More precisely, given two tiles, v at position (x_1, y_1) and u at position (x_2, y_2) respectively such that $|x_1 - x_2| \leq 1$ and $|y_1 - y_2| \leq 1$, we say that:

	$x_1 < x_2$	$x_1 = x_2$	$x_1 > x_2$
$y_1 < y_2$	v is **North-West** of u	v is **North** of u	v is **North-East** of u
$y_1 = y_2$	v is **West** of u	v is u	v is **East** of u
$y_1 > y_2$	v is **South-West** of u	v is **South** of u	v is **South-East** of u

A tile t is represented by the four edges composing it. For notation let t_e be the colour of the tile t along edge e. A tile t can be represented as $t = \{t_{East}, t_{South}, t_{West}, t_{North}\}$. Given an edge e, e^{-1} denotes the opposite edge, e.g given $e = East$ then $e^{-1} = West$.

A *Tile Set* is a set of tiles, all of the same size and such that the edges of each tile are coloured uniquely with respect to orientation. The goal of the *Tiling problem* is to assemble copies of the tiles in a given tile set on an infinite plane ruled into squares of the size of one tile such that:

1. No tile is rotated or reflected.
2. A tile must be placed *exactly* over one square.
3. The colour of adjacent edges must match.
4. Every square must be covered by one tile.

An example to illustrate this is given by Figure 3.1. This problem is *solvable* for a given tile set if and only if such an assembly exists. An assembly is periodic if there exists some finite region of the plane that may be duplicated so as to solve the tiling problem. The *Periodic Tiling Problem* asks if there is a periodic assembly. Both the tiling problem and the periodic tiling problem are classically undecidable problems [5, 11]. Connections between this problem and chemistry are well established [87]. In particular the existence of aperiodic tilings were instrumental in the understanding of *quasicrystals*, structures that are close to being crystals without a unit cell [28, 69]. Indeed, our results can be thought of as asking if the lowest energy state for a given set of ions form a crystal or a quasicrystal.

The main difficulty with encoding the tiling problem into CSP is the concept of *orientation*. In the tiling problem it is integral that each tile is placed under the same

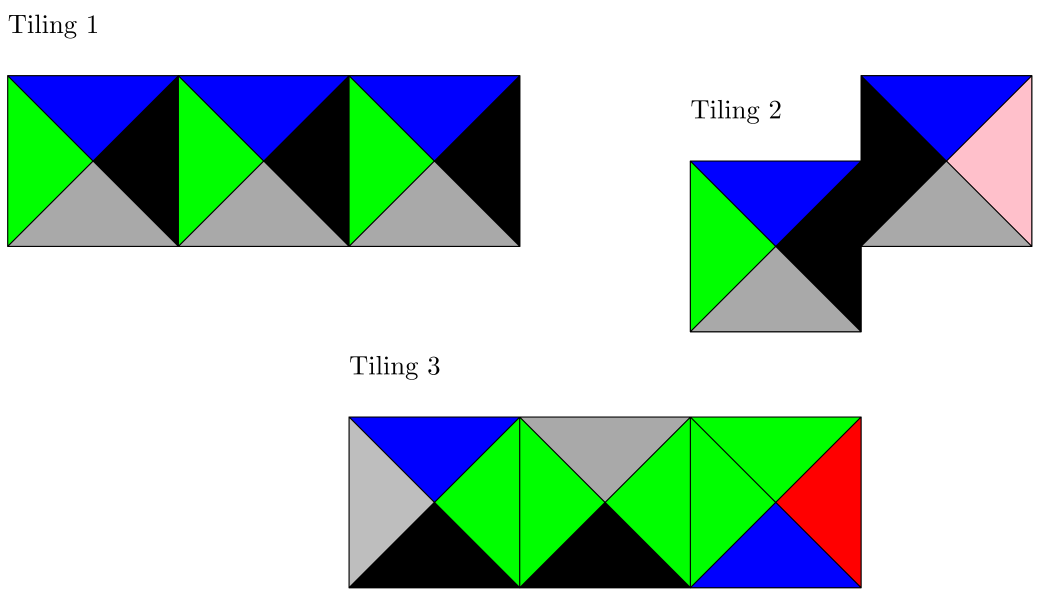

Figure 3.1: Tiling 1 is invalid as the West and East edges of the tiles are not the same colours. Tiling 2 is invalid as the tiles are not properly aligned. Tiling 3 is valid, however it would not be periodic.

orientation. This means that given two adjacent tiles, they must either touch West edge to East edge, or North edge to South edge. As our setting uses only the colours and distance between vertices to determine the pairwise energy, the concept of orientation is difficult to encode. To this end the *k-unique radius* variant of the tiling problem is introduced. Informally a tiling has a k-unique radius if and only if no two copies of a given tile are within a distance of k of each other. Let T be a tiling of $\mathbb{Z}^2$ such that $T(i,j)$ returns the tile at position (i,j). The tiling T has a k-unique radius if and only if for every $(i,j),(x,y) \in \mathbb{Z}^2$, where $D((i,j),(x,y)) \leq k$ the tile $T(i,j)$ is distinct from $T(x,y)$, i.e. $T(i,j) \neq T(x,y)$ where $D((i,j),(x,y))$ returns the Manhattan distance between (i,j) and (x,y), i.e. $D((i,j),(x,y)) = |i-x| + |j-y|$. An example is provided in Figure 3.2.

Problem 8. *The periodic tiling problem with a k-unique radius.*

Input:	A set of tiles, $\mathcal{T}$, and integer k
Question:	Does there exist a periodic tiling of the plane made from $\mathcal{T}$ such that given any tile t at position (x,y) there exists no other copy of t within a distance of k from (x,y)?

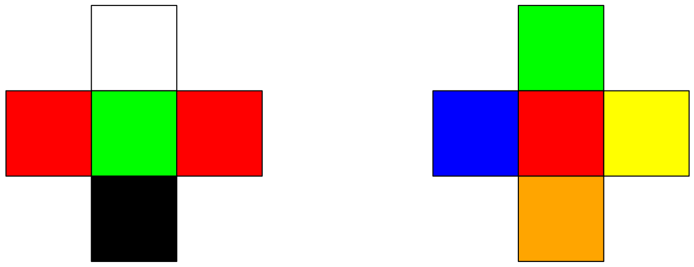

Figure 3.2: An example of an invalid 1 unique tiling (West) and a valid one (East). Note that each colour represents a class of tile. The West instance contains two tiles of the red class within a radius of 1 from the central (green) tile. In contrast every tile within a distance of 1 from the central tile (red) tile of the East tiling is distinct.

Proposition 6. *The periodic k-unique radius tiling problem is undecidable for any $k \in \mathbb{N}$.*

Proof. It follows from the undecidability of the periodic domino problem that this problem is undecidable. Given a set of $\mathcal{T}$ of tiles with the set of colours $\mathcal{C}$, a new set of tiles $\mathcal{T}'$ and colours $\mathcal{C}'$ are constructed as follows. For each colour $c \in \mathcal{C}$, a set of k^2 colours are added to $\mathcal{C}'$ labelled $c_{i,j}$ for each $i, j \in [k]$. For each tile $t \in \mathcal{T}$ where $t = \{t_{East}, t_{South}, t_{West}, t_{North}\}$ a set of k^8 tiles are constructed containing every tile of the form $\{(t_{East})_{i_1,j_1}, (t_{South})_{i_2,j_2}, (t_{West})_{i_3,j_3}, (t_{North})_{i_4,j_4}\}$, for every $i_1, i_2, i_3, i_4, j_1, j_2, j_3, j_4 \in [k]$. Using these sets observe that any tiling T of $\mathcal{T}$ can be converted to a k-unique tiling of $\mathcal{T}'$ by replacing the tile at position x, y of T with the tile $\{(T[x, y]_{East})_{x,y \bmod k}$, $(T[x, y]_{South})_{x,y \bmod k}$, $(T[x, y]_{West})_{x,y \bmod k}$, $(T[x, y]_{North})_{x,y \bmod k}\}$. In the other direction, given a k-unique tiling T' of $\mathcal{T}'$, a tiling from $\mathcal{T}$ can be constructed by replacing each in T' with the corresponding tile from $\mathcal{T}$. □

Problem 9. *The fixed period k-unique tiling problem.*

Input:	A set of tiles, $\mathcal{T}$, integer k, and pair of dimension n_1, n_2
Question:	Does there exist a periodic tiling of the plane of size $n_1 \times n_2$ over $\mathcal{T}$ where every tile within a distance of k for every other tile is distinct

Proposition 7. *The fixed period k-unique tiling problem is NP-hard.*

Proof. Following the same arguments as in Proposition 6, the fixed period tiling problem can be reduced to the fixed period k-unique tiling problem. As the fixed period tiling problem is known to be NP-hard by reduction from the Halting Problem [5, 11, 64], the fixed period k-unique tiling problem is NP-hard. □

3.2 Tiling with Overlapping Digitised Rhombuses

This section covers the problem of completely tiling the integer grid $\mathbb{Z}^2$ using *overlapping digitised rhombuses*. Before the problem can be defined, several concepts must first be introduced. Informally, a *digitised rhombus* with radius r can be thought of as a set of mono-chromatically coloured tiles organised as a rhombus. In this section, we use the *Manhattan distance*, formally given two points (x_1, y_1) and (x_2, y_2) the distance between them is given by $|x_1 - x_2| + |y_1 - y_2|$. We assume that each tile is coloured from some given set of colours $\mathcal{C}$. Figure 3.3 provides an example of such a rhombus.

Definition 18. *A* ***digitised rhombus*** *of radius r is the mapping from the grid $\{(x, y) : x, y \in \mathbb{Z}, |x| + |y| \leq r\}$ to a set of colours $\mathcal{C}$.*

Given a rhombus R and position $(x, y) \in \{(x, y) : x, y \in \mathbb{Z}, |x| + |y| \leq r\}$, $R_{i,j}$ is used to denote the colour mapped by R to position (i, j), i.e. the colour of the tile at position (i, j) in the rhombus. See Figure 3.4 for an example. A rhombus is *distinctly coloured* if $R_{i,j} \neq R_{l,m}$ for every pair of positions $(i, j), (l, m) \in \{(x, y) : (x, y) \in \mathbb{Z}^2, |x| + |y| \leq r\}$ where $(i, j) \neq (l, m)$, i.e. if the colour assigned to every distinct position in the rhombus is unique. An example is of a distinctly coloured digitised rhombus with a radius of 2 is provided in Figure 3.3.

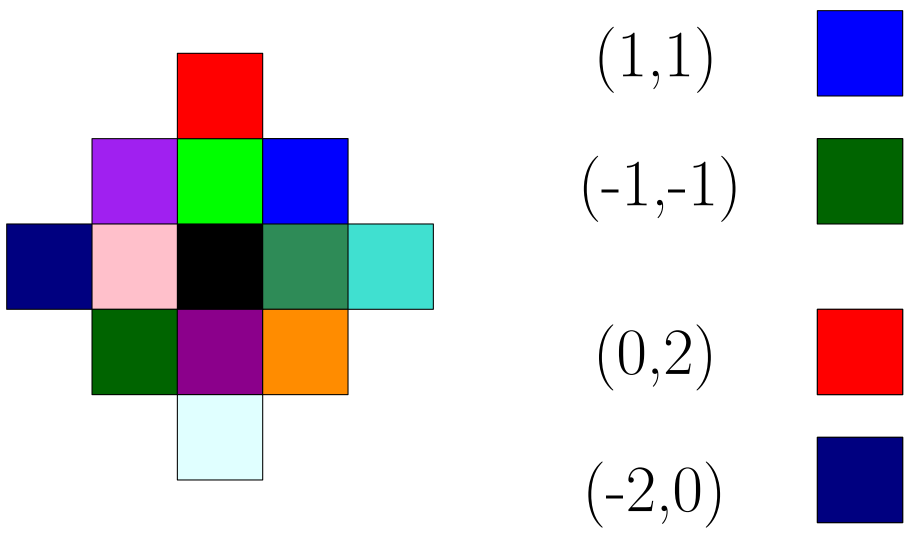

Figure 3.3: An example of an distinctly coloured 2-radius digitised rhombus (Left), along with examples of the colours at some given positions (Right).

In this section, we use a set of rhombuses $\mathcal{R}$ analogously to the set of tiles $\mathcal{T}$ used in tiling problems. More precisely, given the integer grid $\mathbb{Z}^2$, the assignment of a rhombus R to the position $(x, y) \in \mathbb{Z}^2$ is equivalent to colouring every vertex within a radius of r of (x, y) using R. Given a rhombus R assigned to position $(x, y) \in \mathbb{Z}^2$, the position (x', y') at a distance of no more than r from (x, y) is coloured $R_{x'-x,y'-y}$.

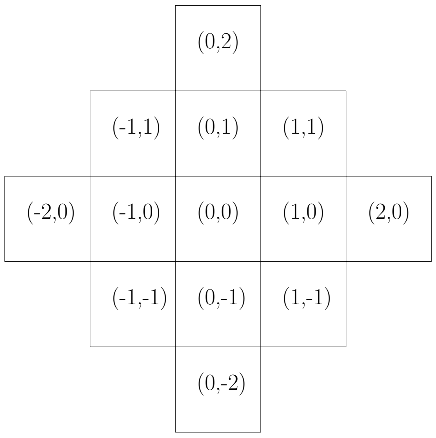

Figure 3.4: An example of a the co-ordinates for a digitised rhombus. Note that $R_{i,j}$ denotes the tile i tiles East and j tiles North of the central tile of R, denoted $R_{0,0}$. Negative values of i are used to denote tiles West of the centre and negative values of j are used to denote tiles South of the centre.

The focus in this work is on *overlapping rhombuses*. Given a pair of digitised rhombuses of radius $r \in \mathbb{N}$, R and S at positions (x_1, y_1) and (x_2, y_2), R overlaps S if the distance between R and S, denoted $D((x_1, y_1), (x_2, y_2))$, is no more than $2r$. The *overlap* between R and S are the set of positions that are assigned colours by both R and S. This corresponds to the set of positions $\{(a, b) \in \mathbb{Z}^2, D((a, b), (x_1, y_1)) \leq r$ and $D((a, b), (x_2, y_2)) \leq r\}$. Informally, R and S properly overlap when centred at (x_1, y_1) and (x_2, y_2) if every position in the overlap is assigned the same colour by both R and S. See Figure 3.5 for an example.

Definition 19. *Let R and S be a pair of r-radius digitised rhombuses centred on positions (x_1, y_1) and (x_2, y_2) respectively. Rhombuses R and S* ***properly overlap*** *if and only if $R_{i-x_1,j-y_1} = S_{i-x_2,j-y_2}$ for every position $(i, j) \in \{(a, b) \in \mathbb{Z}^2, D((a, b), (x_1, y_1)) \leq r$ and $D((a, b), (x_2, y_2)) \leq r\}$*

Note that any two rhombuses at a distance greater than $2 \cdot r$ from each other properly overlap following Definition 19 as the set $\{(a, b) \in \mathbb{Z}^2, D((a, b), (x_1, y_1)) \leq r$ and

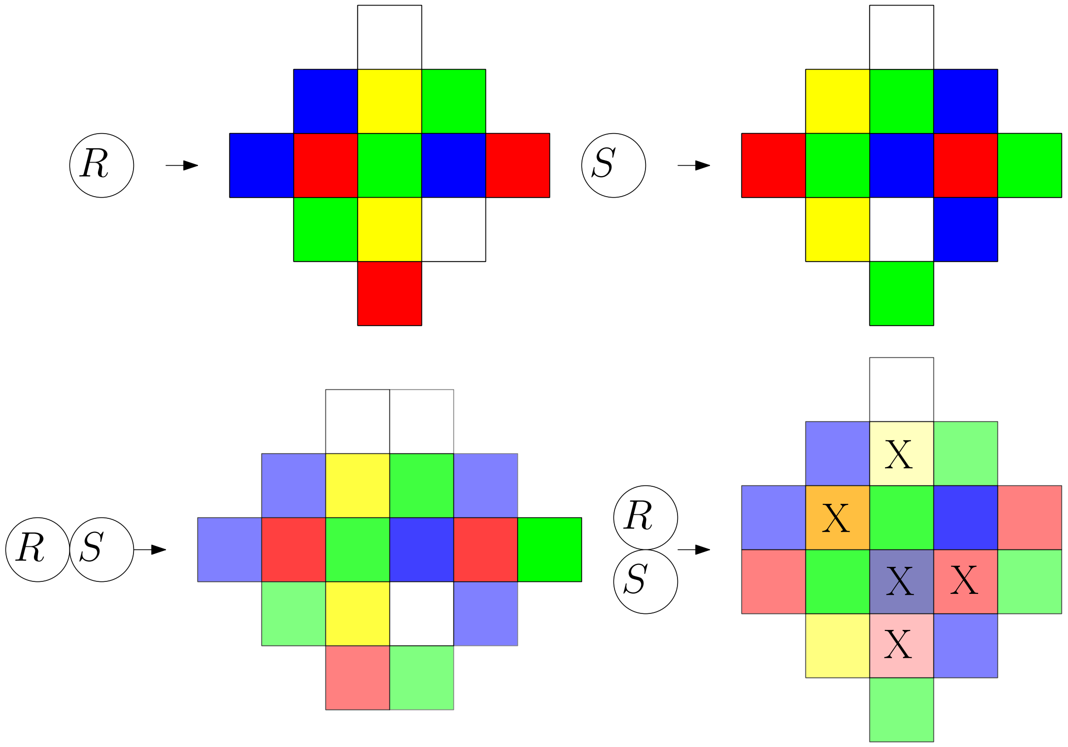

Figure 3.5: An example outlining how two rhombuses R and S may properly overlap (left) and when not (right). Note the in the right example, there are several conflicts where the colours assigned by R and S do not match.

$D((a,b),(x_2,y_2)) \leq r\}$ would be empty. An *assignment* of rhombuses $\mathcal{R}$ to the integer grid $\mathbb{Z}^2$ can be thought of as a complete colouring of $\mathbb{Z}^2$ using the rhombuses in $\mathcal{R}$ as the colours. Infromally an assignment is valid if and only if a rhombus is centred on every vertex in $\mathbb{Z}^2$ and every pair of rhombuses properly overlap.

Definition 20. *An* ***assignment*** *A of rhombuses from the set $\mathcal{R}$ to the integer grid $\mathbb{Z}^2$ is a mapping from $\mathcal{R}$ to $\mathbb{Z}^2$ such that $A : x, y \in \mathbb{Z} \mapsto R$ for every $x, y \in \mathbb{Z}$. Let $A(x,y) : \mathbb{Z}^2 \mapsto \mathcal{R}$ return the rhombus assigned to position $(x,y) \in \mathbb{Z}^2$. An assignment A is* valid *if and only if for every pair of positions $(x_1,y_1),(x_2,y_2) \in \mathbb{Z}^2$, the rhombus $A(x_1,y_1)$ properly overlaps the rhombus $A(x_2,y_2)$.*

Definition 21. *An assignment A from $\mathcal{R}$ to $\mathbb{Z}^2$ is* ***periodic*** *if there exists some pair of integers $a, b \in \mathbb{Z}$ such that for every $(x,y) \in \mathbb{Z}^2$, $A(x,y) = A(x \bmod a, y \bmod b)$.*

Problem 10. *The all-distinct r-discretised rhombus periodic complete assignment problem.*

Input:	A set $\mathcal{R}$ of r-radius digitised rhombuses
Question:	Does there exist a valid periodic assignment of $\mathcal{R}$ to $\mathbb{Z}^2$?

Lemma 2. *The all-distinct r-discretised rhombus periodic complete assignment problem is undecidable for any $r \geq 1$.*

Proof. This Lemma follows naturally from the periodic tiling problem with a 1-unique radius. Let $\mathcal{T}$ be a set of tiles. A set of rhombuses $\mathcal{R}(\mathcal{T})$ is constructed through a two step process. First, the set of colours $\mathcal{C}(\mathcal{T})$ is constructed such that every tile $t \in \mathcal{T}$ corresponds directly to the colour $c_t \in \mathcal{C}(\mathcal{T})$. To construct $\mathcal{R}(\mathcal{T})$, let $\mathbf{S}$ be the set of valid tilings with a 1-unique radius of the grid $\{(x, y) : x, y \in \mathbb{Z}, |x| + |y| \leq 1\}$ using the tiles from $\mathcal{T}$. For every tiling $s \in \mathbf{S}$, a rhombus R is constructed such that $R_{i,j}$ returns the colour corresponding to the tile at position (i, j) of s.

Given a valid assignment A from the set of rhombuses $\mathcal{R}(\mathcal{T})$ to $\mathbb{Z}^2$, a valid tiling with a 1-unique radius of $\mathcal{T}$ can be derived as follows. For every position $(x, y) \in \mathbb{Z}^2$, $A(x, y)$ is replaced with the tiling corresponding to $A(x, y)_{0,0}$. Note that as A is a valid assignment, every tile the is directly adjacent to (x, y) must be both unique and form a valid tiling. Therefore, the corresponding tiling must be a valid tiling with a 1-unique radius. Further if A is periodic the corresponding tiling T is also be periodic.

In the other direction, let T be a valid tiling with a 1-unique radius. To construct an assignment A from T, the position (x, y) of T is replaced with the rhombus corresponding to set of tiles surrounding (x, y). Following the above construction, such a rhombus must exist in $\mathcal{R}(\mathcal{T})$ for every position (x, y). Further, any two adjacent rhombuses must properly overlap following the construction. Therefore the corresponding assignment is valid. As before, if T is periodic then the corresponding assignment A is periodic. $\square$

Corollary 2. *The fixed period all-distinct r-discretised rhombus periodic complete assignment problem is NP-hard for any $r \geq 1$.*

Proof. Following the construction used in Lemma 2 and the NP-hardness of Proposition 7, it follows that the fixed period all-distinct r-discretised rhombus periodic complete assignment problem is NP-hard for any $r \geq 1$. $\square$

3.3 Pairwise Energy Minimisation Problem on $\mathbb{Z}^d$

With the undecidability of the all-distinct r-discretised rhombus periodic complete assignment problem established, the next step is to show how to reduce the all-distinct r-discretised rhombus periodic complete assignment problem to the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$. In this section the pairwise energy function is assumed to be a member of the 2-distance common minimal value class, $\mathcal{CMV}(2)$. It should be noted that these results require a potentially infinite amount of colours and size of unit cell. Despite this, the complexity of energy functions in the real world make the lack of any bounds on the number of colours still a useful model, particularly for the so-called "tiling" settings, where it is possible that even a finite number of ion species may be used to generate an infinite number of blocks in continuous space.

At a high level the reduction from the all-distinct r-discretised rhombus periodic complete assignment problem is done by encoding each rhombus as a colour, then tuning the parameters of the pairwise energy function so that a valid colouring of the grid corresponds to a valid assignment of rhombuses.

The main challenge of this encoding comes from the definition of the $\mathcal{CMV}(2)$ class. By the definition of $\mathcal{CMV}(2)$, for any energy function in $\mathcal{CMV}(2)$ the pairwise energy between vertices is determined solely by the distance between vertices, colour of each vertex, and some given set of parameters. This means that given two directly adjacent vertices v_i and v_j, coloured c_i and c_j respectively, the energy between v_i and v_j is the same irrespective of whether v_j is $East, North, West$ or $South$ of v_i. This leads to the primary challenge of encoding the r-discretised rhombus distinctly coloured periodic complete assignment problem into the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$.

For the 1-radius distinctly coloured rhombus assignment problem this constraint proves to be too much. Consider a pair of rhombuses $R_1, R_2 \in \mathcal{R}$ such that R_1 may properly overlap R_2 only when placed directly $East$ of R_1. Let $f(R_1, R_2, 1) = M$, and further, let $f(R, S, 1) \geq M$ for any other pair of rhombuses $R, S \in \mathcal{R}, (R, S) \neq (R_1, R_2)$. As the energy $f(R_1, R_2, 1) = M$, the energy of any pair of adjacent vertices coloured using R_1 and R_2 must be M, regardless of the orientation between the pair. Therefore a complete colouring of the grid using just R_1 and R_2 would have an average energy per vertex of $4 \cdot M$, despite not corresponding to a valid solution to the all-distinct r-discretised rhombus periodic complete assignment problem. An example is given is Figure 3.6. As such, a larger radius is needed to encode into the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$.

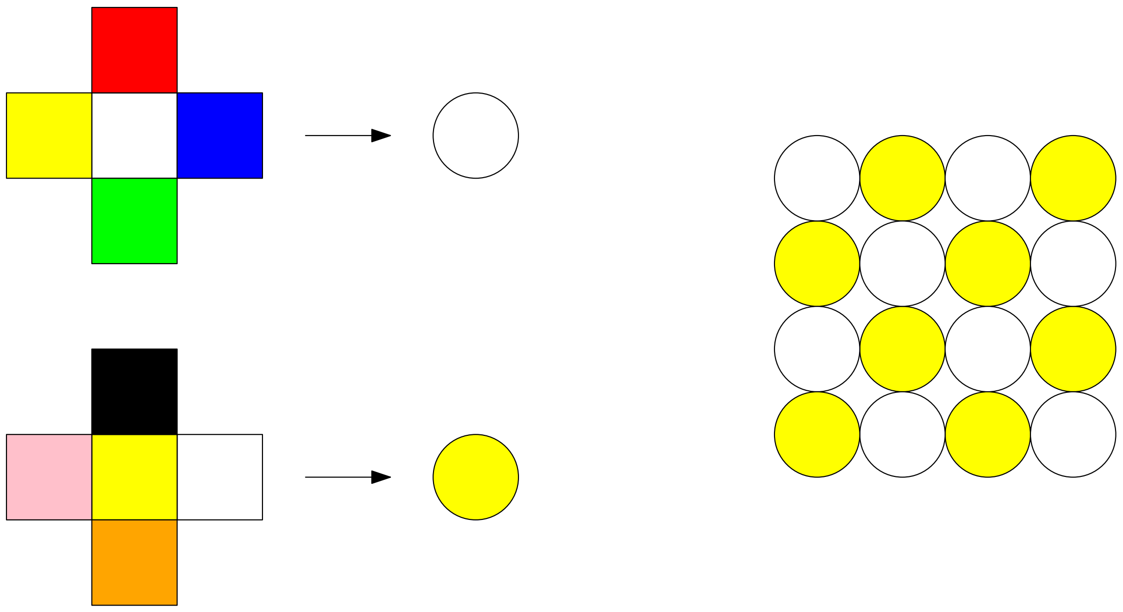

Figure 3.6: An example of a pair of rhombuses which may overlap each other when directly adjacent. Note that the colouring using these rhombuses would have an energy of $4m$ despite not corresponding to a valid assignment of rhombuses.

Construction. Let $\mathcal{R}$ be a set of distinctly coloured rhombuses. For every rhombus $r \in \mathcal{R}$ a colour C_r is added to the set of colours $\mathcal{C}(\mathcal{R})$. For each pair of colours C_i and C_j corresponding to the rhombuses i and j respectively, the energy $f(C_i, C_j, r)$ is set as follows:

1. If i and j properly overlap when i is centred at some position directly adjacent to j then $f(C_i, C_j, 1) = M$, and $f(C_i, C_j, r) > M$ for any $r > 1$.
2. If i and j properly overlap when i is centred at some position diagonally adjacent to j then $f(C_i, C_j, \sqrt{2}) = M$, and $f(C_i, C_j, r) > M$ for either $r = 1$ or $r = 2$.
3. If i and j properly overlap when i is centred at some position peripherally adjacent to j then $f(C_i, C_j, 2) = M$, and $f(C_i, C_j, r) > M$ for $r < 2$.
4. Otherwise $f(c_i, C_j, r) > M$ for any distance r.

Note that by the definition of $\mathcal{CMV}(2)$, $f(C_i, C_j, r) = 0$ for any $r > 2$ and pair $C_i, C_j \in \mathcal{C}(\mathcal{R})$. Using this construction, Lemma 3 shows that a valid assignment of $\mathcal{R}$ to $\mathbb{Z}^2$ can be used to construct a valid colouring of $\mathbb{Z}^2$ using $\mathcal{C}(\mathcal{R})$. Lemma 4 complements Lemma 3 by showing that given such a colouring of $\mathbb{Z}^2$, there must exist a valid assignment of rhombuses to $\mathbb{Z}^2$.

Lemma 3. *Let A be a valid assignment of the set of distinctly coloured 2-radius rhombuses $\mathcal{R}$ to $\mathbb{Z}^2$ with a period of $n_1 \times n_2$. Given such an assignment there exists a periodic colouring of $\mathbb{Z}^2$ using the set of colours $\mathcal{C}(\mathcal{R})$ with an average energy per vertex of $12 \cdot M$.*

Proof. Let U be a unit cell of dimensions $n_1 \times n_2$ such that $U(x, y)$ returns the colour $C_{A(x,y)} \in \mathcal{C}(\mathcal{R})$. Let (x_1, y_1) and (x_2, y_2) be a pair of positions in $\mathbb{Z}^2$ at a distance of no more than 2 apart. Following the above construction, $f(C_{A(x_1,y_1)}, C_{A(x_2,y_2)}, D((x_1, y_1), (x_2, y_2)))$ is set to M provided $A(x_1, y_1)$ properly overlaps $A(x_2, y_2)$ at a distance of $D((x_1, y_1), (x_2, y_2))$. As A is a valid assignment, every pair of vertices within a distance of 2 must properly overlap, hence $f(C_{A(x_1,y_1)}, C_{A(x_2,y_2)}, D((x_1, y_1), (x_2, y_2))) = M$. Therefore, each vertex will interact with 12 other vertices, giving an average energy per vertex of $12 \cdot M$ satisfying the original statement. □

Lemma 4. *Let U be a unit cell of dimensions $n_1 \times n_2$ colouring $\mathbb{Z}^2$ with the colour set $\mathcal{C}(\mathcal{R})$ such that the average energy per vertex is $12 \cdot M$. Given such a unit cell there exists a valid assignment of the set of distinctly coloured 2-radius rhombuses $\mathcal{R}$ to $\mathbb{Z}^2$.*

Proof. Let A be an assignment constructed from U such that $A(x, y)$ returns the rhombus corresponding to $U(x, y)$. To show that A is a valid assignment, let v be some vertex in $\mathbb{Z}^2$, and let $v_{North}, v_{South}, v_{East}$ and v_{West} be the vertices immediately to the $North, South, East$ and $West$ of v respectively. Further, let R_v be the rhombus assigned to v, R_{North} be the rhombus assigned to v_{North}, R_{South} be the rhombus assigned to v_{South}, R_{East} be rhombus assigned to v_{East}, and R_{West} be rhombus assigned to v_{West} As the interaction energy between v and each of $v_{North}, v_{East}, v_{South}$ and v_{West} in the unit cell must be M, $R_{North}, R_{East}, R_{South}$ and R_{West} must all be rhombuses that may properly overlap R_v when centred directly adjacent to R_v. Further as the interaction between v_i and v_j in the unit cell must be M for every $i, j \in \{East, North, West, South\}$, the corresponding rhombuses R_i and R_j must be distinct. Therefore, the set of rhombuses $R_{North}, R_{East}, R_{South}$ and R_{West} must correspond to a set of rhombuses that may mutually overlap R_v. For any such set of 4 rhombuses to properly overlap, note that the central tile of each rhombus must be coloured uniquely relative to each other rhombus. Further, as there are only four possible colours the central tile for each rhombus may have and still be directly adjacent to R_v, we may assume without loss of generality that the colour of the central position R_i corresponds to the colour at position $i \in \{East, North, West, South\}$ relative to the centre of R_v, i.e. that the central position of R_{East} corresponds to the colour at position $(R_v)_{(1,0)}$.

Let $v_{North-East}, v_{North-West}, v_{South-West}$ and $v_{South-East}$ be the set of vertices diagonally adjacent to v. Note that the rhombus $R_{North-East}$ must properly overlap both R_{East} and R_{West} when placed directly adjacently. The only possible such rhombus is one for which the central position is coloured with $(R_v)_{(1,1)}$. By repeating this for each diagonally adjacent vertex, we see that the set of diagonally vertices must be assigned a set of rhombuses such that the each overlaps with every other vertex either diagonally or directly adjacent to v.

Finally, let $v_{East-East}, v_{North-North}, v_{West-West}$ and $v_{South-South}$ be the set of vertices peripherally adjacent to v such that $v_{East-East}$ denotes the vertex directly *East* of v_{East}. As the vertex $v_{East-East}$ is at a distance of 2 from v, the central position of $R_{East-East}$ must be coloured with one of the four colours that are allowed at a distance of 2 from the centre of R_v. Further, as $v_{East-East}$ is at a distance of 1 from v_{East}, the central position of $R_{East-East}$ must be coloured with one of the four colours that are allowed at a distance of 1 from the centre of R_{East}. As every position within each rhombus must be uniquely coloured, there is only a single possible colour that satisfies both of these conditions, being the colour $(R_v)_{(2,0)}$. Repeating this argument for each other peripheral vertex, the peripheral vertices must all be coloured by some rhombus that may properly overlap with R_v when centred on the same point on the grid.

These same arguments may be extended to every other tile in the overlap between any pair of rhombuses. Therefore every rhombus within a radius of 2 of v must properly overlap with R_v. Hence given such a unit cell, a valid periodic assignment of rhombuses from $\mathcal{R}$ may be derived by replacing each colour with the corresponding rhombus. □

Theorem 1. *The pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$ is undecidable for any pairwise energy function in the 2-distance common minimal value class ($\mathcal{CMV}(2)$).*

Proof. Following Lemmas 3 and 4, there exists a valid colouring of $\mathbb{Z}^2$ with an average energy per vertex of $12M$ of $\mathcal{C}(\mathcal{R})$ if and only if there exists a valid assignment of $\mathcal{R}$ to $\mathbb{Z}^2$. As the all-distinct r-discretised rhombus periodic complete assignment problem is undecidable, the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$ is undecidable. □

Corollary 3. *The Charge-Neutral pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$ is undecidable for the Buckingham-Coulomb Potential.*

Proof. Following the construction used by Theorem 1, the pairwise energy minimisation problem over $\mathcal{C}$ on $\mathbb{Z}^d$ is undecidable for the Buckingham-Coulomb Potential. Following Proposition 1, the construction of Buckingham-Coulomb such that it fits into the class of $\mathcal{CMV}(2)$ allows two directly adjacent vertices to have an energy of -1 if and only if they have opposing charges. Therefore, the construction used by Theorem 1 for the version of the Buckingham-Coulomb potential presented in Proposition 1 also admits a charge neutral unit cell. This may further be extended to include an additional colour c_0 to represent empty space such that the interaction between c_0 and any other colour in $\mathcal{C}$ is 0. Note that as the construction above requires every position to be coloured with a vertex having an interaction of -1 with every neighbour, there can be no position coloured c_0. $\square$

Chapter 4

Hardness of Crystal Structure Prediction

This chapter considers the two different abstractions of CSP presented in Section 2.1. Section 4.1 provides hardness results for the fixed period pairwise energy minimisation problem. This focuses on showing NP-hardness for the charge neutral case for 1 or more dimensions, as well providing a parameterised algorithm for the one dimensional case, parameterised by the cut off distance. Section 4.2 provides hardness results for the deletion based approach. This section shows NP-hardness for the deletion based problems presented in Section 2.1.4. Table 4.1 provides a summary of the main results presented in this chapter.

4.1 Pairwise Energy Minimisation for the Fixed-Period Grid Colouring Problem

In this section we analyse the complexity of the fixed period pairwise energy minimisation problem and the charge-neutral fixed period pairwise energy minimisation problem for the one dimensional case. This section is split into two parts. Section 4.1.1 shows that the one dimensional case of the charge-neutral fixed period pairwise energy minimisation problem is NP-hard to both solve and approximate within any positive factor. Section 4.1.2 provides a parameterised algorithm for the fixed period pairwise energy minimisation problem in 1D, parameterised by the cut-off distance of the energy function.

Theorem	Summary	Setting
Theorem 2	NP-hardness to solve within any positive factor by reduction from the independent set problem.	The charge-neutral fixed period pairwise energy minimisation problem for the Buckingham-Coulomb potential.
Theorem 3	An $O(n^3 \cdot q^{3(d+1)})$ time algorithm where q is the number of colours and d the cutoff distance.	The 1 dimensional fixed period pairwise energy minimisation problem for any energy function with a cut off distance of d (including the class $\mathcal{CMV}$ (d)) and unit cell of size n.
Theorem 4	NP-Completeness by reduction from the clique problem.	k-CHARGE REMOVAL, $k_{\geq}$-CHARGE REMOVAL, and MINIMAL-$k_{\geq}$-CHARGE REMOVAL, under any energy function in $\mathcal{CF}$, vertices of charge $\pm c$ for a given c and an unbounded number of ion species.
Theorem 5	NP-Completeness by extension of Theorem 4.	k-CHARGE REMOVAL, $k_{\geq}$-CHARGE REMOVAL, and MINIMAL-$k_{\geq}$-CHARGE REMOVAL, under any energy function in $\mathcal{CF}$, any bounded set of charges and an unbounded number of ion species.
Theorem 6	Non-approximability in Polynomial time for any factor of $n^{1-\epsilon}$, for $\epsilon > 0$	$k_{\geq}$-CHARGE REMOVAL, for any energy function in $\mathcal{CF}$.
Theorem 7	Reduction to max-weight-k-clique.	k-CHARGE REMOVAL or MINIMAL-$k_{\geq}$-CHARGE REMOVAL under any computable energy function, vertices of charge $\pm c$ for a given c, and a unbounded number of ion species.
Theorem 8	NP-Completeness by reduction from independent set on penny graphs.	k-CHARGE REMOVAL, $k_{\geq}$-CHARGE REMOVAL, and MINIMAL-$k_{\geq}$-CHARGE REMOVAL, under the Buckingham-Coulomb potential energy function, vertices of charge ± 1, and two species of ion.
Theorem 9	NP-Completeness by reduction from the knapsack problem.	Minimal-$k_{\geq}$-Charge Removal and $k_{\geq}$-Charge Removal, under the Coulomb potential energy function, unbounded number of vertex weights and unbounded number of ion species.

Table 4.1: Summary of the main results from this chapter.

4.1.1 Buckingham-Coulomb Potential with Charge Neutrality

In this section we consider only the Buckingham-Coulomb potential in 1D, showing that the charge-neutral fixed period pairwise energy minimisation problem is NP hard both to solve and to approximate within any positive factor. It should be noted that NP-hardness can be shown for any function in $\mathcal{CMV}(2)$ for any grid of at least 2 dimensions by reduction from the fixed period k-unique tiling problem by way of Corollary 3, however this reduction does not provide any hardness on the problem of approximation.

The reduction presented in this section uses the *k-independent set problem* to show that it is NP-hard to approximate the optimal solution to the charge neutral fixed period pairwise energy minimisation problem for a set of n ion species within any factor greater than 0. The k-independent set problem asks, for a graph $G = (E, V)$, if there exists set S of k vertices from V such that $(i, j) \notin E$ for every $i, j \in S$. At a high level, the reduction works by representing each vertex with an ion. The energy between ions is set to have a high penalty cost penalty between ions representing adjacent vertices, and a negative cost between other ions.

Charge-Neutral Fixed Period Pairwise Energy Minimisation Problem Instance Construction. Given an instance of the k-independent set problem for the graph $G = (E, V)$, a positive ion species P_i is created for every vertex $i \in V$ with a charge of $+1$. Additionally, a negative ion species N_k with a charge of $-k$ is created. In this setting, the unit cell is fixed to a size of $k+1$ with the grid $\mathbb{Z}$. This effectively corresponds to a word of $k+1$ symbols. Further for the cell to be charge balanced it must either be empty, or have a single ion of the species N_k, and k positive ions. Due to the translational symmetry of this setting, it is assumed that the ion N_k is located at the centre of the unit cell.

In general, given two ions i and j where $i \neq j$, the energy between P_i and P_j is set to either be 0 if $(i, j) \notin E$, or greater than some penalty value $c > 0$ if $(i, j) \in E$ or $i = j$. Further the energy between P_i and N_k is set to no more than some negative value g. The energy between each pair of ions is set as follows.

Between P_i and N_k. To ensure that the energy between P_i and N_k is no more than g, let the cutoff distance be $\lceil \frac{k+1}{2} \rceil$. Using this distance, let $A_{i,N_k} = 0$. This gives the interaction energy between P_i and N_k as $-\frac{C_{i,N_k}}{r^6} - \frac{1}{r}$. Therefore by setting C_{i,N_k} to $g \cdot 8(k+1)^3$, the interaction energy at any distance less than or equal to $\sqrt{2(k+1)}$ is no more than g. Note that the cutoff distance of $\lceil \frac{k+1}{2} \rceil$ ensures that N_k interacts with between 1 and 4 ions in the neighbouring unit cells.

Between P_i and P_j where $(i,j) \in E$ or $i = j$. Given two positive ions representing adjacent vertices in G, the goal is to set the energy to have a large penalty at any distance within the unit cell. To this end, let the cutoff distance between P_i and P_j be $d_{ij} = \lceil \sqrt{2k+1} \rceil$. Note that this cutoff distance ensures that P_i interacts with P_j at any position with the unit cell. By setting $A_{ij} = c, B_{ij} = 0$ and $C_{ij} = 0$, the energy between P_i and P_j at a distance of r_{ij} is $c + \frac{1}{r_{ij}}$.

Between P_i and P_j where $(i,j) \notin E$ and $i \neq j$. Given two positive ions representing non-adjacent vertices, the pairwise interaction energy is set to be 0 at any distance by either setting the cut-off distance to 0, ignoring the interaction, or setting the cutoff distance to 1 and setting $A_{ij} = B_{ij} = 0$, and $C_{ij} = 1$ in order to counteract the Coulomb potential.

Between N_k and N_k. As in the previous case, given two negative ions the pairwise interaction energy is set to 0.

Theorem 2. *The charge-neutral fixed period pairwise energy minimisation problem can not be approximated within any factor greater than* 0 *in polynomial time unless* $P = NP$.

Proof. Using the construction above, observe that the unit cell with minimal energy must either have k positive ions and 1 negative ions, or be empty. Consider a k-independent set instance such that there exists an independent set of size k in the graph G. As a simple lower bound on the interaction energy, let each of the k positive ions be at a distance of 1 from N_k. Note that at this distance each positive ion only interacts with N_k once. In this case the total energy of the unit cell is $k \cdot g \cdot 8(k+1)^3 \approx g \cdot 8(k+1)^4$ with the average energy per ion being $g \cdot 8(k+1)^3$. If there is no such independent set, there must be at least one pairwise interaction with an energy of at least c, setting the average interaction energy to $g \cdot 8(k+1)^3 + \frac{2C}{k+1}$. Hence by setting $c \geq |-g \cdot 8 \cdot (k+1)^4|$, any approximation algorithm that can solve the charge-neutral fixed period pairwise energy minimisation problem within any factor greater than 0 would also be able to solve the k-independent set problem. Therefore the charge-neutral fixed period pairwise energy minimisation problem can not be approximated within a factor greater than 0 within polynomial time unless $P = NP$. $\square$

4.1.2 A Parameterised Algorithm for the One Dimensional Fixed Period Pairwise Energy Minimisation Problem

Following the hardness results for the charge balanced one dimensional case of the fixed period pairwise energy minimisation problem, the obvious question to ask is if there exists any parameterised algorithm for solving the fixed period pairwise energy minimisation problem on $\mathbb{Z}$. In this section we answer this question in the affirmative with an algorithm that can solve the fixed period pairwise energy minimisation problem in $O(n^3 \cdot q^{3 \cdot d})$ where d is the *cut off distance*, the distance for which $f(c_1, c_2, r, \overline{\theta}) = 0$ for any $r \geq d$. For notation, the d-cut off distance property, $\mathcal{CF}(d)$, is introduced.

Definition 22. *A function $f(c_1, c_2, r, \theta) : c_1, c_2 \in \mathcal{C}, r \in \mathbb{R}, \theta \in \mathbb{R}^m \mapsto \mathbb{R}$ has the **d-distance cut-off property** $\mathcal{CF}(d)$ property if $f(c_1, c_2, r, \theta) = 0$ for every $r \geq d$.*

Note that $\mathcal{CF}(d)$ is a generalisation of $\mathcal{CMV}(d)$, with any function with the $\mathcal{CMV}(d)$ property also having the $\mathcal{CF}(d)$ property. As such the functions provided in this section can be applied to the any function in $\mathcal{CMV}(d)$, such as the Buckingham-Coulomb or Ising models. In particular, this may be useful to "tiling" approaches to solving CSP such as MC-EMMA where the problem is reduced to that of ordering blocks in a one dimensional setting.

High Level Overview: At a high level, the idea is to construct a directed acyclic graph containing q^{d+1} source and sink nodes corresponding to every possible structure of length $d+1$. This graph is designed such that the lowest weight path from the source to the sink corresponds to the lowest energy colouring for the fixed period pairwise energy minimisation problem instance. This is done by constructing a set of q^{d+1} vectors corresponding to every possible way of colouring d consecutive vertices from a set of q colours on the 1-dimensional integer grid. To make a graph, a set of n-layers is constructed, with each layer containing a vertex for each vector. Each vertex is connected only to vertices in the next layer, with the constraint that the vertex v in layer i corresponding to the vector $(l_1, l_2, \dots, l_{d+1})$ is connected only to vertices in layer $i+1$ corresponding to vectors of the form $(l_2, l_3, \dots, l_{d+1}, x)$, for some colour $x \in \mathcal{C}$.

Construction: Given an instance of the fixed period pairwise energy minimisation problem for the one dimensional grid with dimension n, and set of colours $\mathcal{C}$, a graph G is constructed. Let $\mathcal{V}(d, \mathcal{C}) = \{(x_1, x_2, \dots, x_{d+1}) | x_1, x_2, \dots, x_{d+1} \in \mathcal{C}\}$. For notation given $l \in \mathcal{V}(d, \mathcal{C})$, l_i is used to denote the colour of the i^{th} position of l, i.e. given $l = (1, 2, 1, 2)$,

$l_2 = 2$ while $l_3 = 1$. For every $i \in [n]$ and $l \in \mathcal{V}(d, \mathcal{C})$ the vertex $v_{i,l}$ is constructed and added to the set V of vertices. Given a pair vertices $v_{i,l}, v_{j,k} \in V$, the edge $v_{i,l}, v_{j,k}$ is added to the set of edges E if and only if $i+1 = j$ and $l_2, l_3, \dots, l_{d+1} = k_1, k_2, \dots, k_d$. The weight of $(v_{i,l}, v_{j,k})$, denoted $w(v_{i,l}, v_{j,k})$, is equal to $\sum\limits_{i=1}^{d} f(k_1, k_{i+1}, i, \overline{\theta})$. This means that each edge $(v_{i,l}, v_{j,k})$ corresponds to the pairwise interaction energy between k_1 and each subsequent vertex in k. In order to account for the energy from the first vector, an additional set of q^{d+1} vertices labelled v_l for every $l \in \mathcal{V}(d, \mathcal{C})$. The vertex v_l has only a single edge connecting it to $v_{1,l}$, weighted as before. Hence by constructing a path of length n starting at some vertex v_l and ending at the vertex $v_{n,l}$ the weight of the path with correspond to the total pairwise energy of the corresponding unit cell. Thus by finding such a path with minimum energy the solution to the fixed period pairwise energy minimisation problem may be found. Using the above construction, the solution to the fixed period pairwise energy minimisation problem instance is found by determining the shortest path from each vertex of the form v_l to the vertex $v_{n,l}$ for every $l \in \mathcal{V}(d, \mathcal{C})$.

Theorem 3. *There exists an algorithm to solve the fixed period pairwise energy minimisation problem in $O(n^3 \cdot q^{3\cdot(d+1)})$ time for any function with the $\mathcal{CF}(d)$ property.*

Proof. Using the above construction, the solution to the corresponding fixed period pairwise energy minimisation problem instance can be found by using all simple all pairs shortest path algorithm. Using a classic algorithm for finding the shortest path between all pairs such as the Floyd–Warshall algorithm, the paths may be found in $O(|V|^3)$ time. Note that the number of vertices equals $(n+1) \cdot q^{d+1}$ giving a total complexity of $O((n+1)^3 \cdot q^{3\cdot(d+1)}) \approx O(n^3 \cdot q^{3\cdot(d+1)})$. □

4.2 Energy Minimisation of Structures under the Deletion Operation

4.2.1 NP-Hardness for an Unbounded Number of Ion Species

This subsection focuses on the class of potential functions $\mathcal{CF}$. It is assumed that the energy function for all cases is an arbitrary function in $\mathcal{CF}$ for which the parameters required by the ions to result in the energy from their pairwise interaction to be any arbitrary a are known. NP-completeness for k-CHARGE REMOVAL as well as for the generalisations to MINIMAL-$k_{\geq}$-CHARGE REMOVAL and $k_{\geq}$-CHARGE REMOVAL is shown when there are bounds on value of the charges (either quantity of charges, or the maximum value). Further, these problems are shown to hard to approximate within a factor of $n^{1-\epsilon}$ for any $\epsilon \geq 0$ in polynomial time for bounded values of charges, where n is the number of ions in the unit cell. It may be noted that in the case the charges are not bounded, MINIMAL-$k_{\geq}$-CHARGE REMOVAL remains NP-hard, however as it is not in NP it is not NP-complete. Along with the hardness results we provide a polynomial time reduction from both k-CHARGE REMOVAL and MINIMAL-$k_{\geq}$-CHARGE REMOVAL to max-weight-k-clique, under the restriction that all vertices have a charges of $\pm c$ for some non zero $c \in \mathbb{Z}$.

Theorem 4. *k-CHARGE REMOVAL, MINIMAL-$k_{\geq}$-CHARGE REMOVAL and $k_{\geq}$-CHARGE REMOVAL are NP-Complete for any energy function in $\mathcal{F}$ for vertices with charges of $\pm c$, for any natural number c.*

Proof. k-CHARGE REMOVAL and $k_{\geq}$-CHARGE REMOVAL are in NP by Proposition 4, and as the vertex charges are bounded, MINIMAL-$k_{\geq}$-CHARGE REMOVAL is in NP by Lemma 1. Hardness is established via a reduction from the k-Clique problem. This is shown by reduction to k-CHARGE REMOVAL, noting that any satisfying solution to k-CHARGE REMOVAL also satisfies $k_{\geq}$-CHARGE REMOVAL and MINIMAL-$k_{\geq}$-CHARGE REMOVAL.

In the k-Clique problem, henceforth k-CLIQUE, the input is a graph, G, and a natural number, k. The goal is to find a clique of size k in G, or report that no such clique exists. A clique is a set of vertices in a graph such that all vertices in the set are adjacent to each other.

Given an instance of k-CLIQUE, $I = (G, k) = ((E, V), k)$, where $n = |V|$, an instance, I', of k-CHARGE REMOVAL is constructed as follows. A unit cell of arbitrary size is chosen. Within this cell $2n$ unique positions are chosen at arbitrary points in the unit cell. In the first n positive ions are placed and in the last n negative ions are placed. Each ion has its own unique specie. Every vertex $v_i \in V$ corresponds to two ions, i^+ and i^- with charges c and $-c$ respectively. For two ions i and j associated with v_i and v_j respectively the parameters $\overline{\theta}(i, j)$ are set so as to satisfy the following:

$$f(\overline{\theta}(i,j), D(i,j)) = \begin{cases} -1 & v_i = v_j \text{ or } (v_i, v_j) \in E \\ P & \text{otherwise.} \end{cases}$$

Where $P \in \mathbb{R}$ is some arbitrarily high penalty value that may be treated as effectively being equal to ∞ for any practical purpose. The definition of $\mathcal{CF}$ guarantees that there exists parameters satisfying these conditions irrespective of the positions, and thus the distance r_{ij}, of the ions. Note that there are $k(2k-1)$ edges in a clique of size $2k$. Let $g = k(2k-1)$ and let $k' = n - k$. To remove k' positive and k' negative ions $c \cdot k'$ vertices must be removed.

The corresponding crystal graph $G' = (V', E')$ is constructed as described in the preliminaries. Let the vertices $v_i^+, v_i^- \in V'$ represent the ions corresponding to $v_i \in V$. $v_i^\pm$ is used to denote either v_i^+ or v_i^-, where the charge of the vertex doesn't matter i.e. we are only concerned with the vertex in G that $v_i^\pm$ corresponds to. From the definition of the energy function, $wt(v_i^\pm, v_j^\pm) = -1$ if $i = j$ or $(v_i, v_j) \in E$, and P otherwise.

We claim that I is satisfiable if and only if I' is satisfiable. First consider the case that I is satisfiable. In this case $c \cdot k'$ vertices may be removed from I', leaving only the vertices corresponding to the clique in I, denoted A. As all vertices in A correspond to adjacent vertices in G, the energy is -1 multiplied by the number of edges, giving a total energy of $-k(2k-1)$, satisfying the k-CHARGE REMOVAL instance. Conversely if there does not exist a clique of size k in G then any subset of charges $A \subseteq V'$ of cardinality k clearly must contain at least one edge with a weight of P, making I unsatisfiable. □

This may be extended to other graph problems relatively easily. One example of this would be the MAX-WEIGHT K-CLIQUE problem. The MAX-WEIGHT K-CLIQUE problem takes as input a weighted graph G, a natural number k, and a goal value g. The problem is to report if a clique of size k exists such that the sum of the weights of the edges is at least g. Using the above construction, a crystal graph G' may be created from G. From this the weights on the edges may be adjust as follows:

$$f(\overline{\theta}(i,j), D(i,j)) = \begin{cases} -wt(v_i, v_j) & v_i \neq v_j \text{ and } (v_i, v_j) \in E \\ -c & v_i = v_j \\ P & \text{otherwise.} \end{cases}$$

Where $P \in \mathbb{R}$ is some arbitrary large penalty value, $wt(v_i, v_j)$ denotes the energy between vertices v_i and v_j in G and c is some constant such that $\nexists (v_i, v_j) \in E$ where $wt(v_i, v_j) \geq c$. The goal value for the k-CHARGE REMOVAL instance is chosen as $-(k \cdot c + v)$. The correctness of this reduction follows from the arguments in Theorem 4.

Theorem 5. k-CHARGE REMOVAL *remains NP-hard for a given set of allowed charges with unique magnitude and an energy function within $\mathcal{CF}$.*

Proof. The construction of Theorem 4 may be extended to the case the set of vertices is limited to any set of allowed charges. Two charges are chosen from this set, c and d where $|c| > |d|$ and $c \cdot d < 0$ such that the difference between the absolute value of the charges, $|c| - |d|$, is minimised. The same steps as in Theorem 4 are followed for the construction to get an initial crystal graph $G = (V, E)$ and k'. Note that I has a *deficiency* of $n(|c| - |d|)$, where n is the number of ions in the initial construction, meaning that some set of vertices must be added to make the cell neutral. To handle the deficiency two sets of dummy vertices with charges of c and d are created.

The first set is to deal with the deficiency that would be left from a clique of size k. To construct these, a natural number t is chosen such that there exists a pair of natural numbers t_c and t_d where $t_c|c| = t$ and $t_d|d| = k(|c| - |d|) + t$. Using these, t_c vertices with a charge of c and t_d vertices with a charge of c and t_d with a weight of d are added. From the definition of $\mathcal{CF}$, the energy between them and all ions in G and between each other is set as 0.

The second set of dummy vertices are to counteract the overall deficiency in the unit cell. A natural number u is chosen such that there exists a pair of natural numbers u_c and u_d, where $u = |c|(u_c + t_c)$ and $u + n(|c| - |d|) = |d|(u_d + t_d)$. u_c vertices with a charge of c and u_d vertices with a charge of d are added. The potential energy between between them and all other vertices, including the set of previously added dummy vertices, is P.

To ensure that the optimal set of ions to be left with is a clique of size k as well as all of the dummy vertices added in the first step, the following is done. The goal energy remains the same as from Theorem 4. Observe that the only way to achieve this is to leave vertices corresponding to a clique of size at least k. As there are $c(n + u_c + t_c)$ vertices for one set, and the goal is to be left with $c(k + t_c)$, a value k' is chosen to remove as $c(n + u_c - k)$. In the case that exactly k'-charges are removed, either the dummy vertices or some other vertices corresponding to a clique of size greater than k, ensuring the set remains neutral is left. In the at-least-k' case, some dummy vertices may also be removed provided the cell remains neutral.

From the arguments in Theorem 4 this is sufficient to ensure the new instance is satisfiable if and only if the original k-CLIQUE instance is. Therefore these problems are NP-hard, even in the case that there are distinct charges c and d, $|c| \neq |d|$. □

Theorem 6. *For any $\epsilon > 0$, $k_{\geq}$-CHARGE REMOVAL for $k = 0$ cannot be approximated within a factor of $n^{1-\epsilon}$, where n is the number of ions, in polynomial time unless $P = NP$.*

Proof. This result follows from the results of Håstad [46], who showed an approximation bound of $n^{1-\epsilon}$ for Max-Clique. Using the reductions from Theorems 4 and 5 let the minimum energy after a removal of at least 0 vertices be e. Note that this corresponds to the lowest potential energy of the instance. From the reductions, it is clear the $e = -k$, where k is the size of the clique, or P if the remaining ions do not correspond to a clique. Note that as P may be arbitrarily large, given an approximation algorithm for any instance of $k_{\geq}$-CHARGE REMOVAL with $k = 0$ an approximation algorithm for Max-Clique may be derived that approximates the instance of Max-Clique to the same factor as it approximates the $k_{\geq}$-CHARGE REMOVAL instance. Therefore any bounds on the approximation of Max-Clique must also apply to this problem, hence $k_{\geq}$-CHARGE REMOVAL can not be approximated within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$ within polynomial time unless $P = NP$. □

While there are simple reductions to other NP-Complete problems such as Integer Programming, embedding this problem into many classical problems is made difficult due to the problem of maintaining the neutrality of the unit cell. To this end, Theorem 7 provides a polynomial time reduction to show how a restricted version of k-CHARGE REMOVAL may be embedded into MAX-WEIGHT K-CLIQUE.

Theorem 7. *k-CHARGE REMOVAL can be reduced to MAX-WEIGHT K-CLIQUE in polynomial time, under the restriction that vertices are limited to charges of $\pm c$ and the energy function is computable within polynomial time.*

Proof. Note that, given charges of $\pm c$, a valid solution to MINIMAL-$k_{\geq}$-CHARGE REMOVAL is either valid for k-CHARGE REMOVAL, or there is no valid solutions to k-CHARGE REMOVAL. Taking as input an instance of MINIMAL-$k_{\geq}$-CHARGE REMOVAL with charges of $\pm c$ with the corresponding crystal graph, it is claimed that this instance may be represented as an instance of the weighted generalisation of k-CLIQUE.

In weighted k-clique, denoted WEIGHTED-K-CLIQUE, the input is a weighted graph, a goal value v, and a natural number k. An instance of WEIGHTED-K-CLIQUE is satisfiable if and only if there exist a clique of size k such the sums of the weight of the edges in the clique is at least v.

Given an instance of MINIMAL-$k_{\geq}$-CHARGE REMOVAL $I = (G, k) = ((V, E), k)$, an instance I' of WEIGHTED-K-CLIQUE is created as follows. A value k' is chosen as $\frac{|V^+|-k}{c}$ rounded down to the nearest natural number. The reason for this choice is to ensure that

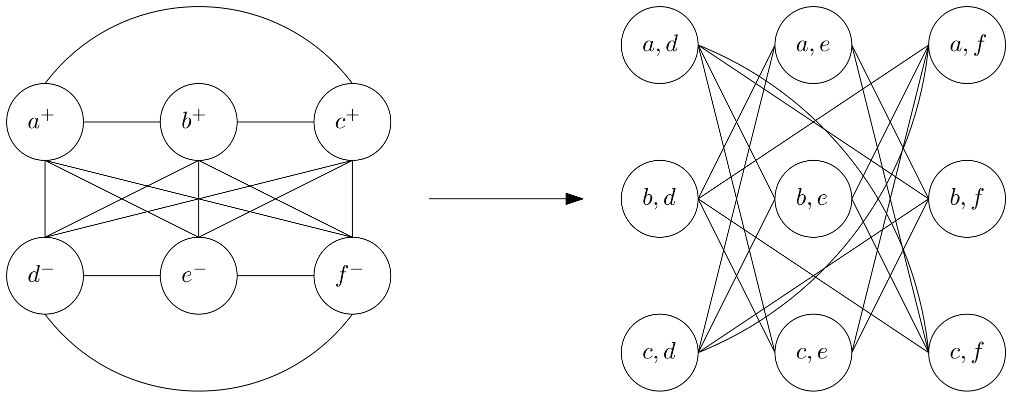

Figure 4.1: Example of the construction from $k_{\geq}$-CHARGE REMOVAL to clique. Note that vertices a, b, and c have a positive weight, while d, e, and f have a negative weight. Also note that any clique of size 3 corresponds to the original graph.

the optimal clique has size equal to the number of vertices left after removing k charges. Note that if $(|V^+|-k)(\mod c) \not\equiv 0$, then there is no valid solution to k-CHARGE REMOVAL, however there may still be some valid solution to MINIMAL-$k_{\geq}$-CHARGE REMOVAL. A new graph $G' = (V', E')$ is created which is initially empty. For each pair of vertices with different charges a new associated vertex in V' is created. An edge is created between each new vertex if and only if the corresponding charges are all unique, i.e. given the set of charges $V^+ = (v_i, v_j)$, and $v^- = (v_k, v_l)$ an edge would be placed between the new vertex representing (v_i, v_k) and the one representing (v_j, v_l), but not from either to the vertex representing (v_i, v_l). Give two connected vertices corresponding to charges (v_i, v_k) and (v_j, v_l) the edged is assigned a value of $-(f(\overline{\theta}(i,j), D(i,j)) + f(\overline{\theta}(i,l), D(i,l)) + f(\overline{\theta}(j,k), D(j,k)) + f(\overline{\theta}(k,l), D(k,l)) + \frac{f(\overline{\theta}(i,j),D(i,j))+f(\overline{\theta}(j,l),D(j,l))}{k'-1})$ to the edge between them. The intuition behind this is for the edge to maintain the weights of the edges in G. $\frac{f(\overline{\theta}(i,k),D(i,k))+f(\overline{\theta}(j,l),D(j,l))}{k'-1}$ is added to this so that within a clique of size k', the edge between the two vertices is fully represented. An example of this construction is shown in Figure 4.1, omitting weights for legibility.

It is now be claimed that any clique of size m corresponds to a neutrally weighted subset $A \subseteq V$, where $\left|\sum_{v_i \in A^+} wt(v_i)\right| = mc$. This is shown by noting that vertices are only connected if they do not represent a common vertex. As such a clique of size m must contain m unique positively weighted and m unique negatively charged vertices for the corresponding vertices to be connected as a clique. Therefore by selecting any clique of size k' in this graph, there is a valid structure left with exactly k' unique positively weighted and k' unique negatively charged vertices. From the definition of k' this corresponds to an subgraph of G after a minimal removal of k.

It may now be claimed that a maximum weight clique of size k' corresponds to the best subset of ions after a k Charge Removal. Note that given a clique with total weight w corresponds to a set of ions with total energy $-w$. It is a straightforward extension to see that a maximum weight clique corresponds to a minimum energy subset of ions. This is seen by noting that by choosing k' as the size of the clique, the corresponding arrangement $A \subseteq V$ has $\left|\sum_{v_i \in A^+} wt(v_i)\right| = c \cdot k'$. From the definition of k', this requires $\left|\sum_{v_i \in A^+} wt(v_i)\right| \leq \left|\sum_{v_i \in V^+} wt(v_i)\right| - k$, which satisfies the requirements for a k Charge Removal. Conversely the definition of k' ensures that the removal must be minimal. Therefore the optimal solution to the WEIGHTED-K-CLIQUE instance must correspond to an optimal solution to the MINIMAL-$k_{\geq}$-CHARGE REMOVALk-CHARGE REMOVAL instance. Similarly any valid solution to the WEIGHTED-K-CLIQUE instance corresponds to some solution to the MINIMAL-$k_{\geq}$-CHARGE REMOVAL instance. □

4.2.2 NP-Hardness for 2-Species with Buckingham-Coulomb Potential

In Section 4.2.1 NP-hardness was shown for the case that there was an unbounded number of species, and NP-completeness in the case that there is a bounded number of charge values. This is strengthened by considering instances with only two unique species. Only the Buckingham-Coulomb potential function with charges of ± 1 is considered in this section. All three problems are again considered, noting that for charges of ± 1 k-CHARGE REMOVAL is equivalent to MINIMAL-$k_{\geq}$-CHARGE REMOVAL. NP-hardness is shown by a reduction from Independent Set problem denoted INDEPENDENT-SET, on penny graphs - adapting it to the Euclidean settings of a crystal graph of ions within a unit cell. The Independent Set problem takes as input a graph, G, and a natural number k. The goal is to find an *independent set*, i.e. a set of vertices such that no two are adjacent, of size k in G, or report that one does not exist. A *Penny graph* is a graph where each vertex may be drawn as a unit circle such that no two circles overlap, and an edge between two vertices exists if and only if the corresponding circles are adjacent to each other, i.e. they intersect at only a single point. Finding an independent set on this class of graphs was shown to be NP-hard by Cerioli et al. [18]. The NP-hardness result for this problem is shown by a reduction from MAX-DEGREE 3 PLANAR VERTEX COVER, shown to be NP-Complete by Garey and Johnson [35].

Construction of the k-CHARGE REMOVAL Instance: Let $I = (G, k)$ be an instance of INDEPENDENT-SET where $G = (V, E)$ is a planar graph with a maximum degree of 3 and $k \in \mathbb{N}$ is the size of the target independent set. An instance of k-CHARGE REMOVAL is created as follows. Using Theorem 1.2 from Cerioli et al. create a new penny graph realisation, G', and a new natural number k. The class of graphs created by this process is denoted as the *long orthogonal penny graphs.* The radius of each circle for G' is chosen as $\frac{n}{2}$.

A region of space in $\mathbb{R}^3$ with a height of at least 1 and a width and length allowing G' to be drawn is created. This space is the parallelepiped for the unit cell. In this space, two copies of G' are drawn such that one is directly above the other at a distance of 1. For every circle in G' two ions are created and placed at the centre of the circles, one in the lower copy of G' and the other in the higher copy. Each ion is labelled with the vertex from G' it corresponds to. In this context *pair* refers to the two ions in the new crystal graph CG, labelled with the same vertex from G'. Two pairs are *neighbouring* if they represent vertices that are adjacent in G'. The lower ions are assigned the positive specie and the upper ions the negative. An example of this arrangement is provided in Figure 4.2. Note that the minimum distance between two pairs in the same plane that are non-adjacent for circles with a radius of $\frac{n}{2}$ is $\sqrt{2}n$, as shown in Figure 4.3.

The positive and negative species are assigned charges of $+1$ and -1 respectively. In general there are 3 sets of parameters to choose determining the interaction between two positive ions, two negative ions, and one positive and one negative ion. For simplicity, the parameters determining the interaction between two positive ions and the parameters determining the interaction between two negative ions are treated as being the same. Informally, this means the energy between two negative ions at a distance of r from each other is the same as the energy between two positive ions at a distance of r. For brevity, 1 and 2 are used to denote the positive and negative specie respectively. With this notation, the parameters that may be set are $A_{11}, B_{11}, C_{11}, A_{12}, B_{12}$, and C_{12}.

An independent set is said to be *left* if the ions left after a removal of k' charges have labels corresponding to an independent set in G'. Let $k' = n - k$, be the number of charges that are required to be removed to be left with an independent set of size k. Note that as the charge of each ion has a magnitude of one, a removal of k' can only be achieved by removing k' positive and k' negative ions. The goal energy for the construction is set as $g = (k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$. To simplify the equations regarding the interaction between planes, $\widehat{r}$ is used to denote $\sqrt{r^2+1}$. To ensure that an independent set is left of size k if and only if one exists, the following three inequalities must be satisfied:

$$\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} + \frac{1}{n} + \frac{A_{12}}{e^{B_{12}\widehat{n}}} - \frac{C_{12}}{\widehat{n}^6} - \frac{1}{\widehat{n}} \geq \left| \frac{A_{12}}{e^{B_{12}}} - C_{12} - 1 \right| \tag{4.1}$$

$$n^2 \left| \frac{A_{11}}{e^{B_{11}r}} - \frac{C_{11}}{r^6} + \frac{1}{r} + \frac{A_{12}}{e^{B_{12}\widehat{r}}} - \frac{C_{12}}{\widehat{r}^6} - \frac{1}{\widehat{r}} \right| \leq \left| \frac{A_{12}}{e^{B_{12}}} - C_{12} - 1 \right|, \qquad r \geq \sqrt{2}n \tag{4.2}$$

$$\frac{A_{11}}{e^{B_{11}r}} - \frac{C_{11}}{r^6} + \frac{1}{r} + \frac{A_{12}}{e^{B_{12}\widehat{r}}} - \frac{C_{12}}{\widehat{r}^6} - \frac{1}{\widehat{r}} > 0, \qquad r \geq \sqrt{2}n \tag{4.3}$$

At a high level, these inequalities are used as follows. Inequality 4.1 ensures that the positive interaction between some ion i and the adjacent pair is greater than the interaction between i and the other ion j in the same pair. This ensures that the cost of keeping two adjacent pairs is greater than the negative energy gained by keeping it. Inequality 4.2 ensures that the total positive interaction energy between some ion i and every ion further than n is no more than the negative energy given by keeping i with the other ion in the pair. This ensures that, given an pair of ions representing a vertex that is not adjacent to any other pairs, it is better to keep the pair in the structure rather than to remove the pair. Finally Inequality 4.3 is used to ensure that the interaction between two ions on the same plane leads to a positive energy penalty no matter the distance. This is used to bound the total energy from above.

The following Lemmas use these inequalities as follows. Lemma 5 show that there exists some parameters for the Buckingham-Coulomb potential satisfying the inequalities, allowing them to be used as a tool when considering the subsequent Lemmas. Lemmas 6 and 7 assume that it is preferable to choose k' pairs over any other set of charges in the arrangement. Lemma 6 shows that the optimal removal results in an independent set. Lemma 7 provides an upper bound on the interaction energy for this setting. Finally Lemma 8 proves that it is always preferable to choose k' pairs over any other set of charges in the arrangement.

Lemma 5. *There exists, for any structure created from a long orthogonal penny graph, some parameters for the Buckingham-Coulomb potential such that Inequalities (4.1, 4.2)* *and (4.3) are satisfied.*

Proof. Values are chosen for A_{12}, B_{12} and C_{12} such that the energy for any pair of ions of opposite vertex at a distance of 1 is -1. This is achieved by choosing a value of $\frac{1}{2n^2}$ for A_{12}, 0 for B_{12}, and $\frac{1}{2n^2}$ for C_{12}. This simplifies the energy equation to:

$$U^{BC}(r) = \frac{A_{11}}{e^{B_{11}r}} - \frac{C_{11}}{r^6} + \frac{1}{r} + \frac{1}{2n^2} - \frac{1}{2n^2\hat{r}^6} - \frac{1}{\hat{r}}.$$

To satisfy Inequality 4.1, $U^{BC}(n) > 1$. This may be satisfied by choosing values for A_{11}, B_{11}, and C_{11} such that $\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} = 1$, noting that $\frac{1}{2n^2} - \frac{1}{2n^2\hat{r}^6} + \frac{1}{r} - \frac{1}{\hat{r}} > 0$ for all positive distances greater than 1. This is satisfied by solving the equation $\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} = 1$, choosing $A_{11} = \frac{C_{11}e^{B_{11}n}}{n^6} + e^{B_{11}n}$.

Inequality (4.2) requires that at a distance of at least $\sqrt{2}n$ the total energy is no more than $\frac{1}{n^2}$. This is satisfied at a distance of $\sqrt{2}n$ by ensuring that the $\frac{A_{11}}{e^{B_{11}\sqrt{2}n}} - \frac{C_{11}}{8n^6} = 0$, which is satisfied after substituting in the appropriate value for A_{11} with $C_{11} = \frac{e^{B_{11}n}}{e^{B_{11}\sqrt{2}n}\left(\frac{1}{(\sqrt{2}n)^6} - \frac{e^{B_{11}n}}{n^6 e^{B_{11}\sqrt{2}n}}\right)}$, which simplifies to $\frac{8n^6}{e^{(\sqrt{2}-1)B_{11}\cdot n} - 8}$. Finally, consider the value of B_{11}. Note that the value of both A_{11} and C_{11} depend greatly on B_{11}, with a small increase in B_{11} leading to a very rapid increase in the value of A_{11} and a rapid decrease in C_{11}. Similarly, the value of the energy given by $\frac{A_{11}}{e^{B_{11}r}} - \frac{C_{11}}{r^6}$ rapidly decreases initially before converging at approximately 0, noting that the first derivative with respect to r of this equation for a given n is $-B_{11}\frac{A_{11}}{e^{B_{11}r}} + \frac{6C_{11}}{r^7}$. As such by choosing a suitably large B_{11} Inequality (4.2) is easily satisfied, one natural choice for this would be $B_{11} = n$.

Note that both $\frac{A_{11}}{e^{B_{11}r}}$ and $\frac{C_{11}}{r^6}$ strictly decrease, therefore if $\left|\frac{A_{11}}{e^{B_{11}r}}\right| \leq \frac{1}{2n^2} - \frac{1}{r} + \frac{1}{\hat{r}}$ and $\left|\frac{C_{11}}{r^6}\right| \leq \frac{1}{r} - \frac{1}{\hat{r}}$ both Inequalities (4.2) and (4.3) are satisfied. From the value of B_{11} this becomes $\frac{n^6}{e^{(\sqrt{2}-1)n^2} - 8}$, which is positive and less than $\frac{1}{n^2}$ for any $n \geq 6$. Note that $\frac{1}{r} - \frac{1}{\hat{r}} \geq \frac{1}{r^6}$ for any $r \geq 1.3$, hence it is clear that $\frac{C_{11}}{r^6} \leq \frac{1}{r} - \frac{1}{\hat{r}}$ for $n \geq 6$. Considering $\frac{A_{11}}{e^{B_{11}r}}$ at a distance of $\sqrt{2}n$, the equation becomes $\frac{A_{11}}{e^{n^2\sqrt{2}}} = \frac{C_{11}}{n^6 e^{n^2\sqrt{2}}} + \frac{1}{e^{(\sqrt{2}-1)n^2}}$, from the previous arguments it follows that this is considerably less than $\leq \frac{1}{2n^2} - \frac{1}{r} + \frac{1}{\hat{r}}$ for $n \geq 6$.

Note that due to the constant $\frac{1}{2n^2}$ term there is a positive value for any distance greater than $\sqrt{2}n$, satisfying Inequality (4.3). □

Lemma 6. *Inequalities (4.1) and (4.2) are sufficient to ensure that an independent set is left if one exists in the original* INDEPENDENT-SET *instance.*

Proof. Inequality (4.1) ensures that if there are two pairs corresponding to adjacent vertices, the total energy always decreases by removing one of the pairs. Inequality (4.2) complements (4.1) by ensuring that given a pair corresponding to a vertex with no adjacent neighbours, the total energy would increase by removing it. This holds even in the case that all other pairs are at a distance of $\sqrt{2}n$. Inequalities (4.1) and (4.2) combined means that the global minimum total energy for any subset is the maximum independent set. Note that the total energy decreases with the cardinality of the given independent set. $\square$

Lemma 7. *Given k pairs, the energy is less than $(k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$ if and only if the pairs correspond to an independent set of size k, for $\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1 < 0$.*

Proof. Given k pairs, the energy between the ions in each pair is $\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1$, for a total of $k(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$. Inequality (4.2) ensures that the maximum energy gained from pairs of ions corresponding to non-adjacent circles is at most $|((\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1))|$. Inequality (4.3) ensures that having vertices on the same plane leads to a slight positive charge. From this it follows that the maximum energy a set of ions corresponding to an independent set is $(k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$. Conversely, from Inequality (4.1) it is known that if there is a pair of adjacent circles the total energy must be greater than $(k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$.

Note that for $k_{\geq}$-CHARGE REMOVAL that if greater than k' pairs were removed this energy could not be achieved as the minimum energy would be $(k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$ for the interaction within pairs. As there is a positive interaction between pairs, the total energy must be slightly greater than this for any $k > 1$. Therefore the total energy is less than $(k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$ if and only if there is an independent set of size k left. Note that under the choice of variables from Lemma 5, the upper bound is $-(k-1)$. $\square$

Lemma 8. *When removing k' charges from the construction from a long orthogonal penny graph, it is always preferable to remove pairs provided that Inequalities (4.1-4.3) hold.*

Proof. Assume that this statement is false, there must be some assignment, where it is preferable to remove some set of at least two vertices, v_i^+ and v_j^-, that do not form a pair with any ions that have be removed. Assume that there are t positive and t negative vertices in the graph. If instead v_j^- was left in, while v_i^+ was removed, the remaining energy would change by at least $-1 + t\left(\frac{A_{11}}{e^{B_{11}\sqrt{2}n}} - \frac{C_{11}}{(\sqrt{2}n)^6} + \frac{1}{\sqrt{2}n} + \frac{A_{12}}{e^{B_{12}\widehat{\sqrt{2}n}}} - \frac{C_{12}}{\widehat{\sqrt{2}n}^6} - \frac{1}{\widehat{\sqrt{2}n}}\right)$. From the arguments in Lemma 6 and the construction in Lemma 5 this leads to a decrease in total energy, making it preferable and therefore contradicting the assumption. Note that given a positively charged vertex of the maximum degree, in this case 3, it could contribute at most $\frac{3}{2n^2} - \frac{3}{2n^8} - \frac{3}{n}$ which has an absolute value less than 1 for any $n \geq 3$. Therefore, by contradiction this holds. □

Theorem 8. k-CHARGE REMOVAL, MINIMAL-$k_{\geq}$-CHARGE REMOVAL *and* $k_{\geq}$-CHARGE REMOVAL *are NP-Complete when limited to only two species of ion and restricted to the Buckingham-Coulomb potential energy function.*

Proof. Building on the results from Lemmas 5, 6, 7, and 8, the next step is to show NP-Completeness. Lemma 6 shows that, provided Inequalities (4.1) and (4.2) hold, the optimal solution is to leave an independent set. From Lemma 5 it follows that these inequalities are satisfiable for any graph under the given construction, noting that the assignment of parameters gives an energy of -1 within pairs. Lemma 7 shows that the upper bound is reachable if and only if an independent set has been left. It follows from Lemma 8 that the assumption that it is preferable to remove a set of pairs over any other set of charges holds when the inequalities also do.

Therefore there is a satisfiable instance of k-CHARGE REMOVAL or any generalisation if and only if the instance of INDEPENDENT SET for the maximum degree 3 planar graph instance is satisfiable. Conversely if the INDEPENDENT SET instance is satisfiable, the corresponding k-CHARGE REMOVAL instance is satisfied by leaving the vertices corresponding to the independent set in the long orthogonal penny graph construction. Hence under these restriction all three problems are NP-complete. Note that this may be extended to vertices with charges $\pm c$ for any given c. □

Corollary 4. *It is NP-hard to approximate the optimal solution tok*-CHARGE REMOVAL, MINIMAL-$k_{\geq}$-CHARGE REMOVAL, *or* $k_{\geq}$-CHARGE REMOVAL *within a factor* $1 + \frac{3}{n-k-1}$ *for the Buckingham-Coulomb potential.*

Proof. Observe that following Lemma 7, given a set of k ion pairs corresponding to an independent set is $(k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$, or in terms of the k-CHARGE REMOVAL problem $(n-k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)$. In the case that there exists some pair of ion-pairs representing adjacent vertices, there must be an energy penalty of $4\left(\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} + \frac{1}{n} + \frac{A_{12}}{e^{B_{12}\hat{n}}} - \frac{C_{12}}{\hat{n}^6} - \frac{1}{\hat{n}}\right)$. Therefore, a lower bound on the energy in the case that there is at least some pair of ions adjacent to each other is $(n-k)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1) + 4\left(\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} + \frac{1}{n} + \frac{A_{12}}{e^{B_{12}\hat{n}}} - \frac{C_{12}}{\hat{n}^6} - \frac{1}{\hat{n}}\right) \geq$

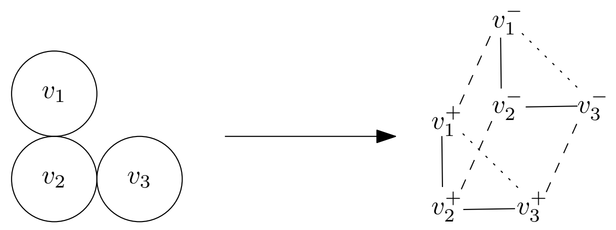

Figure 4.2: Example of the construction of an arrangement from a penny graph, om this example v_1 and v_2 are adjacent, as are v_2 and v_3, but not v_1 and v_3

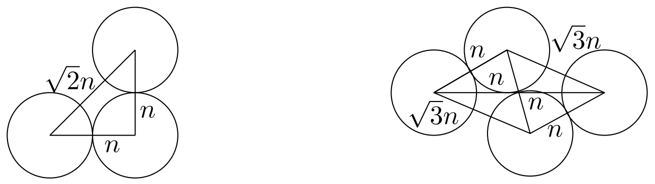

Figure 4.3: Illustration of the distances between the centre of non-adjacent pennies using the construction of Cerioli et al. [18].

$(n-k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1) + 4\left(\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} + \frac{1}{n} + \frac{A_{12}}{e^{B_{12}\hat{n}}} - \frac{C_{12}}{\hat{n}^6} - \frac{1}{\hat{n}}\right)$. Therefore any approximation algorithm that can achieve an approximation ratio smaller than:

$$\frac{(n-k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1) + 3\left(\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} + \frac{1}{n} + \frac{A_{12}}{e^{B_{12}\hat{n}}} - \frac{C_{12}}{\hat{n}^6} - \frac{1}{\hat{n}}\right)}{(n-k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)} =$$

$$1 + \frac{3\left(\frac{A_{11}}{e^{B_{11}n}} - \frac{C_{11}}{n^6} + \frac{1}{n} + \frac{A_{12}}{e^{B_{12}\hat{n}}} - \frac{C_{12}}{\hat{n}^6} - \frac{1}{\hat{n}}\right)}{(n-k-1)(\frac{A_{12}}{e^{B_{12}}} - C_{12} - 1)} \geq 1 + \frac{3}{n-k-1}$$

would be able to find the optimal solution to the underlying k-independent set problem in polynomial time unless $P = NP$. □

4.2.3 NP-Hardness for the Coulomb Potential with Unbounded Charges

The final case that is considered in this work is when the energy function is the Coulomb potential. NP-hardness for this case is shown by a reduction from KNAPSACK to $k_{\geq}$-CHARGE REMOVAL. Note that with an unbounded number charge values this problem is not in NP for MINIMAL-$k_{\geq}$-CHARGE REMOVAL due to Proposition 5 and is trivially NP-hard for k-CHARGE REMOVAL due to Corollary 1. This reduction requires using an unbounded number of charge values, thus it follows from proposition 5 that it is NP-hard to verify if a solution to an instance of $k_{\geq}$-CHARGE REMOVAL is minimal. In this reduction it is shown that an instance of $k_{\geq}$-CHARGE REMOVAL such that the set of ions left correspond to the items for the knapsack instance if and only if there is a set satisfying the knapsack instance.

Theorem 9. *$k_{\geq}$-CHARGE REMOVAL and MINIMAL-$k_{\geq}$-CHARGE REMOVAL remains NP-hard when the energy function is limited to the Coulomb potential.*

Proof. In the knapsack problem, henceforth KNAPSACK, the input is a bag with capacity C, and a set of items S. Each item $i \in S$ has a weight w_i, and a value p_i. In this problem the goal it to find the subset $S' \subseteq S$ such that $\sum\limits_{i \in S'} p_i$ is maximised conditional on $\sum\limits_{i \in S'} w_i \leq C$. Alternatively this may phrased as a decision problem by taking some goal value g and asking if there is an S' such that $\sum\limits_{i \in S'} p_i \geq g$.

NP-completeness for $k_{\geq}$-CHARGE REMOVAL and MINIMAL-$k_{\geq}$-CHARGE REMOVAL is shown by a reduction from the knapsack problem. Given an instance, I, of the knapsack problem as described above, an instance, I', of $k_{\geq}$-CHARGE REMOVAL is created as follows. For every $i \in S$, two charges are created denoted v_i^+ and v_i^- and labelled with the corresponding item. These are assigned a charge of w_i to v_i^+ and $-w_i$ to v_i^-.

The values u and α are defined such that u is some value such that there does not exist any pair of items, i and j, such that $p_i > p_j$ but $p_j + u \geq p_i$, and u is less than the smallest *unit of precision* for the value of the items. Using this, α is defined as some value satisfying the inequality $u > \frac{4n^2 w_{max}^2}{\alpha}$, where w_{max} is the weight of the heaviest item. This ensures that α is some distance such that if all vertices are at least α away from each other there is a difference of no more than u in energy, which is sufficient to ensure that vertices at that distance may be safely ignored.

These vertices are now placed such that, for each item, the distance between the two vertices v_i^+ and v_i^- has a potential energy of $-p_i$. Recall that $U_{ij}^C = \frac{q_i q_j}{r_{ij}}$. This is achieved by placing them at a distance of $\frac{w_i^2}{p_i}$. Each of the pairs of vertices representing an item is placed in a line so that the distance between any two pairs is no less than α. An example of this construction is provided in Figure 4.4.

The value k is chosen as $k = \left(\sum\limits_{i \in S} w_i\right) - c$, ensuring that there are no more than c vertices left after removing k, corresponding to a valid assignment for the knapsack instance. Finally, the goal value is chosen as $g' = -g + u$. It follows from this construction

Item	*Weight*	*Value*
I_1	9	3
I_2	6	2
I_3	3	3

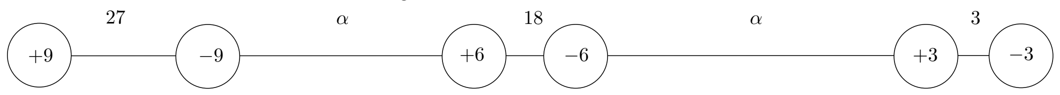

Figure 4.4: Example of construction of the structure from the knapsack instance. In this $u < 1$ and $\alpha > 2916$

that any removal of $k_{\geq}$ charges are a valid packing in terms of the capacity.

If the $k_{\geq}$-CHARGE REMOVAL instance is satisfiable then there must be some valid packing of no more than g' energy. As the interaction between vertices corresponding to different items is trivially small, the only way to achieve this is to choose a set of vertex pairs with an energy between them no more than g'. As the energy between pairs is equal to the value of the items, the only way this is achieved this is to have items corresponding to a packing with value at least g. Conversely if the $k_{\geq}$-CHARGE REMOVAL instance is not satisfiable, there does not exist a packing of value g by the same arguments.

Similarly if the KNAPSACK instance is satisfiable then the $k_{\geq}$-CHARGE REMOVAL instance may be satisfied by removing all charges not corresponding to a satisfying packing of the KNAPSACK instance. Finally if the KNAPSACK instance is not satisfiable then by the previous arguments the KCR instance also can not be satisfied. Therefore this problem is NP-Complete. Note that as the weights on all items is positive, with a corresponding negative energy in I', given a non-minimal satisfying solution there exists some minimal satisfying solution. Therefore MINIMAL-$k_{\geq}$-CHARGE REMOVAL is also NP-hard. $\square$

Chapter 5

The k-Centre Problem on Combinatorial Crystals

With the problem of directly solving CSP being seemingly intractable, the obvious question is if there exists an alternative approach to solve the problem. In the following chapters, we focus on the idea of *sampling* potential crystals structures. At a high level, the goal of this problem is to choose a diverse set of crystal structures, with the hope that by choosing a diverse set, a structure close to the optimal can be found. Here we focus on the *local structures*, representing the interactions between ions that are as close as possible. The motivation for this approach comes from the energy functions which we look at which have a rapid decrease in energy as distance increases. For example, the Coulomb potential defined as $\frac{q_i \cdot q_j}{r_{ij}}$ tends rapidly towards 0. As such, finding local structures provides a strong basis for exploring the space of possible solutions.

We use the k-centre problem as a basis to formalise these notions as a computer science problem. The k-centre problem takes as input a weighted graph $G = (V, E)$ and integer k, with the goal of finding a set $\mathbf{S}$ of k vertices from V minimising $\max_{v \in V} \min_{u \in \mathbf{S}} D(v, u)$ where $D(v, u)$ returns the distance between vertices v and u. To use the k-centre problem as a basis for this setting it is necessary to define a distance between words emphasising local differences.

This chapter is divided into four sections. Section 5.1 provides the key definitions for this chapter, including the distance used and some fundamental results for the k-centre problem in this setting. Section 5.2 provides the first approximation algorithm for solving this problem in the one dimensional case for unconstrained necklaces, using de Bruijn sequences as a basis. Section 5.3 provides a less precise but more general approximation algorithm, covering the other settings discussed in this thesis, including bracelets, necklaces with constrained Parikh vectors, necklaces with forbidden subwords and multidimensional necklaces. Finally Section 5.4 provides further discussion on the usage of this problem in the context of CSP.

5.1 The Overlap Distance and the k-Centre Problem

In this section we formally define the k-centre problem for cyclic words. At a high level, the input to our problem is an alphabet of size q, a vector of dimensions $\overline{\mathbf{n}} = (n_1, n_2, \ldots, n_d)$ that defines the size of the d-dimensional words, and a positive integer k. Note that in the one dimensional case $\overline{\mathbf{n}}$ may be given as a single scalar value, n. The goal is to choose a set $\mathbf{S}$ of k necklaces from the set $\mathcal{N}_q^{\overline{\mathbf{n}}}$ such that the maximum distance between any necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ and the set $\mathbf{S}$ is minimised. Since there is no standard notion of distance between necklaces, our first task is to define one. To this end, we introduce the *overlap distance*, which aims to capture similarity between crystalline materials emphasising local differences. At a high level, the overlap distance between two necklaces is the inverse of the *overlap coefficient* between them, in this case 1 minus the overlap coefficient. This can be seen as a natural distance based "bag-of-words" techniques used in machine learning [36].

Overlap Distance for Necklaces. Our definition of the overlap distance depends of the well studied *overlap coefficient*, defined for a pair of set A and B as $\frac{|A \cap B|}{\min(|A|,|B|)}$. For notation let $\mathfrak{C}(A, B)$ return the overlap coefficient between two sets A and B. Observe that $\mathfrak{C}(A, B)$ returns a rational value between 0 and 1, with 0 indicating no common elements and 1 indicating that either $A \subseteq B$ or $B \subseteq A$. In the context of necklaces the overlap coefficient $\mathfrak{C}(\tilde{\mathbf{w}}, \tilde{\mathbf{v}})$ is defined as the overlap coefficient between the multisets of all subwords of $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$. For some necklace $\tilde{\mathbf{w}}$ of dimensions $\overline{\mathbf{n}}$, the multiset of subwords of dimensions $\overline{\ell}$ contains all $\bar{u} \sqsubseteq_{\overline{\ell}} \bar{w}$. For each subword $\bar{u}$ appearing m times in $\tilde{\mathbf{w}}$, m copies of $\bar{u}$ are added to the multiset. This gives a total of N subwords of dimensions $\overline{\ell}$ for any $\overline{\ell}$, where $N = n_1 \cdot n_2 \cdot \ldots \cdot n_d$. For example, given the necklace represented by $aaab$, the multiset of subwords of length 2 are $\{aa, aa, ab, ba\} = \{aa \times 2, ab, ba\}$. The multiset of all subwords is the union of the multisets of the subwords for every vector of dimensions, having a total size of N^2; see Figure 5.1.

	word *ababab*	word *abbabb*	Intersection
1	$a \times 3, b \times 3$	$a \times 2, b \times 4$	5
2	$ab \times 3, ba \times 3$	$ab \times 2, bb \times 2, ba \times 2$	4
3	$aba \times 3, bab \times 3$	$abb \times 2, bba \times 2, bab \times 2$	2
4	$abab \times 3, baba \times 3$	$abba \times 2, bbab \times 2, babb \times 2$	0
5	$ababa \times 3, babab \times 3$	$abbab \times 2, bbabb \times 2, babba \times 2$	0
6	$ababab \times 3, bababa \times 3$	$abbabb \times 2, bbabba \times 2, babbab \times 2$	0
Total			11

Figure 5.1: Example of the overlap coefficient calculation for a pair of words *ababab* and *abbabb*. There are 11 common subwords out of the total number of 36 subwords of length from 1 till 6, so $\mathfrak{C}(ababab, abbabb) = \frac{11}{36}$ and $\mathfrak{O}(ababab, abbabb) = \frac{25}{36}$.

A	*aaaa*	B	*aaab*	C	*aabb*
D	*abab*	E	*abbb*	F	*bbbb*

$\tilde{\mathbf{w}} \backslash \tilde{\mathbf{v}}$	A	B	C	D	E	F
A	0	$\frac{10}{16}$	$\frac{13}{16}$	$\frac{14}{16}$	$\frac{15}{16}$	1
B	$\frac{10}{16}$	0	$\frac{9}{16}$	$\frac{10}{16}$	$\frac{12}{16}$	$\frac{15}{16}$
C	$\frac{13}{16}$	$\frac{9}{16}$	0	$\frac{10}{16}$	$\frac{8}{16}$	$\frac{13}{16}$
D	$\frac{14}{16}$	$\frac{10}{16}$	$\frac{10}{16}$	0	$\frac{6}{16}$	$\frac{14}{16}$
E	$\frac{15}{16}$	$\frac{12}{16}$	$\frac{8}{16}$	$\frac{10}{16}$	0	$\frac{10}{16}$
F	1	$\frac{15}{16}$	$\frac{13}{16}$	$\frac{14}{16}$	$\frac{8}{16}$	0

Figure 5.2: Example of the overlap distance $\mathfrak{D}(\langle \tilde{\mathbf{w}} \rangle, \langle \tilde{\mathbf{v}} \rangle)$ for all binary cyclic words of length 4.

To use the overlap coefficient as a distance between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$, the overlap coefficient is inverted so that a value of 1 means $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$ share no common subwords while a value of 0 means $\tilde{\mathbf{w}} = \tilde{\mathbf{v}}$. The overlap distance (see example in Figure 5.1) between two necklaces $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$ is $\mathfrak{O}(\tilde{\mathbf{w}}, \tilde{\mathbf{v}}) = 1 - \mathfrak{C}(\tilde{\mathbf{w}}, \tilde{\mathbf{v}})$. Proposition 8 shows that this distance is a metric distance.

Proposition 8. *The overlap distance for necklaces is a metric distance.*

Proof. Let $\tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{c}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$, for some arbitrary vector $\overline{\mathbf{n}} \in \mathbb{N}^d$ and $q \in \mathbb{N}$. In order for the overlap distance to satisfy the metric property, $\mathfrak{O}(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ must be less than or equal to $\mathfrak{O}(\tilde{\mathbf{a}}, \tilde{\mathbf{c}}) + \mathfrak{O}(\tilde{\mathbf{b}}, \tilde{\mathbf{c}})$. Rewriting this gives $1 - \mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) \leq 2 + \mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) - \mathfrak{C}(\tilde{\mathbf{b}}, \tilde{\mathbf{c}})$ which can be rewritten in turn as $\mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) + \mathfrak{C}(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}) \leq 1 + \mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$. Observe that if $\mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{c}}) + \mathfrak{C}(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}) > 1$ then $\frac{|\tilde{\mathbf{a}} \cup \tilde{\mathbf{c}}|}{N^2} + \frac{|\tilde{\mathbf{b}} \cup \tilde{\mathbf{c}}|}{N^2} > 1$, meaning that $|\tilde{\mathbf{a}} \cup \tilde{\mathbf{c}}| + |\tilde{\mathbf{b}} \cup \tilde{\mathbf{c}}| > N^2$. This implies that $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ share at least $|\tilde{\mathbf{a}} \cup \tilde{\mathbf{c}}| + |\tilde{\mathbf{b}} \cup \tilde{\mathbf{c}}| - N^2$ subwords. Therefore $\mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{n}})$ must be at least $\mathfrak{C}(\tilde{\mathbf{a}}, \tilde{\mathbf{n}}) + \mathfrak{C}(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}) - 1$. Hence $\mathfrak{O}(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) \leq \mathfrak{O}(\tilde{\mathbf{a}}, \tilde{\mathbf{c}}) + \mathfrak{O}(\tilde{\mathbf{b}}, \tilde{\mathbf{c}})$. □

The k-Centre Problem. The goal of the k-Centre problem for necklaces is to select a set of k necklaces of dimensions $\overline{\mathbf{n}}$ over an alphabet of size q that are "central" within the set of necklaces $\mathcal{N}_q^{\overline{\mathbf{n}}}$. Formally the goal is to choose a set $\mathbf{S}$ of k necklaces such that the maximum distance between any necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ and the nearest member of $\mathbf{S}$ is minimised. Given a set of necklaces $\mathbf{S} \subset \mathcal{N}_q^{\overline{\mathbf{n}}}$, we use $\mathfrak{D}(\mathbf{S}, \mathcal{N}_q^{\overline{\mathbf{n}}})$ to denote the maximum overlap distance between any necklace in $\mathcal{N}_q^{\overline{\mathbf{n}}}$ and its closest necklace in $\mathbf{S}$.

Problem 11. *k-Centre problem for necklaces.*

Input: A vector of d-dimensions $\overline{\mathbf{n}} \in \mathbb{Z}^d$, an alphabet Σ of size q, and an integer $k \in \mathbb{Z}$.

Question: What is the set $\mathbf{S} \subseteq \mathcal{N}_q^{\overline{\mathbf{n}}}$ of size k minimising $\mathfrak{D}(\mathbf{S}, \mathcal{N}_q^{\overline{\mathbf{n}}})$?

There are two major challenges we have to overcome in order to solve Problem 11, the exponential size of $\mathcal{N}_q^{\overline{\mathbf{n}}}$, and the lack of structural, algorithmic, and combinatorial results for multidimensional necklaces. We show that the conceptually simpler problem of verifying whether a set of necklaces is a solution for Problem 12 is NP-hard for any dimension d.

Problem 12. *The k-Centre verification problem for necklaces. Given a set of k necklaces $\mathbf{S} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ and a distance ℓ, does there exist some necklace $\tilde{\mathbf{v}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ such that $\mathfrak{O}(\tilde{\mathbf{s}}, \tilde{\mathbf{v}}) > \ell$ for every $\tilde{\mathbf{s}} \in \mathbf{S}$?*

Input: A vector of d-dimensions $\overline{\mathbf{n}} \in \mathbb{Z}^d$, an alphabet Σ of size q, an integer $k \in \mathbb{Z}$, and rational distance $\ell \in \mathbb{Q}$.

Question: Does there exists a set $\mathbf{S} \subseteq \mathcal{N}_q^{\overline{\mathbf{n}}}$ of size k such that $\mathfrak{D}(\mathbf{S}, \mathcal{N}_q^{\overline{\mathbf{n}}}) \leq \ell$?

Theorem 10. *Given a set of k necklaces $\mathbf{S} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ and a distance ℓ, it is NP-hard to determine if there exists some necklace $\tilde{\mathbf{v}} \in \mathcal{N}_q^{\overline{\mathbf{n}}}$ such that $\mathfrak{O}(\tilde{\mathbf{s}}, \tilde{\mathbf{v}}) > \ell$ for every $\tilde{\mathbf{s}} \in \mathbf{S}$ for any dimension d.*

Proof. We prove the claim via a reduction from the Hamiltonian cycle problem on bipartite graphs to Problem 12 in one dimension. Note that if the problem is hard in the 1D case, then it is also hard in any dimension $d \geq 1$ by using the same reduction for necklaces of dimensions $(n_1, 1, 1, \ldots, 1)$. Let $G = (V, E)$ be a bipartite graph containing an even number $n \geq 6$ of vertices. The alphabet Σ is constructed with size n such that there is a one to one correspondence between each vertex in V and symbol in Σ. Using Σ a set $\mathbf{S}$ of necklaces is constructed as follows. For every pair of vertices $u, v \in V$ where $(u, v) \notin E$, the necklace corresponding to the word $(uv)^{n/2}$ is added to the set of centres $\mathbf{S}$. Further the word v^n, for every $v \in V$, is added to the set $\mathbf{S}$.

For the set $\mathbf{S}$, we ask if there exists any necklace in $\mathcal{N}_q^n$ that is further than a distance of $1 - \frac{3}{n^2}$. For the sake of contradiction, assume that there is no Hamiltonian cycle in G, and further that there exists a necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^{\mathbf{n}}$ such that the distance between $\tilde{\mathbf{w}}$ and every necklace $\tilde{\mathbf{v}} \in \mathbf{S}$ is greater than $1 - \frac{3}{n^2}$. If $\tilde{\mathbf{w}}$ shares a subword of length 2 with any necklace in $\mathbf{S}$ then $\tilde{\mathbf{w}}$ would be at a distance of no less than $1 - \frac{3}{n^2}$ from $\mathbf{S}$. Therefore, as every subword of length 2 in $\mathbf{S}$ corresponds to a edge that is not a member of E, every subword of length 2 in $\tilde{\mathbf{w}}$ must correspond to a valid edge.

As $\tilde{\mathbf{w}}$ can not correspond to a Hamiltonian cycle, there must be at least one vertex v for which the corresponding symbol appears at least 2 times in $\tilde{\mathbf{w}}$. As G is bipartite, if any cycle represented by $\tilde{\mathbf{w}}$ has length greater than 2, there must exist at least one vertex u such that $(v, u) \notin E$. Therefore, the necklace $(uv)^{n/2}$ is at a distance of no more than $\frac{n^2}{3}$ from $\tilde{\mathbf{w}}$. Alternatively, if every cycle represented by $\tilde{\mathbf{w}}$ has length 2, there must be some vertex v that is represented at least 3 times in $\tilde{\mathbf{w}}$. Hence in this case $\tilde{\mathbf{w}}$ is at a distance of no more than $1 - \frac{3}{n^2}$ from the word $v^n \in \mathbf{S}$. Therefore, there exists a necklace at a distance of greater than $1 - \frac{3}{n^2}$ if and only if there exists a Hamiltonian cycle in the graph G. Therefore, it is NP-hard to verify if there exists any necklace at a distance greater than l for some set $\mathbf{S}$. □

The combination of this negative result with the exponential size of $\mathcal{N}_q^{\mathbf{n}}$ relative to $\overline{\mathbf{n}}$ and q makes finding an optimal solution for Problem 11 exceedingly unlikely. As such the remainder of our work on the k-centre problem for necklaces focuses on approximation algorithms. Lemma 9 provides a lower bound on the optimal distance.

Lemma 9. *Let $\mathbf{S} \subseteq \mathcal{N}_q^{\mathbf{n}}$ be an optimal set of k centres minimising $\mathfrak{D}(\mathbf{S}, \mathcal{N}_q^{\mathbf{n}})$ then $\mathfrak{D}(\mathbf{S}, \mathcal{N}_q^{\mathbf{n}}) \geq 1 - \frac{\log_q(k \cdot N)}{N}$.*

Proof. We first prove the lemma for the one-dimensional case, then extend the proof to the multidimensional setting. Recall that the distance between any pair of necklaces $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ is determined by the overlap coefficient and by extension the number of shared subwords between $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$. Hence the distance between the furthest necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^n$ and the optimal set $\mathbf{S}$ is bound from bellow by determining an upper bound on the number of shared subwords between $\tilde{\mathbf{w}}$ and the words in $\mathbf{S}$. For the remainder of this proof let $\tilde{\mathbf{w}}$ to be the necklace furthest from the optimal set $\mathbf{S}$. Further for the sake of determining an upper bound, the set $\mathbf{S}$ is treated as a single necklace $\tilde{\mathbf{S}}$ of length $n \cdot k$. This may be thought of as the necklace corresponding to the concatenation of each necklace in $\mathbf{S}$. Note that the length of $\mathbf{S}$ is $k \cdot n$. As the distance between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{S}}$ is no more than the distance between $\tilde{\mathbf{w}}$ and any $\tilde{\mathbf{v}} \in \mathbf{S}$, the distance between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{S}}$ provides a lower bound on the distance between $\tilde{\mathbf{w}}$ and $\mathbf{S}$.

In order to determine the number of subwords shared by $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{S}}$, consider first the subwords of length 1. In order to guarantee that $\tilde{\mathbf{w}}$ shares at least one subword of length 1, $\tilde{\mathbf{S}}$ must contain each symbol in Σ, requiring the length of $\tilde{\mathbf{S}}$ to be at least q. Similarly, in order to ensure that $\tilde{\mathbf{w}}$ shares two subwords of length 1 with $\tilde{\mathbf{S}}$, $\tilde{\mathbf{S}}$ must contain 2 copies of every symbol on Σ, requiring the length of $\tilde{\mathbf{S}}$ to be at least $2q$. More generally for $\tilde{\mathbf{S}}$ to share i subwords of length 1 with $\tilde{\mathbf{w}}$, $\tilde{\mathbf{S}}$ must contain i copies of each symbol in Σ, requiring the length of $\tilde{\mathbf{S}}$ to be at least $i \cdot q$. Hence the maximum number of subwords of length 1 that $\tilde{\mathbf{w}}$ can share with $\tilde{\mathbf{S}}$ is either $\lfloor \frac{n \cdot k}{q} \rfloor$, if $\lfloor \frac{n \cdot k}{q} \rfloor \leq n$, or n otherwise.

In the case of subwords of length 2, the problem becomes somewhat more complicated. Note that in order to share a single word of length 2, it is not necessary to to have every subword of length 2 appear as a subword of $\tilde{\mathbf{w}}$. Instead, it is sufficient to use only the prefixes of the canonical representations of each necklace. For example, given the binary alphabet $\{a, b\}$, every necklace has either aa, ab or bb as the prefix of length 2. Note that any necklace of length 2 followed by the largest symbol q in the alphabet $n-2$ times belongs to the set N_q^n. As such, a simple lower bound on the number of prefixes of the canonical form of necklaces is the number of necklaces of length 2, which in turn is bounded by $\frac{q^2}{2}$. Noting that these prefixes in $\tilde{\mathbf{S}}$ may overlap, in order to ensure that $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{w}}$ share at least one subword of length 2, the length of $\tilde{\mathbf{S}}$ must be at least $\frac{q^2}{2}$. Similarly, for $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{w}}$ to share i subwords of length 2, the length of $\tilde{\mathbf{S}}$ must be at least $\frac{i \cdot q^2}{2}$. Hence the maximum number of subwords of length 2 that $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{w}}$ can share is either $\lfloor \frac{2n \cdot k}{q^2} \rfloor$, if $\lfloor \frac{2n \cdot k}{q^2} \rfloor \leq n$, or n otherwise. More generally, in order for $\tilde{\mathbf{S}}$ to share at least one subword of length j with $\tilde{\mathbf{w}}$, the length of $\tilde{\mathbf{S}}$ must be at least $\frac{q^j}{j}$. Further the maximum number of subwords of length j that $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{w}}$ can share is either $\lfloor \frac{j \cdot n \cdot k}{q^j} \rfloor$, if $\lfloor \frac{j \cdot n \cdot k}{q^j} \rfloor \leq n$ or n otherwise.

Using these observations, the maximum length of a common subword that $\tilde{\mathbf{w}}$ can share with $\tilde{\mathbf{S}}$ is the largest value l such that $\frac{q^l}{l} \leq n \cdot k$. By noting that $\frac{q^l}{l} \geq \frac{q^l}{n}$, a upper bound on l can be derived by rewriting the inequality $\frac{q^l}{n} \leq n \cdot k$ as $l = 2\log_q(n \cdot k)$. Note further that, for any value $l' > l$, there must be at least one necklace that does not share any subword of length l' with $\tilde{\mathbf{S}}$ as $\tilde{\mathbf{S}}$ can not contain enough subwords to ensure that this is the case. This bound allows an upper bound number of shared subwords between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{S}}$ to be given by the summation $\sum_{i=1}^{2\log_q(n \cdot k)} \min(\lfloor \frac{i \cdot n \cdot k}{q^i} \rfloor, n) \leq n \cdot \log_q(n \cdot k) + \frac{\log_q(k \cdot n)}{q-1} \approx \frac{q \cdot n \log_q(k \cdot n)}{q-1} \approx n \log_q(k \cdot n)$.

Using this bound, the distance between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{S}}$ must be no less than $1 - \frac{\log_q(k \cdot n)}{n}$.

The same arguments can be applied to the multidimensional case. Let $\overline{\mathbf{m}} = (m_1, m_2, \ldots, m_d)$ be a vector of d-dimensions such that $M = m_1 \cdot m_2 \cdot \ldots \cdot m_d$. The largest value of M such that $\tilde{\mathbf{S}}$ can contain every subword with M positions is $2\log_q(n \cdot k)$. The upper bound on the number of words of dimensions $\overline{\mathbf{m}}$ is $\frac{q^M}{M}$. Let $F(x, \overline{\mathbf{m}})$ return the size of the set $[\overline{\mathbf{m}}]$, i.e. the number of vectors with x positions that are less than or equal to $\overline{\mathbf{m}}$ in each dimension. Using this notation, the maximum number of shared subwords between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{S}}$ is $\sum\limits_{i=1}^{M} F(i, \overline{\mathbf{m}}) \cdot \frac{i \cdot N \cdot k}{q^i}$. Note that $\sum\limits_{i=1}^{M} F(i, \overline{\mathbf{m}}) \cdot \frac{i \cdot N \cdot k}{q^i} \leq \sum\limits_{i=1}^{M} \frac{i \cdot N \cdot k}{q^i}$. Therefore, the upper bound on the number of common subwords in the multidimensional setting is $N \log_q(k \cdot N)$, giving a bound on the distance of $1 - \frac{\log_q(k \cdot N)}{N}$. □

Sections 5.2 and 5.3 provide two approximation algorithms for the k-centre problem using Lemma 9 as a lower bound. The first of these is $1 + (\frac{\log_q(kN)}{N - \log_q(kN)} - \frac{\log_q^2(kN)}{2N(N - \log_q(kN))})$-approximate with a running time $O(N \cdot k)$, but it requires access to the de-Bruijn hypertori of the multidimensional necklaces; this is a generalisation of de-Bruijn sequences. When $d = 1$, there exists an efficient algorithm for computing the de-Bruijn sequence. However, for $d > 1$, no algorithm is known for computing a de-Bruijn hypertori. Therefore, we develop a second algorithm that is $1 + (\frac{\log_q(kN)}{N - \log_q(kN)} - \frac{\log_q^2(k)}{2N(N - \log_q(kN))})$-approximation with a running time $O(N^6)$, requiring techniques presented in Section 8.2.

The main idea behind both algorithms is to try to find the largest vector of dimensions $\overline{\ell} = (l_1, l_2, \ldots, l_d)$ such that every subword of dimensions $\overline{\ell}$ appears at least once in some word within the set. In this setting $\overline{\mathbf{m}}$ is larger than $\overline{\ell}$ if $m_1 \cdot m_2 \cdot \ldots \cdot m_d > l_1 \cdot l_2 \cdot \ldots \cdot l_d$. This is motivated by observing that if two necklaces share a subword of length l, they must also share 2 subwords of length $l - 1$, 3 of length $l - 2$, and so on. Lemma 10 provides an upper bound for the overlap distance between any necklace in $|\mathcal{N}_q^n|$ and the set $\mathbf{S}$ containing all subwords of length l.

Lemma 10. *Given $\tilde{\mathbf{w}}, \tilde{\mathbf{v}} \in \mathcal{N}_q^{\mathbf{n}}$ sharing a common subword $\bar{a}$ of dimensions $\overline{\mathbf{m}}$, let $x_i = n_i \cdot m_i$ if $n_i = m_i$, and $x_i = \frac{m_i(m_i+1)}{2}$ otherwise. The distance between $\bar{w}$ and $\bar{v}$ is bounded from above by $\mathfrak{O}(w, v) \leq 1 - \frac{\prod_{i=1}^{d} x_i}{N^2} \leq 1 - \frac{M^2}{N^2}$ where $N = n_1 \cdot n_2 \cdot \ldots \cdot n_d$ and $M = m_1 \cdot m_2 \cdot \ldots \cdot m_d$.*

Proof. Note that the minimum intersection between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$ is the number of subwords of $\bar{a}$, including the word $\bar{a}$ itself. To compute the number of subwords of $\bar{a}$, consider the number of subwords starting at some position $\bar{\mathbf{j}} \in [|\bar{a}|]$. Assuming that $|\bar{a}|_i < n_i$ for every $i \in [d]$, the number of subwords starting at $\bar{\mathbf{j}}$ corresponds to the size of the set

Sequence:	0000001000011000101000111001001011001101001111010101110110111111
Centre	Word
1	000000100001100010100
2	101000111001001011001
3	110011010011110101011
4	000000 0101110110111111

Figure 5.3: Example of how to split the de Bruijn sequence of order 6 between 4 centres. Highlighted parts are the shared subwords between two centres.

$[\bar{\mathbf{j}}, |\bar{a}|]$, equal to $\prod_{i=1}^{d} m_i - |\bar{a}|_i$. This gives the number of shared subwords as being at least $\sum_{\bar{\mathbf{j}} \in [|\bar{a}|]} \prod_{i \in [d]} m_i - |\bar{a}|_i \geq \sum_{j \in [M]} j \geq \frac{M^2}{2}$. Therefore, the distance between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$ is no more than $1 - \frac{M^2}{2N^2}$. □

5.2 Approximating the k-Centre Problem using de-Bruijn Sequences

In this section we provide our first approximation algorithm that requires access to de-Bruijn sequences for the 1D case and to de-Bruijn hypertori for higher dimensions. The main idea is to determine the largest de-Bruijn sequence that can fit into the set of k-centres. As the de Bruijn sequence of order l contains every word in Σ^l as a subword, by representing the de Bruijn sequence of order l in the set of centres we ensure that every necklace shares a subword of length l with the set of k-centres. Therefore, by determining the longest sequence that can be represented by k centres, an upper bound on the distance between the furthest necklace and the set of k-centres is derived.

Definition 23. *A **de Bruijn hypertorus** of order $\overline{\mathbf{n}}$ is a cyclic d-dimensional word $\bar{T}$ containing, as a subword, every word of dimensions $\overline{\mathbf{n}}$ over the alphabet Σ of size q. Further, each such word of dimensions $\overline{\mathbf{n}}$ over the alphabet Σ appears exactly once as a subword of T.*

Lemma 11. *There exists an $O(n \cdot k)$ time algorithm for the k-Centre problem on $\mathcal{N}_q^n$ such that every word in $\mathcal{N}_q^n$ shares a common subword of length at least $\log_q(n \cdot k)$ with one or more centres. Further, no word in $\mathcal{N}_q^n$ is at a distance of more than $1 - \frac{\log_q^2(k \cdot n)}{2n^2}$ from the nearest centre.*

Proof. The high level idea of this algorithm is to spilt a de Bruijn sequence of order λ between the k centres. The motivation behind this approach is to represent every word of length λ as a subword of at least one centre. Note that the length of the de Bruijn sequence of order λ is q^{λ}.

Given a de Bruijn sequence $\bar{s}$, naively splitting $\bar{s}$ into k words may lead to subwords being lost. For example, take the de Bruijn sequence of order 4 over the alphabet $\{a, b\}$ *aaaabaabbababbbb*, dividing this between two words of length 8 results in the samples *aaaabaab* and *bababbbb*, missing the words $aabb, abba$, and $bbaa$. In order to account for this, the sequence is split into centres of size $n - \lambda + 1$, with the final $\lambda - 1$ symbols of the i^{th} centre being shared with the $(i + 1)^{th}$ centre. In this manner, the first centre is generated by taking the first n symbols of the de Bruijn sequence. To ensure that every subword of length λ occurs, the first $\lambda - 1$ symbols of the second centre is the same as the last $\lambda - 1$ symbol of the first centre. Repeating this, the i^{th} centre is the subword of length n starting at position $i(n - \lambda + 1) + 1$ in the de Bruijn sequence. An example of this is given in Figure 5.3.

The leaves the problem of determining the largest value of λ such that $q^{\lambda} \leq k \cdot (n - \lambda + 1)$. The inequality $q^{\lambda} \leq k \cdot (n - \lambda + 1)$ can be rearranged in terms of λ as $\lambda \leq \log_q(k \cdot (n+1) - k \cdot \lambda)$. Noting that λ must be no more than $\log_q(k \cdot n)$, this upper bound on the value of λ can be rewritten as $\log_q(k \cdot (n + 1 - \log_q(k \cdot n))) \approx \log_q(k \cdot n)$. Using Lemma 10, along with $\log_q(k \cdot n)$ as an approximate value of λ gives an upper bound on the distance between between each necklace in $\mathcal{N}_q^n$ and the set of samples of $1 - \frac{\log_q^2(kn)}{2n^2}$.

As the corresponding de Bruijn sequence can be computed in no more than $O(k \cdot n)$ time [90] and the set of samples can be further derived from the sequence in at most $O(k \cdot n)$ time, the total complexity is at most $O(k \cdot n)$. Note that any algorithm that outputs such a set of centres takes at most $\Omega(k \cdot n)$ time. $\square$

Theorem 11. *Problem 11 in 1D can be approximated in $O(n \cdot k)$ time with an approximation factor of $1 + f(n, k)$ where $f(n, k) = \frac{\log_q(k \cdot n)}{n - \log_q(k \cdot n)} - \frac{\log_q^2(kn)}{2n(n - \log_q(kn))}$ and $f(n, k) \to 0$ for $n \to \infty$.*

Proof. Recall from Lemma 9 that the overlap distance is bounded by $1 - \frac{\log_q(k \cdot n)}{n}$. Using the lower bound of $1 - \frac{\log_q^2(kn)}{2n^2}$ given by Lemma 11 gives an approximation ratio of $\frac{1 - \frac{\log_q^2(kn)}{2n^2}}{1 - \frac{\log_q(k \cdot n)}{n}} = \frac{2n^2 - \log_q^2(kn)}{2n^2 - 2n \log_q(kn)} = 1 + \frac{2n \log_q(kn) - \log_q^2(kn)}{2n^2 - 2n \log_q(kn)} = 1 + \frac{\log_q(kn)}{n - \log_q(kn)} - \frac{\log_q^2(kn)}{2n(n - \log_q(kn))}$. Note that $f(n, k) =$

$\frac{2n\log_q(kn)-\log_q^2(kn)}{2n^2-2n\log_q(kn)}$ converges to 0 when $n \to \infty$ for a constant $k < q^n/n$. $\square$

Theorem 12. *Let T be a d-dimensional de Bruijn hyper torus of dimensions $(x, x, \ldots, x)$. There exist k subwords of T that form a solution to the k-centre problem for $\mathcal{N}_q^{(y,y,\ldots,y)}$ with an approximation factor of $1 + f(n,k)$ where $f(n,k) = \frac{\log_q(kN)}{N-\log_q(k\cdot N)} - \frac{\log_q^2(k\cdot N)}{2N(N-\log_q(k\cdot N))}$, $f(n,k) \to 0$, $N \to \infty$.*

Proof. Recall from Lemma 9 that the lower bound on the distance between the centre and every necklace in $\mathcal{N}_q^{\mathbf{n}}$ is $1 - \frac{log_q(k\cdot N)}{N}$. As in Theorem 11, the goal is to find the largest de Bruijn torus that can "fit" into the centres. To simplify the reasoning, the de Bruijn hyper tori here is limited to those corresponding to the word where the length of each dimension is the same. Formally, the de Bruijn hypertori are restricted to be of the dimensions $m_1 = m_2 = \ldots = m_j = \sqrt[j]{N}$ for some $j \in [d]$, giving the total number of positions in the tori as M. Similarly, the centres is assumed to have dimensions $n_1 = n_2 = \ldots = n_d = \sqrt[d]{N}$, giving N total positions.

Observe that the largest torus that can be represented in the set of centres has M positions such that $q^M \leq k \cdot N^{(d-j)/d}(\sqrt[d]{N} - \sqrt[j]{M} + 1)^j$. This can be rewritten to give $M \leq \log_q(k \cdot N^{(d-j)/d}(\sqrt[d]{N} - \sqrt[j]{M} + 1)^j)$. Noting that M is of logarithmic size relative to N, this is approximately equal to $M \leq \log_q(k \cdot N)$. Using Lemma 10, the minimum distance between any necklace in $\mathcal{N}_q^{\mathbf{n}}$ is $1 - \frac{\log_q^2(kN)}{2N^2}$. This is compared to the optimal solution, following the arguments from Theorem 11 giving a ratio of $1 + f(N,k)$ where $f(N,k) = \frac{2\cdot N\log_q(k\cdot N)-\log_q^2(k\cdot N)}{2\cdot N^2-2\cdot N\cdot\log_q(k\cdot N)} = \frac{\log_q(kN)}{N-\log_q(kN)} - \frac{\log_q^2(kN)}{2N(N-\log_q(kN))}$. $\square$

For both cases table providing some explicit examples of the approximation ratio for different values of n and k is given in Table 5.1. While this provides a good starting point for solving the k-Centre problem for $\mathcal{N}_q^{\mathbf{n}}$, results on generating de Bruijn tori are highly limited, focusing on the cases with small dimensions [19, 49, 51, 52, 53]. As such an alternate approach is needed.

5.3 Approximating the k-Centre Problem using Prefix Trees

In this section we present our second approximation algorithm. At a high level our algorithm works as follows. It recursively builds a tree of possible necklace prefixes, starting with the empty string, in a breadth first manner, continuing until there are k such prefixes. Once these prefixes have been generated, the centres are built as necklaces containing

$k\backslash n$	1	2	3	4	5	6	7	8
1	1.0	1.75	1.8242	1.75	1.6657	1.59388	1.53532	1.4875
2	1.0	1.0	4.54496	2.875	2.322	2.04096	1.86822	1.75
3	1.0	1.0	1.0	5.76696	3.17774	2.48677	2.15592	1.95785
4	1.0	1.0	1.0	1.0	4.61912	3.00217	2.43963	2.14583
5	1.0	1.0	1.0	1.0	7.98402	3.65337	2.73732	2.32623
6	1.0	1.0	1.0	1.0	27.84082	4.54496	3.06221	2.50535
7	1.0	1.0	1.0	1.0	1.0	5.88615	3.4276	2.68724
8	1.0	1.0	1.0	1.0	1.0	8.19368	3.84946	2.875

$k\backslash n$	1	2	3	4	5	6	7	8
1	1.0	1.18333	1.19493	1.18333	1.16897	1.15565	1.144	1.13393
2	1.41667	1.41667	1.34509	1.29167	1.25296	1.22393	1.20138	1.18333
3	1.8242	1.59388	1.44797	1.36238	1.30633	1.26659	1.23682	1.2136
4	2.33333	1.75	1.53018	1.41667	1.34644	1.29825	1.2629	1.23575
5	3.09914	1.89704	1.6006	1.46153	1.379	1.32369	1.28372	1.25334
6	4.54496	2.04096	1.66333	1.50021	1.40664	1.34509	1.30113	1.26799
7	8.75423	2.18549	1.72065	1.53449	1.4308	1.36364	1.31615	1.28059
8	1.0	2.33333	1.77396	1.56548	1.45235	1.38007	1.32939	1.29167

Table 5.1: Table of approximation ratio for the algorithm given in Theorem 11 for different values of n and k for a binary alphabet (top) and an alphabet of size 8 (below). Note that when $k \geq q^n$ the approximation ratio is 1 as every necklace can be represented in the set.

these prefixes. Our algorithm relies on the operations of efficiently counting and ranking multidimensional necklaces. However, there are no known algorithms for these operations for bracelets, constrained necklaces, and multidimensional necklaces. Chapters 6, 7 and 8 provide such algorithms. For now, we assume that the number of members of a set of cyclic words sharing a given prefix can be computed.

The k-Centres selection based on a tree of prefixes: At a high level, this prefix algorithm works by finding a set of k prefixes, i.e. a set of k words corresponding to prefixes of the canonical forms of the cyclic words. The algorithm recursively builds the tree of possible prefixes, starting with the empty string, in a breadth first manner, continuing until there are k such prefixes. Once these prefixes have been generated, the centres are built as necklaces containing these prefixes.

This is achieved as follows. At each step there is the set of prefixes $\mathbf{P}$ of length l such that $|\mathbf{P}| < k$. Let $\mathbf{P}'$ be the set of prefixes of length $l+1$. Observe that every prefix in $\mathbf{P}'$ is of the form $\bar{p} : x$ for some $x \in \Sigma$ and $\bar{p} \in \mathbf{P}$. Additionally, for every $\bar{p} \in \mathbf{P}$, there is at least one member of $\mathbf{P}'$, $\bar{p}'$, such that $\bar{p}' = \bar{p} : x$ for some $x \in \Sigma$. Therefore $\mathbf{P}'$ is generated by considering each prefix in $\mathbf{P}$. Given $\bar{p} \in \mathbf{P}$ and $x \in \Sigma$, $\bar{p} : x$ bellongs to $\mathbf{P}'$ if and only if it is the prefix of cyclic word belonging to the relevant set. The set $\mathbf{P}'$ is generated by repeating this process for every $\bar{p} \in \mathbf{P}$, $x \in \Sigma$. Once the size of $\mathbf{P}'$ is greater than k, the algorithm terminates using the prefixes in $\mathbf{P}$ as a basis to construct the set of k centres. For each $\bar{p} \in \mathbf{P}$, a centre is generated by appending an arbitrary subword following the prefix.

This approach is utilised for Multidimensional Necklaces, Bracelets Parikh Vectors solving systems of linear equations, Necklaces with restricted Parikh vectors, Necklaces with forbidden subwords, and Fixed Content Multidimensional Necklaces. In each case the number of words in the relevant setting with a given prefixes must be tracked. This is only possible through use of novel ranking procedures that are discussed in the following chapters.

Theorem 13. *There exists a polynomial-time algorithm to construct k centres of $\mathcal{N}_q^{\mathbf{n}}$, $\mathcal{B}_q^n$, $\mathbf{P}(n, A, \overline{\mathbf{C}})$, $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$, $\mathcal{N}_q^n(\mathcal{F})$ or $\mathcal{N}_{\overline{\mathbf{P}}}^{\mathbf{n}}$ that is an approximation of the optimal solution by a factor of $1+\epsilon$ where $\epsilon = \frac{\log_q(kN)}{N - \log_q(kN)} - \frac{\log_q^2(k)}{2N(N-\log_q(kN))}$ and $\epsilon \to 0$ for $N \to \infty$.*

Proof. Let λ be the length of the prefixes at the termination of the algorithm. To determine the length of λ, observe that each centre corresponds to a prefix of length λ. Therefore, this becomes the problem of determining the largest value of λ such that the size of the set is less than k. An upper bound on the size of the set of necklace prefixes of length λ is q^{λ}. As this must be less than or equal to k, this is rewritten as $\lambda \leq \log_q(k)$. Using Lemma 10, an upper bound on the distance between each word and the nearest centre is $1 - \frac{\log_q(k)(\log_q(k)+1)}{2N^2}$, which is bounded by $\frac{\log_q^2(k)}{2N^2}$. Lemma 9 gives a lower bound on the distance between every necklace in $\mathcal{N}_q^{\mathbf{n}}$ and the nearest centre in the centre of $1 - \frac{\log_q(k \cdot N)}{N}$. Therefore, this algorithm gives an approximation of the optimal solution by a factor of $\frac{1 - \frac{\log_q^2}{k} 2N^2}{1 - \frac{\log_q^2(k)}{2N^2}}$, which is simplified to a factor of $1 + \frac{2N \log_q(kN) - \log_q^2(k)}{2N^2 - 2N \log_q(kN)}$. Note that $\epsilon = \frac{2N \log_q(kN) - \log_q^2(k)}{2N^2 - 2N \log_q(kN)}$ $= \frac{\log_q(kN)}{N - \log_q(kN)} - \frac{\log_q^2(k)}{2N(N - \log_q(kN))}$ converges to 0 when $n \to \infty$ for any fixed k. To show that the method terminates in polynomial time, we note that the time to compute the number of necklaces with a given prefix can be determined as follows:

- The number of bracelets sharing a given prefix can be computed in $O(n^4 \cdot q^2)$ time (shown in Lemma 28).
- The number of Parikh vectors solving the system of linear equations $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$ can be found in $O(C \cdot n \cdot q^2)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ (shown in Lemma 30).
- The number of necklaces with Parikh vectors solving the system of linear equations $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$ sharing a given prefix can be found in $O(C \cdot n^2 \cdot q$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ (shown in Lemma 33).
- The number of necklaces containing no forbidden subwords from a set $\mathcal{F}$ sharing a given prefix can be found in $O(\log^2(n) \cdot n^8 \cdot |\mathcal{F}|^2)$ time (shown in Lemma 42).
- The number of multidimensional necklaces sharing a given prefix can be found in $O(N^5)$ time (shown in Lemma 55).
- The number of fixed-content multidimensional necklaces sharing a given prefix can be found in $O(n^{6+q})$ time (shown in Lemma 56).

As an additional cost, for every $i \leq \lambda + 1$, at most k centres are checked. To determine the longest λ, let there be p_i prefixes of length i. Observe that there are at least $p_i + q - 1$ words of length $i = 1$. Therefore the number of prefixes of length i is at least $(i+1)q - i$. Therefore the longest length is $\frac{k-q}{q-1}$. Thus the maximum number prefixes that need to be checked is $k \cdot \frac{k-q}{q-1}$, giving an additional cost of $O(k^2)$ to determining the set of k samples. $\square$

5.4 The k-Centre Problem for Crystal Structure Prediction

This section considers in more detail how to apply the k-centre problem to the context of CSP. There are two primary methods that the k-centre technique may be applied to, either generating a set of starting samples which may then be used by an alternative method such as MC-EMMA or FUSE or using the k-centre problem to directly solve CSP.

In both cases, the overlap distance is used for comparison. As mentioned in Section 5.1, the main motivation behind this distance is to sample a large number of local substructures. These are prioritised as local interactions correspond to the majority of the energy in crystal structures.

In the first case, these local structures can be used to inform the CSP algorithm potentially good local structures. Many current algorithms use a purely random sample, which can lead to structures corresponding to the same global structure, for instance by not accounting for equivalence under cyclic shifts. Additionally, by selecting a diverse starting population, the algorithms have a greater degree of information allowing enabling better decisions to be made regarding the global composition. Figures 5.4 and 5.5 provide an example of using the k-centre problem for Parikh vectors to provide a set of starting structures for the MC-EMMA system.

The case of directly solving CSP is more involved. There are two possible high level directions that the k-centre approach can be used for solving CSP. The first of these is the "pure" version, in which the goal is to choose a set of centres such that the optimal solution is *close* to one of the centres. The idea in this case would be to first calculate the energy based on the unit cells explicitly represented by the samples, then to use a more powerful technique such as *energy relaxation* on a small number of good structures to determine the global optimal. Energy relaxation can be thought of as determining the local minimum that can be reached by adjusting the positions ions from some fixed unit cell through a series of small moves. While this technique is highly powerful, it is very computationally expensive, meaning that it is best used sparingly.

A second more computational approach is to iteratively take samples from progressively smaller sets. This can be thought of as "zooming in" on good potential structures. The high level idea behind this approach is to take a comparatively small number of initial samples, followed by choosing a second set of samples close to the centres with a low energy. The motivation behind this approach would be to attempt to avoid costly energy relaxation calculations while still being able to find the global minimum.

Input

Chemical Formula: $4 \cdot (Y + 2 \cdot B + 2 \cdot C + 5 \cdot O + 13 \cdot Fe)$

Layers:

P_1	$0 \cdot Y, 4 \cdot B, 0 \cdot C, 0 \cdot O, 4 \cdot Fe$	N_1	$0 \cdot Y, 0 \cdot B, 0 \cdot C, 4 \cdot O, 8 \cdot Fe$
P_2	$0 \cdot Y, 0 \cdot B, 4 \cdot C, 0 \cdot O, 4 \cdot Fe$	N_2	$0 \cdot Y, 0 \cdot B, 0 \cdot C, 4 \cdot O, 4 \cdot Fe$
P_3	$4 \cdot Y, 0 \cdot B, 0 \cdot C, 0 \cdot O, 0 \cdot Fe$	N_3	$0 \cdot Y, 0 \cdot B, 0 \cdot C, 4 \cdot O, 4 \cdot Fe$
P_4	$4 \cdot Y, 0 \cdot B, 0 \cdot C, 0 \cdot O, 4 \cdot Fe$	N_4	$0 \cdot Y, 0 \cdot B, 0 \cdot C, 4 \cdot O, 4 \cdot Fe$
P_5	$2 \cdot Y, 0 \cdot B, 2 \cdot C, 0 \cdot O, 4 \cdot Fe$	N_5	$0 \cdot Y, 0 \cdot B, 0 \cdot C, 4 \cdot O, 4 \cdot Fe$
P_6	$1 \cdot Y, 0 \cdot B, 3 \cdot C, 0 \cdot O, 4 \cdot Fe$		
P_7	$3 \cdot Y, 0 \cdot B, 1 \cdot C, 0 \cdot O, 4 \cdot Fe$		
P_8	$2 \cdot Y, 0 \cdot B, 2 \cdot C, 0 \cdot O, 0 \cdot Fe$		
P_9	$1 \cdot Y, 0 \cdot B, 3 \cdot C, 0 \cdot O, 0 \cdot Fe$		
P_{10}	$3 \cdot Y, 0 \cdot B, 1 \cdot C, 0 \cdot O, 0 \cdot Fe$		

Figure 5.4: An example of the input for the MC-EMMA model of CSP. The input consists of a chemical formula (top) and set of precomputed three dimensional layers with either a positive charge (labelled P_1 to P_{10}) or negative charge (labelled N_1 to N_5). By satisfying the equation, the charge of the global structure will be 0. Note that despite N_2, N_3, N_4 and N_5 having the same chemical formula, each layer will represent a different placement of the ions, leading to different possibilities of the global composition.

Centre	Necklace	Parikh Vector
1	$P_1N_1P_1N_1P_5N_1P_7N_2P_7N_2$	$(2,0,0,0,1,0,2,0,0,0,3,2,0,0,0)$
2	$P_1P_1P_2P_5P_5N_1N_1N_1N_2N_2$	$(2,1,0,0,2,0,0,0,0,0,3,2,0,0,0)$
3	$P_1P_7N_1P_8N_1P_7P_1N_1N_2N_1$	$(2,0,0,0,0,0,2,1,0,0,4,1,0,0,0)$
4	$P_1P_5N_1P_2N_1P_1N_1P_5N_1N_4$	$(2,1,0,0,1,0,0,1,0,0,4,0,0,1,0)$
5	$P_1N_1P_1N_1P_2N_1P_8N_1P_8N_1$	$(2,1,0,0,0,0,0,2,0,0,5,0,0,0,0)$
6	$P_1P_9P_7N_1P_1N_5N_1P_5N_1N_1$	$(2,0,0,0,1,0,1,0,1,0,4,0,0,0,1)$
7	$P_1P_8P_9N_1N_1N_1P_1P_7N_1N_1$	$(2,0,0,0,0,0,1,1,1,0,5,0,0,0,0)$
8	$P_1N_1P_5N_1P_1N_1P_9P_9N_1N_1$	$(2,0,0,0,1,0,0,0,2,0,5,0,0,0,0)$
9	$P_1P_5P_1P_7P_7N_1N_1N_1N_3N_3$	$(2,0,0,0,1,0,2,0,0,0,3,0,2,0,0)$
10	$P_1P_1N_5N_4P_2P_5P_5N_1N_1N_1$	$(2,1,0,0,2,0,0,0,0,0,3,0,0,1,1)$

Figure 5.5: An example of the k-centre problem for the MC-EMMA setting using the input from Figure 5.4. In this case each layer from the input is represented by some symbol. The output is a set of 10 centres over this new alphabet represented by necklaces each with a unique Parikh vector such that the sum of elements equals to goal formula. By choosing both diversity in terms of Parikh vectors and local structures, a larger number of local substructures can be investigated than would be possible via a random sampling. These necklaces can be transformed into crystal structures by replacing each symbol with the corresponding precomputed slice.

Chapter 6

Ranking and Unranking Bracelets

Following the k-centre algorithms provided in Chapter 5, it is necessary to show how to rank bracelets in polynomial time. As the problem of ranking bracelets is a major challenge in its own right, it is worth laying out a road map for the remainder of this chapter. Section 6.1 provides a high level overview for the main result of this chapter, namely the approach taken to rank bracelets. Section 6.2 provides some key preliminary results. Sections 6.3 and 6.4 give auxiliary results on ranking palindromic and enclosing bracelets, as defined in Chapter 2. Finally Section 6.5 summarises the main contribution from this chapter.

6.1 Ranking Bracelets

In this section we present the main result of this chapter: the first efficient algorithm for ranking bracelets. In what follows, we tacitly assume that we are ranking a word $\bar{v}$ of length n. The time-complexity of the ranking algorithm is $O(q^2 \cdot n^4)$, where q is the size of the alphabet and n is the length of the considered bracelets. The key part of the algorithm is to compute the rank of the word $\bar{v}$ with respect to the set of bracelets by finding three other ranks: the rank over all necklaces, the rank over palindromic necklaces, and the rank over enclosing apalindromic necklaces.

A bracelet can correspond to two apalindromic necklaces, or to exactly one palindromic necklace. If a bracelet $\hat{\mathbf{b}}$ corresponds to two necklaces $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$, then it is important to take into account the lexicographical positions of these two necklaces $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ with respect to a given word $\bar{v}$. There are three possibilities: $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ could be less than $\bar{v}$; $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ encloses $\bar{v}$, e.g. $\tilde{\mathbf{l}}_b < \bar{v} < \tilde{\mathbf{r}}_b$, or both of necklaces $\tilde{\mathbf{l}}_b$ and $\tilde{\mathbf{r}}_b$ are greater than $\bar{v}$. This is visualised in Figure 6.1. Therefore the number of bracelets smaller than a given word w can be calculated by adding the number of palindromic necklaces less than $\bar{v}$, enclosing bracelets smaller than $\bar{v}$ and half of all other apalindromic and non-enclosing necklaces smaller than $\bar{v}$. Let us define the following notation is used for the rank of $\bar{v} \in \Sigma^n$ for sets of bracelets and necklaces.

- $RN(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *necklaces* of length n over Σ.
- $RP(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *palindromic necklaces* over Σ.
- $RB(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *bracelets* of length n over Σ.
- $RE(\bar{v})$ denotes the rank of $\bar{v}$ with respect to the set of *bracelets enclosing* $\bar{v}$.

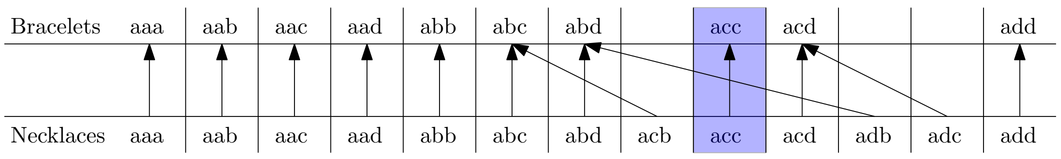

Figure 6.1: In this example the top line represents the set of bracelets and the bottom line the set of necklaces, with arrows indicated which necklace corresponds to which bracelet. Assuming we wish to rank the word *acc* (highlighted), *abc* and *acb* are apalindromic necklaces smaller than *acc*, while *abd* encloses *acc*. All other necklaces are palindromic.

In Lemma 12 below, we show that $RB(\bar{v})$ can be expressed via $RN(\bar{v})$, $RP(\bar{v})$ and $RE(\bar{v})$. The problem of computing $RN(\bar{v})$ has been solved in quadratic time [92], so the goal of the chapter is to design efficient procedures for computing $RP(\bar{v})$ and $RE(\bar{v})$.

Lemma 12. *The rank of a word $\bar{v} \in \Sigma^n$ with respect to the set of bracelets of length n over the alphabet Σ is given by $RB(\bar{v}) = \frac{1}{2}\left(RN(\bar{v}) + RP(\bar{v}) + RE(\bar{v})\right)$.*

Proof. Simply dividing the number of necklaces by 2 undercounts the number of bracelets, while doing nothing overcounts the number of bracelets. Therefore to get the correct number of bracelets, those bracelets corresponding to only 1 necklace must be accounted for. A bracelet $\hat{\mathbf{a}}$ corresponds to 2 necklaces smaller than $\bar{v}$ if and only if $\hat{\mathbf{a}}$ does not enclose $\bar{v}$ and $\hat{\mathbf{a}}$ is apalindromic. Therefore the number of bracelets corresponding to 2 necklaces is $\frac{1}{2}\left(RN(\bar{v}) - RP(\bar{v}) - RE(\bar{v})\right)$. The number of bracelets enclosing $\bar{v}$ is equal to $RE(\bar{v})$. The number of bracelets corresponding to palindromic necklaces is equal to $RP(\bar{v})$.

Therefore the total number of bracelets is $\frac{1}{2}\left(RN(\bar{v}) - RP(\bar{v}) - RE(\bar{v})\right) + RP(\bar{v}) + RE(\bar{v}) = \frac{1}{2}\left(RN(\bar{v}) + RP(\bar{v}) + RE(\bar{v})\right)$. □

Lemma 12 provides the basis for ranking bracelets. Theorem 14 uses Lemma 12 to get the complexity of the ranking process. The remainder of this chapter proves Theorem 14, starting with the complexity of ranking among palindromic necklaces in Section 6.3 followed by the complexity of ranking enclosing bracelets in Section 6.4.

Theorem 14. *Given a word $\bar{v} \in \Sigma^n$, the rank of $\bar{v}$ with respect to the set of bracelets of length n over the alphabet Σ, $RB(\bar{v})$, can be computed in $O(q^2 \cdot n^4)$ time.*

The remainder of this chapter proves Theorem 14. For simplicity, the word $\bar{v}$ is assumed to be a necklace representation. It is well established how to find the lexicographically largest necklace smaller than or equal to some given word. Such a word can be found in quadratic time using an algorithm form [92]. Note that the number of necklaces less than or equal to $\bar{v}$ corresponds to the number of necklaces less than or equal to the lexicographically largest necklace smaller than $\bar{v}$. From Lemma 12 it follows that to rank $\bar{v}$ with respect to the set of bracelets, it is sufficient to rank $\bar{v}$ with respect to the set of necklaces, palindromic necklaces, and enclosing bracelets. The rank with respect to the set of palindromic necklaces, $RP(\bar{v})$ can be computed in $O(q \cdot n^3)$ using the techniques given in Theorem 16 in Section 6.3. The rank with respect to the set of enclosing bracelets, $RE(\bar{v})$ can be computed in $O(q^2 \cdot n^4)$ as shown in Theorem 17 in Section 6.4. As each of these steps can be done independently of each other, the total complexity is $O(q^2 \cdot n^4)$.

This complexity bound is a significant improvement over the naive method of enumerating all bracelets, requiring exponential time in the worst case. New intuition is provided to rank the palindromic and enclosing cases. The main source of complexity for the problem of ranking comes from having to consider the lexicographic order of the word under reflection. New combinatorial results and algorithms are needed to count the bracelets in these cases.

Before showing in detail the algorithmic results that allow bracelets to be efficiently ranked, it is useful to discus the high level ideas. Lemma 12 shows our approach to ranking bracelets by dividing the problem into the problems of ranking necklaces, palindromic necklaces and enclosing bracelets. For both palindromic necklaces and enclosing bracelets, we derive a *canonical form* using the combinatorial properties of these objects.

Using these canonical forms, the number of necklaces smaller than $\bar{v}$ is counted in an iterative manner. In the palindromic case, this is done by counting the number of necklaces greater than $\bar{v}$, and subtracting this from the total number of palindromic necklaces. In the enclosing case, this is done by directly counting the number of necklaces smaller than $\bar{v}$. For both cases, the counting is done by way of a tree comprised of the set of all prefixes of words of the canonical form. By partitioning the internal vertices of the trees based on the number of children of the vertices, the number of words of the canonical form may be derived in an efficient manner, forgoing the need to explicitly generate the tree. This allows the size of these partitions to be computed through a dynamic programming approach. It follows from these partitions how to count the number of leaf nodes, corresponding to the canonical form.

Theorem 15. *The z^{th} bracelet of length n over Σ can be computed in $O(n^5 \cdot q^2 \cdot \log(q))$.*

Proof. The unranking process is done through a binary search using the ranking algorithm as a black box. Let $\bar{\alpha}$ be a word which is the bracelet representation of the z^{th} bracelet. The value of $\bar{\alpha}$ is determined iteratively, starting with the first symbol and working forwards. The first symbol of $\bar{\alpha}$ is determined preforming a binary search over $\bar{\alpha}$. For $x \in \bar{\alpha}$, the words $x1^{n-1}$ and xq^{n-1} are generated, where 1 is the smallest symbol in Σ and q the largest. If $RB(x1^{n-1}) \leq z \leq RB(xq^{n-1})$, then the first symbol of $\bar{\alpha}$ is x, otherwise the new value of x is chosen by standard binary search, being greater than x if $z > RB(xq^{n-1})$ and less than x if $z < RB(x1^{n-1})$. The i^{th} symbol of $\bar{\alpha}$ is done in a similar manner, generating the words $\bar{\alpha}_{[1,i-1]}x1^{n-i-1}$ and $\bar{\alpha}_{[1,i-1]}xq^{n-i-1}$, converting $\bar{\alpha}_{[1,i-1]}x1^{n-i-1}$ to a necklace representation using Algorithm 1 due to Sawada and Williams [92]. Repeating this for all n symbols leaves $\bar{\alpha}$ as being the bracelet representation of the z^{th} smallest bracelet, i.e. the bracelet with $z - 1$ smaller bracelets. As the binary search takes $\log(q)$ operations for each of the n symbols, requiring $O(q^2 \cdot n^4)$ time to rank for each symbol at each position. Therefore the total complexity is $O(n^5 \cdot q^2 \cdot \log(q))$ time. □

6.2 Bounding Subwords

For both the palindromic and enclosing cases the number of necklaces smaller than $\bar{v} \in \Sigma^n$ is computed by iteratively counting the number of words of length n for which no subword is smaller than $\bar{v}$. The set of such words, denoted by $\mathbf{S}_n$, is analysed iteratively as well, since it can have an exponential size. In order to relate $\mathbf{S}_i$ to $\mathbf{S}_{i+1}$, we split $\mathbf{S}_i$ into parts using the positions of length i subwords of $\bar{v}$ with respect to the lexicographic order on

$\mathbf{S}_i$. Informally, every $\bar{w} \in \mathbf{S}_i$ can be associated with the unique lower bound from $\mathbf{S}(\bar{v}, i)$, which is used to identify the parts leading us to the following definition.

Definition 24. *Let $\bar{w}, \bar{v} \in \Sigma^*$ where $|\bar{w}| \leq |\bar{v}|$. The word $\bar{w}$ is **bounded** (resp. strictly bounded) by $\bar{s} \sqsubseteq_{|\bar{w}|} \bar{v}$, if $\bar{s} \leq \bar{w}$ (resp. $\bar{s} < \bar{w}$) and there is no $\bar{u} \sqsubseteq_{|\bar{w}|} \bar{v}$ such that $\bar{s} < \bar{u} \leq \bar{w}$.*

The aforementioned parts $\mathbf{S}_i(\bar{s})$ contain all words $\bar{w} \in \mathbf{S}_i$ such that $\bar{s} \sqsubseteq_{|\bar{w}|} \bar{v}$. The key observation is that words of the form $x\bar{w}$ for all $\bar{w} \in \mathbf{S}_i$ and some fixed symbol $x \in \Sigma$ belong to the same set $\mathbf{S}_{i+1}(\bar{s}')$, where $\bar{s}' \sqsubseteq \bar{v}$. The same is true for words of the form $\bar{w}x$. Thus, we can compute the corresponding $\bar{s}'$ for all pairs of $\bar{s}$ and x in order to derive sizes of $\mathbf{S}_{i+1}(\bar{s}')$. Moreover, this relation between $\bar{s}$, x, and $\bar{s}'$ is independent of i allowing us to store this information in two $n^2 \times k$ arrays XW and WX. Both arrays are indexed by the words $\bar{s} \sqsubseteq \bar{v}$ and characters $x \in \Sigma$. Given a word $\bar{w}$ strictly bounded by $\bar{s}$, $XW[\bar{s}, x]$ contains the word $\bar{s}' \sqsubseteq_{|\bar{s}|+1} \bar{v}$ strictly bounding $x\bar{w}$. Similarly, $WX[\bar{s}, x]$ contains the word $\bar{s}' \sqsubseteq_{|\bar{s}|+1} \bar{v}$ strictly bounding $\bar{w}x$. By precomputing these arrays, the cost of determining these words can be avoided during the ranking process. In order to compute these arrays, the following technical lemmas are needed.

Lemma 13. *Let $\bar{w}, \bar{v} \in \Sigma^*$, $|\bar{w}| < |\bar{v}|$, let $x \in \Sigma$ and let $\bar{s} \sqsubseteq_{|\bar{w}|} \bar{v}$ be the subword of $\bar{v}$ that bounds $\bar{w}$. The word $\bar{s}' \sqsubseteq \bar{v}$ bounds $x\bar{w}$ if and only if $\bar{s}'$ bounds $x\bar{s}$.*

Proof. Let $\bar{s}' \sqsubseteq \bar{v}$ bound $x\bar{w}$. Since $\bar{s} \leq \bar{w}$, we have $x\bar{s} \leq x\bar{w}$. For the sake of contradiction assume that $x\bar{s}$ is bounded by $\bar{u} < \bar{s}'$. If $\bar{u}_1 < x$ then $\bar{s}'_1 = x$ as for any smaller value of $\bar{s}'_1$, $\bar{u}$ would not bound $x\bar{w}$. Under this assumption $\bar{s} < \bar{s}'_{[2,|\bar{s}'|]} \leq \bar{w}$, in which case $\bar{s}'_{[2,|\bar{s}'|]}$ would bound $\bar{w}$, contradicting this assumption. If $\bar{u}_1 = x$, then again $\bar{s} < \bar{s}'_{[2,|\bar{s}'|]} < \bar{w}$, in which case $\bar{s}'_{[2,|\bar{s}'|]}$ bounds $\bar{w}$ contradicting the original assumption that $\bar{s}$ bounds $\bar{w}$.

In the other direction, let $\bar{s}'$ bound $x\bar{s}$. If $\bar{s}'$ does not bound $x\bar{w}$ then there must exist some word $\bar{u}$ bounding $x\bar{w}$. As $x\bar{s} < \bar{u} < x\bar{w}$, $\bar{u}_1 = x$ hence $\bar{u} = x\bar{u}'$. Therefore $\bar{u}'$ bounds $\bar{w}$, contradicting our original assumption. Hence $\bar{s}'$ bounds $x\bar{w}$ if and only if $\bar{s}'$ bounds $x\bar{s}$ where $\bar{s}$ bounds $\bar{w}$. □

Lemma 14. *Let $\bar{w}, \bar{v} \in \Sigma^*$, let $x \in \Sigma$ and let $\bar{s} \sqsubseteq \bar{v}$ be the subword of $\bar{v}$ that bounds $\bar{w}$. Let $\bar{s}' \sqsubseteq \bar{v}$ bound $\bar{w}x$. Either $\bar{s}'$ bounds $\bar{s}x$, or $\bar{s}' = \bar{s}y$ for $y > x$.*

Proof. Let $\bar{u}$ bound $\bar{s}x$. If $\bar{u} \neq \bar{s}'$ then as $\bar{w}x \geq \bar{s}' > \bar{s}x \geq \bar{u}$, if $\bar{s}'_{[1,|\bar{s}|]} > \bar{s}$ then $\bar{s}'_{[1,|\bar{s}|]}$ must bound $\bar{w}$, contradicting the assumption that $\bar{s}$ bounds $\bar{w}$. Therefore the only possible value of $\bar{s}' > \bar{s}x$ is when $\bar{s}' = \bar{s}y$ for some $y > x$. □

Lemma 15. *Let $\bar{w}, \bar{u}, \bar{v} \in \Sigma^*$, let $x \in \Sigma$ and let $\bar{s} \sqsubseteq \bar{v}$ be the subword of $\bar{v}$ that strictly bounds both $\bar{w}$ and $\bar{u}$. The word $\bar{s}' \sqsubseteq \bar{v}$ which bounds $\bar{w}x$ also bounds $\bar{u}x$.*

Proof. For the sake of contradiction assume $\bar{u}x$ is bounded by $\bar{t}$. This implies that $\bar{s}' < \bar{w}x < \bar{t} \leq \bar{u}x$. Following Lemma 14, $\bar{t} = \bar{s}y$ for $y > x$. However, as $\bar{w} > \bar{s}$, $\bar{t}$ must be less than $\bar{w}$ and hence $\bar{t}$ would be a better bound for $\bar{w}$. □

Proposition 9. *Let $\bar{v} \in \Sigma^n$. The array $XW[\bar{s} \sqsubseteq \bar{v}, x \in \Sigma]$ such that $XW[\bar{s}, x]$ strictly bounds $x\bar{w}$ for every $\bar{w}$ strictly bounded by $\bar{s}$ can be computed in time $O(q \cdot n^3 \cdot \log(n))$.*

Proof. Given some pair of arguments $\bar{s} \sqsubseteq \bar{v}, x \in \Sigma$, the word bounding $x\bar{s}$ can be found through a binary search on $\mathbf{S}(\bar{v}, |\bar{s}|+1)$. As each comparison takes at most $O(n)$ operations, and at most $\log(n)$ comparisons are needed, each entry can be computed in $O(n \log(n))$ operations. As there are $O(n^2)$ subwords of $\bar{v}$ and q characters in Σ, there is at most $O(q \cdot n^3 \log(n))$ operations needed. □

Proposition 10. *Let $\bar{v} \in \Sigma^n$. The array $WX[\bar{s} \sqsubseteq \bar{v}, x \in \Sigma]$, such that $WX[\bar{s}, x]$ strictly bounds $\bar{w}x$ for every $\bar{w}$ strictly bounded by $\bar{s}$, can be computed in $O(q \cdot n^3 \cdot \log(n))$ time.*

Proof. For some pair pair of arguments $\bar{s} \sqsubseteq \bar{v}, x \in \Sigma$, let $\bar{w}$ be the smallest word greater than $\bar{s}$. The word bounding $\bar{w}x$ can be found through a binary search of $\mathbf{S}(\bar{v}, |\bar{s}| + 1)$. Following Lemma 15, given any word $\bar{u}$ strictly bounded by $\bar{s}$, $\bar{u}x$ is also bounded by the same word bounding $\bar{w}x$. As in Proposition 9, each comparison takes at most $O(n)$ operations, with the search requiring at most $\log(n)$ comparisons. As there are $n^2 \cdot q$ arguments, at most $O(q \cdot n^3 \log(n))$ operations are needed to compute every value of WX. □

6.3 Computing the Rank $RP(\bar{v})$

To rank palindromic necklaces, it is crucial to analyse their combinatorial properties. This section focuses on providing results on determining unique words representing palindromic necklaces. We study two cases depending on whether the length n of a palindromic necklace is even or odd. The reason for this division can be seen by considering examples of palindromic necklaces. If equivalence under the rotation operation is not taken into account, then a word is palindromic if $\bar{w} = \bar{w}^R$. If the length n of $\bar{w}$ is odd, then if $\bar{w} = \bar{w}^R$, $\bar{w}$ can be written as $\bar{\phi}x\bar{\phi}^R$, where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$. For example, the word *aaabaaa* is equal to $\bar{\phi}x\bar{\phi}^R$, where $\bar{\phi} = aaa$ and $x = b$. If the length n of w is even, then if $\bar{w} = \bar{w}^R$, $\bar{w}$ can be written as $\bar{\psi}\bar{\psi}^R$, where $\bar{\psi} \in \Sigma^{n/2}$. For example the word $aabbaa$ is equal to $\bar{\psi}\bar{\psi}^R$, where $\bar{\psi} = aab$.

Once rotations are taken into account, the characterisation of palindromic necklaces becomes more difficult. It is clear that any necklace $\tilde{\mathbf{a}}$ that contains a word of the form $\bar{\phi}x\bar{\phi}^R$ or $\bar{\phi}\bar{\phi}^R$ is palindromic. However this check does not capture every palindromic necklace. Let us take, for example, the necklace $\tilde{\mathbf{a}} = ababab$, which contains two words $ababab$ and $bababa$. While $ababab$ can neither be written as $\bar{\phi}x\bar{\phi}^R$ nor $\bar{\phi}\bar{\phi}^R$, it is still palindromic as $\langle ababab^R \rangle = \langle bababa \rangle = ababab$. Therefore a more extensive test is required. As the structure of palindromic words without rotation is different depending on the length being either odd or even, it is reasonable to split the problem of determining the structure of palindromic necklaces into the cases of odd and even length.

The number of palindromic necklaces are counted by computing the number of these characterisations. This is done by constructing trees containing every prefix of these characterisations. As each vertex corresponds to the prefix of a word, the leaf nodes of these trees correspond to the words in the characterisations. By partitioning the tree in an intelligent manner, the number of leaf nodes and therefore number of these characterisations can be computed. In the odd case this corresponds directly to the number of palindromic necklaces, while in the even case a small transformation of these sets is needed.

6.3.1 Odd Length Palindromic Necklaces

Starting with the odd-length case, Proposition 11 shows that every palindromic necklace of odd length contains **exactly one word** that can be written as $\bar{\phi}x\bar{\phi}^R$ where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$. This fact is used to rank the number of bracelets by constructing a tree representing every prefix of a word of the form $\bar{\phi}x\bar{\phi}^R$ that belongs to a bracelet greater than $\bar{v}$.

Proposition 11. *A necklace $\tilde{\mathbf{w}}$ of odd length n is palindromic if and only if there exists exactly one word $\bar{u} = \bar{\phi}x\bar{\phi}^R$ such that $\bar{u} \in \tilde{\mathbf{w}}$, where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$.*

Proof. Let $\bar{v} \in \tilde{\mathbf{w}}$. If $\bar{v}$ is of the form $\bar{\phi}x\bar{\phi}^R$, then clearly we have that $\bar{v} = \bar{v}^R$. In the other direction, for the sake of contradiction assume $\tilde{\mathbf{w}}$ is a palindromic necklace of odd length n such that no word $\bar{v} \in \tilde{\mathbf{w}}$ is of the form $\bar{\phi}x\bar{\phi}^R$. Note that the cardinality of $\tilde{\mathbf{w}}$ is equal the period of the words in $\tilde{\mathbf{w}}$. As the length of the words in $\tilde{\mathbf{w}}$ is odd, so to must be the length of the period. Given a word $\bar{v} \in \tilde{\mathbf{w}}$, if $\bar{v} \neq \bar{v}^R$ then the size of $\tilde{\mathbf{w}}$ is equal to $|\tilde{\mathbf{w}} \setminus \{\bar{v}, \bar{v}^R\}| + 2$. As the size of $\tilde{\mathbf{w}}$ is odd, there must be at least one word $\bar{v} \in \tilde{\mathbf{w}}$ where $\bar{v} = \bar{v}^R$. For $\bar{v} = \bar{v}^R$, $\bar{v}_1 = \bar{v}_n, \bar{v}_2 = \bar{v}_{n-1}, \ldots, \bar{v}_{\frac{n-1}{2}} = \bar{v}_{\frac{n+3}{2}}$. Therefore this word can be expressed as $\bar{\phi}x\bar{\phi}^R$ where $\bar{\phi} = \bar{v}_{[1,(n-1)/2]}$ and $x = \bar{v}_{(n+1)/2}$.

For the remainder of this proof $\bar{u}_i$ is used to denote the character at position $(i \bmod n)+1$ in the word $\bar{u}$. For the sake of contradiction, assume that there exists some pair of words $\bar{u}, \bar{v} \in \tilde{\mathbf{w}}$ such that $\bar{u} \neq \bar{v}$ and both $\bar{u} = \bar{u}^R$ and $\bar{v} = \bar{v}^R$. As both $\bar{u}$ and $\bar{v}$ belong to the same necklace class, there must exist some rotation r such that $\langle \bar{u} \rangle_r = \bar{v}$. Further, as $\bar{v} = \bar{v}^R$, $\langle \bar{u} \rangle_r = \bar{v}^R$. Therefore, $\bar{u}_{r+i} = \bar{v}_i, \bar{u}_{n-1-r+i} = \bar{v}_i$, $\bar{u}_{r+i} = \bar{v}_{n-i-1}$, and $\bar{u}_{n-1-r+i} = \bar{v}_{n-i-1}$. Further $\bar{u}_i = \bar{u}_{n-i+1}$ and $\bar{v}_i = \bar{v}_{n-i+1}$. Therefore $\bar{v}_i = \bar{u}_{r+i} = \bar{u}_{n-r-i+1} = \bar{v}_{2n-2r-i-1} = \bar{v}_{n-(2n-2r-i-1)-1} = \bar{u}_{3r-n+i} = \bar{u}_{3r+i}$. Therefore $\bar{u}_i = \bar{u}_{2r+i}$ implying that $\bar{v} = \langle \bar{v} \rangle_{2r}$. Therefore the period of $\bar{u}$ must be equal to some common divisor of $2r$ and n. As the length of n is odd, the greatest divisor equals to $GCD(r, n)$. As such the period must be a factor of r, meaning that $\bar{u} = \langle \bar{u} \rangle_r = \bar{v}$, contradicting the assumption that $\bar{u} \neq \bar{v}$. Therefore there is exactly one word in $\tilde{\mathbf{w}}$ of the form $\bar{\phi}x\bar{\phi}^R$. □

Corollary 5. *The number of palindromic necklaces of odd length n over Σ equals $q^{(n+1)/2}$.*

Proof. It follows from Proposition 11 that for every palindromic necklace $\tilde{\mathbf{w}}$ of length n, there exists exactly one word $\bar{\phi} \in \Sigma^*$ and symbol $x \in \Sigma$ such that $\bar{\phi}x\bar{\phi}^R$. Hence, the number of palindromic necklaces equals the number of words of the form $\bar{\phi}x\bar{\phi}^R$ with length n. Note that for the length of $\bar{\phi}x\bar{\phi}^R$ to be n, the length of $\bar{\phi}$ must be $\frac{n-1}{2}$. Therefore the number of values of $\bar{\phi}$ is $q^{(n-1)/2}$. As there are q values of x, the number of values of $\bar{\phi}x\bar{\phi}^R$ is $q^{(n+1)/2}$. □

The problem now becomes to rank a word $\bar{v}$ with respect to the odd length palindromic necklaces utilising their combinatorial properties. Let $\bar{v} \in \Sigma^n$ be a word of odd length n. We define the set $\mathcal{PO}(\bar{v})$, where $\mathcal{PO}$ stands for palindromic odd length. The set $\mathcal{PO}(\bar{v})$ contains one word representing each palindromic bracelet of odd length n that is greater than $\bar{v}$.

$$\mathcal{PO}(\bar{v}) := \left\{ \bar{w} \in \Sigma^n : \bar{w} = \bar{\phi}x\bar{\phi}^R,\ \text{where } \langle \bar{w} \rangle > \bar{v},\ \bar{\phi} \in \Sigma^{(n-1)/2},\ x \in \Sigma \right\}.$$

As each word corresponds to a unique palindromic necklace of length n greater than $\bar{v}$, and every palindromic necklace greater than $\bar{v}$ corresponds to a word in $\mathcal{PO}(\bar{v})$, the number

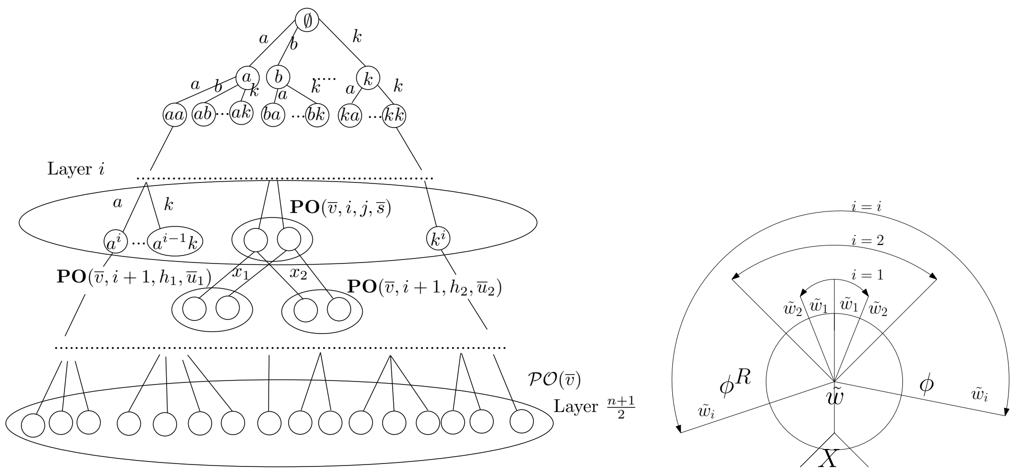
Layer i
$\mathbf{PO}(\bar{v}, i, j, \bar{s})$
$\mathbf{PO}(\bar{v}, i+1, h_1, \bar{u}_1)$
$\mathbf{PO}(\bar{v}, i+1, h_2, \bar{u}_2)$
$\mathcal{PO}(\bar{v})$
Layer $\frac{n+1}{2}$
$i = i$
$i = 2$
$i = 1$
ϕ^R
ϕ
$\tilde{w}$
X

Figure 6.2: (Left) The relationship between $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ with the tree $\mathcal{TO}(\bar{v})$ and $\mathbf{PO}(\bar{v})$. (right) Example of the order for which characters are assigned. Note that at each step the choices for the symbol $\tilde{\mathbf{w}}_i$ is constrained in the no subword of $\tilde{\mathbf{w}}_{[1,i]}\tilde{\mathbf{w}}^R_{[1,i]}$ is greater than or equal to $\bar{v}$.

of palindromic necklaces greater than $\bar{v}$ is equal to $|\mathcal{PO}(\bar{v})|$. Using this set the number of necklaces less than $\bar{v}$ can be counted by subtracting the size of $\mathcal{PO}(\bar{v})$ from the total number of odd length palindromic necklaces, equal to $q^{(n+1)/2}$ (Corollary 5).

High level idea for the Odd Case. Here we provide a high level idea for the approach we follow for computing $\mathcal{PO}(\bar{v})$. Let $\bar{v}$ have a length n. Since $\mathcal{PO}(\bar{v})$ only contains words of the form $\bar{\phi}x\bar{\phi}^R$, where $\bar{\phi} \in \Sigma^{(n-1)/2}$ and $x \in \Sigma$, we have that $\bar{w}_i = \bar{w}_{n-i}$ for every $\bar{w} \in \mathcal{PO}(\bar{v})$.As the lexicographically smallest rotation of every $\bar{w} \in \mathcal{PO}(\bar{v})$ must be greater than $\bar{v}$, it follows that any word rotation of $\bar{w}$ must be greater than $\bar{v}$ and therefore every subword of $\bar{w}$ must also be greater than or equal to the prefix of $\bar{v}$ of the same length. This property is used to compute the size of $\mathcal{PO}(\bar{v})$ by iteratively considering the set of prefixes of each word in $\mathcal{PO}(\bar{v})$ in increasing length representing them with the tree $\mathcal{TO}(\bar{v})$. As generating $\mathcal{TO}(\bar{v})$ directly would require an exponential number of operations, a more sophisticated approach is needed for the calculation of $|\mathcal{PO}(\bar{v})|$ based on partial information.

As the tree $\mathcal{TO}(\bar{v})$ is a tree of prefixes, vertices in $\mathcal{TO}(\bar{v})$ are referred to by the prefix they represent. So $\bar{u} \in \mathcal{TO}(\bar{v})$ refers to the unique vertex in $\mathcal{TO}(\bar{v})$ representing $\bar{u}$. The root vertex of $\mathcal{TO}(\bar{v})$ corresponds to the empty word. Every other vertex $\bar{u} \in \mathcal{TO}(\bar{v})$ corresponds to a word of length i, where i is the distance between $\bar{u}$ and the root vertex.

Given two vertices $\bar{p}, \bar{c} \in \mathcal{TO}(\bar{v})$, $\bar{p}$ is the parent vertex of a child vertex $\bar{c}$ if and only if $\bar{c} = \bar{p}x$ for some symbol $x \in \Sigma$. The i^{th} layer of $\mathcal{TO}(\bar{v})$ refers to all representing words of length i in $\mathcal{TO}(\bar{v})$. The size of $\mathcal{PO}(\bar{v})$ is equivalent to the number of unique prefixes of length $\frac{n+1}{2}$ of words of the palindromic form $\bar{\phi}x\bar{\phi}^R$ in $\mathcal{PO}(\bar{v})$. This set of prefixes corresponds to the vertices in the layer $\frac{n+1}{2}$ of $\mathcal{TO}(\bar{v})$. Therefore the maximum depth of $\mathcal{TO}(\bar{v})$ is $\frac{n+1}{2}$.

To speed up computation, each layer of $\mathcal{TO}(\bar{v})$ is partitioned into sets that allow the size of $\mathcal{PO}(\bar{v})$ to be efficiently computed. This partition is chosen such that the size of the sets in layer $i + 1$ can be easily derived from the size of the sets in layer i. As these sets are tied to the tree structure, the obvious property to use is the number of children each vertex has. As each vertex $\bar{u} \in \mathcal{TO}(\bar{v})$ represents a prefix of some word $\bar{w} \in \mathcal{PO}(\bar{v})$, the number of children of $\bar{u}$ is the number of symbols $x \in \Sigma$ such that $\bar{u}x$ is a prefix of some word in $\mathcal{PO}(\bar{v})$. Recall that every word in $\bar{w} \in \mathcal{PO}(\bar{v})$ has the form $\bar{\phi}x\bar{\phi}^R$, and that there is no subword of $\bar{w}$ that is less than $\bar{v}$. Therefore if $\bar{u} \in \mathcal{TO}(\bar{v})$, there must be no subword of $\bar{u}^R\bar{u}$ that is less than $\bar{v}$. Hence the number of children of $\bar{u}$ is the number of symbols $x \in \Sigma$ such that no subword of $x\bar{u}^R\bar{u}x$ is less than the prefix of $\bar{v}$ of the same length. As $\bar{u}^R\bar{u}$ has no subword less than $\bar{v}$, $x\bar{u}^R\bar{u}x$ has a subword that is less than $\bar{v}$ only if either (1) $x\bar{u}^R\bar{u}x < \bar{v}$ or (2) there exists some suffix of length j such that $(\bar{u}^R\bar{u})_{[2i-j,2i]} = \bar{v}_{[1,j]}$ and $x < \bar{v}_{j+1}$. For the first condition, let $\bar{s} \sqsubseteq_{2i} \bar{v}$. By the definition of strictly bounding subwords (Definition 24), $x\bar{u}^R\bar{u}x < \bar{v}$ if and only if $x\bar{s}x < \bar{v}$. Note that this ignores any word $\bar{u}$ where $\bar{u}^R\bar{u} \sqsubseteq \bar{v}$. The restriction to strictly bounded words is to avoid the added complexity caused by Proposition 9, where the word that bounds $x\bar{s}x$ might not be the word that bounds $x\bar{u}^R\bar{u}x$. For the second property, let j be the length of the longest suffix of $\bar{u}^R\bar{u}$ that is a prefix of $\bar{v}$. From Lemma 1 due to Sawada and Williams [92], there is some suffix of $\bar{u}^R\bar{u}x$ that is smaller than $\bar{v}$ if and only if $x < \bar{v}_{j+1}$. The i^{th} layer of $\mathcal{TO}(\bar{v})$ is partitioned into n^2 sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$, for every $i \in [\frac{n+1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$.

Definition 25. *Let $i \in [\frac{n+1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$. The set $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ contains every prefix $\bar{u} \in \mathcal{TO}(\bar{v})$ of length i where (1) the longest suffix of $\bar{u}^R_{[1,i]}\bar{u}_{[1,i]}$ which is a prefix of $\bar{v}$ has a length of j and (2) The word $\bar{u}^R_{[1,i]}\bar{u}_{[1,i]}$ is strictly bounded by $\bar{s}$.*

An overview of the properties used by $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ is given in Figures 6.2 and 6.3. It follows from the earlier observations that each vertex in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ has the same number of children. Lemma 16 strengthens this observation, showing that given $\bar{a}, \bar{b} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$, $\bar{a}x \in \mathbf{PO}(\bar{v}, i + 1, j', \bar{s}')$ if and only if $\bar{b}x \in \mathbf{PO}(\bar{v}, i + 1, j', \bar{s}')$.

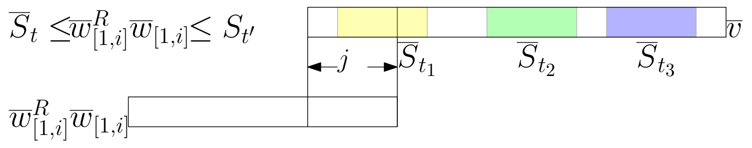
$\overline{S}_t \leq \overline{w}^R_{[1,i]}\overline{w}_{[1,i]} \leq S_{t'}$
$\overline{v}$
j
$\overline{S}_{t_1}$
$\overline{S}_{t_2}$
$\overline{S}_{t_3}$
$\overline{w}^R_{[1,i]}\overline{w}_{[1,i]}$

Figure 6.3: Visual representation of the properties of $\bar{w}^R_{[1,i]}\bar{w}_{[1,i]} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$.

The remainder of this section establishes how to count the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ and the number of children vertices for each vertex in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$. The first step is to formally prove that all vertices in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ have the same number of children vertices. This is shown in Lemma 16 by proving that given two vertices $\bar{a}, \bar{b} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$, if the vertex $\bar{a}' = \bar{a}x$ for $x \in \Sigma$ belongs to the set $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, so to does $\bar{b}' = \bar{b}x$.

Lemma 16. *Let $\bar{a}, \bar{b} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$ and let $x \in \Sigma$. If the vertex $\bar{a}' = \bar{a}x$ belongs to $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, the vertex $\bar{b}' = \bar{b}x$ also belongs to $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$. Furthermore the value of j' and $\bar{s}'$ can be computed in constant time from the values of $j, \bar{s}$ and x.*

Proof. By the definition of the set $\mathbf{PO}(\bar{v}, i, j, \bar{s})$, the last j symbols of $\bar{a}^R\bar{a}$ and $\bar{b}^R\bar{b}$ are equal to $\bar{v}_{[1,j']}$. Therefore if $j' > 0$, x must be equal to $\bar{v}_{j+1}$, satisfying this observation. On the other hand, if $j' = 0$ then x must be greater than $\bar{v}_{j+1}$. Following Lemmas 15 and 13, if $\bar{s}'$ bounds $x\bar{a}^R\bar{a}x$ and $\bar{s}$ bounds both $\bar{a}^R\bar{a}$ and $\bar{b}$, then $\bar{s}'$ also bounds $x\bar{b}^R\bar{b}x$. Hence $\bar{b}'$ must also belong to $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$.

To compute the value of j' and $\bar{s}'$ in constant time, assume that the arrays XW and WX as defined in Section 6.2. Note that if $x < \bar{v}_{j+1}$, there is no such value of j' or $\bar{s}'$ as the suffix of $x\bar{a}^R\bar{a}x$ of length $j+1$ is smaller than $\bar{v}$, contradicting the definition of the set. If $x = \bar{v}_{j+1}$ then the value of j' must be $j+1$. Otherwise, the value of j' is 0 following Lemma 1 of Sawada and Williams [92]. The value $\bar{s}'$ can be derived using WX and XW by finding the word $\bar{u} = WX[\bar{s}, x]$ that bounds $\bar{s}x$, then $\bar{s}' = XW[\bar{u}, x]$ that bounds $x\bar{u}$. Therefore the value of j' and $\bar{s}'$ can be computed in constant time. □

Computing the size of $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$. Lemma 16, provides enough information to compute the size of $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ once the size of $\mathbf{PO}(\bar{v}, i-1, j, \bar{s})$ has been computed for each value of $j \in [2(i-1)]$ and $\bar{s} \in \mathbf{S}(\bar{v}, 2(i-1))$. At a high level, the idea is to create an array, *SizePO*, storing the size of the $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ for every value of $i \in [\frac{n-1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$. For simplicity, let the value of $SizePO[i, j, \bar{s}]$ be the size of $|\mathbf{PO}(\bar{v}, i, j, \bar{s})|$.

Lemma 17 formally provides the method of computing $SizePO[i, j, \bar{s}]$ for every $j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$ once $SizePO[i-1, j', \bar{s}']$ has been computed for every $j' \in [2i-2]$ and

$\bar{s} \sqsubseteq_{2i-2} \bar{v}$. Observe that each vertex $a \in \mathbf{PO}(\bar{v}, i, j', \bar{s}')$ represents a prefix $\bar{a}'x$ where $\bar{a}'$ is either in $\mathbf{PO}(\bar{v}, i-1, j, \bar{s})$, for some value of j and $\bar{s}$, or $\bar{a}' \sqsubseteq \bar{v}$. Using this, the high level idea is to derive the values of j' and $\bar{s}'$ for each $j \in [2(i-1)], \bar{s} \in \mathbf{S}(\bar{v}, 2(i-1))$ and $x \in \Sigma$. Once the values j' and $\bar{s}'$ have been derived, the value of $SizePO[i, j', \bar{s}']$ is increased by the size of $\mathbf{PO}(\bar{v}, i-1, j, \bar{s})$. Repeating this for every value of $j, \bar{s}$ and x leaves the value of $SizePO[i, j', \bar{s}']$ as the number of vertices in $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ representing words of the form $\bar{a}x$ where $\bar{a} \not\sqsubseteq \bar{v}$. As each set $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ may have children in at most q sets $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$, the number of vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ with a parent vertex in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ can be computed in $O(q \cdot n^2)$ by looking at every argument of $j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$.

To account for the vertices in $\mathbf{PO}(\bar{v}, i, j', \bar{s}')$ of the form $\bar{b}x$ where $\bar{b}^R\bar{b} \sqsubseteq \bar{v}$, a similar process is applied to each pair $\bar{s} \in \mathbf{S}(\bar{v}, 2(i-1))$ and $x \in \Sigma$. For each pair, the values $\bar{s}'$ and j' are derived in the same manner as Lemma 16 utilising the tables XW and WX. Once derived, the value of $SizePO[i, j', \bar{s}']$ is increased by one, to account for the vertex $\bar{s}x$. As the values of j' and $\bar{s}'$ can be computed in $O(n)$ time from the value of x and $\bar{s}$, the number of vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ where the parent vertex is a subword of $\bar{v}$ can be computed in $O(q \cdot n^2)$ time.

Lemma 17. *Given the size of* $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ *for* $i \in \left[\frac{n-3}{2}\right]$ *and every* $j \in [2i], \bar{s} \sqsubseteq_{2i} \bar{v}$*, the size of* $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ *for every* $j' \in [2i+2], \bar{s}' \sqsubseteq_{2i+2} \bar{v}$ *can be computed in* $O(q \cdot n^2)$ *time.*

Proof. Assume that WX and XW have been precomputed. Further assume that the array $SizePO$ has be initialised such that $SizePO[i, j, \bar{s}] = |\mathbf{PO}(\bar{v}, i, j, \bar{s})|$ for every value of $j \in [2i]$ and $\bar{s} \in \mathbf{S}(\bar{v}, 2i)$, and $SizePO[i+1, j, \bar{s}] = 0$ for every $j' \in [2i+2]$, and $\bar{s}' \in \mathbf{S}(\bar{v}, 2i)$.

The first step is to count the number of vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ representing words of the form $\bar{a}x$ where $\bar{a} \not\sqsubseteq \bar{v}$. This is done by checking each $j \in [2i], \bar{s} \in \mathbf{S}(\bar{v}, 2i)$, and $x \in \Sigma$. For each $j, \bar{s}$ and x, the values j' and $\bar{s}'$ are derived in constant time as in Lemma 16. Following Lemma 16, every vertex $\bar{a} \in \mathbf{PO}(\bar{v}, i, j, \bar{s})$ has some child vertex in $\bar{a}' \in \mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ such that the last symbol of the word $\bar{a}'$ is equal to x. Therefore the value of $SizePO[i+1, j', \bar{s}']$ is increased by the value of $SizePO[i, j, \bar{s}]$. Repeating this for every value of $j, \bar{s}$ and x leaves the value of $SizePO[i+1, j', \bar{s}']$ equal to the number of vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ of the form $\bar{a}x$ where $\bar{a} \not\sqsubseteq \bar{v}$. As there are n possible value of both j and $\bar{s}$, and q values of x, this process takes $O(n^2 \cdot q)$ operations.

To compute the number vertices in $\mathbf{PO}(\bar{v}, i+1, j', \bar{s}')$ of the form $\bar{b}x$ where $\bar{b} \sqsubseteq \bar{v}$, a similar process is applied to each pair $\bar{s} \in \mathbf{S}(\bar{v}, 2i)$ and $x \in \Sigma$. Formally, for each pair of $\bar{s}$ and x, the first step is to check that $\bar{s} = \bar{s}^R$. This can be done in linear time by comparing the two strings. This check ensures that new word is palindromic. The second check is that $x\bar{s}x \not\sqsubseteq \bar{v}$. This is to ensure that the new word is not counted in the next layer. This can be done by finding the word $\bar{s}'$ in the same manner as in Lemma 16, and checking if the word $\bar{u}'$ preceding $\bar{s}'$ in the ordered set $\mathbf{S}(\bar{s}, 2i+2)$ is equal to $x\bar{s}x$. Let j be the length of the longest suffix of $\bar{s}$ that is a prefix of $\bar{v}$. The value of j can be found in linear time by using a simple pattern matching algorithm on $\bar{s}$ and recording the final state. The value of j' can be found form the value of j and x using Lemma 16 in constant time. Once j' and $\bar{s}'$ have been derived, the value of $SizePO[i+1, j', \bar{s}']$ can be increased by 1. As there are n possible values of $\bar{s}, q$ possible values of x, and at most $O(n)$ operations are required for each pair, this process takes $O(n^2 \cdot q)$ operations. Therefore the total complexity is $O(n^2 \cdot q)$. □

Once the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ has been computed for every $i \in [\frac{n-1}{2}], j \in [2i], \bar{s} \in \mathbf{S}(\bar{v}, 2i)$, the final step is to compute $|\mathcal{PO}(\bar{v})|$. The high level idea is to determine the number of vertices in $\mathcal{PO}(\bar{v})$ are children of a vertex in $\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$.The set $\mathbf{X}(\bar{v}, j, \bar{s}) \subseteq \Sigma$ is introduced to help with this goal. Let $\mathbf{X}(\bar{v}, j, \bar{s})$ contain every symbol $x \in \Sigma$ such that $\bar{a}x\bar{a}^R \in \mathcal{PO}(v)$ where $\bar{a} \in \mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$. By the definition of $\mathbf{X}(\bar{v}, j, \bar{s})$, $|\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})|$ equals the number of words $\bar{w} \in \mathcal{PO}(\bar{v})$ where$(\bar{w}_1 \dots \bar{w}_{(n-1)/2}) \in \mathbf{PO}(\bar{v}$, i, j ,$\bar{s})$. Lemma 18 shows how to compute the size of $\mathbf{X}(\bar{v}, j, \bar{s})$ in $O(q \cdot n)$ time.

Lemma 18. *Let $\mathbf{X}(\bar{v}, j, \bar{s})$ contain every symbol in Σ such that $\bar{a}x\bar{a}^R \in \mathcal{PO}(v)$ where $\bar{a} \in \mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$. The size of $\mathbf{X}(\bar{v}, j, \bar{s})$ can be computed in $O(q \cdot n)$ time.*

Proof. The size of $\mathbf{X}(\bar{v}, j, \bar{s})$ can be computed in a direct manner by checking if $x \in \mathbf{X}(\bar{v}, j, \bar{s})$ for each $x \in \Sigma$. Given some $x \in \Sigma$, note that if $x < \bar{v}_{j+1}$ then there exists some rotation of $\langle \bar{a}x\bar{a}^R \rangle$ that is smaller than $\bar{v}$. Let $x \geq \bar{v}_{j+1}$. For x to be a member of $\mathbf{X}(\bar{v}, j, \bar{s})$ observe that for $\langle \bar{a}x\bar{a}^R \rangle$ to be greater than $\bar{v}$, $\bar{v}_{[1,j]}x\bar{a}^R\bar{a}$ must be greater than v. Using the bound given by $\bar{s}$ gives $\langle \bar{a}x\bar{a}^R \rangle > \bar{v}_{[1,j]}x\bar{s}$. Therefore if $\bar{v}_{[1,j]}x\bar{s} \geq \bar{v}$, $x \in \mathbf{X}(\bar{v}, j, \bar{s})$. In the other hand, if $\bar{v}_{[1,j]}x\bar{s} < \bar{v}$, then note that $\bar{s} < \bar{v}_{[j+2,n+j]}$. Therefore $\bar{a}^R\bar{a} < \bar{v}_{[j+2,n+j]}$ as it is bounded by $\bar{s}$. Hence $\langle \bar{a}x\bar{a}^R \rangle < \bar{v}$. Therefore, $x \in \mathbf{X}(\bar{v}, j, \bar{s})$ if and only if $\bar{v}_{[1,j]}x\bar{s} \geq \bar{v}$. As this can be checked in $O(n)$ steps by directly comparing the two words, and there are q values of z to check, the total complexity is $O(q \cdot n)$. □

Converting *SizePO* to $|\mathcal{PO}(\bar{v})|$. The final step in computing $\mathcal{PO}(\bar{v})$ is to convert the cardinality of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ to the size of $\mathcal{PO}(\bar{v})$. Lemma 19 provides a formula for counting the size of $\mathcal{PO}(\bar{v})$. Combining this formula with the techniques given in Lemma 17 an algorithm for computing the size of $\mathcal{PO}(\bar{v})$ directly follows.

It follows from Lemma 16 that the number of words in $\mathcal{PO}(\bar{v})$ with a prefix in $\mathbf{PO}\left(\bar{v}, \frac{n-1}{2}, j, \bar{s}\right)$ is equal to the cardinality of $\mathbf{PO}\left(\bar{v}, \frac{n-1}{2}, j, \bar{s}\right)$ multiplied by the size of $\mathbf{X}(\bar{v}, j, \bar{s})$. Similarly the number of words in $\mathcal{PO}(\bar{v})$ with a prefix $\bar{u}$ of length $\frac{n-1}{2}$ where $\bar{u}^R\bar{u} \sqsubseteq \bar{v}$ can be determined using $\mathbf{X}(\bar{v}, j, \bar{u}^R\bar{u})$. The main difference in this case is that if $\bar{u}^R\bar{u} = \bar{v}_{[j+2,n+j]}$, where j is the length of the longest suffix of $\bar{u}^R\bar{u}$ that is a prefix of $\bar{v}$, then the number of words in $\mathcal{PO}(\bar{v})$ where $\bar{u}$ is a prefix is 1 fewer than for the number of words strictly bounded by $\bar{u}^R\bar{u}$, i.e. $|\mathbf{X}(\bar{v}, J(\bar{s}, \bar{v}), \bar{s})| - 1$. Lemma 19 provides the procedure to compute $|\mathcal{PO}(\bar{v})|$.

Lemma 19. *Let $J(\bar{s}, \bar{v})$ return the length of the longest suffix of $\bar{s}$ that is a prefix of $\bar{v}$. The size of $\mathcal{PO}(\bar{v})$ is equal to*

$$\sum_{\bar{s}\in\mathbf{S}(\bar{v},n-1)} \left(\sum_{j=1}^{n-1} |\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}\left(\bar{v}, \tfrac{n-1}{2}, j, \bar{s}\right)| \right) + \begin{cases} 0 & \bar{s} \neq \phi\phi^R \\ |\mathbf{X}(\bar{v}, J(\bar{s}, \bar{v}), \bar{s})| & \bar{s} \neq \bar{v}_{[j+2,n+j]} \\ |\mathbf{X}(\bar{v}, J(\bar{s}, \bar{v}), \bar{s})| - 1 & \bar{s} = \bar{v}_{[j+2,n+j]} \end{cases}$$

Further this can be computed in $O(q \cdot n^3 \cdot \log(n))$ time.

Proof. From Lemma 18 the size of the set $\mathbf{X}(\bar{v}, j, \bar{s})$ can be computed in $O(n \cdot q)$ operations. By the definition of $\mathbf{X}(\bar{v}, j, \bar{s})$, $|\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})|$ is the number of words $\bar{w} \in \mathcal{PO}(\bar{v})$ where $\bar{w}_{[1,(n-1)/2]} \in \mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$. Therefore
$\sum_{\bar{s}\in\mathbf{S}(\bar{v},n-1)} \left(\sum_{j=1}^{n-1} |\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})| \right)$ counts every word $\bar{w} \in \mathcal{PO}(\bar{v})$ where $\bar{w}_{[1,(n-1)/2]} \in \mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$ for some arguments $j \in [|\mathbf{v}| - 1], \bar{s} \in \mathbf{S}(\bar{v}, n-1)$. As there are n^2 possible values of j and $\bar{s}$, and computing $|\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})|$ requires $O(q \cdot n)$ steps, the total complexity of counting $\sum_{\bar{s}\sqsubseteq\bar{v}} \left(\sum_{j=1}^{n-1} |\mathbf{X}(\bar{v}, j, \bar{s})| \cdot |\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})| \right)$ is $O(n^3 \cdot q)$.

For words of the form $\bar{\phi}^R x \bar{\phi}$ where $\bar{\phi}^R\bar{\phi} \sqsubseteq \bar{v}$ note that for every character in $\mathbf{X}(\bar{v}, j, \bar{s})$, $\langle x\bar{\phi}\bar{\phi}^R \rangle \geq \bar{v}$. Further, as $\langle \bar{\phi}^R\bar{\phi}x \rangle = \bar{v}$ only when $\bar{\phi}^R\bar{\phi} = \bar{v}_{[j+2,|v|+j]}$ and $x = \bar{v}_{j+1}$, the number of words of this form is $|\mathbf{X}(\bar{v}, j, \bar{s})|$, when $\bar{\phi}^R\bar{\phi} \neq \bar{v}_{[j+2,|v|+j]}$, and $|\mathbf{X}(\bar{v}, j, \bar{s})| - 1$ otherwise. As the conditions can be checked in $O(n)$ time, $\mathbf{X}(\bar{v}, j, \bar{s})$ can be computed in $O(n \cdot q)$ time, and there are $O(n)$ subwords in $\mathbf{S}(\bar{v}, n-1)$, the total complexity of computing

$\sum_{\bar{s}\in\mathbf{S}(\bar{v},n)} \begin{cases} |\mathbf{X}(\bar{v}, J(\bar{s},\bar{v}),\bar{s})| & \bar{s} = \phi\phi^R \text{ and } \bar{s} \neq \bar{v}_{[j+2,n+j]} \\ |\mathbf{X}(\bar{v}, J(\bar{s},\bar{v}),\bar{s})| - 1 & \bar{s} = \phi\phi^R \text{ and } \bar{s} = \bar{v}_{[j+2,n+j]} \\ 0 & \bar{s} \neq \phi\phi^R \end{cases}$ is $O(n^2 \cdot q)$. Therefore the total complexity of computing the size of $\mathcal{PO}(\bar{v})$ from the array $SizePO[i,j,\bar{s}]$ is $O(n^3 \cdot q)$. In order to compute the array $SizePO$ a total of $O(q \cdot n^3 \cdot \log(n))$ operations are needed. Hence the total complexity is $O(q \cdot n^3 \cdot \log(n))$. □

6.3.2 Even Length Palindromic Necklaces

Section 6.3.1 shows how to rank $\bar{v}$ within the set of odd length palindromic necklaces. This leaves the problem of counting even length palindromic necklaces. As in the odd case, the first step is to determine how to characterise these words. Proposition 12 shows that every palindromic necklace has at least one word of either the form $\bar{\phi}\bar{\phi}^R$, where $\bar{\phi} \in \Sigma^{n/2}$, or $x\bar{\phi}y\bar{\phi}^R$, where $x, y \in \Sigma$ and $\bar{\phi} \in \Sigma^{(n/2)-1}$. Proposition 12 is strengthened by Propositions 13 and 14, showing that each palindromic necklace of even length has no more than two words of either form. Lemmas 22, 23, 24 and 25 use these results a similar manner to Section 6.3.1 to count the number of palindromic necklaces of even length.

Proposition 12. *A necklace $\tilde{\mathbf{w}}$ of even length n is palindromic if and only if there exists some word $\bar{u} \in \tilde{\mathbf{w}}$ where either (1) $\bar{u} = x\bar{\phi}y\bar{\phi}^R$ where $x, y \in \Sigma$ and $\bar{\phi} \in \Sigma^{(n/2)-1}$, or (2) $\bar{u} = \bar{\phi}\bar{\phi}^R$ where $\bar{\phi} \in \Sigma^{n/2}$.*

Proof. Given a word $\bar{u}$ of the form $x\bar{\phi}y\bar{\phi}^R$ where $\bar{u} \in \tilde{\mathbf{w}}$, $\bar{u}^R$ is equal to $\bar{\phi}y\bar{\phi}^R x$. Following this observation $\bar{u} = \langle \bar{u}^R \rangle_1$. Therefore for every word in $\tilde{\mathbf{w}}$ the reflection is also in $\tilde{\mathbf{w}}$. Similarly, given a word $\bar{u} \in \tilde{\mathbf{w}}$ of the form $\bar{\phi}\bar{\phi}^R$, $\bar{u} = \bar{i}^R$, therefore for every word in the necklace $\tilde{\mathbf{w}}$, the reflection is also in $\tilde{\mathbf{w}}$.

In the other direction, let $\tilde{\mathbf{w}}$ be a palindromic necklace of even length n. If there is any word $\bar{u} \in \tilde{\mathbf{w}}$ such that $\bar{u} = \bar{u}^R$, then the word must be of the form $\bar{\phi}\bar{\phi}^R$. Therefore for the sake of contraction, assume every word $\bar{u} \in \tilde{\mathbf{w}}$ must not be equal to $\bar{u}^R$. As $\tilde{\mathbf{w}}$ is palindromic, there exists some rotation i such that $\bar{u} = \langle \bar{u}^R \rangle_i$. Therefore $\bar{u}_1 = \bar{u}_{n-i}, \bar{u}_2 = \bar{u}_{n-i-1} \dots \bar{u}_{n-i} = \bar{u}_1$ and $\bar{u}_{n-i+1} = \bar{u}_n \dots \bar{u}_1 = \bar{u}_{n-i+1}$. This splits $\bar{u}$ into 2 subwords, $\bar{s}$ and $\bar{t}$, where $\bar{s} = \bar{u}_{[1,n-i]}$ and $\bar{t} = \bar{u}_{[n-i+1,n]}$ where $\bar{s} = \bar{s}^R$ and $\bar{t} = \bar{t}^R$. Note that $\bar{s}_1\bar{t}\bar{s}_{n-i} = \bar{s}_{n-i}\bar{t}^R\bar{s}_1$ and $\bar{t}_1\bar{s}\bar{t}_{n-i} = \bar{t}_{n-i}\bar{s}^R\bar{t}_1$.

To show the structural claim, there are two cases to consider depending on the value of i and $\frac{n}{2}$. If i is odd then lengths of $\bar{s}$ and $\bar{t}$ are even. Two new words $\bar{s}'$ and $\bar{t}'$ are defined where $\bar{s}' = \bar{s}_{[(i/2)+1,i]}\bar{t}_{[1,(n-i)/2]}$ and $\bar{t}' = \bar{t}_{[(n-i)/2+1,n-i]}\bar{s}_{[1,i/2]}$. By the definition of $\bar{s}$ and $\bar{t}$, $\bar{s}'^R = \bar{t}^R_{[1,(n-i)/2]}\bar{s}^R_{[i,i/2+1]} = \bar{t}_{[(n-i)/2+1,n-i]}\bar{s}_{[1,i/2+1]} = \bar{t}'$. Therefore this word can be rotated to a word of the form $\bar{\phi}\bar{\phi}^R$.

If i is even then the lengths of $\bar{s}$ and $\bar{t}$ are odd. As before 2 words $\bar{s}'$ and $\bar{t}'$ are constructed of length $\frac{n}{2}-1$ where $\bar{s}' = \bar{s}_{[i/2+1,i]}\bar{t}_{[1,(n-i)/2-1]}$ and $\bar{t}' = \bar{t}_{\frac{n-i}{2}+1}\ldots\bar{t}_{n-i}\bar{s}_1\ldots\bar{s}_{\frac{i}{2}+1}$. As before, $\bar{s}'^R = \bar{t}^R_{[1,(n-i]/2-1]}\bar{s}^R_{[i/2+1,i]} = \bar{t}_{[(n-i)/2+1,n-i]}\bar{s}_{[1,i/2-1]} = v'$. Letting $x = \bar{t}_{\frac{n-i}{2}}$ and $y = \bar{s}_{\frac{i}{2}}$, then there is some rotation of $\bar{u}$ of the form $x\bar{\phi}y\bar{\phi}^R$. □

Proposition 13. *The word $\bar{u} \in \Sigma^*$ equals both $x\bar{\phi}y\bar{\phi}^R = \bar{\psi}\bar{\psi}^R$ if and only if $\bar{u} = x^n$.*

Proof. Starting with $x\bar{\phi}y\bar{\phi}^R = \bar{\psi}\bar{\psi}^R$ as $x\bar{\phi} = \bar{\psi}$, $x\bar{\phi}y\bar{\phi}^R = x\bar{\phi}\bar{\phi}^R x$. This implies $\bar{\phi}_1 = x$ allowing this to be rewritten as $xx\bar{\phi}'\bar{\phi}'^R xx = x\bar{\phi}y\bar{\phi}^R$, implying that $\bar{\phi}'_1 = x$. Repeating this gives $x\bar{\phi}y\bar{\phi}^R = xxx\ldots x$. □

Proposition 14. *For an even length palindromic necklace $\tilde{\mathbf{a}}$ there are at most two words $\bar{w}, \bar{u} \in \tilde{\mathbf{a}}$ where either (1) $\bar{w}$ and $\bar{u}$ are of the form $x\bar{\phi}y\bar{\phi}^R$ where $x, y \in \Sigma$ and $\bar{\phi} \in \Sigma^{(n/2)-1}$ or (2) $\bar{w}$ and $\bar{u}$ are of the form $\bar{\phi}\bar{\phi}^R$ where $\bar{\phi} \in \Sigma^{n/2}$.*

Proof. From Proposition 12 there must be at least 1 word of either form. Proposition 13 shows that a word may only be of the form $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\psi}\bar{\psi}^R$ if and only if $\bar{w} = x^n$. Let $\bar{w}$ and $\bar{v}$ be two words such that $\bar{w}, \bar{v} \in \tilde{\mathbf{a}}$ and $\bar{w} \neq \bar{v}$ where $\tilde{\mathbf{a}}$ is a necklace of even length. There are two cases based on the form of $\bar{w}$ and $\bar{v}$.

Case 1: $\bar{w} = x\bar{\phi}y\bar{\phi}^R$, $\bar{v} = a\bar{\psi}b\bar{\psi}^R$, $\bar{v} = \langle\bar{w}\rangle_r$. Let r be the smallest rotation where $\langle\bar{w}\rangle_r \neq \bar{w}$ and $\langle\bar{w}\rangle_r = a\bar{\psi}b\bar{\psi}^R$. Therefore $\bar{v}_i = \bar{v}_{n-i+1} = \bar{w}_{n-i+r} = \bar{w}_{n-n+i-r} = \bar{w}_{i-r} = \bar{v}_{n+i-2r} = \bar{v}_{i-2r}$. Therefore, $\bar{v}_i = \bar{v}_{i+2r} = \bar{v}_{i+4r} = \ldots = \bar{v}_i$. Therefore $\bar{w}$ has a period of no more than $p = GCD(2r, n)$. If $GCD(2r, n) \leq 2r$, then the period must be no more than r. If the period is r then $\bar{w} = \bar{v}$, contradicting the assumption that they are not equal. Otherwise, $\langle\bar{w}\rangle_r = \langle\bar{w}\rangle_{r-p}$, contradicting the assumption that r is the smallest rotation for which the rotation of $\bar{w}$ equals $x\bar{\psi}y\bar{\psi}^R$, for some arguments of $x, y \in \Sigma$ and $\bar{\psi} \in \Sigma^*$. Therefore the period must be $2r$. Hence let $r > s$ be some rotation such that $\bar{w} \neq \langle\bar{w}\rangle_s \neq \langle\bar{w}\rangle_r$. As $\langle\bar{w}\rangle_r = \bar{w}^R, \langle\bar{w}\rangle_{s+r} = (\langle\bar{w}\rangle_s)^R$. As the period is $2r$, if $s + r > 2r$ then the rotation $s - r$ is equivalent to the rotation by s contradicting the assumption that r is the smallest rotation for which $\langle\bar{w}\rangle_r = x\bar{\psi}y\bar{\psi}^R$, for some arguments of $x, y \in \Sigma$ and $\bar{\psi} \in \Sigma^*$. Therefore the only word satisfying $\bar{v}_i = \bar{v}_{i-2r}$ is when $r = \frac{n}{2}$, making $\bar{v} = y\bar{\phi}^R x\bar{\phi}$.

Case 2: $\bar{w} = \bar{\phi}\bar{\phi}^R$, $\bar{v} = \bar{\psi}\bar{\psi}^R$, $\bar{v} = \langle \bar{w} \rangle_r$. For the sake of contradiction, let r be the smallest rotation such that $\bar{w} \neq \langle \bar{w} \rangle_r$ and $\langle \bar{w} \rangle_r = \bar{\psi}\bar{\psi}^R$. Therefore $\bar{v}_i = \bar{w}_{i+r \bmod n}$, further $\bar{w}_i = \bar{v}_{n+i-r \bmod n}$, $\bar{w}_i = \bar{w}_{n-i+1}$ and $\bar{v}_i = \bar{v}_{n-i+1}$. These equations can be rearranged to give $\bar{v}_i = \bar{v}_{n-i+1} = \bar{w}_{r+n-i-1} = \bar{w}_{n-r-n+i+1-1} = \bar{w}_{i-r} = \bar{v}_{i-2r} = \bar{v}_i$. Repeated application of $\bar{v}_i = \bar{v}_{i-2r} = \bar{v}_{i-4r} = \ldots = \bar{v}_{i-s \cdot r}$ shows that $\bar{w}$ must have a period of no more than $p = GCD(2r, n)$. Therefore $\bar{w}$ can be rewritten as $\bar{u}^{n/p} = \bar{\phi}\bar{\phi}^R$. If $\frac{n}{p}$ is even then $\bar{u} = \bar{u}^R$. Assume for the sake of contradiction that there is some rotation t such that $r < t < 2r, \bar{w} \neq \langle \bar{w} \rangle_t \neq \bar{v}$ and $\langle \bar{w} \rangle_t$ is of the form $\bar{\phi}\bar{\phi}^R$. If $\bar{u} = \bar{u}^R$, then $\langle \bar{u} \rangle_t = (\langle \bar{u} \rangle_t)^R$. Hence the rotation by $2r - t$ is equivalent to the rotation by t, contradicting the assumption that r is the smallest rotation. If the period of $\bar{w}$ is smaller than $2r$ it must be a factor of r, hence $\langle \bar{w} \rangle r = \bar{w}$ contradicting the assumption that $\bar{w} \neq \bar{v}$. Therefore the period must be $2r$, implying that if $\bar{w} = \bar{\phi}\bar{\phi}^R$ then $\bar{v} = \bar{\phi}^R\bar{\phi}$. If $\frac{n}{p}$ is odd then as $\bar{u}^{n/p} = \bar{\phi}\bar{\phi}^R, \bar{u}_{[1,r]} = \bar{u}_{r+1,2r}$. Therefore the period is at most r, contradicting the assumption that $p = GCD(2r, n)$. In this case the arguments from the even case apply again. □

Propositions 12, 13 and 14 show that every palindromic necklace of even length has 1 or 2 words of either the form $x\bar{\phi}y\bar{\phi}^R$ or $\bar{\phi}\bar{\phi}^R$. To count the number of words of each form, the problem is split into two sub problems, counting words of the form $x\bar{\phi}y\bar{\phi}^R$ and counting the number of words of the form $\bar{\phi}\bar{\phi}^R$. This is done using the same basic ideas as in Section 6.3.1. Two new sets $\mathcal{PE}(\bar{v})$ and $\mathcal{PS}(\bar{v})$ are introduced, serving the same function as $\mathcal{PO}(\bar{v})$ for words of the from $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\phi}\bar{\phi}^R$ respectively.

$$\mathcal{PE}(\bar{v}) := \left\{ \bar{w} \in \Sigma^n : \bar{w} = x\bar{\phi}y\bar{\phi}^R, \text{ where } \langle \bar{w} \rangle > \bar{v}, \bar{\phi} \in \Sigma^{(n/2)-1}, x, y, \in \Sigma \right\}$$
$$\mathcal{PS}(\bar{v}) := \left\{ \bar{w} \in \Sigma^n : \bar{w} = \bar{\phi}\bar{\phi}^R, \text{ where } \langle \bar{w} \rangle > \bar{v}, \bar{\phi} \in \Sigma^{(n/2)-1} \right\}$$

Unlike the set $\mathcal{PO}(\bar{v})$ in Section 6.3.1 the sets $\mathcal{PE}(\bar{v})$ and $\mathcal{PS}(\bar{v})$ do not correspond directly to bracelets greater than $\bar{v}$. For notation let $\mathcal{GE}(\bar{v})$ and $\mathcal{GS}(\bar{v})$ denote the number of bracelets greater than $\bar{v}$ of the form $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\phi}\bar{\phi}^R$ respectively. The number of even length necklaces greater than $\bar{v}$ equals $\mathcal{GE}(\bar{v}) + \mathcal{GS}(\bar{v}) - (q - \bar{v}_1)$, where $q - \bar{v}_1$ denotes the number of symbols in Σ greater than $\bar{v}_1$. Before showing how to compute the size of these sets, it is useful to first understand how they are used to compute the rank amongst even length palindromic necklaces. Lemmas 21 and 20 shows how to covert the cardinalities of these sets into the number of even length palindromic necklaces smaller than $\bar{v}$. The main idea is to use the observations given by Propositions 12 and 14 to determine how many even length palindromic necklaces have either one or two words of the form $x\bar{\phi}y\bar{\phi}^R$ or $\bar{\phi}\bar{\phi}^R$.

Proposition 15. *Let $l = \frac{n+2}{4}$ if $\frac{n}{2}$ is odd or $l = \frac{n}{4}$ if $\frac{n}{2}$ is even. The number of even length palindromic necklaces is given by $\frac{1}{2}\left(q^{n/2}(q+2) + q^l\right) - q$.*

Proof. First consider the number of words of the form $x\bar{\phi}y\bar{\phi}^R$. Let $\bar{w}, \bar{u} \in \tilde{\mathbf{w}}$ be a pair of words of the form $x\bar{\phi}y\bar{\phi}^R$ such that $\bar{w} \neq \bar{u}$ and $\langle \bar{w} \rangle_r = \bar{u}$. Following Proposition 14, if $2r < n$ and $\frac{n}{2r}$ is odd, then $\bar{w} = x\bar{\phi}y\bar{\phi}^R = \bar{\psi}^t$ for some word $\bar{\psi}$ of even length and $t = \frac{n}{2r}$. Therefore $\bar{\psi} = \bar{\psi}^R$ and further $\bar{\psi} = \langle \bar{\psi} \rangle_r$, therefore there is only a single word of the form $x\bar{\phi}y\bar{\phi}$. On the other hand if $2r < n$ and $\frac{n}{2r}$ is even then $\bar{w} = x\bar{\phi}y\bar{\phi}^R = \bar{\psi}^t$ for $t = \frac{n}{2r}$ and some word $\bar{\psi}$ of length $2r$. In this case, as t must be at least 2, $x\bar{\phi}y\bar{\phi}^R = x\bar{\phi}x\bar{\phi}$, therefore $y = x$ and $\bar{\phi} = \bar{\phi}^R$. Further as $\bar{u} = (\bar{\psi}_r)^t$, $\bar{\psi} = \bar{\psi}^R$, therefore $\bar{u} = \bar{w}$, hence there is only a single word of the form $x\bar{\phi}y\bar{\phi}^R$. Therefore the period of $\bar{w}$ must be n and hence there are only two words of the form $x\bar{\phi}y\bar{\phi}^R$ if and only if $x\bar{\phi} \neq y\bar{\phi}^R$.

Using this basis, the number of even length palindromic necklaces with one words of the form $x\bar{\phi}y\bar{\phi}^R$ equals the number of words of the form $x\bar{\phi}x\bar{\phi}$. This is equal to $q^{n/2}$. As the number of words with 2 representations of the form $x\bar{\phi}y\bar{\phi}^R$ is $q^{(n/2)+1}$, the number of necklaces with any word of the form $x\bar{\phi}y\bar{\phi}^R$ is $\frac{1}{2}\left(q^{n/2+1} + q^{n/2}\right)$.

Proposition 14 shows that, given $\bar{w}, \bar{u} \in \tilde{\mathbf{w}}$ of the form $\bar{\phi}\bar{\phi}^R$, $\bar{w} \neq \bar{u}$ if and only if $\bar{u} = \langle \bar{w} \rangle_{n/2}$ and $\bar{\phi} \neq \bar{\phi}^R$. Therefore the number of necklaces with 1 word of the form $\bar{\phi}\bar{\phi}^R$ is equal to the number of values of $\bar{\phi}$ for which $\bar{\phi} = \bar{\phi}^R$. If $|\bar{\phi}|$ is odd, this is equal to $q^{(n+2)/4}$ and $q^{n/4}$ if $|\bar{\phi}|$ is even. Hence the number of necklaces with two representations of the form $\bar{\phi}\bar{\phi}^R$ is $\frac{1}{2}(q^{n/2} - q^l)$, where $l = \frac{n+2}{4}$ if $\frac{n}{2}$ is odd or $l = \frac{n}{4}$ if $\frac{n}{2}$ is even. Therefore the total number of necklaces with any word of the form $\bar{\phi}\bar{\phi}^R$ is $\frac{1}{2}\left(q^{n/2} + q^l\right)$. Recalling from Proposition 13 that a word is of both forms if and only if it is of the form x^n, there are q necklaces that would be counted by both equations. Therefore the total number of even length necklaces are $\frac{1}{2}\left(q^{n/2+1} + q^{n/2} + q^{n/2} + q^l\right) - q$. □

Lemma 20. *The number of necklaces greater than $\bar{v}$ containing at least one word of the form $x\bar{\phi}y\bar{\phi}^R$ is given by* $GE(\bar{v}) = \frac{1}{2}\left(|\mathcal{PE}(\bar{v})| + \begin{cases} |\mathcal{PO}(\bar{v}_{[1,n/2]})| & \frac{n}{2} \textit{ is odd.} \\ GE(\bar{v}_{[1,n/2]}) & \frac{n}{2} \textit{ is even.} \end{cases}\right)$.

Proof. It follows that the number of necklaces of the form $x\bar{\phi}y\bar{\phi}^R$ that are greater than $\bar{v}$ equals to the number of necklaces with one word of the form $x\bar{\phi}y\bar{\phi}^R$, plus the number of necklaces with two words of the form $x\bar{\phi}y\bar{\phi}^R$. The number of words of the form $x\bar{\phi}y\bar{\phi}^R$ greater than $\bar{v}$ equals the size of $\mathcal{PE}(\bar{v})$. As a necklace has only one word of the form

$x\bar{\phi}y\bar{\phi}^R$ if and only if $x\bar{\phi} = y\bar{\phi}^R$. This leaves the problem of counting the number of words of the form $x\bar{\phi}x\bar{\phi}$ in necklaces greater than $\bar{v}$. If $\frac{n}{2}$ is odd, then $\bar{\phi}$ can be rewritten as $\bar{\psi}\bar{\psi}^R$. In this case, the goal becomes to fine the number of words of the form $x\bar{\psi}\bar{\psi}^R x\bar{\psi}\bar{\psi}^R$ in bracelets greater than $\bar{v}$, which equals $|\mathcal{PO}(\bar{v}_{[1,n/2]})|$. On the other hand, if $\frac{n-2}{2}$ is odd then $\bar{\phi}$ can be rewritten as $\bar{\psi}y\bar{\psi}^R$. In this case, the goal becomes to fine the number of words of the form $x\bar{\psi}y\bar{\psi}^R x\bar{\psi}y\bar{\psi}^R$ in bracelets greater than $\bar{v}$, which equals the number of words of the form $x\bar{\psi}y\bar{\psi}^R$ that are bracelets greater than $\bar{v}$. This is given by $GE(\bar{v}_{[1,n/2]})$. Therefore the total number of necklaces of the form $x\bar{\phi}y\bar{\phi}^R$ greater than $\bar{v}$ is given by:

$$GE(\bar{v}) = \frac{1}{2}\left(|\mathcal{PE}(\bar{v})| + \begin{cases} |\mathcal{PO}(\bar{v}_{[1,n/2]})| & \frac{n}{2} \text{ is odd.} \\ GE(\bar{v}_{[1,n/2]}) & \frac{n}{2} \text{ is even.} \end{cases}\right)$$

□

Lemma 21. *The number of necklaces greater than $\bar{v}$ containing at least one word of the form $\bar{\phi}\bar{\phi}^R$ is given by* $GS(\bar{v}) = \frac{1}{2}\left(|\mathcal{PE}(\bar{v})| + \begin{cases} |\mathcal{PO}(\bar{v})| & \frac{n}{2} \textit{ is odd.} \\ GS(\bar{v}_{[1,n/2]}) & \frac{n}{2} \textit{ is even.} \end{cases}\right)$.

Proof. Similar to Lemma 20, this Lemma is proven in a combinatorial manner by looking at the two cases where there is only a single word of the form $\bar{\phi}\bar{\phi}^R$. Recall that there is a single word of this form if and only if $\bar{\phi} = \bar{\phi}^R$. Therefore, the number of necklaces with a single word of the form $\bar{\phi}\bar{\phi}^R$ equals the number of palindromic words of length $\frac{n}{2}$. Hence if $\frac{n}{2}$ is even, the number of such words is $GS(\bar{v}_{[1,n/2]})$. On the other hand, if $\frac{n}{2}$ is odd, the number of such words is $|\mathcal{PO}(\bar{v})|$. Using the same arguments as in Proposition 15:

$$GE(\bar{v}) = \frac{1}{2}\left(|\mathcal{PE}(\bar{v})| + \begin{cases} |\mathcal{PO}(\bar{v})| & \frac{n}{2} \text{ is odd.} \\ GS(\bar{v}_{[1,n/2]}) & \frac{n}{2} \text{ is even.} \end{cases}\right)$$

□

High Level Idea for the Even Case: Lemmas 20 and 21 show how to use the sets $\mathcal{PS}(\bar{v})$ and $\mathcal{PE}(\bar{v})$ to get the number of necklaces of the form $x\bar{\phi}y\bar{\phi}^R$ and $\bar{\phi}\bar{\phi}^R$ respectively. This leaves the problem of computing the size of both sets. This is achieved in a manner similar to the one outlined in Section 6.3.1. At a high level the idea is to use two trees analogous to $\mathcal{TO}(\bar{v})$ as defined in Section 6.3.1. The tree $\mathcal{TE}(\bar{v})$ is introduced to compute the cardinality of $\mathcal{PE}(\bar{v})$ and the tree $\mathcal{TS}(\bar{v})$ is introduced to compute the cardinality of

$\mathcal{PS}(\bar{v})$. As in Section 6.3.1, the trees $\mathcal{TE}(\bar{v})$ and $\mathcal{TS}(\bar{v})$ contain every prefix of a word in $\mathcal{PS}(\bar{v})$ or $\mathcal{PE}(\bar{v})$ respectively. The leaf vertices of these trees correspond to the words in these sets.

To compute the size of $\mathcal{PE}(\bar{v})$ using $\mathcal{TE}(\bar{v})$, the same approach as in Section 6.3.1 is used. A word $\bar{u}$ of length less than $\frac{n}{2}$ is a prefix of some word in $\mathcal{PE}(\bar{v})$ if and only if no subword of $(\bar{u}_{[1,|\bar{u}|-1]})^R\bar{u}$ is less than the prefix of $\bar{v}$ of the same length. This is slightly different from the odd case, where $\bar{u} \in \mathcal{PE}(\bar{v})$ if and only if there is no subword of $\bar{u}^R\bar{u}$ smaller than the prefix of $\bar{v}$ of the same length. To account for this difference the sets $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ are introduced as analogies to the sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$.

Definition 26. *Let $i \in [\frac{n+1}{2}], j \in [2i]$ and $\bar{s} \sqsubseteq_{2i} \bar{v}$. The set $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ contains every word $\bar{u} \in \mathcal{TE}(\bar{v})$ of length i where (1) the longest suffix of $(\bar{u}_{[1,i-1]})^R\bar{u}_{[1,i]}$ which is a prefix of $\bar{v}$ has a length of j and (2) the word $(\bar{u}_{[1,i-1]})^R\bar{u}_{[1,i]}$ is strictly bounded by $\bar{s} \sqsubseteq_{2i-1} \bar{v}$.*

As in Section 6.3.1, the size of $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ is computed via dynamic programming. The array $SizePE$ is introduced, storing the size of $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ for every value of $i \in \left[\frac{n}{2}\right], j \in [2i-1]$ and $\bar{s} \sqsubseteq_{2i-1} \bar{v}$. Let $SizePE$ be and $n \times n \times n$ array such that $SizePE[i, j, \bar{s}] = |\mathbf{PE}(\bar{v}, i, j, \bar{s})|$. Lemma 22 shows that the techniques used in Lemma 17 can be used to compute $SizePE$ in $O(q \cdot n^3 \log(n))$ time. This is done by proving that the properties established by Lemma 16 regarding the relationship between the sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ also hold for the sets $\mathbf{PE}(\bar{v}, i, j, \bar{s})$. As words in $\mathcal{PS}(\bar{v})$ are of the form $\bar{\phi}\bar{\phi}^R$, a word $\bar{u}$ is in $\mathcal{TS}(\bar{v})$ if and only if no subword of $\bar{u}^R\bar{u}$ is less than the prefix of $\bar{v}$ of the same length. Note that this corresponds to the same requirement as the odd case. As such the internal vertices in the tree $\mathcal{TS}(\bar{v})$ may be partitioned in the same way as those of $\mathcal{TO}(\bar{v})$. Lemma 24 shows how to convert the array $SizePO$ as defined is Section 6.3.1 to the size of $\mathcal{PS}(\bar{v})$.

Lemma 22. *Given $\bar{u}, \bar{w} \in \mathbf{PE}(\bar{v}, i, j, \bar{s})$ and $x \in \Sigma$. If $\bar{u}x \in \mathbf{PE}(\bar{v}, i+1, j', \bar{s}')$ then $\bar{v}x \in \mathbf{PE}(\bar{v}, i+1, j', \bar{s}')$. Further the values of j' and $\bar{s}'$ can be computed in constant time from the values of $j, \bar{s}$ and x. Therefore the array $SizePE[i, j, \bar{s}]$ can be computed for every value $i \in \left[\frac{n}{2}\right], j \in [2i-1]$ and $\bar{s} \sqsubseteq_{2i-1} \bar{v}$ in $O(q \cdot n^3 \cdot \log(n))$ time.*

Proof. Note that these are the same properties as proven in Lemma 16. As the arguments j and $\bar{s}$ serve the same function for both $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ and $\mathbf{PO}(\bar{v}, i, j, \bar{s})$, the arguments from Lemma 16 can be applied directly to this setting.

Following the above arguments, the techniques employed in Lemma 17 can be applied to computing the value of $PE[i, j, \bar{s}]$ for every argument $i \in \left[\frac{n}{2}\right], j \in [2i-1]$ and $\bar{s} \sqsubseteq_{2i-1}$

$\bar{v}$. The only modification needed is to account for the change the form of the words in $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ versus those in $\mathbf{PO}(\bar{v}, i, j, \bar{s})$. As the words in $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ have the form $\bar{\phi}^R x \bar{\phi}$, rather than $\bar{\phi}^R \bar{\phi}$, the set $\mathbf{PE}(\bar{v}, i, j, \bar{s})$ represents words of length $2i - 1$. □

Lemma 23. *Let $\bar{v} \in \Sigma^n$. The size of $\mathcal{PE}(\bar{v})$ can be computed in $O(q \cdot n^3 \cdot \log(n))$ time.*

Proof. Note that for every word $\bar{w} \in \mathcal{PE}(\bar{v})$, either $\bar{w}_{[1,(n/2)]} \in \mathbf{PE}(\bar{v}, \frac{n}{2}, j, \bar{s})$ or $(\bar{w}_{[2,(n/2)-1]})^R \bar{w}_{[1,(n/2)-1]} \sqsubseteq \bar{v}$. Following the arguments in Lemmas 18 and 19, the number of words $\bar{w} \in \mathcal{PE}(\bar{v})$ where $\bar{w}_{[1,(n/2)]} \in \mathbf{PE}(\bar{v}, \frac{n}{2}, j, \bar{s})$ for some given values of $j \in [n-1]$ and $\bar{s} \sqsubseteq_{n-1} \bar{v}$ is equal to the number of symbols $x \in \Sigma$ where $\langle \bar{w}_{[1,(n/2)]} x \bar{w}^R_{[1,(n/2)-1]} \rangle > \bar{v}$. Using the same techniques laid out in Lemma 18, the set of such symbols can be computed in $O(q \cdot n)$ time. It follows that given the array PE, the number of words $\bar{w} \in \mathcal{PO}(\bar{v})$ where $(\bar{w}_1, \bar{w}_2, \ldots . \bar{w}_{n/2}) \in \mathbf{PE}(\bar{v}, \frac{n}{2}, j, \bar{s})$ can be computed in $O(n^2 \cdot q)$ operations by checking every combination of $j \in \left[\frac{n}{2}\right]$, $\bar{s} \in \mathbf{S}(\bar{v}, n-1)$ and $x \in \Sigma$.

Similarly if $\bar{w} \in \mathbf{S}(\bar{v}, n-1)$, then $\bar{w}x \in \mathcal{PE}(\bar{v})$ if and only if $\bar{w} = \bar{\phi}^R x \bar{\phi}$ and $\bar{w}\langle x \rangle > \bar{v}$. Each subword $\bar{s} \in \mathbf{S}(\bar{v}, n-1)$ may be checked in $O(n^2)$ operations by first checking that $\bar{s} = \bar{s}^R$, then finding the smallest rotation of $\bar{s}x$ and comparing it to $\bar{v}$. As there are n words in $\mathbf{S}(\bar{v}, n-1)$ and q symbols in Σ, this takes $O(n^3 \cdot q)$ operations. Computing the arrays PE, WX and XW takes $O(n^3 \cdot q \cdot \log(n))$ time, hence the total complexity is $O(n^3 \cdot q \cdot \log(n))$. □

The size of $\mathcal{PS}(\bar{v})$ is calculated in a similar manner. As the words in $\mathcal{PS}(\bar{v})$ are of the form $\bar{\phi}\bar{\phi}^R$, the prefixes of length i correspond to subwords of length $2i$ with the form $\bar{u}^R\bar{u}$. Note that these are the same as the prefixes used in Section 6.3.1 for odd length palindromic necklaces. As such, the sets $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ are used to partition internal vertices of the tree $\mathcal{TS}(\bar{v})$. Lemma 24 shows how to use these sets to compute the size of $\mathcal{PS}(\bar{v})$.

Lemma 24. *Let $\bar{v} \in \Sigma^n$. The size of $\mathcal{PS}(\bar{v})$ can be computed in $O(q \cdot n^3 \cdot \log(n))$ time.*

Proof. For every word $\bar{w} \in \mathcal{PS}(\bar{v})$ there are two cases to consider:

- **Case 1:** $(\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)-1]} \sqsubseteq \bar{v}$.
- **Case 2:** There exists some set $\mathbf{PO}(\bar{v}, i, \frac{n}{2} - 1, j, \bar{s})$ such that $\bar{w}_{[1,(n/2)-1]} \in \mathbf{PS}(\bar{v}, \frac{n}{2} - 1, j, \bar{s})$.

The number of words in the first case can be computed by considering every subword $\bar{s} \in \mathbf{S}(\bar{v}, n-2)$ and $z \in \Sigma$ where $\bar{s} = \bar{s}^R$ and $\langle z\bar{s}z \rangle > \bar{v}$. Note both of the above conditions can be checked in at most $O(n)$ operations. If both conditions hold, then $\bar{s}$ and z correspond to exactly one word in $\mathcal{PS}(\bar{v})$. As there are n possible values of $\bar{s}$ and q values of z therefore the number of words in this case can be computed in $O(n^2 \cdot q)$ operations.

The number of words in the second case can be computed by considering every vale of $j \in [n-2], \bar{s} \in \mathbf{S}(\bar{v}, n-2)$ and $z \in \Sigma$. Let $\bar{w}_{[1,(n/2)-1]} \in \mathbf{PS}(\bar{v}, \frac{n}{2}-1, j, \bar{s})$. The word $z(\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)-1]} z \in \mathcal{PS}(\bar{v})$ if and only if $\langle z(\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)-1]} z\rangle > \bar{v}$. This is the case if and only if $\bar{v}_{[1,j]} zz\bar{s} > \bar{v}$ which can be checked in $O(n)$ time. If $\bar{v}_{[1,j]} zz\bar{s} > \bar{v}$, then there are $PS[\frac{n}{2}-1, j, \bar{s}]$ prefixes in $\mathbf{PS}(\bar{v}, i, j, \bar{s})$ such that
$\langle z(\bar{w}_{[1,(n/2)-1]})^R \bar{w}_{[1,(n/2)-1]} z\rangle > \bar{v}$. As there are n values of j and $\bar{s}$ and q values of z the number of words in this case can be computed in $O(n^3 \cdot q)$ operations. Finally, in order to compute this case in $O(n^3 \cdot q)$ steps, the array PS must be precomputed, requiring $O(q \cdot n^3 \cdot \log(n))$ operations. Therefore the total complexity is $O(q \cdot n^3 \cdot \log(n))$. □

Combining Lemmas 23 and 24 with Lemmas 20 and 21 provides the tools to compute the rank of $\bar{v}$ among even length palindromic necklaces. Lemma 25 shows how to combine these values to get the rank of $\bar{v}$ among even length palindromic necklaces.

Lemma 25. *The rank of $\bar{v} \in \Sigma^n$ among even length palindromic necklaces can be computed in $O(q \cdot n^3 \cdot \log(n)^2)$ time.*

Proof. From Proposition 15, the number of even length palindromic necklaces is equal to $\frac{1}{2}\left(q^{n/2+1} + 2q^{n/2} + q^l\right) - q$, where $l = \frac{n+2}{4}$ if $\frac{n}{2}$ is odd, or $l = \frac{n}{4}$ if $\frac{n}{2}$ is even. Lemma 20 provides an equation to count the number of necklaces greater than $\bar{v}$ containing at least one word of the form $x\bar{\phi}y\bar{\phi}^R$. The equation given by Lemma 20 requires the size of $\mathcal{PE}(\bar{v})$ to be computed, needing at most $O(q \cdot n^3 \cdot \log(n))$ operations, and either $|\mathcal{PE}(\bar{v}_{[1,n/2]})|$ or $GE(\bar{v}_{[1,n/2]})$. As both $|\mathcal{PE}(\bar{v})|$ and $|\mathcal{PO}(\bar{v})|$ require $O(q \cdot n^3 \cdot \log(n))$ operations, the total complexity comes from the number of such sets that must be considered. As the prefixes of $\bar{v}$ that need to be computed is no more than $\log_2(n)$, the total complexity of computing $GE(\bar{v})$ is $O(q \cdot n^3 \cdot \log^2(n))$. Similarly as the complexity of computing $\mathcal{PS}(\bar{v})$ is $O(q \cdot n^3 \cdot \log(n))$, the complexity of computing $GS(\bar{v})$ is $O(q \cdot n^3 \cdot \log^2(n))$. □

Theorem 16. *Give a word $\bar{v} \in \Sigma^n$, the rank of $\bar{v}$ with respect to the set of palindromic necklaces, $RP(\bar{v})$, can be computed in $O(q \cdot n^3 \cdot \log^2(n))$ time.*

Proof. The number of odd length palindromic necklaces is given by Proposition 5 as $q^{(n-1)/2}$. Lemma 19 shows that the size of set $\mathcal{PO}(\bar{v})$, corresponding to the number of odd length palindromic bracelets, can be computed in $O(q \cdot n^3 \cdot \log(n))$ time. By subtracting the size of $\mathcal{PO}(\bar{v})$ from $q^{(n-1)/2}$, the rank of $\bar{v}$ can be computed in $O(q \cdot n^3 \cdot \log(n))$ time. Lemma 25 shows that of $RP(\bar{v})$ can be computed in $O(q \cdot n^3 \cdot \log^2(n))$ time if the length of $\bar{v}$ is even. Hence the total complexity is $O(q \cdot n^3 \cdot \log^2(n))$. □

6.4 Enclosing Bracelets

Following Lemma 12 and Theorem 16, the remaining problem is counting the number of enclosing words. This section provides a technique to count the number of necklaces enclosing some word $\bar{v}$. As in the palindromic case, the structure of these words are first analysed so that a more efficient algorithm can be derived.

Proposition 16. *The bracelet representation of every bracelet $\hat{\mathbf{w}}$ enclosing the word $\bar{v} \in \Sigma^n$ can be written as $\bar{v}_{[1,i]}x\bar{\phi}$ where; $x \in \Sigma$ is a symbol that is strictly smaller than $\bar{v}_{[i+1]}$, and $\bar{\phi} \in \Sigma^*$ is a word such that every rotation of $(\bar{v}_{[1,i]}x\bar{\phi})^R$ is greater than $\bar{v}$.*

Proof. For the sake of contradiction let $\hat{\mathbf{w}}$ be a bracelet enclosing $\bar{v}$ such that the bracelet representation of $\hat{\mathbf{w}}$, $\bar{a}$ can not be written as $\bar{v}_{[1,i]}x\bar{\phi}$. Let $\bar{b} = \langle \bar{a}^R \rangle$. By the definition of an enclosing necklace, $\bar{a} < \bar{v} < \bar{b}$. If $\bar{a}_1 < \bar{v}_1$, then $\bar{b}_1 < \bar{v}_1$. Similarly if $\bar{b}_1 > \bar{v}_1$ then $\bar{a}_1 > \bar{v}_1$. Hence $\bar{a}_1 = \bar{v}_1 = \bar{b}_1$. Therefore there exists some non zero value of i such that $\bar{a}_{[1,i]} = \bar{v}_{[1,i]}$.

Let i be the length of the longest shared prefix of $\bar{v}$ and $\bar{a}$, i.e. the largest value such that $\bar{v}_{[1,i]} = \bar{a}_{[1,i]}$. If the symbol $\bar{a}_{i+1} > \bar{v}_{i+1}$ $\bar{a} > \bar{v}$ contradicting the assumption that $\bar{a} < \bar{v}$. Similarly if $\bar{a}_{i+1} = \bar{v}_{i+1}$, there is a longer shared prefix. Therefore $\bar{a}_{i+1} < \bar{v}_{i+1}$.

As this word can be written as $\bar{v}_{[1,i]}x\bar{\phi}$, it must be assumed that some rotation of $(\bar{v}_{[1,i]}x\bar{\phi})^R$ is less than or equal to $\bar{v}$. If this is the case, $\hat{\mathbf{w}}$ does not enclose $\bar{v}$, as both necklace classes are smaller than or equal to $\bar{v}$. Therefore the bracelet representation of every bracelet $\hat{\mathbf{w}}$ enclosing the word $\bar{v} \in \Sigma^n$ can be written as stated. □

Proposition 17. *Given a bracelet $\hat{\mathbf{w}}$ enclosing the word $\bar{v} \in \Sigma^n$ of the form $\bar{v}_{[1,j]}x\bar{\phi}$ as given in Proposition 16. The value of x must be greater than or equal to $\bar{v}_{[(j+1) \bmod l]}$ where l is the length of the longest Lyndon word that is a prefix of $\bar{v}_{[1,j]}$.*

Proof. For the sake of contradiction assume that $x < \bar{v}_{[(j+1) \bmod l]}$. Following Theorem 2.1 due to Cattell et. al. [17], the subword $\bar{v}_{[j-(j \bmod l),j]} = \bar{v}_{[1,j \bmod l]}$. Therefore if $x < \bar{v}_{j+1 \bmod l}$ then the subword $\bar{v}_{[1,j \bmod l]}x < \bar{v}_{[1,l]}$. In this case, there is a smaller rotation of $\bar{v}_{[1,j]}x\bar{\phi}$, contradicting our assumption the $\bar{v}_{[1,j]}x\bar{\phi}$ is the smallest rotation. Hence x must be greater than or equal to $\bar{v}_{j+1 \bmod l}$. □

High Level Idea for the Enclosing Case: As in Sections 6.3.1 and 6.3.2, the main idea is to use the structure from Proposition 16 as a basis to count the number of enclosing bracelets. For each value of i and x, the number of possible values of $\bar{\phi}$ are counted. This is done in a recursive manner, working backwards from the last symbol. For each combination of i and x, the key properties to observe are that (1) every suffix of $\bar{\phi}$ must be greater than or equal to $\bar{v}_{[1,i]}x$ and (2) every rotation of $\bar{\phi}^R x \bar{v}_{[1,i]}^R$ is greater than $\bar{v}$.

These observations are used to create a tree, $\mathcal{TEN}(\bar{v}, i, x)$, where each vertex represents a suffix of some possible value of $\bar{\phi}$. Equivalently, the vertices of $\mathcal{TEN}(\bar{v}, i, x)$ can be thought of as representing the prefixes of $\bar{\phi}^R$. The leaf vertices of $\mathcal{TEN}(\bar{v}, i, x)$ represent the possible values of $\bar{\phi}$. As in Section 6.3, each layer of $\mathcal{TEN}(\bar{v}, i, x)$ is grouped into sets based on the lexicographical value of the reflection of the suffixes, and the prefixes of the suffixes. Let $t \in [|\bar{w}| - i]$, $j \in [t + i + 1]$ and $\bar{s} \sqsubseteq_{t+i+1} \bar{v}$. For the t^{th} layer of $\mathcal{TEN}(\bar{v}, i, x)$, the set $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ is introduced containing a subset of the vertices at layer t. The idea is to use the values of j and $\bar{s}$ to divide the prefixes at layer t by lexicographic value and suffix respectively. Let $\bar{u} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ be a suffix of some word $\bar{w}$ such that $\bar{v}_{[1,i]}x\bar{w}$ is a bracelet enclosing $\bar{v}$. To ensure that the necklace represented by the reflection is strictly greater than $\bar{v}$, j is used to track the longest prefix of $\bar{u}^R$ that is a prefix of $\bar{v}$. To ensure that there is no rotation of $x\bar{v}_{[1,i]}^R \bar{w}^R$, the subword $\bar{s} \sqsubseteq_t \bar{v}$ is used to bound the value of $\bar{u}^R$. Formally, $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ contains every suffix $\bar{u} \in \mathcal{TEN}(\bar{v}, i, x)$ of length i where (1) the longest prefix of $\bar{u}^R$ that is also a prefix of $\bar{v}$ and (2) the subword $\bar{s} \sqsubseteq_t \bar{v}$ bounds $\bar{u}^R$.

As in Section 6.3 the number of leaf vertices are calculated by determining the size of the sets $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ at layer $|\bar{v}| - i - 2$, and the number of children of each set. To determine the size of the sets, two key observations must be made. The first is that given the word $\bar{u} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ and the symbol $y \in \Sigma$, if $y\bar{u} \in \mathcal{TEN}(\bar{v}, i, x)$ then there exists some pair $j' \in [n], \bar{s}' \sqsubseteq_{|\bar{u}|+1} \bar{v}$ such that $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$. Secondly, if $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$, then $y\bar{w} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for every $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$. These observations are proven in Lemma 26, as well as showing how to determine the values of j' and $\bar{s}'$.

Lemma 26. *Given $\bar{u} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ and symbol $y \in \Sigma$, the pair $j' \in [n], \bar{s}' \sqsubseteq_{|\bar{u}|+1} \bar{v}$ such that $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ can be computed in constant time. Further, if $y\bar{u} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$, then $y\bar{w} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for every $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$.*

Proof. Assume that the array XW given in Section 6.2 has been precomputed. Following the same arguments as presented in Lemma 16, the value of j' is either $j+1$, if $y = \bar{v}_{j+1}$, or 0 otherwise. Similarly, the value of $\bar{s}'$ is equal to the value given by $XW[\bar{s}, \bar{w}_1]$. Note that if $y\bar{s}' < \bar{v}$ then there is no such value of $\bar{s}'$. Similarly if $y < \bar{v}_{j+1}$ then there no value of j'. To show that $y\bar{w} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for every $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$, recall from Lemma 13 that if $\bar{s}'$ bounds $y\bar{s}$, then $\bar{s}'$ bounds $y\bar{w}$ for every $\bar{w}$ bounded by $\bar{s}$. Similarly, if j' is the length longest suffix of $\bar{u}^R y$ that is a prefix of $\bar{v}$, j' must also be the length of the longest suffix of $\bar{w}^R y$ that is a prefix of $\bar{v}$. □

From Lemma 26, the size of $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ are computed using the sizes of $\mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ for $j' \in [0, n]$ and $\bar{s}' \in \mathbf{S}(\bar{v}, |\bar{s}| + 1)$. To compute the value of $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$, an array SE of size $q \times n \times n \times n^2$ is introduced such that the value of $SE[x, i, j, \bar{s}] = |\mathcal{E}(\bar{v}, i, x, j, \bar{s})|$.

Lemma 27. *Let $\bar{v} \in \Sigma^n$. Let SE be a $n \times n^2$ array such that $SE[x, i, j, \bar{s}] = |\mathcal{E}(\bar{v}, i, x, j, \bar{s})|$ for $j \in [0, n]$ and $\bar{s} \sqsubseteq \bar{v}$. Every value of $SE[x, i, j, \bar{s}]$ is computed in $O(q^2 \cdot n^4)$ time.*

Proof. Initially the value of $SE[j, \bar{s}]$ is set to 0. Observe that every word $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ where $|\bar{w}| > 1$ can be written as $z\bar{w}'$ for $\bar{w}' \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$. From Lemma 26, the value of j' and $\bar{s}'$ can be calculated in constant time. Therefore to efficiently compute the values of SE, it is reasonable to start by computing the size of $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ for every $i \in [0, n], x \in \Sigma, j \in [0, n]$ and $\bar{s} \in \mathbf{S}(\bar{v}, n-1)$. Given $i \in [0, n], x \in \Sigma, j \in [0, n]$ and $\bar{s} \in \mathbf{S}(\bar{v}, n-1)$, the size of $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ is computed directly by checking each value of $z \in \Sigma$. If $z \geq (\bar{v}_{[1,i]}x)_{j+1 \bmod i+1}$ and $x\bar{s} > \bar{v}$ then the value of $SE[i, x, j, \bar{s}]$ is incremented by 1, otherwise it remains the same.

Once the value of $SE[i, x, j, \bar{s}]$ has been computed for every value of $i \in [1, n], x \in \Sigma, j \in [0, n]$ and $\bar{s} \in \mathbf{S}(\bar{v}, n-1)$, the next step is to compute the value of $SE[i', x', j', \bar{s}']$ for every $i' \in [1, n], x' \in \Sigma, j' \in [0, n]$ and $\bar{s}' \in \mathbf{S}(\bar{v}, n-2)$. This is done by looking at each value of $i \in [1, n], x \in \Sigma, j \in [0, n], \bar{s} \in \mathbf{S}(\bar{v}, n-1)$ and $z \in \Sigma$ and determining the values of j' and $\bar{s}'$ for which $z\bar{w} \in \mathcal{E}(\bar{v}, i, x, j', \bar{s}')$ where $\bar{w} \in \mathcal{E}(\bar{v}, i, x, j, \bar{s})$ following Lemma 26. Once the value of j' and $\bar{s}'$ has been determined, $SE[x, i, j', \bar{s}']$ is increased by $SE[x, i, j, \bar{s}]$. By repeating this for every value of $i \in [1, n], x \in \Sigma, j \in [0, n], \bar{s} \in \mathbf{S}(\bar{v}, n-1)$ and $z \in \Sigma$ leaves the value of $SE[x, i, j', \bar{s}']$ as the size of $\mathcal{E}(\bar{v}, i, x, j', \bar{s}')$.

Let $t \in [1, n-1]$. Once every value of $SE[x, i, j, \bar{s}]$ for every value of $i \in [1, n], x \in \Sigma, j \in [0, n]$, and $\bar{s} \in \mathbf{S}(\bar{v}, t)$, the value of $SE[x', i', j', \bar{s}']$ is computed for every $i \in [1, n], x \in \Sigma, j \in [0, n], \bar{s} \in \mathbf{S}(\bar{v}, t-1)$. This is done by determining the value of j' and $\bar{s}'$ for each combination of $i \in [1,n], x \in \Sigma, j \in [0,n], \bar{s} \in \mathbf{S}(\bar{v},t)$ and $z \in \Sigma$ following Lemma 26. Once the value of j' and $\bar{s}'$ has been determined, $SE[x,i,j',\bar{s}']$ is increased by $SE[x,i,j,\bar{s}]$. By repeating this for every value of $i \in [1,n], x \in \Sigma, j \in [0,n], \bar{s} \in \mathbf{S}(\bar{v},t)$ and $z \in \Sigma$ leaves the value of $SE[x,i,j',\bar{s}']$ as the size of $\mathcal{E}(\bar{v},i,x,j',\bar{s}')$.

Repeating this for every value of t from $n-1$ to 1 completely computes the array SE. In order to compute this array, observe that for each of the $O(n)$ values of t, there are $O(n)$ values of i, j and $\bar{s}$ to check alongside $O(q)$ values of x and z. As each combination only needs to be checked once, and the process of determining j' and $\bar{s}'$ can be done in constant time, the total complexity is $O(n^4 \cdot q^2)$. □

Once SE has been computed, the number of enclosing words can be computed using SE and each valid combination of i and x. This is done in a direct manner. Note that the number of possible values of $\bar{\phi}$ such that $\bar{v}_{[1,i]}x\bar{\phi}$ represents a bracelet enclosing $\bar{v}$ is equal to $SE[x,i,j,\bar{s}]$ where j is the longest suffix of $\bar{v}_{[2,i]}x$ that is a prefix of $\bar{v}$ and $\bar{s}$ is the subword that bounds $x\bar{v}^R_{[1,i]}$. As both values can be computed naively in $O(n^2)$ operations, the complexity of this problem comes predominately from computing SE.

Theorem 17. *The number of bracelets enclosing $\bar{v} \in \Sigma^n$ can be computed in $O(n^4 \cdot q^2)$.*

Proof. From Lemma 27 the array SE may be computed in $O(n^4 \cdot q^2)$ operations. Using SE, let $i \in [1,n]$ and $x \in \Sigma$. Further let l be the length of the longest Lyndon word that is a prefix of $\bar{v}_{[1,i]}$. If the value of x is less than $\bar{v}_{i+1 \bmod l}$ or greater than or equal to $\bar{v}_{i+1}$ then there is no bracelet represented by $\bar{v}_{[1,i]}x\bar{\phi}$. Similarly if $x\bar{v}^R_{[1,i]} < \bar{v}_{[1,i+1]}$, then any bracelet of the form $\bar{v}_{1,i}x\bar{\phi}$ does not enclose $\bar{v}$. Otherwise, the number of enclosing bracelets represented by $\bar{v}_{[1,i]}x\bar{\phi}$ is equal to $SE[x,i,j,\bar{s}']$ where j is the longest suffix of $\bar{v}_{[2,i]}x$ that is a prefix of $\bar{v}$ and $\bar{s}$ is the subword that bounds $x\bar{v}^R_{[1,i]}$. By summing the value of $SE[x,i,j,\bar{s}']$ for each value of $i \in [1,n]$ and $x \in \Sigma$ such that $\bar{v}_{[1,i]}x$ is the prefix of the representation of some bracelet enclosing $\bar{v}$ gives the number of enclosing bracelets.

Therefore $RE(\bar{v}) = \sum\limits_{i \in [1,n-1]} \sum\limits_{x \in \Sigma} \begin{cases} 0 & x\bar{v}^R_{[1,i]} < \bar{v} \\ 0 & x \leq \bar{v}_{i+1 \bmod l} \text{ or } x > \bar{v}_{i+1} \\ SE[x,i,j,\bar{s}'] & \textit{Otherwise.} \end{cases}$ □

6.5 Ranking Bracelets

The tools are now available to prove Theorem 14 and show that it is possible to rank a word $\bar{v} \in \Sigma^n$ with respect to the set of bracelets of length n over the alphabet Σ in $O(q^2 \cdot n^4)$

steps.

Proof of Theorem 14 .

Proof. To rank bracelets, it is sufficient to use the results of ranking $\bar{v}$ with respect to necklaces, palindromic necklaces and bracelets enclosing $\bar{v}$, combining them as shown in Lemma 12. Sawada et. al. provided an algorithm to rank v with respect to necklaces in $O(n^2)$ time. It follows from Theorem 16 that the rank with respect to palindromic necklaces can be computed in $O(q \cdot n^3)$ time. Theorem 17 shows that the rank with respect to bracelets enclosing v can be computed in $O(q^2 \cdot n^4)$ time. As combining these results can be done in $O(1)$ steps, therefore the overall complexity is $O(q^2 \cdot n^4)$. Further, this shows that following Lemma 28 the number of bracelets sharing a given prefix $\bar{p}$ can be computed in at most $O(n^4 \cdot q^2)$ time. □

Theorem 18. *Given a word $\bar{v} \in \Sigma^n$, the rank of $\bar{v}$ with respect to the set of bracelets of length n over the alphabet Σ, $RB(\bar{v})$, can be computed with no more than $O(q \cdot n^4)$ space.*

Proof. Following Theorem 14, the space complexity of computing the rank of $\bar{v}$ is equivalent to the greatest space complexity of computing the rank of $\bar{v}$ among the sets of necklaces, palindromic necklaces, and enclosing bracelets. As all three sets require the set of bounding subwords to be precomputed in order to run within the given time bounds, a total of $O(n^2 \cdot q)$ space complexity is needed, dominating the naive $O(n^2)$ space complexity of ranking necklaces.

For Palindromic necklaces, observe that the size of $\mathbf{PO}(\bar{v})$ requires only the sizes of $\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$ and $\mathbf{X}(\bar{v}, j, \bar{s})$ to be computed for every $j \in [n-1]$ and $\bar{s} \sqsubset_{n-1} \bar{v}$, requiring a total of $O(n^2)$ space. Further, the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ can be computed from the set of bounding subwords and the sizes of $\mathbf{PO}(\bar{v}, i-1, j', \bar{s}')$ for every $j' \in [2i-2], \bar{s}' \sqsubset_{2i-2} \bar{v}$. As the sizes of $\mathbf{PO}(\bar{v}, i-1, j', \bar{s}')$ only need to be stored until the size of $\mathbf{PO}(\bar{v}, i, j, \bar{s})$ has been computed for every $j \in [2i], \bar{s} \sqsubset_{2i} \bar{v}$. Therefore only a total of $O(n^2)$ space complexity is needed to compute the size of $\mathbf{PO}(\bar{v}, \frac{n-1}{2}, j, \bar{s})$ for every $j \in [n-1]$ and $\bar{s} \sqsubset_{n-1} \bar{v}$. The same argument may be applied to the even case with the additional cost of storing the size of $O(\log(n))$ sets, giving a total space complexity in the even case of $O(n^2)$. By extension, the space complexity of computing the rank among palindromic necklaces is $O(n^2)$.

In the enclosing case, the rank of $\bar{v}$ can be computed from the size of the sets $\mathcal{E}(\bar{v}, i, x, j, \bar{s})$ for every $x \in \Sigma, i \in [1, n-1], j \in [n]$, and $\bar{s} \sqsubset \bar{v}$, requiring $O(q \cdot n^4)$ space. Therefore the total space complexity required is $O(q \cdot n^4)$ for both ranking within the set of enclosing bracelets and the general set of bracelets. □

Corollary 6. *The z^{th} bracelet of length n over Σ can be computed with no more than $O(q \cdot n^4)$ space.*

Proof. Observe that at each step in the unranking process, it is necessary to rank two bracelets, each requiring $O(q \cdot n^4)$ space. Further, as it is never necessary to store more than the current prefix, and some negligible amount of information concerning the current state of the binary search, the space complexity is dominated by the $O(q \cdot n^4)$ term. Hence the total space complexity is $O(q \cdot n^4)$. □

Using the ranking method given by Theorem 14, a method to derive the number of bracelets sharing a given prefix in polynomial time can be derived. Lemma 28 provides a basis for determining the number of bracelets sharing a given prefix.

Lemma 28. *The number of bracelets sharing a given prefix $\bar{p}$ can be determined in $O(n^4 \cdot q^2)$ time.*

Proof. This is done by comparing the rank of the smallest and largest necklaces with a given prefix. The canonical form of the largest necklace is found using the word $\bar{w}$ where $\bar{w}_i = \bar{p}_{i \bmod |\bar{p}|}$. If $\bar{w}$ is not a necklace, then using the algorithm outlined in Theorem 28, the smallest word $\bar{w}' > \bar{w}$ such that $\bar{w}'$ is the canonical form of some necklace, can be found in $O(n)$ time. In the other direction, the word $\bar{v}$ representing the largest necklace with $\bar{p}$ is defined as $\bar{v} = \bar{p} : q^{n-|\bar{p}|}$, where q is the largest symbol in the alphabet Σ.

In order to determine the number of bracelets between these two necklaces, the ranking algorithm that is presented in Chapter 6 is used. As $RB(\bar{w})$ returns the number of bracelets smaller than $\bar{w}$, the number of bracelets between $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{v}}$ equals $RB(\bar{v}) - RB(\bar{w}) + 1$. Theorem 14 shows that $RB(\bar{w})$ can be computed in at most $O(n^4 \cdot q^2)$ time, where n is the length of $\bar{w}$ and q the size of the alphabet. Therefore the number of bracelets sharing a given prefix $\bar{p}$ can be computed in at most $O(n^4 \cdot q^2)$ time. □

Chapter 7

Ranking and Unranking Constrained Necklaces

In this chapter we consider two sets of possible constraints on necklaces. The first of these is the set of necklaces corresponding to positive integer solutions to linear Diophantine equations with positive weights. These constraints not only provide a useful tool for necklaces, but have the independent motivation of sampling solutions to linear equations. In particular, these solutions match the problems present in tools such as MC-EMMA or FUSE, where the goal is to find combinations of blocks that match a given chemical formula. The second set we look at is the set of necklaces with forbidden subwords. These are simply necklaces that do not contain as a subword any word from some given set. This set can be used to provide a means to avoid known bad combinations, by eliminating them as potential centres.

The remainder of this chapter is organised as follows. Section 7.1 provides results for counting, ranking and unranking n-weight Parikh vectors solving diophantine equations, i.e. Parikh vectors solving some given set of linear diophantine equations where the sum of the entries equals n. Section 7.2 generalises the results from Section 7.1 from Parikh vectors to necklaces with the given Parikh vectors. Explicitly, Section 7.2 provides results for counting, ranking, and unranking necklaces of length n with Parikh vectors solving some given set of Diophantine equations. Finally, Section 7.3 provides algorithms for ranking and unranking necklaces under a given set of forbidden subwords.

7.1 n-Weight Parikh Vectors

The first setting that this chapter looks at is counting the number of n-weight Parikh vectors solving some system of linear equations. Recall from Section 2.2.1 that a Parikh vector $\overline{\mathbf{P}}$ has weight n if the sum of entries $P_1 + P_2 + \ldots + P_q = n$. Note that every entry in a Parikh vector must be either 0 or some positive integer. This section is interested in Parikh vectors of weight n that can solve a given system of m linear equations for q variables, determined by some matrix of size $m \times q$ $A \in \mathbb{N}^{m,q}$ and vector $\overline{\mathbf{C}} \in \mathbb{N}^m$ of length m. A Parikh vector $\overline{\mathbf{P}}$ of length q solves such a system if $A \cdot \overline{\mathbf{P}} = \overline{\mathbf{C}}$.

This direction is motivated by the problem of choosing blocks in settings such as FUSE or MC-EMMA, without any consideration for the local structures. As such, our systems of linear equations can be thought of as corresponding to chemical equations, with the vector $\overline{\mathbf{C}}$ denoting the number of each ion species appearing in the global structure such that $\overline{\mathbf{C}}_i$ denotes how many ions of species i must appear in the overall structure. Similarly $A_{i,j}$ specifying how many times the j^{th} block contains the i^{th} species.

Parikh words, the canonical form of the Parikh vectors, are used as a basis for comparison. Recall that the Parikh word corresponding to a given n-weight Parikh vector $\overline{\mathbf{P}}$ of length q is the word $1^{P_1} : 2^{P_2} : \ldots : q^{P_q}$. For a given system of linear equations, the value of q is simply the number of variables in the system.

The first problem this section considers is that of counting the number of Parikh vectors with weight n solving a given set of linear equations defined by the matrix $A \in \mathbb{N}^{m,q}$ and vector $\overline{\mathbf{C}} \in \mathbb{N}^m$, i.e. determining the size of the set $\mathbf{P}(n, A, \overline{\mathbf{C}})$. The key observation to make is that the number of vectors in $\mathbf{P}(n, A, \overline{\mathbf{C}})$ where $x_i \geq 1$ equals $|\mathbf{P}(n-1, A, \overline{\mathbf{C}}) - w(i)|$, where $w(i)$ returns the vector of length q corresponding to the i^{th} column of A.

In order to count the size of $\mathbf{P}(n, A, \overline{\mathbf{C}})$ precisely, the number of Parikh words is used as a way of uniquely representing each vector. As the i^{th} symbol of each Parikh word is no smaller than the $(i-1)^{th}$ symbol, it is reasonable to compute the number of such words based on the number of suffixes. Using this observation as a basis, let $Count(n, A, \overline{\mathbf{C}}, i)$ return the number of solutions to $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$ where the first $i-1$ variables are set to 0, i.e. $\overline{\mathbf{x}}_1 = \overline{\mathbf{x}}_2 = \ldots = \overline{\mathbf{x}}_{i-1} = 0$. Note that the number of vectors in $\mathbf{P}(n, A, \overline{\mathbf{C}})$ corresponding to a Parikh word for which the first symbol is i equals $Count(n-1, A, \overline{\mathbf{C}} - \mathrm{w}(i), i)$. Recall from the preliminaries that $\mathrm{w}(i)$ returns the vector corresponding to the i^{th} column of A. More generally, note that the size of $Count(n, A, \overline{\mathbf{C}}, i)$ is $\sum\limits_{j=i}^{q} Count(n-1, A, \overline{\mathbf{C}} - \mathrm{w}(j), j)$.

Further, observe that $Count(0, A, \overline{\mathbf{C}}, i)$ equals either 1, if $\overline{\mathbf{C}}_1 = \overline{\mathbf{C}}_2 = \ldots = \overline{\mathbf{C}}_q = 0$ or 0 otherwise. For notation, let $\overline{\mathbf{C}} = 0$ if $\overline{\mathbf{C}}_1 = \overline{\mathbf{C}}_2 = \ldots = \overline{\mathbf{C}}_q = 0$ and $\overline{\mathbf{C}} \neq 0$ if $\overline{\mathbf{C}}_i \neq 0$ for any $i \in [q]$. Using these observations, the following recursive formula for $Count(n, A, \overline{\mathbf{C}}, i)$ is derived:

$$Count(n, A, \overline{\mathbf{C}}, i) = \begin{cases} \sum\limits_{j=i}^{q} Count(n-1, A, \overline{\mathbf{C}} - \mathrm{w}(j), j) & n > 0 \\ 1 & n = 0, \overline{\mathbf{C}} = 0 \\ 0 & n = 0, \overline{\mathbf{C}} \neq 0 \end{cases}$$

Lemma 29. *The size of $\mathbf{P}(n, A, \overline{\mathbf{C}})$ can be computed in $O(C \cdot n \cdot q^2)$ operations, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$, and the number of variables in $\overline{\mathbf{x}}$ is q.*

Proof. Note that $|\mathbf{P}(n, A, \overline{\mathbf{C}})| = Count(n, A, \overline{\mathbf{C}}, 1)$. Observe that $Count(n, A, \overline{\mathbf{C}}, i)$ can be computed in $O(q)$ operations if $Count(n-1, A, \overline{\mathbf{Q}}, j)$ has been computed for every $j \in [q]$ and $\overline{\mathbf{Q}} \in [\overline{\mathbf{C}}]$. Further note that $Count(0, A, \overline{\mathbf{C}}, i)$ can be computed in m operations by checking that $\overline{\mathbf{C}}_j = 0$ for every $j \in [m]$. Using these observations, a dynamic programming approach is derived. Starting with $t = 0$, the value of $Count(t, A, \overline{\mathbf{Q}}, i)$ is computed for every combination of $\overline{\mathbf{Q}} \in [\overline{\mathbf{C}}]$, $t \in [0, n]$, $i \in [q]$ in increasing value of t. Note that the number of possible values of $[\overline{\mathbf{Q}}]$ equals the number of vectors of the from $(x_1, x_2, \ldots, x_q)$ where $0 \leq x_i \leq \overline{\mathbf{C}}_i$, giving a total of $(\overline{\mathbf{C}}_1 + 1) \cdot (\overline{\mathbf{C}}_2 + 1) \cdot \ldots \cdot (\overline{\mathbf{C}}_m + 1)$ possible values which is of order $O(C)$. Therefore, as there are $O(C)$ such values of $\overline{\mathbf{Q}}$, n values of t and q values of i, the total number of values of $Count(t, A, \overline{\mathbf{Q}}, i)$ is $O(C \cdot n \cdot q)$. Further, as each computation requires at most q operations, the total complexity of computing the size of $\mathbf{P}(n, A, \overline{\mathbf{C}})$ is $O(C \cdot n \cdot q^2)$. □

This leaves the problem of *ranking* n-weight Parikh vectors solving a system of linear equations. Let $\mathbf{R}(\bar{w})$ denote the set of Parikh vectors in $\mathbf{P}(n, A, \overline{\mathbf{C}})$ corresponding to a Parikh word smaller than some given word $\bar{w} \in \Sigma(A)^n$. The size of $\mathbf{R}(\bar{w})$, denoted $|\mathbf{R}(\bar{w})|$, is computed by partitioning $\mathbf{R}(\bar{w})$ into n sets denoted $\mathbf{A}(\bar{w}, i)$ such that $\mathbf{A}(\bar{w}, i)$ contains every word $\bar{v} \in \mathbf{R}(\bar{w})$ where $\bar{w}_{[1,i-1]} = \bar{v}_{[1,i-1]}$ and $\bar{w}_i > \bar{v}_i$. In this way, the set $\mathbf{A}(\bar{w}, i)$ can be thought of as the set of potential suffixes to the word $\bar{w}_{[1,i]}$ such that for every $\bar{v} \in \mathbf{A}(\bar{w}, i)$, $\bar{v} \leq \bar{w}_{[i+1,n]}$ and $A \cdot \mathrm{P}(\bar{w}_{[1,i]} : \bar{v}) = \overline{\mathbf{C}}$. Using these sets, the size of $\mathbf{R}(\bar{w})$ is given by $|\mathbf{R}(\bar{w})| = \sum\limits_{i=1}^{n} \mathbf{A}(\bar{w}, i)$.

7.1.1 Computing the Number of Suffixes of Parikh Words

This sections shows how to count the size of the set $\mathbf{A}(\bar{w}, i)$. Observe that every word $\bar{v} \in \mathbf{A}(\bar{w}, i)$ can be written as $\bar{w}_{[1,i-1]} : \bar{v}_{[i,n]}$ where $A \cdot \mathrm{P}(\bar{v}_{[i,n]}) = \overline{\mathbf{C}} - A \cdot \mathrm{P}(\bar{w}_{[1,i-1]})$. Further, as $\bar{v} < \bar{w}$, $\bar{v}_i < \bar{w}_i$. Similarly, as $\bar{v}$ is a Parikh word, $\bar{v}_i \geq \bar{w}_{i-1}$. Note that $\bar{v}_{[i,n]}$ must itself be a Parikh word of size $n - i$ solving the system of linear equations defined by matrix $A \in \mathbb{N}^{m,q}$ and the vector $\overline{\mathbf{C}} - A \cdot \mathrm{P}(\bar{w}_{[1,i-1]})$. Therefore the number of possible values of $\bar{v}_{[i,n]}$, and hence the size of $\mathbf{A}(\bar{w}, i)$, corresponds to $\sum\limits_{j=\bar{w}_{i-1}}^{\bar{w}_i - 1} Count(n-i, A, A \cdot \mathrm{P}(\bar{w}_{[1,i-1]}) - \mathrm{w}(j), j)$. Hence the rank of $\bar{w}$ equals:

$$|\mathbf{R}(\bar{w})| = \sum_{i=1}^{n} \sum_{j=\bar{w}_{i-1}}^{\bar{w}_i - 1} Count(n - i, A, \overline{\mathbf{C}} - A \cdot \mathrm{P}(\bar{w}_{[1,i-1]}) - \mathrm{w}(j), j)$$

Theorem 19. *The rank of $\bar{w}$ among the set $\mathbf{P}(n, A, \overline{\mathbf{C}})$ can be computed in at most $O(C \cdot n \cdot q^2)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables in the system of linear equations.*

Proof. Using the same dynamic programming approach as in Lemma 29, the value of $Count(i, A, \overline{\mathbf{Q}}, j)$ can be computed for every $\overline{\mathbf{Q}} \in [\overline{\mathbf{C}}], i \in [n]$ and $j \in [q]$, in at most $O(C \cdot n \cdot q^2)$ time. Using the equation $|\mathbf{R}(\bar{w})| = \sum\limits_{i=1}^{n} \sum\limits_{j=\bar{w}_{i-1}}^{\bar{w}_i - 1} Count(n-i, A, \overline{\mathbf{C}} - A \cdot \mathrm{P}(\bar{w}_{[1,i-1]}) - \mathrm{w}(j), j)$, the size of $\mathbf{R}(\bar{w})$ can be computed in $O(n \cdot q)$ time once the size of $Count(i, A, \overline{\mathbf{Q}}, j)$ has been computed for every value of $\overline{\mathbf{Q}} \in [\overline{\mathbf{C}}], i \in [n]$ and $j \in [q]$. As the precomputation of $Count(i, A, \overline{\mathbf{Q}}, j)$ can be done separately to the computation of $|\mathbf{R}(\bar{w})|$, the total time complexity of this process is at most $O(C \cdot n \cdot q^2)$ time. □

7.1.2 Unranking n-Weight Parikh Vector Solutions to Linear Diophantine Equations

Complementing the ranking results given in Theorem 19, the next problem we consider is that of *unranking* the set $\mathbf{P}(n, A, \overline{\mathbf{C}})$. Recall that the unranking problem asks for the i^{th} member of an ordered set, in this case the set of Parikh words corresponding to the Parikh vectors in $\mathbf{P}(n, A, \overline{\mathbf{C}})$. The unranking process is done by iteratively computing the prefix of the i^{th} Parikh vector in increasing length of prefix of the Parikh word. Note that this is the same high level approach as used in the bracelet setting in Theorem 15. Lemma

30 provides the technical background for both the unranking approach, and the k-centre problem.

Lemma 30. *The number of Parikh vectors in* $\mathbf{P}(n, A, \overline{\mathbf{C}})$ *where the prefix of the Parikh word is* $\bar{a} \in \Sigma(A)^*$ *can be computed in* $O(C \cdot n \cdot q^2)$ *time, where* $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ *and* q *is the number of variables in the system of linear equations.*

Proof. Following Theorem 19, the number of Parikh vectors corresponding to Parikh words with a prefix smaller than $\bar{a}$ can be computed using $|\mathbf{R}(\bar{a} : \bar{a}_{|\bar{a}|}^{n-|\bar{a}|})|$. Similarly the number of Parikh vectors represented with a word with a prefix smaller than or equal to $\bar{a}$ is given by $|\mathbf{R}(\bar{a} : q^{n-|\bar{a}|})| + 1$. Therefore the total number of words sharing a given prefix $\bar{a}$ is given by $|\mathbf{R}(\bar{a} : q^{n-|\bar{a}|})| + 1 - |\mathbf{R}(\bar{a} : \bar{a}_{|\bar{a}|}^{n-|\bar{a}|})|$. As the sizes of $\mathbf{R}(\bar{a} : q^{n-|\bar{a}|})$ and $\mathbf{R}(\bar{a} : \bar{a}_{|\bar{a}|}^{n-|\bar{a}|})$ can be computed in $O(C \cdot n \cdot q^2)$ time, the total complexity of this process is $O(C \cdot n \cdot q^2)$. □

Lemma 30 provides the basis for both the k-centre problem and the unranking process. The k-centre problem for this set is discussed in Theorem 13. For the unranking problem, at a high level, the idea is to use take the prefix of length j and extend it using a binary search on the alphabet $\Sigma(A)$. By repeating this process n times, the i^{th} solution can be found.

Theorem 20. *The* i^{th} *Parikh vector in the set* $\mathbf{P}(n, A, \overline{\mathbf{C}})$ *can be found in* $O(C \cdot n^2 \cdot q^2 \cdot \log(q)))$ *time where* $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ *and* q *is the number of variables in the system of linear equations.*

Proof. Let $\bar{w}$ be the Parikh word corresponding to the i^{th} Parikh vector in $\mathbf{P}(n, A, \overline{\mathbf{C}})$. In order to determine the first symbol of $\bar{w}$, observe that $|\mathbf{R}(\bar{w}_1^n)| \leq i \leq |\mathbf{R}(\bar{w}_1 : q^{n-1})|$. By preforming a binary search on the set of symbols $\Sigma(A)$, the value of the first symbol can be determined in $\log(q)$ steps, each requiring two words to be ranked giving a total of $O(C \cdot n \cdot q^2 \cdot \log(q)))$ time. More generally, the j^{th} symbol of $\bar{w}$ can be determined from the first $j-1$ symbols. As with the first symbol, the j^{th} can be determined as the symbol $x \in \Sigma(A)$ where $|\mathbf{R}(\bar{w}_{[1,j-1]} : x^{n-j+1})| \leq i \leq |\mathbf{R}(\bar{w}_{[1,j-1]} : x : q^{n-j})|$. As there are n symbols to determine for $\bar{w}$, the total amount of time required to compute the i^{th} Parikh vector is $O(C \cdot n^2 \cdot q^2 \cdot \log(q))$. □

7.2 Necklaces with n-Weight Parikh Vectors

7.2.1 Counting Necklaces with n-Weight Parikh Vectors

Following the results from the Section 7.1, the next step is to generalise to the set of necklaces corresponding to Parikh vectors solving a given set of linear equations. As with the set $\mathbf{P}(n, A, \overline{\mathbf{C}})$, we consider three problems, those of counting, ranking and unranking the set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$. The first problem we consider is that of computing the size of $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$. Recall that the set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ contains every necklace $\tilde{\mathbf{w}} \in \mathcal{N}_q^n$ such that $A \cdot \mathrm{P}(\tilde{\mathbf{w}}) = \overline{\mathbf{C}}$ for some $m \times q$ matrix $A \in \mathbb{Z}^{m,q}$ and vector $\overline{\mathbf{C}} \in \mathbb{Z}^m$ of length m. For notation, let $Q(A, \overline{\mathbf{C}}, n)$ return the number of words $\bar{w} \in \Sigma(A)$ such that $A \cdot \mathrm{P}(\bar{w}) = \overline{\mathbf{C}}$ Following the classical arguments for counting the number of necklaces such as those provided by Knuth et al. [59], the size of $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ is given by:

$$|\mathcal{N}_q^n(A, \overline{\mathbf{C}})| = \frac{1}{n} \sum_{d|n} \phi\left(\frac{n}{d}\right) Q\left(A, \frac{\overline{\mathbf{C}}}{d}, d\right)$$

Where $\phi(x)$ is Euler's totient function, returning the number of positive integers co-prime to x and $\frac{\overline{\mathbf{C}}}{d} = \left(\frac{\overline{\mathbf{C}}_1}{d}, \frac{\overline{\mathbf{C}}_2}{d}, \dots, \frac{\overline{\mathbf{C}}_q}{d}\right)$. This leaves the problem of computing $Q(A, \overline{\mathbf{C}}, n)$. This is done in a recursive manner. Observe that the number of words counted by $Q(A, \overline{\mathbf{C}}, n)$ for which the first symbol is $x \in \Sigma(A)$ equals $Q(A, \overline{\mathbf{C}} - \mathrm{w}(x), n-1)$. As in Section 7.1, $\mathrm{w}(x)$ is used to denote the vector corresponding to the x^{th} column of $A \in \mathbb{N}^{m,q}$. Therefore, for $n \geq 1$, $Q(A, \overline{\mathbf{C}}, n) = \sum_{x \in \Sigma(A)} Q(A, \overline{\mathbf{C}} - \mathrm{w}(x), n-1)$. The value of $Q(A, \overline{\mathbf{C}}, 0)$ is either 1, if $\overline{\mathbf{C}} = 0$, or 0 if $\overline{\mathbf{C}}_i \neq 0$ for any $i \in [q]$. Hence $Q(A, \overline{\mathbf{C}}, n)$ is given by:

$$Q(A, \overline{\mathbf{C}}, n) = \begin{cases} \sum_{x \in \Sigma(A)} Q(A, \overline{\mathbf{C}} - \mathrm{w}(x), n-1) & n > 0 \\ 1 & n = 0 \text{ and } \overline{\mathbf{C}} = 0 \\ 0 & n = 0 \text{ and } \overline{\mathbf{C}} \neq 0 \end{cases}$$

Lemma 31. *The value of $Q(A, \overline{\mathbf{C}}, n)$ can be computed in $O(q \cdot n \cdot C)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables.*

Proof. Using the above equation, the value of $Q(A, \overline{\mathbf{C}}, n)$ may be computed in $O(q)$ time if the value of $Q(A, \overline{\mathbf{P}}, n-1)$ has been computed for every $\overline{\mathbf{P}} \in [\overline{\mathbf{C}}]$. Further, the value of $Q(A, \overline{\mathbf{P}}, 0)$ can be computed in $O(m)$ time. Therefore, by computing $Q(A, \overline{\mathbf{P}}, n)$ in increasing value of n for all C possible values of $\overline{\mathbf{P}} \in [\overline{\mathbf{C}}]$, every value of $Q(A, \overline{\mathbf{P}}, n)$ can be computed in $O(q \cdot n \cdot C)$ time. □

Theorem 21. *The number of necklaces corresponding to n-weight Parikh vectors solving the system of linear equations defined by the $m \times q$ matrix $A \in \mathbb{N}^{m,q}$ and vector $\overline{\mathbf{C}} \in \mathbb{N}^m$ of length m can be computed in $O(q \cdot n \cdot C)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables.*

Proof. Following Lemma 31, the value of $Q(A, \overline{\mathbf{P}}, i)$ can be computed in at most $O(q \cdot n \cdot C)$ time for every $i \in [n]$ and $\overline{\mathbf{P}} \in [\overline{\mathbf{C}}]$. Similarly, the value of $\phi(i)$ can be computed in at most $O(n)$ time. Assuming the value of $Q(A, \overline{\mathbf{P}}, i)$ has been precomputed for every value of $i \in [n]$ and $\overline{\mathbf{P}} \in [\overline{\mathbf{C}}]$, and the value of $\phi(i)$ precomputed for $i \in [n]$, the size of $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ can be computed in $O(n)$ time. Therefore as the precomputation of $Q(A, \overline{\mathbf{P}}, i)$ dominates this process, the total complexity is $O(q \cdot n \cdot C)$. □

7.2.2 Ranking n-Length Necklaces with Parikh Vector Solving a System of Linear Diophantine Equations

Using the counting approach outlined in Section 7.2.1 as a basis, the next problem we consider is how to rank these necklaces in lexicographical order. Let $RN(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ be the rank of a necklace $\tilde{\mathbf{w}}$ in $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$. At a high level, this rank is computed in the same manner as for unconstrained necklaces, using the set of words belonging to a necklace class smaller than $\tilde{\mathbf{w}}$ as a basis. For notation, let $S(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}})$ be the set of words belonging to a necklace class in $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ smaller than $\tilde{\mathbf{w}}$, i.e. $S(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}}) = \{\langle \tilde{\mathbf{v}} \rangle_i | \tilde{\mathbf{v}} \in \mathcal{N}_q^n(A, \overline{\mathbf{C}}), \tilde{\mathbf{v}} < \tilde{\mathbf{w}}, i \in [n]\}$. The following technical Lemma lays the foundations for ranking within the set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$.

Lemma 32. *The rank of a necklace $\tilde{\mathbf{w}}$ in the set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ is given by the equation* $RN(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}}) = \sum\limits_{d|n} \frac{1}{d} \left(\sum\limits_{l|d} \mu\left(\frac{d}{l}\right) |S(\tilde{\mathbf{w}}, l, A, \frac{l \cdot \overline{\mathbf{C}}}{n})| \right)$

Proof. In order to use the size of $S(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}})$ to compute the rank of $\tilde{\mathbf{w}}$, the subset of aperiodic necklaces in $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ is used. Let $RL(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ be the number of length n aperiodic necklaces in $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ smaller than $\tilde{\mathbf{w}}$. Observe that $RN(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}}) = \sum\limits_{d|n} RL(\tilde{\mathbf{w}}, d, A, \frac{d \cdot \overline{\mathbf{C}}}{n})$.

In order to compute $RL(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$, let $S'(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}}) \subseteq S(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}})$ be the subset of $S(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}})$ containing all aperiodic words in $S(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}})$, i.e. $S'(A, \overline{\mathbf{C}}, n, \tilde{\mathbf{w}}) = \{\bar{v} \in \Sigma(A)^n : A \cdot \mathrm{P}(\bar{v}) = \overline{\mathbf{C}}, \langle \bar{v} \rangle < \tilde{\mathbf{w}}, \bar{v}$ is aperiodic $\}$. Note that as every aperiodic necklace of length n contains n representative in $S'(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ $RL(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}}) = \frac{|S'(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})|}{n}$.

This leaves the problem of computing the size of $S'(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$. Note that the size of $S(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ is given by $|S(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})| = \sum\limits_{d|n} |S'(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})|$. By applying the Möbius inversion formula the size of $S'(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ can be counted in terms of $S(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ as $|S'(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}}) = \sum\limits_{d|n} \mu\left(\frac{n}{d}\right) |S(\tilde{\mathbf{w}}, d, A, \frac{d \cdot \overline{\mathbf{C}}}{n})|$. Putting this together gives $RN(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}}) =$
$\sum\limits_{d|n} \frac{1}{d} \left(\sum\limits_{l|d} \mu\left(\frac{d}{l}\right) |S(\tilde{\mathbf{w}}, l, A, \frac{l \cdot \overline{\mathbf{C}}}{n})| \right)$ □

Lemma 32 leaves the problem of determining the size of $S(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$. The size of $S(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ is computed by partitioning the set into the subsets $A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ where $A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ contains every word $\bar{v} \in S(\tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ where:

- i is the smallest translation of $\bar{v}$ such that $\langle \bar{v} \rangle_i \leq \langle \tilde{\mathbf{w}} \rangle$.
- j is the length of the longest prefix of $\langle \bar{v} \rangle_i$ that is also a prefix of $\langle \tilde{\mathbf{w}} \rangle$, i.e. the largest value such that $(\langle \bar{v} \rangle_i)_{[1,j]} = (\langle \tilde{\mathbf{w}} \rangle)_{[1,j]}$.

The size of $A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ is computed by considering two cases based on the value of $i + j$ relative to n.

Case 1. $i + j < n$**:** In this case every word $\bar{v} \in A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ can be written as $\bar{a} : (\langle \tilde{\mathbf{w}} \rangle)_{[1,j]} : b : \bar{c}$ where:

- $\bar{a}$ is a word of length i such that every suffix of $\bar{a}$ is greater than than prefix of $(\langle \tilde{\mathbf{w}} \rangle)$ of the same length, i.e. $\bar{a}_{[i-l:i]} = (\langle \tilde{\mathbf{w}} \rangle)_{[1:l]}$.
- $b \in \Sigma(A)$ is some symbol smaller than $(\langle \tilde{\mathbf{w}} \rangle)_{j+1}$.
- $\bar{c}$ is unconstrained other than that $A \cdot \mathrm{P}(\bar{a} : (\langle \tilde{\mathbf{w}} \rangle)_{[1,j]} : x : \bar{c}) = \overline{\mathbf{C}}$.

To compute the number of possible values of $\bar{a}$, the function $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}')$ is introduced. Formally, $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$ counts the number of words $\bar{v} \in \Sigma(A)^x$ such that (1) $\bar{v}_{[i-l,i]} > \langle \tilde{\mathbf{w}} \rangle_{[1,l]}$ for every $l \in [0, i-1]$ (2) $\bar{v}_{[1,j]} = \langle \tilde{\mathbf{w}} \rangle_{[1,j]}$ and (3) $A \cdot \mathrm{P}(\bar{v}) = \overline{\mathbf{C}}'$.

The value of $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}')$ can be computed in a similar manner to the $Q(n, A, \overline{\mathbf{C}})$. Note that every word counted by $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}')$ must have, as the $(j+1)^{th}$ symbol some symbol greater than or equal to $\langle \tilde{\mathbf{w}} \rangle_{j+1}$. The number of words with the symbol $x > \langle \tilde{\mathbf{w}} \rangle_{j+1}$ at position $j+1$ equals $\alpha(\tilde{\mathbf{w}}, i-j-1, 0, A, \overline{\mathbf{C}}' - A \cdot \mathrm{P}(\langle \tilde{\mathbf{w}} \rangle_{[1,j]} : x))$. Similarly the number of words counted by $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}')$ where the symbol at position $j+1$ is $\langle \tilde{\mathbf{w}} \rangle_{j+1}$ is given by $\alpha(\tilde{\mathbf{w}}, i, j+1, A, \overline{\mathbf{C}}')$. In the case where $i = j$, the size of $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}')$ is either 1, if $j = 0$ and $\overline{\mathbf{C}}' = 0$, or 0 otherwise. Using these observations the value of $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}')$ is given by:

$$\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}') = \begin{cases} \left(\begin{array}{l} \alpha(\tilde{\mathbf{w}}, i, j+1, A, \overline{\mathbf{C}}') + \\ \displaystyle\sum_{x=(\langle \tilde{\mathbf{w}} \rangle)_{j+1}}^{q} \alpha(\tilde{\mathbf{w}}, i-j-1, 0, A, \overline{\mathbf{C}}' - A \cdot \mathrm{P}(\langle \tilde{\mathbf{w}} \rangle_{[1,j]} : x)) \end{array} \right) & i > j \\ 1 & i = j = 0, \overline{\mathbf{C}}' = 0 \\ 0 & i = j > 0 \\ 0 & i = j = 0, \overline{\mathbf{C}}' \neq 0 \end{cases}$$

Let $\overline{\mathbf{P}}$ be the Parikh vector of $\bar{a}$. The number of values of b is given by:

$$\sum_{b=1}^{(\langle \tilde{\mathbf{w}} \rangle)_{j+1}-1} \begin{cases} 1 & (A \cdot (\overline{\mathbf{P}} + \mathrm{P}(b)))_i \leq \overline{\mathbf{C}}_i, \forall i \in [m] \\ 0 & otherwise \end{cases}.$$

Similarly, given the value of b, the number of possible values of $\bar{c}$ is computed by $Q(n-i-j-1, A, \overline{\mathbf{C}} - (\overline{\mathbf{P}} + \mathrm{P}(b)))$. Let $\beta(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}}') = \sum_{b=1}^{(\langle \tilde{\mathbf{w}} \rangle)_{j+1}-1} Q(n-i-j-1, A, \overline{\mathbf{C}}' - A \cdot \mathrm{P}(b))$. Combining these observations, the size of $A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ is given by:

$$|A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})| = \sum_{\overline{\mathbf{Q}} \in [\overline{\mathbf{C}}]} \alpha(\tilde{\mathbf{w}}, i, 0, A, \overline{\mathbf{Q}}) \cdot \beta(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}} - \overline{\mathbf{Q}})$$

Case 2. $i + j \geq n$**:** In this case, every word $\bar{v} \in A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ can be written as $(\langle \tilde{\mathbf{w}} \rangle)_{[i+j-n,j]} : b : \bar{a} : (\langle \tilde{\mathbf{w}} \rangle)_{[1,i+j-n-1]}$ where:

- Every suffix of $(\langle \tilde{\mathbf{w}} \rangle)_{[i+j-n,j]} : b : \bar{a}$ is greater than the prefix of $\langle \tilde{\mathbf{w}} \rangle$ of the same length.

- $b \in \Sigma(A)$ is a symbol such that $b < (\langle \tilde{\mathbf{w}} \rangle)_{j+1}$.

Let s be the length of the longest suffix of $(\langle \tilde{\mathbf{w}} \rangle)_{[i+j-n,j]}$ that is a prefix of $\langle \tilde{\mathbf{w}} \rangle$. Formally, let s be the largest value such that $((\langle \tilde{\mathbf{w}} \rangle)_{[i+j-n,j]})_{[n-i-s,n-i]} = (\langle \tilde{\mathbf{w}} \rangle)_{[1,s]}$. Note that for every suffix of $(\langle \tilde{\mathbf{w}} \rangle)_{[i+j-n,j]} : b : \bar{a}$ to be greater than the prefix of the same length, b must be greater than or equal to $(\langle \tilde{\mathbf{w}} \rangle)_{s+1}$. The number of possible values of $\bar{a}$ is either $\alpha(\tilde{\mathbf{w}}, n-j-1, 0, A, \overline{\mathbf{C}} - A \cdot (\mathrm{P}((\langle \tilde{\mathbf{w}} \rangle)_{[1,j]} : b)))$, if $b > (\langle \tilde{\mathbf{w}} \rangle)_{s+1}$, or $\alpha(\tilde{\mathbf{w}}, n-j+s, s+1, A, \overline{\mathbf{C}} - A \cdot (P((\langle \tilde{\mathbf{w}} \rangle)_{[1,j+1]})))$ if $b = \langle \tilde{\mathbf{w}} \rangle)_{[1,j+1]}$. This gives the size of $A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})$ as:

$$\begin{aligned}|A(i, j, \tilde{\mathbf{w}}, n, A, \overline{\mathbf{C}})| =& \alpha(\tilde{\mathbf{w}}, n-j+s, s+1, A, \overline{\mathbf{C}} - A \cdot (P((\langle \tilde{\mathbf{w}} \rangle)_{[1,j]} : (\langle \tilde{\mathbf{w}} \rangle)_{[s+1]})))+ \\ & \sum_{b=(\langle \tilde{\mathbf{w}} \rangle)_{s+1}+1}^{(\langle \tilde{\mathbf{w}} \rangle)_{j+1}-1} \alpha(\tilde{\mathbf{w}}, n-j-1, 0, A, \overline{\mathbf{C}} - A \cdot (P((\langle \tilde{\mathbf{w}} \rangle)_{[1,j]} : b)))\end{aligned}$$

Theorem 22. *The rank of a necklace $\tilde{\mathbf{w}}$ among the set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ can be found in $O(q \cdot n^2 \cdot C)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables in the system of linear equations.*

Proof. From Lemma 32, the rank of $\tilde{\mathbf{w}}$ can be computed in at most $O(n^2)$ time if the size of $S(\tilde{\mathbf{w}}, d, A, \frac{d \cdot \overline{\mathbf{C}}}{n})$ has been precomputed for every factor d of n. In order to compute the size of $S(\tilde{\mathbf{w}}, d, A, \frac{d \cdot \overline{\mathbf{C}}}{n})$, it is sufficient to compute the size of $A(i, j, \tilde{\mathbf{w}}, d, A, \overline{\mathbf{C}})$ for every $i, j \in [d]$. Assuming the size of $A(i, j, \tilde{\mathbf{w}}, d, A, \overline{\mathbf{C}})$ has been precomputed for every $i, j \in [d]$, the size of $S(\tilde{\mathbf{w}}, d, A, \frac{d \cdot \overline{\mathbf{C}}}{n})$ can be computed in $O(n^2)$ time. The size of $A(i, j, \tilde{\mathbf{w}}, d, A, \overline{\mathbf{C}})$ may be computed in $O(n^2 \cdot C)$ time if $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{P}})$ and $\beta(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{P}})$ has been computed for every $i, j \in [n]$, $\overline{\mathbf{P}} \in [\overline{\mathbf{C}}]$.

This leaves the problem of computing $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$ and $\beta(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$. The value of $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$ can be computed in $O(q)$ time if the value of $\alpha(\tilde{\mathbf{w}}, i, j+1, A, \overline{\mathbf{C}})$ and $\alpha(\tilde{\mathbf{w}}, i-j-1, 0, A, \overline{\mathbf{C}}')$ has been precomputed for every $\overline{\mathbf{C}}' \in [\overline{\mathbf{P}}]$. Further, the value of $\alpha(\tilde{\mathbf{w}}, i, i, A, \overline{\mathbf{C}}')$ may be computed in $O(m)$ time for any set of arguments. Therefore by starting with $i = j = 0$, the value of $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$ can be computed in a dynamic manner for increasing values of i, j and $\overline{\mathbf{C}}$. As there are $O(n^2 \cdot C)$ possible values of $i, j \in [n]$ and $\overline{\mathbf{C}}' \in [\overline{\mathbf{C}}]$, the total complexity of precomputing each value of $\alpha(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$ is $O(q \cdot n^2 \cdot C)$. Following Lemma 31, the value of $\beta(\tilde{\mathbf{w}}, i, j, A, \overline{\mathbf{C}})$ can be computed in $O(q \cdot n \cdot C)$ time.

Therefore the overall complexity of computing the rank of $\tilde{\mathbf{w}}$ in the set of necklaces with Parikh vectors solving can be $O(q \cdot n^2 \cdot C)$ time. □

7.2.3 Unranking $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$

As with the set $\mathbf{P}(n, A, \overline{\mathbf{C}})$, we consider the problem of unranking the set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$. Following the same steps outlined in Section 7.1.2, the first problem is to count the number of necklaces in $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ sharing a given prefix $\bar{a}$.

Lemma 33. *The number of necklaces in $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ sharing a given prefix $\bar{a} \in \Sigma(A)$ can be computed in $O(q \cdot n^2 \cdot C)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables in the system of linear equations.*

Proof. Observe that the number of necklaces sharing a given prefix is given by $RN(\bar{a} : q^{n-|\bar{a}|}) + 1 - RN(\bar{a}^{|\bar{\mathbf{a}}|/n})$. As the rank of a necklace can be computed in $O(q \cdot n^2 \cdot C)$ time, the number of necklaces sharing a common prefix can be computed in $O(q \cdot n^2 \cdot C)$ time. □

Lemma 33 provides the basis for both the unranking procedure and the k-centre algorithm. The high level idea for the unranking procedure is to use a binary search on $\Sigma(A)$ to determine the j^{th} symbol of the i^{th} necklace in increasing order of j. The following theorem outlines the unranking process.

Theorem 23. *The i^{th} member of $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ can be computed in $O(q \cdot \log(q) \cdot n^3 \cdot C)$ time, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables in the system of linear equations.*

Proof. Following the same process laid out in Theorem 20, the i^{th} necklace $\bar{w}$ can be found by progressively building the prefix of $\bar{w}$ from length 1 to n. At each step, a binary search over the set of symbols in $\Sigma(A)$ requiring $2\log(q)$ necklaces to be ranked. Therefore to compute $\bar{w}$, a total of $2n \cdot \log(q)$ necklace must be ranked. This gives the total complexity of $O(q \cdot \log(q) \cdot n^3 \cdot C)$. □

Corollary 7. *The set $\mathcal{N}_q^n(A, \overline{\mathbf{C}})$ can be generated in lexicographic order in no more than $O(q \cdot \log(q) \cdot n^3 \cdot C)$ time per necklace, where $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$ and q is the number of variables in the system of linear equations.*

Proof. This bound may be achieved by simply using the unranking algorithm for i from 1 to $|\mathcal{N}_q^n(A, \overline{\mathbf{C}})|$. □

7.3 Necklaces with Forbidden Subwords

The second type of constrained necklace that we consider in this section is the set of necklace with forbidden subwords. In many ways this is a more challenging problem compared to either the general or the Diophantine equation cases due to the cyclic nature of necklaces. Unlike with an non-cyclic word, when counting the number of cyclic words without a given forbidden word, it must be ensured that it does not occur for *any* shift, as opposed to just one. This is further complicated when considering multiple forbidden words, where it must be checked that no forbidden word occurs for any translation.

In order to sample from the set of necklaces with forbidden subwords, some notation must be defined. Let $\mathcal{F}$ be the set of words that are forbidden from appearing as subwords within the set of necklaces. We denote by $\mathcal{N}_q^n(\mathcal{F})$ the set of all necklaces over an alphabet of size q of length n without any word from $\mathcal{F}$ appearing as a subword. Ruskey and Sawada [90] computed the size of $\mathcal{N}_q^n(\mathcal{F})$ as

$$|\mathcal{N}_q^n(\mathcal{F})| = \frac{1}{n}\sum_{d|n}\phi(d)|\mathbf{F}_q^{n/d}(\mathcal{F})|, \tag{7.1}$$

where $\mathbf{F}_q^n(\mathcal{F})$ is the set of words of length n over the alphabet of size q containing no subword in $\mathcal{F}$. The number of Lyndon words of length n with not containing any subword in $\mathcal{F}$, denoted $|\mathcal{L}_q^n(\mathcal{F})|$, is given relative to the number of necklaces, using the equation:

$$|\mathcal{L}_q^n(\mathcal{F})| = \sum_{d|n}\mu(d)\cdot|\mathcal{N}_q^{n/d}(\mathcal{F})|. \tag{7.2}$$

Before introducing the functions for ranking and unranking $\mathcal{N}_q^{n/d}(\mathcal{F})$, some theoretical results must be established. Let $\mathbf{T}(\bar{w},\mathcal{F})$ be the set of words belonging to a necklace class in $\mathcal{N}_q^n(\mathcal{F})$ smaller than $\bar{w}$, i.e. $\mathbf{T}(\bar{w},\mathcal{F}) = \{\bar{v}\in\Sigma^{|\bar{w}|}|\bar{v}\in\tilde{\mathbf{v}},\tilde{\mathbf{v}}\in\mathcal{N}_q^{|\bar{w}|}(\mathcal{F}),\langle\tilde{\mathbf{v}}\rangle\leq\bar{w}\}$. Additionally in order to rank words amongst $\mathcal{N}_q^n(\mathcal{F})$ the subset $\mathbf{T}'(\bar{w},\mathcal{F})\subseteq\mathbf{T}(\bar{w},\mathcal{F})$ of aperiodic words is needed. Formally, $\mathbf{T}'(w,F) = \{\bar{v}\in\Sigma^{|\bar{w}|}|\bar{v}\in\tilde{\mathbf{v}},\tilde{\mathbf{v}}\in\mathcal{L}_q^{|\bar{w}|}(\mathcal{F}),\langle\tilde{\mathbf{v}}\rangle\leq\bar{w}\}$.

Lemma 34. *The size of* $\mathbf{T}(\bar{w},\mathcal{F})$ *is given by* $\sum_{d|n}|\mathbf{T}'(\bar{w}_{[1,d]},\mathcal{F})$.

Proof. Observe that every word in the set $\mathbf{T}(\bar{w},\mathcal{F})$ is either aperiodic, in which case it belongs also to the set $\mathbf{T}'(\bar{w},\mathcal{F})$, or it is periodic. If it is periodic, the period must be some value that is a factor of n. Given some word with a period d, if it is smaller than $\bar{w}$, then it occurs in the set $\mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})$. By definition, any word greater than $\bar{w}$ does not appear in any set $\mathbf{T}'(\bar{w}_{[1,d]}, F)$ for any factor d of n. As each set $\mathbf{T}'(\tilde{\mathbf{w}}_{[1,d]}, \mathcal{F})$ consists only of aperiodic words, there can be no word that occurs in both $\mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})$ and $\mathbf{T}'(\bar{w}_{[1,e]}, \mathcal{F})$ for $d \neq e$. Therefore the size of $\mathbf{T}(\bar{w}, \mathcal{F})$ can be computed as $|\mathbf{T}(\bar{w}, \mathcal{F}) = \sum\limits_{d|n} |\mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})$. □

By application of the Möbius inversion formula to $|\mathbf{T}(\bar{w}, \mathcal{F})| = \sum\limits_{d|n} |\mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})|$, an equation for the size of $\mathbf{T}'(\bar{w}, \mathcal{F})$ can be derived as:

$$|\mathbf{T}'(\bar{w}, \mathcal{F})| = \sum_{d|n} \mu\left(\frac{n}{d}\right) |\mathbf{T}(\bar{w}_{[1,d]}, \mathcal{F})|. \tag{7.3}$$

These equations form the basis for ranking $\bar{w}$. Starting with the rank of $\bar{w}$ within the set $\mathcal{L}_q^n(\mathcal{F})$. The rank within the set of Lyndon words is used to compute the rank within the set of necklaces following the same logic outlined in Lemma 34.

Lemma 35. *The number of Lyndon words without any forbidden subword in $\mathcal{F}$ that are smaller than some word $\bar{w}$ is given by $RL(\bar{w}, \mathcal{F}) = \frac{1}{|\bar{w}|} \cdot |\mathbf{T}'(\bar{w}, \mathcal{F})|$.*

Proof. As every Lyndon word is aperiodic, each Lyndon word of length $|\bar{w}|$ has $|\bar{w}|$ unique translations. Hence, for any word $\bar{w}$ each Lyndon word smaller than $\bar{w}$ appears $|\bar{w}|$ times within $\mathbf{T}'(\bar{w}, \mathcal{F})$. □

Lemma 36. *The number of necklaces smaller than w without any forbidden subword in F is equal to $RN(\bar{w}, \mathcal{F}) = \sum\limits_{d||\bar{w}|} \frac{1}{|\bar{w}|} \cdot \mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})$.*

Proof. It follows from Lemma 34 that all necklaces smaller than $\bar{w}$ are either aperiodic, or periodic with a period that is a factor of the length of the necklace. From Lemma 35, the necklaces that are smaller than $\bar{w}$ and are aperiodic is $\frac{1}{|\bar{w}|} \cdot \mathbf{T}'(\bar{w}, \mathcal{F})$. Similarly, the necklaces with a period of some factor d of $|\bar{w}|$ are $\frac{1}{|\bar{w}|} \cdot |\mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})|$. □

The problem now becomes computing the size of $\mathbf{T}(\bar{w}, \mathcal{F})$. To do this, the set is partitioned into the sets $\mathbf{A}(t, j, \mathcal{F})$ such that for every word $\bar{v} \in \mathbf{A}(t, j, \mathcal{F})$ the following hold:

- t is the smallest translation such that $\langle \bar{v} \rangle_t$ is smaller than $\bar{w}$, i.e. $\langle \bar{v} \rangle_t < \bar{w}$.
- Under the translation by t, j is the length of the longest prefix of $\langle \bar{v} \rangle_t$ that is also a prefix of $\bar{w}$, i.e. $(\langle \bar{v} \rangle_t)_{[1,j]} = \bar{w}_{[1,j]}$.

The size of $\mathbf{A}(t, j, \mathcal{F})$ is computed by considering two possible cases based on the value of $t + j$.

Case 1, $t + j < n$: In this case, every word in $\mathbf{A}(t, j, \mathcal{F})$ is of the form $\bar{\beta} : \bar{w}_{[1,j]} : x : \bar{\rho}$ where:

- $\bar{\beta}$ is some word of length t with no forbidden subword such that every suffix is greater than $\bar{w}$;
- $\bar{w}_{[1,j]}$ is the prefix of $\bar{w}$ of length j;
- x is a symbol smaller than $\bar{w}_{j+1}$;
- $\bar{\rho}$ is a word with no restrictions other than having no forbidden substrings.

To compute the number of possible words satisfying $\bar{\rho}$ and $\bar{\beta}$, we define the function $B(\bar{w}, l, t, j, \bar{p}, \bar{s})$. A full definition of this function is given in Section 7.3.1. At a high level, this function works similarly to the class $\mathbf{PO}(\bar{v}, i, j, \mathbf{s})$ from Chapter 6. The function works in two phases. The first phase to compute the number of possible values of $\bar{\beta}$, followed by computing the values of $\bar{\rho}$. The word $\bar{p}$ corresponds to the longest suffix of $\bar{w}_{[1,j]} : x$ that is a prefix of some forbidden word in $\mathcal{F}$. Similarly $\bar{s}$ corresponds to either $\bar{w}_{[1,j]} : x$ if $\bar{w}_{[1,j]} : x$ is a subword of some word in $\mathcal{F}$, or the longest prefix of $\bar{w}_{[1,j]} : x$ that is the suffix of a word in $\mathcal{F}$ otherwise. The values $l, t \in \mathbb{N}$ are used to denote the lengths of $\bar{\rho}$, and $\bar{\beta}$ respectively. Finally j is a helper value for tracking the longest prefix of $\bar{w}$ that is a prefix of the words counted by $B(\bar{w}, l, t, j, \bar{p}, \bar{s})$. Using $B(\bar{w}, l, t, j, \bar{p}, \bar{s})$, the size of $\mathbf{A}(\bar{w}, t, j, \mathcal{F})$ can be computed as:

$$|\mathbf{A}(\bar{w}, t, j, \mathcal{F})| = \sum_{x=1}^{\bar{w}_{j+1}-1} \begin{cases} B(\bar{w}, l, t, j, P(\bar{w}_{[1,j]} : x), S(\bar{w}_{[1,j]} : x)) & \bar{f} \not\sqsubseteq \bar{w}_{[1,j]} : x, \forall \bar{f} \in \mathcal{F} \\ 0 & \textit{Otherwise.} \end{cases}$$

Where $P(\bar{w}_{[1,j]} : x)$ returns the longest suffix of $\bar{w}_{[1,j]} : x$ that is the prefix of some forbidden subword in $\mathcal{F}$, and $S(\bar{w}_{[1,j]} : x)$ returns either $\bar{w}_{[1,j]} : x$, if $\bar{w}_{[1,j]} : x \sqsubseteq \bar{f}$ for some $\bar{f} \in \mathcal{F}$, or the longest prefix of $\bar{w}_{[1,j]} : x$ that is the suffix of some forbidden subword in $\mathcal{F}$ otherwise.

Case 2, $j+t \geq n$: In this case every word in $\mathbf{A}(t,j,\mathcal{F})$ has the form $\bar{w}_{[s,j]} : x : \bar{\beta} : \bar{w}_{[1,s-1]}$. Let δ be the length of the longest prefix of $\bar{w}$ that is a suffix $\bar{w}_{[s,j]}$. If $x < \bar{w}_{\delta+1}$, then the shift by $s-\delta$ would be smaller than $\bar{w}$. Therefore x must be greater than or equal to $\bar{w}_{\delta+1}$. If $x = \bar{w}_{\delta+1}$ then every suffix of $\bar{w}_{[1,\delta+1]} : \bar{\beta}$ must be greater than the prefix of $\bar{w}$ of the same length. The number of such words is computed using $B(\bar{w}, n-j-1, \delta+1, 0, P(\bar{w}_{[1,\delta+1]}, S(\bar{w}_{[1,\delta+1]}))$. Alternatively, if $x > \bar{w}_{\delta+1}$ and $\bar{w}_{[s,j]} : x$ contains no forbiden subword, then the number of possible values for $\bar{\beta}$ is $B(\bar{w}, n-j-1, 0, 0, P(\bar{w}_{[s,j]} : x), S(\bar{w}_{[s,j]} : x))$. From this, the size of $\mathbf{A}(\bar{w},t,j,\mathcal{F})$ can be computed as:

$$|\mathbf{A}(\bar{w},t,j,\mathcal{F})| = B(\bar{w}, n-j-1, \delta+1, 0, P(\bar{w}_{[1,\delta+1]}, S(\bar{w}_{[1,\delta+1]}))+ \sum_{x=\bar{w}_{\delta+1}+1}^{\bar{w}_{j+1}-1} \begin{cases} B(\bar{w}, n-j-1, 0, 0, P(\bar{w}_{[s,j]} : x), S(\bar{w}_{[s,j]} : x)) & \bar{w}_{[s,j]} : x \notin \mathcal{F} \\ 0 & Otherwise. \end{cases}$$

Section 7.3.1 provides the full details on how to compute $B(\bar{w}, l, t, j, \bar{p}, \bar{s})$. Theorem 24 treats $B(\bar{w}, l, t, j, \bar{p}, \bar{s})$ as a black box that can be computed in $O(n^8 \cdot |\mathcal{F}|^2)$ time using Lemma 41.

Theorem 24. *The rank of a word amongst all necklaces without any forbidden subword may be computed in $O(q \cdot n^8 \cdot |\mathcal{F}|^3)$.*

Proof. It follows from Lemma 41 that the value of $B(\bar{w}, l, j, t, \bar{p}, \bar{s})$ can be computed for every value of $l, j, t \in [n]$, $\bar{p} \in \{\bar{f}_{[1,i]} | \bar{f} \in \mathcal{F}, i \in [|\bar{f}|]\}$ and $\bar{s} \in \{\bar{u} \sqsubseteq \bar{f} | \bar{f} \in \mathcal{F}, \bar{u}$ is a suffix of $\bar{f}$ or $|\bar{u}| = n-(t+l)\}$ in $O(n^8 \cdot |\mathcal{F}|^2)$ time. Assuming that these values have been precomputed, the size of $\mathbf{A}(\bar{w},j,t,\mathcal{F})$ can be computed in $O(q \cdot n \cdot |\mathcal{F}|)$ time following the two cases above. As the process of computing the size of $\mathbf{A}(\bar{w},j,t,\mathcal{F})$ is dominated by the precomputation of $B(\bar{w}, l, j, t, \bar{p}, \bar{s})$, the total time to compute $|\mathbf{A}(\bar{w},j,t,\mathcal{F})|$ is $O(n^8 \cdot |\mathcal{F}|^2)$.

Following Lemma 36, the number of Necklaces may be computed by summing the size of $\mathbf{T}'(\bar{w}_{[1,d]}, \mathcal{F})$ for every factor d of n. Note that there are at most $\log_2(n)$ factors of n. The size of $\mathbf{T}'(\bar{w}, \mathcal{F})$ can be computed using Lemma Equation 7.3. In the worst case there are $O(\log n)$ sets of $\mathbf{T}(\bar{w}_{[1,d]}, \mathcal{F})$, each of which taking at most $O(n^8 \cdot |\mathcal{F}|^2)$ time to compute. Putting this together, the total time complexity is $O(q \cdot n^8 \cdot |\mathcal{F}|^3)$. □

7.3.1 Computing the Number of Prefixes Greater than $\bar{w}$ with Forbidden Subwords

This section shows how to compute $B(\bar{w}, l, j, t, \bar{p}, \bar{s})$. For notation let $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ be the set of words counted by $B(\bar{w}, l, j, t, \bar{p}, \bar{s})$. Following the informal definition given above, the set $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ is defined as the set of words such that for every $\bar{v} \in \mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$:

- $\bar{v}$ has a length of $l + t$
- The cyclic word $\bar{w}_{[1,n-l-t-1]} : \bar{u} : \bar{v}$ contains no forbidden subword from the set $\mathcal{F}$ for some word $\bar{u} \in \Sigma^*$ where:
  - The suffix of $\bar{u}$ of length j equals $\bar{w}_{[1,j]}$.
  - $\bar{p}$ is the longest suffix of $\bar{u}$ that is the prefix of some forbidden subword.
  - $\bar{s}$ is either the longest prefix of $\bar{w}_{[1,n-l-t-1]} : \bar{u}$ that is the suffix of a forbidden word, or equals $\bar{w}_{[1,n-l-t-1]} : \bar{u}$ if $\bar{w}_{[1,n-l-t-1]} : \bar{u}$ is the subword of some forbidden word.
- Every suffix of $\bar{v}_{[l+1,l+t]}$ is greater than the prefix of $\bar{w}$ of the same length, i.e. for every $i \in [t], \bar{v}_{[l+i,l+t]} \geq \bar{v}_{[1,t-i+1]}$.
- If $l = 0$, then every suffix of $\bar{u} : \bar{v}$ is greater than the prefix of $\bar{w}$ of the same length.

At a high level the idea the key observation is that for every word $\bar{v} \in \mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$, the suffix $\bar{v}_{[2,l+t]}$ must belong to some set $\mathbf{B}(\bar{w}, l', j', t', \bar{p}', \bar{s}')$. The following Lemmas serves to formalise this observation in a useful way for computing the size of $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$.

Lemma 37. *Let $\bar{v} \in \mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ be a word such that $\bar{v}_{[2,t+l]} \in \mathbf{B}(\bar{w}, l', j', t', \bar{p}', \bar{s}')$, then the values of $l', j', t', \bar{p}'$, and $\bar{s}'$ can be determined from the values of $\bar{v}_1$ and the values of $l, j, t, \overline{\mathbf{P}}, \mathbf{S}$ in $O(n^2 \cdot |\mathcal{F}|)$ time.*

Proof. The value of $\bar{p}'$ is simply the longest suffix of $\bar{p} : \bar{v}_1$ that is a prefix of some forbidden word $\bar{f} \in \mathcal{F}$. Through the use of pattern matching algorithms such as the Knuth-Morris-Pratt algorithm applied to each forbidden subword, this can be computed in $O(n \cdot |\mathcal{F}|)$ time. Similarly, the value of $\bar{s}'$ is either:

- $\bar{s}$, if $|\bar{s}| < n - (t + l)$.
- $\bar{s} : x$ if $\bar{s} : x$ is a subword of some forbidden subword.

- the longest prefix of $\bar{s} : x$ that is a suffix of some forbidden subword otherwise.

As there are n possible subwords for each of the forbidden subwords, using a pattern matching algorithm the total complexity of computing $\bar{s}'$ is $O(n^2 \cdot |\mathcal{F}|)$. The value of l' is either $l-1$, if $l > 0$ or 0 if $l = 0$. Similarly, the value of t' is either t, if $l > 0$ or $t-1$ if $l = 0$. Finally, the value of j' is either 0, if $l > 0$ or $l = 0$ and $\bar{v}_1 \neq \bar{w}_{j+1}$, otherwise $j' = j+1$. As the complexity is dominated by the process of computing $\bar{p}'$ and $\bar{s}$, the overall complexity is $O(n^2 \cdot |\mathcal{F}|)$. □

Lemma 38. *Let $\bar{v} \in \mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ be a word such that $\bar{v}_{[2,t+l]} \in \mathbf{B}(\bar{w}, l', j', t', \bar{p}', \bar{s}')$, then for any word $\bar{v}' \in \mathbf{B}(\bar{w}, l', j', t', \bar{p}', \bar{s}')$, $\bar{v}_1 : \bar{v}' \in \mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$.*

Proof. Following the arguments of Lemma 37, the values of $l', j', t', \bar{p}'$, and $\bar{s}'$ are determined solely by the values of $\bar{w}, l, j, t, \bar{p}, \bar{s}$ and $\bar{v}_1$. Note that any word $\bar{v}' \in \mathbf{B}(\bar{w}, l', j', t', \bar{p}', \bar{s}')$, must also satisfy the conditions that (1) $\bar{p}' : \bar{v}' : \bar{s}'$ contains no word in $\mathcal{F}$ as a subword for and (2) if $|\bar{s}'| = n - (l+t)$ there is no subword of the cyclic word $\bar{w} : \bar{s}'$ that is a forbidden subword. Therefore, $\bar{v}_1 : \bar{v}'$ must also belong to $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$. □

Lemmas 37 and 38 provide the foundational tools to compute $B(\bar{w}, l, j, t, \bar{p}, \bar{s})$. At a high level, the idea is to use a dynamic programming approach similar to Lemma 19, starting with the case where $l = t = 0$ then working with increasing values of l and t. Following Lemma 38, the size of $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ can be computed by looking at every possible value of the first symbol, and the size of the corresponding sets $\mathbf{B}(\bar{w}, l', j', t', \bar{p}', \bar{s}')$, derived using Lemma 37. Lemmas 39 and 40 provide the equations for computing the size of $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ in the case that $l > 0$ and $l = 0$ respectively.

Lemma 39. *For any $l > 0$, the size of $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ is given by:*

$$\sum_{x \in \Sigma} \begin{cases} B(\bar{w}, l(x), j(x), t(x)\bar{p}(x), \bar{s}(x)) & \bar{p} : x \notin \mathcal{F} \\ 0 & \bar{p} : x \in \mathcal{F} \end{cases}$$

Where $l(x), j(x), t(x)\bar{p}(x)$, and $\bar{s}(x)$ are computed as in Lemma 37.

Proof. Following Lemmas 37 and 38, every word $\bar{v} \in \mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ sharing the same first symbol x must have a suffix in $B(\bar{w}, l(x), j(x), t(x)\bar{p}(x), \bar{s}(x))$. Further, there can be no word in $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ starting with any symbol $x \in \Sigma$ where $\bar{p} : x \in \mathcal{F}$. Therefore, the size of $\mathbf{B}(\bar{w}, l, j, t, \bar{p}, \bar{s})$ equals $\sum_{x \in \Sigma} \begin{cases} B(\bar{w}, l(x), j(x), t(x)\bar{p}(x), \bar{s}(x)) & \bar{p} : x \notin \mathcal{F} \\ 0 & \bar{p} : x \in \mathcal{F} \end{cases}$ □

Lemma 40. *The size of* $\mathbf{B}(\bar{w},0,j,t,\bar{p},\bar{s})$ *is given by:*

$$\sum_{x=\bar{w}_{j+1}}^{q} \begin{cases} B(\bar{w},0,j(x),t(x)\bar{p}(x),\bar{s}(x)) & \bar{p}:x \notin \mathcal{F} \\ 0 & \bar{p}:x \in \mathcal{F} \end{cases}$$

Where $\bar{w}, j(x), t(x)\bar{p}(x)$, *and* $\bar{s}(x)$ *are computed as in Lemma 37.*

Proof. Note that as $l = 0$, by the definition of the set $\mathbf{B}(\bar{w},0,j,t,\bar{p},\bar{s})$, $\bar{w}_{[1,j]}\bar{v} > \bar{w}$ for any $\bar{v} \in \mathbf{B}(\bar{w},0,j,t,\bar{p},\bar{s})$. Therefore, the first symbol of every word in $\mathbf{B}(\bar{w},0,j,t,\bar{p},\bar{s})$ must be no smaller than $\bar{w}_{j+1}$. Other than this restriction, the same arguments from Lemma 39 can be applied to this setting, giving the equation:

$$|\mathbf{B}(\bar{w},0,j,t,\bar{p},\bar{s})| = \sum_{x=\bar{w}_{j+1}}^{q} \begin{cases} B(\bar{w},0,j(x),t(x)\bar{p}(x),\bar{s}(x)) & \bar{p}:x \notin \mathcal{F} \\ 0 & \bar{p}:x \in \mathcal{F} \end{cases}$$

□

Lemmas 39 and 40 provide the equations needed to compute the size of $\mathbf{B}(\bar{w},l,j,t,\bar{p},\bar{s})$. Lemma 41 compliments these Lemmas by providing an upper bound on the time complexity to compute the size of $\mathbf{B}(\bar{w},l,j,t,\bar{p},\bar{s})$.

Lemma 41. *The size of* $\mathbf{B}(\bar{w},l,j,t,\bar{p},\bar{s})$ *can be computed in* $O(q \cdot n^8 \cdot |\mathcal{F}|^3)$ *time.*

Proof. Following Lemmas 39 and 40, the size of $\mathbf{B}(\bar{w},0,j,t,\bar{p},\bar{s})$ can be computed in $O(q{\cdot}n^2{\cdot}|\mathcal{F})$ time if the size of $\mathbf{B}(\bar{w},l(x),j(x),t(x),\bar{p}(x),\bar{s}(x))$ is known for every $x \in \Sigma$. Note further than $\mathbf{B}(\bar{w},0,j,0,\bar{p},\bar{s})$ can be computed in $O(n{\cdot}|\mathcal{F}|^2)$ time by noting that it equals 0, if either $j > 0$ or $\bar{p}:\bar{s}$ contains a forbidden subword, or 1 if none of these conditions hold. Therefore, a dynamic programming approach can be employed starting with $\mathbf{B}(\bar{w},0,j,0,\bar{p},\bar{s})$ for every $j \in [n], \bar{p} \in \{\bar{f}_{[1,i]} | \bar{f} \in \mathcal{F}, i \in [|\bar{f}|]\}$ and $\bar{s} \in \{\bar{u} \sqsubseteq \bar{f} | \bar{f} \in \mathcal{F}, \bar{u}$ is a suffix of $\bar{f}$ or $|\bar{u}| = n - (t+l)\}$.

From this base case, the size of $\mathbf{B}(\bar{w},l,j,t,\bar{p},\bar{s})$ is then computed in increasing value of l and t, such that $\mathbf{B}(\bar{w},l,j,t,\bar{p},\bar{s})$ is computed before either $\mathbf{B}(\bar{w},l+1,j',t,\bar{p}',\bar{s}')$ or $\mathbf{B}(\bar{w},l,j',t+1,\bar{p}',\bar{s}')$ for any $j' \in [n], \bar{p}' \in \{\bar{f}_{[1,i]} | \bar{f} \in \mathcal{F}, i \in [|\bar{f}|]\}$ and $\bar{s}' \in \{\bar{u} \sqsubseteq \bar{f} | \bar{f} \in \mathcal{F}, \bar{u}$ is a suffix of $\bar{f}$ or $|\bar{u}| = n-(t+l+1)\}$. As there are $O(n)$ possible value of l, j and t, $O(n{\cdot}|\mathcal{F})$ possible values of $\bar{p}$ and $O(n^2{\cdot}|\mathcal{F}|)$ possible values of $\bar{s}$, this dynamic programming approach requires the size of $O(n^6 \cdot |\mathcal{F}|^2)$ sets to be computed. As each set takes at most $O(q \cdot n^2 \cdot |\mathcal{F})$ time, the total complexity of computing the size of $\mathbf{B}(\bar{w},l,j,t,\bar{p},\bar{s})$ is $O(q \cdot n^8 \cdot |\mathcal{F}|^3)$. □

7.3.2 Unranking Necklaces with Forbidden Subwords

Using the same approach as in Theorems 15, 20 and 23 the set of necklaces with forbidden subwords may be unranked by first determine the number necklaces sharing a given prefix. To this end, Lemma 42 is introduced to serve an analogous function to Lemmas 30 and 33.

Lemma 42. *The number of necklaces with Forbidden with no subword in the set $\mathcal{F}$ sharing a prefix $\bar{a}$ can be computed in $O(q \cdot n^8 \cdot |\mathcal{F}|^3)$ time.*

Proof. Note that the ranking algorithm outlined in Theorem 24 does not require that the input word is free of forbidden subwords, only that the input is the canonical form of some necklace. Hence the number of words sharing a given prefix can be determined by taking the difference between the lexicographically smallest and largest necklaces sharing a given prefix. The smallest necklace $\bar{v}$ can be found in $O(n)$ time using Theorem 28. The largest necklace is simply $\bar{a} : q^{n-|\bar{a}|}$. Therefore the number of necklaces sharing a given prefix is given by $Rank(\bar{a} : q^{n-|\bar{a}|}, \mathcal{F}) - Rank(\bar{v}, \mathcal{F})$. From Theorem 24, the value of $Rank(\bar{a} : q^{n-|\bar{a}|}, \mathcal{F})$ can be computed in $O(q \cdot n^8 \cdot |\mathcal{F}|^3)$ time. □

Using Lemma 42 as a basis, the i^{th} necklace with the set of forbidden subwords $\mathcal{F}$ can be unranked using the same process as in Theorems 20 and 23.

Theorem 25. *The i^{th} necklace in $\mathbf{N}_q^n(\mathcal{F})$ can be found in $O(\log(q) \cdot \log^2(n) \cdot n^9 \cdot |\mathcal{F}|^2)$ time.*

Proof. Let $\bar{w}$ be the canonical representation of the i^{th} necklace in the set $\mathbf{N}_q^n(\mathcal{F})$. The first symbol $\bar{w}_1$ can be computed by by performing a binary search over Σ to find the symbol x such that $Rank(x^n) \leq i \leq Rank(x : q^{n-1})$. Using a binary search approach allows this to be done in $\log(q)$ comparisons, giving a total time complexity of $O(\log(q) \cdot q \cdot n^8 \cdot |\mathcal{F}|^3)$. Once the first symbol has been determined the second symbol can be found using the same approach.

In general, once the first $j - 1$ symbols have been determined, the j^{th} symbol can be determined by finding the symbol $x \in \Sigma$ as follows. Let $\bar{v}$ be the canonical form of the lexicographically smallest necklace such that $\bar{v}_{[1,j]} = \bar{w}_{[1,j-1]} : x$. The j^{th} symbol of $\bar{w}$ is x if and only if $Rank(\bar{v}, \mathcal{F}) \leq i \leq Rank(\bar{w}_{[1,j-1]} : x : q^{n-j})$.

As each symbol can be determined in $\log(q)$ comparisons, and a total of n symbols need to be determined, a total of $n \cdot \log(q)$ words need to be ranked. Therefore, the total complexity of determining the i^{th} necklace in $\mathbf{N}_q^n(\mathcal{F})$ is $O(\log(q) \cdot q \cdot \log^2(n) \cdot n^9 \cdot |\mathcal{F}|^3)$ time. □

Chapter 8

Polynomial Time Algorithms for Multidimensional Necklaces

The final object we look at are the class of multidimensional necklaces. As this work has been the first to study this object, there are many fundamental to results to generalise from the one-dimensional case to the multidimensional setting. In this chapter we provide generalisations for several of these fundamental operations on multidimensional necklaces. Section 8.1 provides closed form equations for counting the number of necklaces, Lyndon words, and atranslational necklaces. These results are extended in Subsection 8.1.1 to the fixed content setting. Section 8.2 provides our algorithm for ranking multidimensional necklaces, with Subsection 8.2.2 extending our ranking result to the fixed content case. Section 8.3 provides our techniques for generating and unranking multidimensional necklaces, the unranking result being extended to the fixed content setting in the same section.

8.1 Counting of Multidimensional Necklaces

This section provides a comprehensive overview of the equations for counting the number of necklaces, Lyndon words, and atranslational words. For both necklaces and Lyndon words, explicit counting is done by application of the Pólya enumeration theorem to the group operations defined in Chapter 2. The equations below are classical formulas for counting the number of one-dimensional necklaces and one-dimensional Lyndon words respectively. A classical proof for the Necklace Equation is provided by Graham et. al. [42], while Perrin [80] provides a proof of the Lyndon word Equation.

$$|\mathcal{N}_q^n| = \frac{1}{n}\sum_{d|n} \phi\left(\frac{n}{d}\right) q^d. \tag{8.1}$$

$$L_q^n = \frac{1}{n}\sum_{d|n} \mu\left(\frac{n}{d}\right) q^d. \tag{8.2}$$

Where ϕ is Euler's totient function and μ is the Möbius function. Formally, $\phi(n)$ gives the number of natural numbers smaller than n which are co-prime to n, and $\mu(n)$ returns -1, 0, or 1 depending on the prime factorisation of n. These equations form the starting point for counting multidimensional necklaces. Recall from the preliminaries that multidimensional necklaces of dimensions $\overline{\mathbf{n}}$ are equivalence classes of words in $\Sigma^{\overline{\mathbf{n}}}$ under the group $Z_{\overline{\mathbf{n}}} = Z_{n_1} \times Z_{n_2} \times \ldots \times Z_{n_d}$ where $\times$ denotes the direct product and Z_x the cyclic group of order x. A straightforward way to compute the number of necklaces of dimensions $\overline{\mathbf{n}}$ is by using the *Pólya enumeration formula*, giving:

$$|\mathcal{N}_q^{\overline{\mathbf{n}}}| = \frac{1}{N}\sum_{g \in Z_{\overline{\mathbf{n}}}} q^{c(g)}.$$

Where $g = (g_1, g_2, \ldots, g_d)$ is some group action in $Z_{\overline{\mathbf{n}}}$ and $c(g)$ returns the number of cycles from the group action g. Since $Z_{\overline{\mathbf{n}}}$ is formed by the direct product of the cyclic groups, for each group action g we have that $g = (g_1, g_2, \ldots, g_d)$, where $1 \leq i_j \leq n_j$. Therefore, the number of necklaces, $|\mathcal{N}_q^{\overline{\mathbf{n}}}|$, is rewritten as:

$$|\mathcal{N}_q^{\overline{\mathbf{n}}}| = \frac{1}{N}\sum_{g_1=1}^{n_1}\sum_{g_2=1}^{n_2}\cdots\sum_{g_d=1}^{n_d} q^{c((g_1,g_2,\ldots g_d))}$$

In order to determine the value of $c(g)$, consider the permutation induced by g. Given some position $\mathbf{j} = (j_1, \ldots, j_d)$, let $\mathbf{j}'$ be the position following $\mathbf{j}$ in the cycle induced by g, i.e. $\mathbf{j}' = \mathbf{j} \cdot g$. The coordinate of $\mathbf{j}'$ in the i^{th} dimension is equal to the coordinate in the i^{th} dimension of $\mathbf{j}$ shifted by g_i. Since this is a cyclic operation, this shift is done modulo the length of dimension i, n_i. This gives $j_i' = (j_i + g_i) \bmod n_i$.

Let g^t denote the group action made by applying t times operation g to the identity operation I, i.e. $I \cdot g \cdot g \ldots \cdot g$. The length of the cycle induced by some cyclic shift g is the smallest value $t > 0$ such that $\mathbf{j} \cdot g^t = \mathbf{j}$. In other words, the length of the cycle equals the number of times g must be applied to itself to become the identity operation. The length of this cycle is therefore the smallest t such that for every i, $(\mathbf{j}_i + t \cdot g_i) \bmod n_i \equiv \mathbf{j}_i$. To compute this, note that t must be divisible by the smallest value l_i for each dimension such that $(\mathbf{j}_i + l_i \cdot g_i) \bmod n_i \equiv \mathbf{j}_i$. As such, the smallest value t may have is the least common multiple of every l_i. For any smaller non-zero value, there is some dimension i for which $(\mathbf{j}_i + t \cdot g_i) \bmod n_i \not\equiv \mathbf{j}_i$. By the properties of modular addition, it is clear that every cycle has the same length. Therefore, the number of cycles of length t is $\frac{N}{t}$.

This is rewritten as follows. Observe that the only possible values for l_i are divisors of n_i. For each divisor f_i of n_i, there are $\phi(\frac{n_i}{f_i})$ values for which $f_i = l_i$. As this is independent in each dimension, this is used to derive the following equation for the number of necklaces:

$$|\mathcal{N}_q^{\overline{\mathbf{n}}}| = \frac{1}{N} \sum_{f_1 | n_1} \phi\left(f_1\right) \sum_{f_2 | n_2} \phi\left(f_2\right) \dots \sum_{f_d | n_d} \phi\left(f_d\right) q^{\frac{N}{\operatorname{lcm}\left(f_1, f_2, \dots, f_d\right)}}.$$

The necklace counting formula is used to compute the number of Lyndon words through repeated application of the Möbius inversion formula, giving:

$$L_q^{\overline{\mathbf{n}}} = \sum_{f_1 | n_1} \mu\left(\frac{n_1}{f_1}\right) \sum_{f_2 | n_2} \mu\left(\frac{n_2}{f_2}\right) \dots \sum_{f_d | n_d} \mu\left(\frac{n_d}{f_d}\right) |\mathcal{N}_q^{f_1, f_2 \dots f_d}|$$

Related to the concept of aperiodic necklaces are *atranslational necklaces.* Recall that a necklace $\tilde{\mathbf{w}}$ is atranslational if there exists no cyclic shift $g \in Z_{\overline{\mathbf{n}}}$ such that $g \neq (n_1, n_2, \dots, n_d)$ and $\langle \tilde{\mathbf{w}} \rangle_g = \langle \tilde{\mathbf{w}} \rangle$. Note that while every atranslational word is aperiodic, not every aperiodic word is atranslational. Lemma 43 formally characterises the aperiodic words that are not atranslational.

Lemma 43. *Every word $\bar{w} \in \mathbf{L}_q^{\overline{\mathbf{n}}}$ is either in $\mathbf{A}_q^{\overline{\mathbf{n}}}$ or of the form $\bar{u}^p : \langle \bar{u}^p \rangle_g : \dots : \langle \bar{u}^p \rangle_{g^{t-1}}$ where:*

- *g is a translation where $g_d = p$ and there exists no translation $r < g$ where $\langle \bar{u}^p \rangle_r = \bar{u}^p$.*
- *$\bar{u} \in \mathbf{L}_q^{(r/p, n_d - 1, \dots, n_1)}$. $t = \frac{n_d}{r}$ and is the smallest value greater than 0 such that $g^t = I$.*

Proof. For the sake of contradiction let $\bar{w} \in \mathbf{L}_q^{\overline{\mathbf{n}}}$ be an aperiodic word that is neither atranslational nor of the form $\bar{u}^p : \langle \bar{u}^p \rangle_g : \dots : \langle \bar{u}^p \rangle_{g^{t-1}}$ for $\bar{u} \in \mathbf{L}_q^{(r/p, n_d - 1, \dots, n_1)}$. As $\bar{w}$ is not atranslational, let g be the translation such that $\bar{w} = \langle \bar{w} \rangle_g$. Further let $\bar{u}$ be the prefix of $\bar{w}$ corresponding to the first g_d slices. If $\bar{u} \notin \mathbf{L}_q^{(r/p, n_d - 1, \dots, n_1)}$ then $\bar{u}$ has some period which that is also the period of $\bar{w}$. Otherwise note that $\langle \bar{w} \rangle = \bar{w}$. Therefore as $\langle \bar{w} \rangle_g = \bar{w}$, $\langle \bar{w}_{[g_d+1,2g_d]} \rangle_{(g_1,g_2,\dots,g_{d-1})} = \bar{u}$. More generally, $\langle \bar{w}_{[(l-1)\cdot g_d+1,l\cdot g_d]} \rangle_{(g_1,g_2,\dots,g_{d-1})^l} = \bar{u}$. This allows $\bar{w}$ to be written as $\bar{u} : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})} : \dots : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})^{t-1}}$. Note that if $t < \frac{n_d}{g_d}$ then $\langle \bar{w} \rangle_g = \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})} : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})^2} : \dots : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})^{t+1}}$, therefore $\langle \bar{w} \rangle_g = \bar{w}$ if and only if $\bar{u} = \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})}$. If $\bar{u} = \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})}$, then $\bar{w} = \bar{u} : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})} : \dots : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})^t} = \bar{u}^t$. Hence in this case $\bar{w}$ would be periodic. Therefore for $\bar{w}$ to be aperiodic and not a translational it must be of the form $\bar{u} : \langle \bar{u} \rangle_{(g_1,\dots,g_d)} : \dots : \langle \bar{u} \rangle_{(g_1,\dots,g_d)^{t-1}}$. In the other direction, if $\bar{w} = \bar{u} : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})} : \dots : \langle \bar{u} \rangle_{(g_1,g_2,\dots,g_{d-1})}$ and $\bar{u} \in \mathbf{A}_q^{(n_1,n_2,\dots,n_{d-1},r)}$ then $\bar{w} \in \mathbf{A}_q^{\mathbf{n}}$. Similarly if $\bar{u} \notin \mathbf{A}_q^{(n_1,n_2,\dots,n_{d-1},r)}$ it must be in $\mathbf{L}_q^{(n_1,n_2,\dots,n_{d-1},r)}$. □

Following the characterisation of translational Lyndon words given by Lemma 43, the next obvious question is how to count the number of atranslational words. To do so two further results are needed to reduce the complexity of the counting problem. Lemmas 44 and 45 provide an outline for how to reduce the number of atranslational words that need to be counted.

Lemma 44. *Let $\bar{a}, \bar{b} \in \mathbf{L}_q^{\mathbf{n}}$. Given any integer r and translation $g \in Z_{(n_1,n_2,\dots,n_{d-1})}$ such that $g^t = I$, if $\bar{a}^r : \langle \bar{a}^r \rangle_g : \dots : \langle \bar{a}^r \rangle_{g^{t-1}} \in \mathbf{L}_q^{n_1,n_2,\dots,n_{d-1},m}$ then either $\bar{b}^r : \langle \bar{b}^r \rangle_g : \dots : \langle \bar{b}^r \rangle_{g^{t-1}} \in \mathbf{L}_q^{n_1,n_2,\dots,n_{d-1},m}$ or $\bar{b} = \bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1}}$ and $g'_i + g_i \bmod n_i \equiv 0$ for all $i \in [d]$.*

Proof. For the sake of contradiction assume that $\bar{b}^r : \langle \bar{b}^r \rangle_g : \dots : \langle \bar{b}^r \rangle_{g^{t-1}} \in \mathbf{L}_q^{n_1,n_2,\dots,n_{d-1},m}$ while $\bar{a}^r : \langle \bar{a}^r \rangle_g : \dots : \langle \bar{a}^r \rangle_{g^{t-1}} \notin \mathbf{L}_q^{n_1,n_2,\dots,n_{d-1},m}$. As $\bar{b}^r : \langle \bar{b}^r \rangle_g : \dots : \langle \bar{b}^r \rangle_{g^{t-1}} \in \mathbf{L}_q^{n_1,n_2,\dots,n_{d-1},m}$, g must be some operation such that $g^l \neq I$ for any $l < t$, as otherwise either $\bar{b}^r : \langle \bar{b}^r \rangle_g : \dots : \langle \bar{b}^r \rangle_{g^{t-1}}$ would be periodic, or there would exist some translation smaller than $\bar{b}^r : \langle \bar{b}^r \rangle_g : \dots : \langle \bar{b}^r \rangle_{g^{t-1}}$. Note that $\bar{a}^r : \langle \bar{a}^r \rangle_g : \dots : \langle \bar{a}^r \rangle_{g^{t-1}}$ must be periodic, as otherwise it would belong to $\mathbf{L}_q^{n_1,n_2,\dots,n_{d-1},m}$. As $\bar{a}$ is in $\mathbf{L}_q^{\mathbf{n}}$, either $\bar{a}$ is atranslational or $\bar{a} = \bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1}}$ for some atranslational word $\bar{c}$. If $\bar{a}$ is atranslational, then there is no translation g such that $\bar{a}^r : \langle \bar{a}^r \rangle_g : \dots : \langle \bar{a}^r \rangle_{g^{t-1}}$ is periodic without $\bar{b}^r : \langle \bar{b}^r \rangle_g : \dots : \langle \bar{b}^r \rangle_{g^{t-1}}$ being periodic. On the other hand, if $\bar{a} = \bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'}$ then $(\bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1})^r} : (\bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1})^r+g} : \dots : (\bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1})^r+g^{t-1}}$ must be periodic. For any value of g' where $g' + g \neq I$, $(\bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1})^r} : (\bar{c}^{r'} : \langle \bar{c}^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1})^r+g} : \dots : (\bar{c}^{r'} : \langle c^{r'} \rangle_{g'} : \dots : \langle \bar{c}^{r'} \rangle_{g'^{t'-1})^r+g^{t-1}}$ must be aperiodic. □

Lemma 45. *Let $\overline{\mathbf{n}}$ be a vector of dimensions. Given some value f which is a factor of n_d, and value c which is a factor of f, for any word $\bar{a} \in \mathbf{L}_q^{n_1,n_2,\ldots,n_{d-1},c}$ such that $\bar{a}^r : \langle\bar{a}^r\rangle_g : \ldots : \langle\bar{a}^r\rangle_{g^{t-1}} \in \mathbf{L}_q^{\overline{\mathbf{n}}}$ there exists some word $\bar{b} \in \mathbf{L}_q^{n_1,n_2,\ldots,n_{d-1},f}$ such that $\bar{a}^r : \langle\bar{a}^r\rangle_g : \ldots : \langle\bar{a}^r\rangle_{g^{t-1}} = \bar{b} : \langle\bar{b}\rangle_{g'} : \ldots : \langle\bar{b}\rangle_{g'^{t'-1}}$ where $r \cdot c \leq f$.*

Proof. This claim is shown by considering two cases based on the value of r relative to f. The first case is when $r = \frac{f}{c}$. In this case let $g' = g$ and $\bar{b} = \bar{a}^{r-1} : \langle\bar{a}\rangle_g$. Clearly the Lyndon word $\bar{a}^r : \langle\bar{a}^r\rangle_g : \ldots : \langle\bar{a}^r\rangle_{g^{t-1}}$ is equivalent to $\bar{b} : \langle\bar{b}\rangle_g : \ldots : \langle\bar{b}\rangle_{g^{t-1}}$. In the second case $r < \frac{f}{c}$. If $c{\cdot}r$ is a factor of f, then either the word $\bar{a}^r : \langle\bar{a}^r\rangle_g : \ldots : \langle\bar{a}\rangle_{g^{f/(r\cdot c)}} \in \mathbf{A}_q^{n_1,n_2,\ldots,n_{d-1},f}$ or $\bar{a}^r : \langle\bar{a}^r\rangle_g : \ldots : \langle\bar{a}^r\rangle_{g^{t-1}}$ is periodic, contradicting the initial assumption. If $c \cdot r$ is not a factor of f, then let $r' = \frac{f}{c} \bmod r$ and $t' = \lfloor\frac{f}{cr}\rfloor$. If $\bar{a}^r : \langle\bar{a}^r\rangle_g : \langle\bar{a}^{r'}\rangle_{g^{t'}}$ is not atranslational then $\bar{a}^r : \langle\bar{a}^r\rangle_g : \ldots : \langle\bar{a}^r\rangle_{g^{t-1}}$ must be periodic with a period in dimension d of at least f. Hence $\bar{a}^r : \langle\bar{a}^r\rangle_g : \langle\bar{a}^{r'}\rangle_{g^{t'}} \in A_q^{n_1,n_2,\ldots,n_{d-1},f}$. □

In order to use these characterisations to relate the number of Lyndon words to the number of atranslational words it is important to count the number of possible translations. The main challenge is account for d-dimensional translational Lyndon words made of $(d-1)$-dimensional translation Lyndon words. To this end the set $\mathbf{G}(l, \overline{\mathbf{n}}) = \{(x_1, x_2, \ldots, x_{d-1}) \in [\overline{\mathbf{n}}] : x_i^{n_d/l} \bmod n_i \equiv 0$, and for some dimension i, there exists no value of $j \in [\frac{n_d}{l} - 1]$ such that $x_i^j \bmod n_i \equiv 0\}$ is introduced. This set counts the number of possible translations of a d-dimensional atranslational word of dimensions $(n_1, n_2, \ldots, n_{d-1}, l)$ that may be used to build a d-dimensional Lyndon word of dimensions $\overline{\mathbf{n}}$. The following Lemma provides an important step in the computation of the number of $d-1$-dimensional atranslational words that can be used to build a d-dimensional Lyndon word.

Lemma 46. *Let $\mathbf{G}(l, \overline{\mathbf{n}}) = \{(x_1, x_2, \ldots, x_{d-1}) \in [\overline{\mathbf{n}}] : x_i^{n_d/l} \bmod n_i \equiv 0$, and for some dimension i, there exists no value of $j \in [\frac{n_d}{l} - 1]$ such that $x_i^j \bmod n_i \equiv 0\}$. Given some translation $t \in \mathbf{G}(l, (n_1, n_2, \ldots, n_{d-1}))$, $(t_1, t_2, \ldots, t_{d-2}, \frac{n_{d-1}}{l}) \in \mathbf{G}(l, \overline{\mathbf{n}})$ if and only if $l = 1$ and $n_{d-1} = n_d$.*

Proof. Observe that $\left(\frac{n_i}{l}\right) \cdot n_{i+1} \bmod n_i \equiv 0$. Further note that $\frac{n_i}{l}$ must be the smallest translation such that $t_a \cdot \frac{n_i}{l} \bmod n_a \equiv 0$ for every $a \in [i-1]$ and hence $\frac{n_i}{l}$ must be a factor of n_{i+1}. Additionally, if $n_{i+1} > \frac{n_i}{l}$, then $\frac{n_i}{l}$ exists as some value smaller than n_{i+1} such that $t_a \cdot \frac{n_i}{l} \bmod n_a \equiv 0$. Hence the only possible value of n_{i+1} is $\frac{n_i}{l}$ and further for n_{i+1} to be greater than or equal to n_i, l must be equal to 1 and therefore $n_{i+1} = n_i$. Therefore, given some translation $t \in \mathbf{G}(l, (n_1, n_2, \ldots, n_{d-1}))$, $(t_1, t_2, \ldots, t_{d-2}, \frac{n_{d-1}}{l}) \in \mathbf{G}(l, \overline{\mathbf{n}})$ if and only if $l = 1$ and $n_{d-1} = n_d$. □

Lemma 46 provides the basis for generalising the set $\mathbf{G}(l, \overline{\mathbf{n}})$ to count the number of ways a $d-i$-dimensional atranslational word can be used to form a d-dimensional Lyndon word. More explicitly, consider the i-dimensional atranslational word $\bar{w}$. To use $\bar{w}$ as the translational base of some d-dimensional Lyndon word, note that there must be some translation applied to $\bar{w}$ at every dimension from i to d. Let $\bar{u} = (\bar{w} : \langle \bar{w} \rangle_g : \ldots : \langle \bar{w} \rangle_{g^t}) : \langle (\bar{w} : \langle \bar{w} \rangle_g : \ldots : \langle \bar{w} \rangle_{g^t}) \rangle_h : \ldots : \langle (\bar{w} : \langle \bar{w} \rangle_g : \ldots : \langle \bar{w} \rangle_{g^t}) \rangle_{h^s}$. For $\bar{u}$ to be a Lyndon word, h must not be $(g_1, g_2, \ldots, g_i, n_d/l)$ as $(\bar{w} : \langle \bar{w} \rangle_g : \ldots : \langle \bar{w} \rangle_{g^t}) = \langle (\bar{w} : \langle \bar{w} \rangle_g : \ldots : \langle \bar{w} \rangle_{g^t}) \rangle_{(g_1, g_2, \ldots, g_i, n_d/l)}$.

Using this observation, the following two functions are needed to count the number possible ways an i-dimensional atranslational word can be used to build a d-dimensional word. Let $I(i, l, \overline{\mathbf{n}})$ return the number of dimensions $j \in [i+1, d]$ where there exists some translation $g \in \mathbf{G}(1, (n_1, n_2, \ldots, n_j))$ such that $(g_1, g_2, \ldots, g_{i-1}, \frac{n_i}{l}, 1, 1, \ldots, 1) \in \mathbf{G}(1, \overline{\mathbf{n}})$. The value of $I(i, l, \overline{\mathbf{n}})$ can be computed using Lemma 46 as:

$$I(i, l, (n_1, n_2, \ldots, n_d)) = \begin{cases} 0 & i = d \text{ or } l > 1 \\ 1 + I(i, l, (n_1, n_2, \ldots, n_{d-1})) & n_i = n_d \\ I(i, l, (n_1, n_2, \ldots, n_{d-1})) & n_i \neq n_d \end{cases}$$

The function $H(i, l, \overline{\mathbf{n}}, d)$ is used to return the number of possible sets of translations that can be used to build a d-dimensional Lyndon word from $\bar{w}$. Note that each such set requires $d - i$ translations if $l = n_i$, or $d - i + 1$ translations if $l < n_i$. If $i = d$ then the value of $H(i, l, \overline{\mathbf{n}}, d)$ is either 1, if $l = n_d$, or $|\mathbf{G}(l, \overline{\mathbf{n}})|$ otherwise. If $i < d$, the number of possible translations of dimensions d equals the size of $\mathbf{G}(1, \overline{\mathbf{n}})$ minus the number of dimensions where the translation in the lower dimension can be cancelled out by some translation in a higher dimension. Note that if any translation in dimension i can be cancelled out by some translation in dimensions $j > i$, then following Lemma 46 every translation can be. Therefore the value of $H(i, l, \overline{\mathbf{n}}, d)$ is given by the equation

$$H(i, l, \overline{\mathbf{n}}, d) = \Big\{ (|\mathbf{G}(1, \overline{\mathbf{n}})| - (I(i, l, \overline{\mathbf{n}}))) \cdot (H(i, l, (n_1, n_2, \ldots, n_{d-1}), d - 1))$$

Using the functions $H(i, l, \overline{\mathbf{n}}, d)$ and $I(i, l, \overline{\mathbf{n}})$, the number atranslational words of dimensions $\overline{\mathbf{n}}$ are counted in terms of atranslational words of smaller dimensions and Lyndon words of dimensions $\overline{\mathbf{n}}$. Lemma 47 shows how to express the number of Lyndon words in terms of atranslational words. Lemma 8 builds on this to show how to count the number of atranslational words using Lemma 47.

Lemma 47. *The number of d-dimensional Lyndon words is given in terms of atranslational words as:*

$$\mathbf{L}_q^{\overline{\mathbf{n}}} = |A_q^{\overline{\mathbf{n}}}| + \sum_{i\in[d]}\sum_{l|n_i}\begin{cases}0 & l = n_i\\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right)|\mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l}|\cdot H(i,l,\overline{\mathbf{n}},d) & 1 < l < n_d\end{cases}$$

Proof. Note that every Lyndon word is either atranslational itself, or of the form $\bar{a}^r : \langle\bar{a}^r\rangle_g : \dots : \langle\bar{a}^r\rangle_{g^{t-1}}$ for some $\bar{a} \in \mathbf{L}^{n_1,n_2,\dots,n_{d-1},f}$. Following Lemma 45, every Lyndon word of the form $\bar{a}^r : \langle\bar{a}^r\rangle_g : \dots : \langle\bar{a}^r\rangle_{g^{t-1}}$ can be rewritten as $\bar{b} : \langle\bar{b}\rangle_g : \dots : \langle\bar{v}\rangle_{g^{t-1}}$ for some $\bar{b} \in \mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l\cdot r}$. Let $\bar{a}$ be an atranslational word of dimensions $(n_1, n_2, \dots, n_{d-1}, l)$. For Lyndon words with a d-dimensional translational period there are three cases to consider. If $l = n_d$, then $\bar{a} \in \mathbf{A}_q^{n_1,n_2,\dots,n_d}$. If $\frac{n_d}{l}$ is prime then for every cyclic shift of $X = (x_1, x_2, \dots, x_{d-1})$ where $x_i \in 1 \dots n_i - 1$ such that $x_i^{n_d/l} \bmod n_i \equiv 0$ and for some i $\nexists j \in 1 \dots \frac{n_d}{l} - 1$, the word $\bar{a} : \langle\bar{a}\rangle_X : \dots : \langle\bar{a}^r\rangle_{X^{(n_2/l)-1}} \in \mathbf{L}_q^{\overline{\mathbf{n}}}$. The number of words of the form $\bar{a} : \langle\bar{a}\rangle_g : \dots : \langle\bar{a}\rangle_{g^{(n_d/l)-1}} \in \mathbf{L}_q^{\overline{\mathbf{n}}}$ is $|\mathbf{G}(l, \overline{\mathbf{n}})| \cdot |\mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l}|$.

In the case that $\frac{n_d}{l}$ is not prime, following Lemma 45 there exists some d' such that $\bar{b} = \bar{a} : \langle\bar{a}\rangle_g : \dots : \langle\bar{a}\rangle_{g^{t'}}$ where $\bar{b}$ has dimensions $(n_1, n_2, \dots, l')$. If there are at least two distinct prime factors of $\frac{n_d}{l}$, then note that $\bar{a} : \langle\bar{a}\rangle_g : \dots : \langle\bar{a}\rangle_{g^t}$ is counted for each prime factor. Let p be the number of distinct prime factors. To avoid over counting, every word of size $(n_1, n_2, \dots, n_{d-1}, l)$ needs to be subtracted $p-1$ times. To this end, a new function $P(t)$ is introduced to act as a correction factor.

If $p = 2$ then by setting $P(2) = -1$ the over counting is avoided. If $p = 3$, then as these words were counted three times for each prime factor, then subtracted three times $\frac{n_2}{d\cdot i}$ for each i in the set of prime factors, to avoid under counting these words $P(3)$ must return 1. One special case is when $\frac{n_d}{l}$ has a square prime factor, i^2. In this case as $\frac{n_d}{l\cdot i}$ has the same number of distinct primes, $P(\frac{n_d}{l})$ must return 0. Repeating this argument, $P(s)$ is -1 if s has an even number of prime factors, 1 if s has an odd number of prime factors, and 0 otherwise. Note that this corresponds to $-1(\mu\left(\frac{n_d}{l}\right))$ where $\mu\left(\frac{n_d}{l}\right)$ is the möbius function.

Further, as $P(1) = 1$, both the prime and non-prime cases can be combined into one case.

The same arguments may be applied to the lower dimensional case. Note that the number of possible translations in this case is given by $H(i, l, \overline{\mathbf{n}}, d)$. This gives the number of Lyndon words with a translational period of dimensions $(n_1, n_2, \dots, n_{i-1}, l, 1, 1 \dots, 1)$ as $|\mathbf{A}_q^{(n_1,n_2,\dots,n_{i-1},l,1,1\dots,1)}| \cdot H(i, l, \overline{\mathbf{n}}, d)$, where l is a factor of n_i. In order to account for over counting, the number of possible Lyndon words is multiplied by $\left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right)$. Therefore the total number of Lyndon words of dimensions $\overline{\mathbf{n}}$ is equal to:

$$\mathbf{L}_q^{\overline{\mathbf{n}}} = |A_q^{\overline{\mathbf{n}}}| + \sum_{i\in[d]} \sum_{l|n_i} \begin{cases} 0 & l = n_i \\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right) |\mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l}| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

□

Corollary 8. *The number of atranslational words is given by:*

$$|\mathbf{A}_q^{\overline{\mathbf{n}}}| = |L_q^{\overline{\mathbf{n}}}| - \sum_{i\in[d]} \sum_{l|n_i} \begin{cases} 0 & l = n_i \\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right) |\mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l}| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

Proof. It follows from Lemma 47 that the number of translational words in

$$|\mathbf{L}_q^{\overline{\mathbf{n}}}| = \sum_{i\in[d]} \sum_{l|n_i} \begin{cases} 0 & l = n_i \\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right) |\mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l}| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

Hence the number of atranslational words is

$$|\mathbf{A}_q^{\overline{\mathbf{n}}}| = |\mathbf{L}_q^{\overline{\mathbf{n}}}| - \sum_{i\in[d]} \sum_{l|n_i} \begin{cases} 0 & l = n_i \\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right) |\mathbf{A}_q^{n_1,n_2,\dots,n_{d-1},l}| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

□

From these equations, an upper and lower bound on the number of necklaces is derived.

Lemma 48. *The number of necklaces is bounded by* $\frac{q^N}{N} \leq |\mathcal{N}_q^{\overline{\mathbf{n}}}| \leq q^N$ *where* $\overline{\mathbf{n}}$ *is the dimension vector and* q *is the size of the alphabet.*

Proof. The upper bound comes directly as the number of possible words. Using the above equations, observe that for every word n_i, 1 is a factor. As $\phi(1) = 1$, this gives the number of necklaces as at least $\frac{q^N}{N}$. □

8.1.1 Counting Fixed Content Multidimensional Necklaces

Following the above formulation, the natural question to ask is if there exists similar formulae for the number of fixed content Necklaces, Lyndon words, and atranslational words. Starting with $\mathcal{N}_{\overline{\mathbf{P}}}^{\overline{\mathbf{n}}}$, using the arguments from Graham et. al. [42] the number of necklaces can be computed by considering the possible periodic sub-words. It follows from above that to split along the i^{th} dimension with a period of t_i, $\overline{\mathbf{P}}_j \mod t_i \equiv 0$ for each letter j. For notation, let $\frac{\overline{\mathbf{P}}}{k} = \left(\frac{P_1}{k}, \frac{P_2}{k}, \dots, \frac{P_q}{k}\right)$. Further let $\binom{N}{\overline{\mathbf{P}}}$ denote the multinomial $\binom{N}{P_1, P_2, \dots, P_q}$. For each subword with periods t_1 to t_D there are $\binom{\frac{N}{\text{lcm}(t_1, t_2 \dots t_D)}}{\frac{\overline{\mathbf{P}}}{\text{lcm}(t_1, t_2 \dots t_D)}}$ possible fixed-content words of dimensions $(t_1, t_2, \dots, t_d)$. Therefore the total number of fixed content necklaces is:

$$|\mathcal{N}_{\overline{\mathbf{P}}}^{\overline{\mathbf{n}}}| = \frac{1}{N} \sum_{t_1 | \gcd(n_1, \overline{\mathbf{P}})} \phi\left(\frac{n_1}{d_1}\right) \dots \sum_{t_D | \gcd(n_D, \frac{\overline{\mathbf{P}}}{d_1 \dots d_{D-1}})} \phi\left(\frac{n_D}{d_D}\right) \binom{\frac{N}{\text{lcm}(t_1, t_2 \dots t_D)}}{\frac{\overline{\mathbf{P}}}{\text{lcm}(t_1, t_2 \dots t_D)}} \tag{8.3}$$

Where $\gcd(n, \overline{\mathbf{P}})$ is the greatest common denominator of both n and every value in the vector $\overline{\mathbf{P}}$, i.e. $\gcd(n, P_1, P_2, \dots, P_q)$. The number of fixed content Lyndon words can be counted though repeated application of the Móbius inversion formula using the previous arguments as:

$$L_{\overline{\mathbf{P}}}^{\overline{\mathbf{n}}} = \sum_{d_1 | \gcd(n_1, \overline{\mathbf{P}})} \mu\left(\frac{n_1}{d_1}\right) \dots \sum_{d_D | \gcd(n_D, \frac{\overline{\mathbf{P}}}{d_1 d_2 \dots d_{D-1}})} \mu\left(\frac{n_D}{d_D}\right) |\mathcal{N}_{\overline{\mathbf{P}}}^{d_1, d_2 \dots d_D}| \tag{8.4}$$

Finally the number of atranslational fixed content necklaces is derived using the same arguments as in the unconstrained case. More specifically, the number of atranslational words of dimensions $\overline{\mathbf{v}}$ is given by:

$$|\mathbf{A}_{\overline{\mathbf{P}}}^{\overline{\mathbf{n}}}| = |\mathbf{L}_{\overline{\mathbf{P}}}^{\overline{\mathbf{n}}}| - \sum_{l \mid \gcd(n_d, \overline{\mathbf{P}})} \begin{cases} 0 & l = 1 \\ -\mu\left(\frac{n_d}{l}\right) |\mathbf{A}_{\overline{\mathbf{P}}/l}^{n_1, n_2, \dots, n_{d-1}, n_d/l}| \cdot |\mathbf{G}(l, \overline{\mathbf{n}})| & 1 < l < n_d \\ -\mu\left(\frac{n_d}{l}\right) |\mathbf{L}_{\overline{\mathbf{P}}/l}^{n_1, n_2, \dots, n_{d-1}}| \cdot |\mathbf{G}(l, \overline{\mathbf{n}})| & l = n_d \end{cases} \tag{8.5}$$

8.2 Ranking Multidimensional Necklaces

Recall that the rank of a necklace $\tilde{\mathbf{w}}$ in the set $|\mathcal{N}_q^{\overline{\mathbf{n}}}|$ is the number of necklaces smaller than $\tilde{\mathbf{w}}$ under some ordering, in this case the ordering given in Definition 14. More broadly, we can take any word $\bar{v}$ and determine the number of necklaces that are represented by a word smaller than $\bar{v}$ using the same ordering. In this case, the smallest necklace greater than or equal to $\bar{v}$ is determined using the *NextNecklace* algorithm given in Theorem 28. For the remainder of this section, we assume that we are finding the rank of some word that is the canonical representation of a necklace. Before we provide a high level overview of how this problem is tackled, we need to define a method of comparing two words of different sizes. In this section, two words $\bar{w} \in \Sigma^{\overline{\mathbf{n}}}$ and $\bar{u} \in \Sigma^{\overline{\mathbf{f}}}$ are compared if and only if $n_i \bmod f_i \equiv 0$ for every $i \in [d]$. As such, given such a pair of words $\bar{u}^{\overline{\mathbf{n}}/\overline{\mathbf{f}}}$ is used to denote the word $\bar{u}'$ where $\bar{u}'_{(i_1, i_2, \dots, i_d)} = \bar{u}_{(i_1 \bmod f_1, i_2 \bmod f_2, \dots, i_d \bmod f_d)}$. Using this notation, a comparison between word $\bar{w}$ and $\bar{u}$ is given as:

Definition 27. *Let $\bar{u} \in \Sigma^{(f_1, f_2, \dots, f_d)}$, and $\bar{v} \in \Sigma^{(n_1, n_2, \dots, n_d)}$ where $n_i \bmod f_i \equiv 0$. $\bar{u} < \bar{v}$ if and only if $\bar{u}^{\overline{\mathbf{n}}/\overline{\mathbf{f}}} < \bar{v}$ following Definition 14. Similarly, $\bar{u} > \bar{v}$ if and only if $\bar{u}^{\overline{\mathbf{n}}/\overline{\mathbf{f}}} > \bar{v}$.*

At a high level, the ranking algorithm for a word $\bar{w}$ works by first determining the number of words of dimensions $(f_1, f_2, \dots, f_d)$ smaller than $\bar{w}$, denoted $T(\bar{w}, f_1, f_2, \dots, f_d)$, for every f_i that is factor of n_i. This value is transformed, first from $T(\bar{w}, f_1, f_2, \dots, f_d)$ to the number of aperiodic words smaller than $\bar{w}$, denoted $L(\bar{w}, f_1, f_2, \dots, f_d)$, and finally to the number of atranslational words smaller than $\bar{w}$, $A(\bar{w}, f_1, f_2, \dots, f_d)$. The set $A(\bar{w}, f_1, f_2, \dots, f_d)$ is then translated into the rank of $\bar{w}$ within the set of atranslational necklaces $A_q^{(f_1, f_2, \dots, f_d)}$, denoted $RA(\bar{w}, f_1, f_2, \dots, f_d)$. This rank is than used to calculate the rank within the set of Lyndon words $RL(\bar{w}, f_1, f_2, \dots, f_d)$. Finally, this rank is translated to the necklace rank $RN(\bar{w}, f_1, f_2, \dots, f_d)$. Lemmas 49, and 50 show how to transform the size of the sets $T_{\bar{w},f_1,f_2,\dots,f_d}$ into the size of $A(\bar{w}, n_1, n_2, \dots, n_d)$. Lemmas 51, 52 and 53 show how to transform the size of the sets $A(\bar{w}, f_1, f_2, \dots, f_d)$ into the value $RN(\bar{w}, n_1, n_2, \dots, n_d)$.

In order to compute the size of $T(\bar{w}, f_1, f_2, \dots, f_d)$, we partition the set into the subsets $B(\bar{w}, g, j, f_1, f_2, \dots, f_d)$. Here $B(\bar{w}, g, j, f_1, f_2, \dots, f_d)$ contains the set of words $\bar{v} \in T(\bar{w}, f_1, f_2, \dots, f_d)$ where: (1) g is the smallest translation such that $\langle \bar{v} \rangle_g < \bar{w}$ and (2) j is the length of the longest shared prefix between $\langle \bar{v}^{\mathbf{n}/\mathbf{f}} \rangle_g$ and $\bar{w}'$, i.e. the largest value such that $\left(\langle \bar{v}^{\mathbf{n}/\mathbf{f}} \rangle_g\right)_{[1,j]} = \bar{w}_{[1,j]}$. The size of each set $B(\bar{w}, g, j, f_1, f_2, \dots, f_d)$ is computed by considering the structure of the words in $B(\bar{w}, g, j, f_1, f_2, \dots, f_d)$. This requires the size of two further sets to be computed, the number of non-cyclic words where every suffix is greater than $\bar{w}$, and the number of words of dimensions $(f_1, f_2, \dots, f_{d-1})$ that are smaller than $\bar{w}_{j+1}$. The first of these sets is the more technical, requiring a new recursive technique to be built which is provided in Subsection 8.2.1.

The remainder of this section is structured as follows. Lemmas 49 to 53 cover the theoretical tools needed to rank necklaces. Following this, an overview of the method to compute the size of $T(\bar{w}, f_1, f_2, \dots, f_d)$ is provided. Finally, Subsection 8.2.1 covers the main sub method used in the ranking process. Finally Theorem 26 is restated and formally proven.

Lemma 49. *The size of $L(\bar{w}, n_1, n_2, \dots, n_d)$ can be computed in terms of $T(\bar{w}, f_1, f_2, \dots, f_d)$ using the equation:*

$$L(\bar{w}) = \sum_{f_1|n_1} \mu\left(\frac{n_1}{f_1}\right) \sum_{f_2|n_2} \mu\left(\frac{n_2}{f_2}\right) \dots \sum_{f_d|n_d} \mu\left(\frac{n_d}{f_d}\right) T(\bar{w}, f_1, f_2, \dots, f_d)$$

Proof. Observe that every word in $T(\bar{w}, n_1, n_2, \dots, n_d)$ is either aperiodic, in which case it is in $L(\bar{w}, n_1, n_2, \dots, n_d)$, or periodic, in which case it is in $L(\bar{w}, f_1, f_2, \dots, f_d)$ where f_i is a factor of n_i. Following the same arguments as given in Section 8.3, the size of $T(\bar{w}, n_1, n_2, \dots, n_d)$ is equal to $\sum\limits_{f_1|n_1} \sum\limits_{f_2|n_2} \dots \sum\limits_{f_d|n_d} |L(\bar{w}, f_1, f_2, \dots, f_d)|$. By repeated application of the Möbius inversion formula, the size of $L(\bar{w}, n_1, n_2, \dots, n_d)$ can be computed as:

$$L(\bar{w}, n_1, n_2, \dots, n_d) = \sum_{f_1|n_1} \mu\left(\frac{n_1}{f_1}\right) \sum_{f_2|n_2} \mu\left(\frac{n_2}{f_2}\right) \dots \sum_{f_d|n_d} \mu\left(\frac{n_d}{f_d}\right) T(\bar{w}, f_1, f_2, \dots, f_d)$$

□

Lemma 50. *The size of* $A\left(\bar{w}, n_1, n_2, \ldots, n_d\right)$ *equals*

$$|L(\bar{w}, n_1, n_2, \ldots, n_d)| - \sum_{i \in [d]} \sum_{l | n_i} \begin{cases} 0 & l = n_i \\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right) |A_q^{n_1, n_2, \ldots, n_{d-1}, l}| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

Proof. Following the arguments given in Lemma 47, observe that any Lyndon word in $L(\bar{w}, n_1, n_2, \ldots, n_d)$ is either be atranslational, or of the form $\bar{a} : \langle \bar{a} \rangle_g : \ldots : \langle \bar{a} \rangle_{g^{t-1}}$. In the latter case, let $l = |\bar{a}|_d$. Note that $\bar{a}$ must be either in $A(\bar{w}_{[1,l]}, n_1, n_2, \ldots, n_{d-1}, l)$, if $l > 1$ or $L(\bar{w}_1)$ if $l = 1$. Repeating the same arguments as in Lemma 47 allows the size of $A\left(\bar{w}, n_1, n_2, \ldots, n_d\right)$ to be written as:

$$|L(\bar{w}, n_1, n_2, \ldots, n_d)| - \sum_{i \in [d]} \sum_{l | n_i} \begin{cases} 0 & l = n_i \\ \left(\prod_{t=i+1}^{d-1} -\mu(n_t)\right)\left(-\mu\left(\frac{n_i}{l}\right)\right) |A_q^{n_1, n_2, \ldots, n_{d-1}, l}| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

□

Lemma 51. *The rank* $RA(\bar{w}, n_1, n_2, \ldots, n_d) = \frac{1}{N}|A\left(\bar{w}, n_1, n_2, \ldots, n_d\right)|$.

Proof. Observe that any atranslational necklace of dimensions $\overline{\mathbf{n}}$ has exactly N representations. Therefore the number of atranslational necklaces smaller than $\bar{w}$ is $\frac{1}{N}A\left(\bar{w}\right)$. Hence

$$RA(\bar{w}, n_1, n_2, \ldots, n_d) = \frac{1}{N}A\left(\bar{w}, n_1, n_2, \ldots, n_d\right).$$

□

In order to use the rank $RA(\bar{w}, n_1, n_2, \ldots, n_d)$ to determine the rank $RL(\bar{w}, n_1, n_2, \ldots, n_d)$, it is necessary to consider the special case where $\bar{w}$ is a translational, aperiodic word. Let $\bar{u}$ be the translational period of $\bar{w}$ with dimensions $(g_1, g_2, \ldots, g_{d-1}, \frac{n_i}{l}, 1, 1, \ldots, 1)$ for $g \in \mathbf{G}(l, n_1, n_2, \ldots, n_i)$ and $i \in [d]$. Further, let $\bar{u}[j]$ be the Lyndon word of dimensions $(g_1, g_2, \ldots, g_{i-1}, \frac{n_i}{l}, n_{i+1}, \ldots, n_j)$ such that $\bar{u}[j]_{\bar{\mathbf{i}}} = \bar{w}_{\bar{\mathbf{i}}}$ for $j \in [i+1, d]$. Note that $\bar{u}[j]$ can be written as $\bar{u}[j] : \bar{u}[j-1] : \langle \bar{u}[j-1] \rangle_{r_j} : \ldots : \langle \bar{u}[j-1] \rangle_{r_j^{n_j - 1}}$, for some $r_j \in \mathbf{G}(l_j, (n_1, n_2, \ldots, n_j))$ where $l_j = 1$ if $j > i$ and 1 otherwise. Observe that the number of Lyndon words with a translational period of $\bar{u}$ with dimensions $\overline{\mathbf{n}}$ that are smaller than $\bar{w}$ is equal to the sum of the number of translations in $\mathbf{G}(l_j, n_1, n_2, \ldots, n_j)$

multiplied by $H(i, l_i, \overline{\mathbf{n}})$. Let $S(g, l, (n_1, n_2, \ldots, n_j))$ return the number of translations in $\mathbf{G}(l, (n_1, n_2, \ldots, n_j))$ smaller than g. To this end let $U(\bar{w})$ return either:

- 0 if $\bar{w}$ is either atranslational or periodic.
- $\sum\limits_{j=i}^{d} \begin{cases} S(r_j, l, (n_1, n_2, \ldots, n_j)) & j = i \\ S(r_j, 1, (n_1, n_2, \ldots, n_j)) & \textit{otherwise.} \end{cases}$ if $\bar{w}$ is a Lyndon word with a translational period of g.

Using $U(\bar{w})$, the number of Lyndon words can be computed from $RA(\bar{w}, n_1, n_2, \ldots, n_d)$ as follows.

Lemma 52. *The rank*

$$RL(\bar{w}, n_1, n_2, \ldots, n_d) = RA(\bar{w}, n_1, n_2, \ldots, n_d) + U(\bar{w}) +$$

$$\sum_{i \in [d]} \sum_{l | n_i} \begin{cases} 0 & l = n_i \\ \left(\prod\limits_{t=i+1}^{d-1} -\mu(n_t)\right) \left(-\mu\left(\frac{n_i}{l}\right)\right) |RA(\bar{w}_{[1,l]}, n_1, n_2, \ldots, n_{i-1})| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

Proof. Note that every necklace smaller than $\bar{w}$ is either atranslational, in which case it is counted by $RA(\bar{w}, n_1, n_2, \ldots, n_d)$, or is translational. In the latter case following Lemma 47 for each necklace counted by $RA(\bar{w}_{[1,l]}, n_1, n_2, \ldots, n_{d-1}, l)$, there are $H(i, l, \overline{\mathbf{n}})$ translational necklace counted by $RL(\bar{w}, n_1, n_2, \ldots, n_d)$. Further, if $\bar{w}$ is a translational Lyndon word of the form $\bar{v} : \langle \bar{v} \rangle_g : \ldots : \langle \bar{v} \rangle_g$, then there are there are $U(\bar{w})$ Lyndon words of the form $\bar{v} : \langle \bar{v} \rangle_g : \ldots : \langle \bar{v} \rangle_g$ where $\bar{v}_{\bar{\mathbf{i}}} = \bar{w}_{\bar{\mathbf{i}}}$ for every $\bar{\mathbf{i}} \in [|\bar{v}|]$. Following Lemma 47 $RL(\bar{w}, n_1, n_2, \ldots, n_d)$ is counted in terms of $RA(\bar{w}, n_1, n_2, \ldots, n_{d-1}, l)$ as:

$$RL(\bar{w}, n_1, n_2, \ldots, n_d) = RA(\bar{w}, n_1, n_2, \ldots, n_d) + U(\bar{w}) +$$

$$\sum_{i \in [d]} \sum_{l | n_i} \begin{cases} 0 & l = n_i \\ \left(\prod\limits_{t=i+1}^{d-1} -\mu(n_t)\right) \left(-\mu\left(\frac{n_i}{l}\right)\right) |RA(\bar{w}_{[1,l]}, n_1, n_2, \ldots, n_{i-1})| \cdot H(i, l, \overline{\mathbf{n}}, d) & 1 < l < n_d \end{cases}$$

□

Lemma 53. *The rank* $RN(\bar{w}, n_1, n_2, \dots, n_d) = \sum\limits_{f_1 | n_1} \sum\limits_{f_2 | n_2} \dots \sum\limits_{f_d | n_d} RL(\bar{w}, f_1, f_2, \dots, f_d)$.

Proof. Observe that every necklace counted by $RN(\bar{w}, n_1, n_2, \dots, n_d)$ has a period of $\overline{\mathbf{m}}$ where m_i is a factor of $|\bar{w}|_i$ for every $i \in 1 \dots d$. As $RL(\bar{w}, f_1, f_2, \dots, f_d)$ counts the rank among aperiodic necklaces of dimensions $(f_1, f_2, \dots, f_d)$, the rank among necklaces is given by:

$$RN(\bar{w}, n_1, n_2, \dots, n_d) = \sum_{f_1 | n_1} \sum_{f_2 | n_2} \dots \sum_{f_d | n_d} RL(\bar{w}, f_1, f_2, \dots, f_d)$$

□

This leaves the challenge of computing the size of $T(\ \bar{w},\ f_1,\ f_2,\ \dots,\ f_d)$. To this end, $T(\bar{w}, f_1, f_2, \dots, f_d)$ is partitioned into the sets $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}})$ such that $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}})$ contains every word $\bar{v} \in T(\bar{w}, \overline{\mathbf{f}})$ where:

- g_d is the smallest translation in dimension d of $\bar{v}$ such that $\langle \bar{v} \rangle_{(\theta_1, \theta_2, \dots, \theta_{d-1}, g_d)} < \bar{w}$ for some translation $\theta \in Z_{(f_1, f_2, \dots, f_{d-1})}$.
- j is the largest value such that $(\langle \bar{v}' \rangle_{(\theta_1, \theta_2, \dots, \theta_{d-1}, g_d)})_{[1,j]} = \bar{w}'_{[1,j]}$.

To compute the size of $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}})$, there are two cases to consider based on the values of g_d and j.

Case 1: $g_d + j \leq f_d$. In this case every word $\bar{v} \in \mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}})$ can be written as $\bar{a} : \langle \bar{w}_{[1,j]} : \bar{b} \rangle_\theta : \bar{c}$ where:

- $\bar{a}$ is a $(f_1, f_2, \dots, f_{d-1}, g_d)$ dimensional word for which there exists no translation $r \in Z_{(f_1, f_2, \dots, g_d)}$ such that $(\langle \bar{a} \rangle_r)_{[1, g_d - r_d]} < \bar{w}_{[1, g_d - r_d]}$.
- $\bar{b}$ is some word of dimensions $(f_1, f_2, \dots, f_{d-1})$ that is smaller than $\bar{w}_{j+1}$.
- θ is some translation in $Z_{(f_1, f_2, \dots, f_{d-1})}$.
- $\bar{c}$ is an unrestricted word of dimensions $(f_1, f_2, \dots, f_{d-1}, f_d - (g_d + j + 1))$.

To count the number of words of this form, it is necessary to compute the number of non-cyclic words of dimensions $(f_1, f_2, \dots, f_{d-1}, i)$ where every suffix of length i is greater than $\bar{w}_{[1,i]}$. To this end a new set $\beta(\bar{w}, i, j, f_1, f_2, \dots, f_{d-1})$ is introduced containing every word $\bar{u}$ where:

- The dimensions of $\bar{u}$ are $(f_1, f_2, \ldots, f_{d-1}, i)$.
- There exists no translation $g \in Z((f_1, f_2, \ldots, f_{d-1}))$ where $\langle \bar{u}_{[i-l,i]} \rangle_g \leq \bar{w}_{[1,l]}$.
- The first j slices of $\bar{u}$ are equal to the first j slices of $\bar{w}$, i.e. $\bar{u}_{[1,j]} = \bar{w}_{[1,j]}$.

When it is clear from context $\beta(\bar{w}, i, j, f_1, f_2, \ldots, f_{d-1})$ is denoted $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$. A method to compute the size of $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$ is given in Subsection 8.2.1. Using $|\beta(\bar{w}, i, j, \overline{\mathbf{f}})|$ as a black box, the number of possible values of $\bar{a}$ is $|\beta(\bar{w}, i, j, \overline{\mathbf{f}})|$. Similarly, the number of possible values of $\bar{b}$ is given by $q^{f_1 \cdot f_2 \cdot \ldots \cdot f_{d-1}} - |\beta(\bar{w}_{j+1}, 1, 0, \overline{\mathbf{f}})| - 1$. The number of possible values of θ is equal to the size of the set $\mathbf{\Theta} = \{r \in Z_{\overline{\mathbf{f}}} : \nexists s \in Z_{\overline{\mathbf{f}}}$ where $s < r$ and $\langle \bar{w} \rangle_r = \langle \bar{w} \rangle_s\}$. Finally, the number of values of $\bar{c}$ is given by $q^{f_1 \cdot f_2 \cdot \ldots \cdot f_{d-1} \cdot (f_d - (g_d + j + 1))}$. Therefore the size of $\mathbf{B}(\bar{w}, g, j, f\overline{\mathbf{f}})$ when $g_d + j < n_d$ is given by:

$$|\beta(\bar{w}, g_d, 0, \overline{\mathbf{f}})| \cdot (q^{f_1 \cdot f_2 \cdot \ldots \cdot f_{d-1}} - |\beta(\bar{w}_{j+1}, 1, 0, \overline{\mathbf{f}})| - 1) \cdot |\mathbf{\Theta}| \cdot q^{f_1 \cdot f_2 \cdot \ldots \cdot f_{d-1} \cdot (f_d - (g_d + j + 1))}$$

Case 2: $g_d + j > f_d$. In this case every word $\bar{v} \in \mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}})$ can be written as $\langle \bar{w}_{[j+g_d-f_d,j]} : \bar{b} \rangle_\theta : \bar{a} : \langle \bar{w}_{[1,j+g_d-f_d]} \rangle_\theta$ where:

- $\bar{a}$ is a $(f_1, f_2, \ldots, f_{d-1}, f_d - (j+1))$ dimensional word for which there exists no translation $r \in Z_{(f_1, f_2, \ldots, f_{d-1}, g_d)}$ such that $\langle \bar{a} \rangle_r < \bar{w}_{[1,g_d]}$.
- $\bar{b}$ is some word of dimensions $(f_1, f_2, \ldots, f_{d-1})$ that is smaller than $\bar{w}_{j+1}$.
- θ is a translation in the set $\mathbf{\Theta} = \{r \in Z_{(f_1, f_2, \ldots f_{d-1})} : \nexists s \in Z_{(f_1, f_2, \ldots f_{d-1})}$ where $s < r$ and $\langle \bar{w}_{[1,j]} \rangle_r = \langle \bar{w}_{[1,j]} \rangle_s\}$.

The number of possible values of θ is equal to the size of the set $\mathbf{\Theta}$ as in Case 1. The number of possible values of $\bar{b}$ in this case is somewhat more complicated than in Case 1. Let t be the length of the longest suffix of $\bar{w}_{[j+g_d-f_d,j]}$ such that $\bar{w}_{[j-t,j]} = \bar{w}_{[1,t]}$. To avoid $\langle \bar{v} \rangle_\psi$, for some $\psi \in Z_{(f_1, f_2, \ldots, f_{d-1}, f_d - g_d)}$, being smaller than $\bar{w}$, $\bar{b}$ must be greater than or equal to $\bar{w}_{t+1}$. Note that the number of words greater than $\bar{w}_{t+1}$ is given by $\beta(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}})$. Therefore the number of possible values of $\bar{b}$ as $(q^{f_1 \cdot f_2 \cdot \ldots \cdot f_{d-1} \cdot (f_d - (g_d + j + 1))} - \beta(\bar{w}_{j+1}, 1, 0) - 1) - (q^{n_1 \cdot n_2 \cdot \ldots \cdot f_{d-1} \cdot (f_d - (g_d + j + 1))} - \beta(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}})) = \beta(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}}) - \beta(\bar{w}_{j+1}, 1, 0, \overline{\mathbf{f}}) + 1$. If $\bar{b} = \bar{w}_{t+1}$, the number of possible values of $\bar{a}$ is given by $|\beta(\bar{w}, f_d + t - j, t + 1, \overline{\mathbf{f}})|$. Otherwise the number of possible values of $\bar{a}$ is given by $|\beta(\bar{w}, f_d - j - 1, 0, \overline{\mathbf{f}})|$. Therefore the total number of words of the form $\langle \bar{w}_{[j+g_d-f_d,j]} : \bar{b} \rangle_\theta : \bar{a} : \langle \bar{w}_{[1,j+g_d-f_d]} \rangle_\theta$ is:

$$|\beta(\bar{w}, f_d + t - j, t+1, \overline{\mathbf{f}})| + \big(|\beta(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}})| - |\beta(\bar{w}_{j+1}, 1, 0, \overline{\mathbf{f}})|\big) \cdot |\beta(\bar{w}, f_d - j - 1, 0, \overline{\mathbf{f}})| \cdot |\mathbf{\Theta}|$$

8.2.1 Computing the Number of Prefixes Greater than $\bar{w}$

The size of $T(\bar{w})$ require the size of the set $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$ to be computed. Let $\bar{v} \in \beta(\bar{w}, i, j, \overline{\mathbf{f}})$. Observe that if $v_{j+1} > \bar{w}_{j+1}$, then for any translation $g \in Z((f_1, f_2, \dots, f_{d-1}, j+1))$, $\bar{v}_{[1,j+1]} > \bar{w}_{[1,j+1]}$. Therefore the number of possible values of $\bar{v}_{[j+2,i]} = |\beta(\bar{w}, i-j-1, 0, \overline{\mathbf{f}})|$. Similarly the number of values of $\bar{v}$ where $\bar{v}_{j+1} = \bar{w}_{j+1}$ is $|\beta(\bar{w}, i, j+1, \overline{\mathbf{f}})|$. This allows the size of $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$ to be computed in a recursive manner. In the special case where $j = i$, there is either one word in $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$, if $j = 0$, or none if $j > 0$. Let $NS(\bar{w}, j, \overline{\mathbf{f}})$ return the number of possible slices of dimensions $(f_1, f_2, \dots, f_{d-1}$ that are greater than $\bar{w}_{j+1}$. Using $NS(\bar{w}, j, \overline{\mathbf{f}})$ as a black box, the size of $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$ can be computed as:

$$|\beta(\bar{w}, i, j, \overline{\mathbf{f}})| = \begin{cases} 0 & i = j, j > 0 \\ 1 & i = j = 0 \\ NS(\bar{w}, j, \overline{\mathbf{f}}) \cdot |\beta(\bar{w}, i-j-1, 0, \overline{\mathbf{f}})| + |\beta(\bar{w}, i, j+1, \overline{\mathbf{f}})| & \textit{Otherwise.} \end{cases}$$

This leaves the problem of computing $NS(\bar{w}, j, \overline{\mathbf{f}})$. This is done by considering two cases. First are the set of slices that belong to a necklace class greater than $\bar{w}_{j+1}$. The number of such necklaces can be computed as $|\mathcal{N}_q^{(f_1, f_2, \dots, f_{d-1})}| - RN(\bar{w}_j, f_1, f_2 \dots, f_{d-1})$, i.e. the number of necklaces of dimensions $(f_1, f_2, \dots, f_{d-1})$ minus the necklaces smaller than $\bar{w}_j$. To account for the number of possible translations of each necklace, it is easiest to use the sets of aperiodic words instead. The number of such words are determined by counting the number of atranslational words of dimensions $(f_1, f_2, \dots, f_{i-1}, h_i, 1, \dots, 1)$ for every $i \in [d]$ and factor h_i of f_i. This rank is then multiplied by the number of possible translations, given by $f_1 \cdot f_2 \cdot \dots \cdot f_{i-1} \cdot h_i$, and $H(i, h, (f_1, f_2, \dots, f_d), d)$ to account for the number of necklaces with a translational period in $T(\bar{w}, f_1, f_2 \dots, f_d)$. The second case to consider are translations of $\bar{w}_{j_1}$ greater than $TR(\bar{w}_{j+1})$. This is given by $TP(\bar{w}_{j+1}) - TR(\bar{w}_{j+1})$. This allows the number of necklaces greater than $\bar{w}_j$ along with the number of translations of these necklaces to be counted as:

$$NS(\bar{w}, j, \bar{\mathbf{f}}) = (TP(\bar{w}_{j+1}) - TR(\bar{w}_{j+1})) + \sum_{i \in [d-1]} \sum_{h_i | f_i} RA(\bar{w}_j, \overline{\mathbf{h[i]}})) \cdot |\overline{\mathbf{h[i]}}| \cdot H(i, h, (f_1, f_2, \dots, f_d), d)$$

Where $\overline{\mathbf{h[i]}} = (f_1, \dots, f_{i-1}, h_i, \dots, 1)$ and $|\overline{\mathbf{h[i]}}| = f_1 \cdot f_2 \cdot \dots \cdot f_{i-1} \cdot h_i$.

Theorem 26. *The rank of a d-dimensional necklace with dimensions $\bar{\mathbf{n}}$ can be computed with a complexity of $O(N^5)$ for both time and space.*

Proof. Lemmas 49, 50, 51, 52, and 53 show that to rank $RN(\bar{w})$, the first step is to compute the size of $T(\bar{w}, f_1, f_2 \dots, f_d)$. Following Lemma 50, to compute the size of $A(\bar{w}), f_1, f_2 \dots, f_d$, the set $A(\bar{w}_{[1,l]}, f_1, f_2 \dots, f_{d-1}, l)$ must be computed for every factor l of f_d, alongside the set $L(\bar{w}, f_1, f_2 \dots, f_d)$ and $L(\bar{w}_1, f_1, f_2 \dots, f_{d-1})$. Note that this requires at most $\log_2(n_d)$ sets to be computed. The size of the set $L(\bar{w}, f_1, f_2 \dots, f_{d-1})$ can be computed by computing the size of $T(\bar{w}, h_1, h_2, \dots, h_d)$ where h_i is a factor of f_i. Therefore for $L(\bar{w}, f_1, f_2 \dots, f_{d-1})$, the size of at most $\log_2(N)$ sets $T(\bar{u}, h_1, h_2 \dots, h_d)$ must be computed.

Following the above observations, $T(\bar{w}, n_1, n_2 \dots, n_d)$ can be computed by determining the size of $B(\bar{w}, g, j, n_1, n_2 \dots, n_{d-1})$ using n_d^2 combinations of j and g. For each pair j and g, the size of $\beta(\bar{w}, i, j, n_1, n_2 \dots, n_{d-1})$ must be computed for some value of i. This is done in a dynamic programming approach. Starting with $i = j$, the size of $|\beta(\bar{w}, i, j, n_1, n_2 \dots, n_d)|$ is computed using the previously computed values as a basis. As such, the size of $|\beta(\bar{w}, i, j, n_1, n_2 \dots, n_d)|$ for every pair i and j can be computed in n_d^2 time multiplied by the complexity of computing $NS(\bar{w}, j, n_1, n_2 \dots, n_d)$. To compute $NS(\bar{w}, j, n_1, n_2 \dots, n_d)$, $d \cdot \frac{\log_2 N}{d} = \log_2 \frac{N}{n_d}$ words of dimensions $d-1$ must be ranked.

As there are n_d^2 values of $\beta(\bar{w}, i, j, n_1, n_2 \dots, n_d)$, and $\log_2(\frac{N}{n_d})$ words of dimensions $d-1$ must be ranked for each of the n_d^2 values of $\beta(\bar{w}, i, j, n_1, n_2 \dots, n_d)$, to precompute every value of $\beta(\bar{w}, i, j, n_1, n_2 \dots, n_d)$ $n_d^2 \cdot \log_2(\frac{N}{n_d})$ time is needed, multiplied by the cost of ranking a $d-1$ word. If $d = 2$, then the rank at this step can be computed in $O(n_1^2)$ time using existing algorithms due to Sawada and Williams [92]. Hence the size of $\beta(\bar{w}, i, j, n_1, n_2 \dots, n_d)$ for every value of i and j can be computed in the two dimensional case in $O(n_d \cdot N \cdot \log_2(\frac{N}{n_d}) \cdot n_1^2) = O(N^2 \cdot \log_2(\frac{N}{n_d}))$ time. To get the rank of a two dimensional word, a further n_2^2 time is needed to compute the size of $T(\bar{w}, n_1, n_2 \dots, n_d)$, with $\log_2(N)$ sets of $T(\bar{w})$ to be computed. Therefore the rank of a two dimensional word can be computed in $O(n_2^2 \cdot \log_2(N)N^2 \cdot \log_2(\frac{N}{n_d}))$.

Similarly in the three dimensional case, the set of all values of $\beta(\bar{w}, i, j, n_1, n_2 \ldots, n_d)$ can be computed in $O(n_3^2 \cdot n_2^2 \cdot \log_2(N) \frac{N^2}{n_3^2} \cdot \log_2(n_1)) = O(N^2 \cdot n_2^2 \cdot \log_2(N) \cdot \log_2(n_1))$. Thus the complexity of ranking a three dimensional word is $O(n_3^2 \cdot \log_2(N) \cdot N^2 \cdot n_2^2 \cdot \log_2(N) \cdot \log_2(n_1))$ time. In the more general case, a total of $n_d^2 \cdot \log_2(N)$ words of dimension $d-1$ must be ranked. Using the two and three dimensional cases as a base, the total complexity of ranking a d dimensional word is $O(\left(\prod_{i=2}^{d} n_i^4 \cdot \log_2(n_i)\right) n_1^2) \leq O(N^5)$. $\square$

8.2.2 Ranking Fixed Content Necklaces

The same tools used in the unrestricted case are used in the fixed content case. As before, the goal is to count the number of words of dimensions $\overline{\mathbf{f}}$ that belong to a necklace class smaller than the ranked word $\bar{w}$, with the additional constraint that $F = f_1 \cdot f_2 \cdot \ldots \cdot f_d$ is a factor of P_i for every $P_i \in \overline{\mathbf{P}}$. The main complexity is generalising the previous approach comes from the constraint on the content. Let $\mathbf{T}(\bar{w}, i, j, \overline{\mathbf{f}}, t, \overline{\mathbf{Q}})$ be the set of words of dimensions $\overline{\mathbf{f}}$ with fixed content $\overline{\mathbf{Q}}$ belonging to a necklace class smaller than $\bar{w}$. As in the unconstrained case, this set is subdivided based on two values g_d and j. Formally, the set $\mathbf{B}(\bar{w}, \overline{\mathbf{f}}, \overline{\mathbf{q}}) \subseteq \mathbf{T}(\bar{w}, i, j, \overline{\mathbf{f}}, t, \overline{\mathbf{Q}})$ contain every word $\bar{v} \in \mathbf{T}(\bar{w}, \overline{\mathbf{f}}, \overline{\mathbf{Q}})$ where:

- $h = (h_1, h_2, \ldots, h_{d-1}, g_d)$ is the smallest translation such that $\langle \bar{v} \rangle_h < \bar{w}$.
- j is the largest value such that $(\langle \bar{w} \rangle_h)_{[1,j]} = \bar{w}_{[1,j]}$.
- $\mathrm{P}(\bar{w}_{[1,j]}) + \overline{\mathbf{q}} = \overline{\mathbf{Q}}$.

In order to compute the size of $\mathbf{B}(\bar{w}, \overline{\mathbf{f}}, \overline{\mathbf{q}})$, a generalisation of $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$ is needed. More precisely, due to the constraint on the content, it is necessary not only to count the number of words for which every suffix is greater than $\bar{w}$, as in $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$, but instead to count the number of such suffixes of the words in $\mathbf{B}(\bar{w}, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ for each prefix of $\bar{w}$. To this end, let $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$ return the number of triples $(\bar{x}, \bar{y}, \bar{z})$ where:

- $\bar{x}$ is a word of dimensions $(f_1, f_2, \ldots, f_{d-1}, i)$ such that every suffix of $\bar{x}$ belongs to a necklace class larger than the prefix of $\bar{w}$ of the same length and $\bar{x}_{[1,j]} = \bar{w}_{[1,j]}$.
- $\bar{y}$ is a word of dimension $(f_1, f_2, \ldots, f_{d-1})$ such that $\bar{y} < \bar{w}_t$.
- $\bar{z}$ has dimensions $(f_1, f_2, \ldots, f_{d-1}, l)$.

- $\mathrm{P}(\bar{x} : \bar{y} : \bar{z}) = \overline{\mathbf{q}}$.

As with $\beta(\bar{w}, i, j, \overline{\mathbf{f}})$, the problem of computing $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$ is solved recursively. In effect, the problem is solved in three stages. First, the number of possible values of $\bar{x}$ are computed. Secondly, for each value of $\bar{x}$, the number of possible values of $\bar{y}$ are computed. Finally, the number of possible values of $\bar{z}$ are computed using the remaining symbols.

To compute the number of values of $\bar{x}$, observe that $\bar{x}_{[j+1,i]}$ is counted by either $\gamma(\bar{w}, i-j-1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}} - P(\bar{x}_{j+1}), t, l)$, if $\bar{x}_{j+1} > \bar{w}_{j+1}$, or by $\gamma(\bar{w}, i, j+1, \overline{\mathbf{f}}, \overline{\mathbf{q}} - P(\bar{w}_{j+1}), t, l)$ if $\bar{x}_{j+1} = \bar{w}_{j+1}$. In order to count the number of possible values of $\bar{x}_1$ that are greater than $\bar{w}_{j+1}$, the same approach as in the unrestricted setting is used. Let $V(\overline{\mathbf{q}}, \overline{\mathbf{f}})$ contain every Parikh vector $\overline{\mathbf{q}}'$ where $\overline{\mathbf{q}}'_i \leq \overline{\mathbf{q}}_i$ and $\sum\limits_{i=1}^{q} \overline{\mathbf{q}}'_i = f_1 \cdot f_2 \cdot \ldots \cdot f_{d-1}$. Further let $X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ return the number of values of $\bar{x}_{j+1}$ with a Parikh vector $\overline{\mathbf{q}}$ that are greater than $\bar{w}_{j+1}$. $X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ is computed in a similar manner to $NS(\bar{s}, j, \bar{f})$. Formally:

$$X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}) = \left(\begin{array}{l} (TP(\bar{w}_{j+1}) - RP(\bar{w}_{j+1})) + \\ \sum\limits_{i \in [d-1]} \sum\limits_{h_i | f_i} RA(\bar{w}_{j+1}, \overline{\mathbf{h}[\mathbf{i}]}, \overline{\mathbf{q}}) \cdot (h_i \cdot f_{i-1} \cdot f_{i-2} \cdot \ldots \cdot f_1) \end{array} \right)$$

Therefore the number of possible values of $\bar{x}_{j+1}$ of dimensions $\overline{\mathbf{f}}$ is given by

$$\sum_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{f}})} X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}').$$

Similarly the number of possible values of $\bar{y}$ is the number of words either belonging to a necklace class smaller than $\langle \bar{w}_{j+1} \rangle$, or belonging to the same necklace class as $\langle \bar{w}_{j+1} \rangle$, while having a smaller rotation. Note that the number of such words for a given Parikh vector $\overline{\mathbf{q}}$ is given by $\binom{F}{\overline{\mathbf{q}}} - X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}) - 1$. Finally, the number of possible words of dimensions $(f_1, f_2, \ldots, f_{d-1}, l)$ with the Parikh vector $\overline{\mathbf{q}}$ is given by $\binom{F}{\overline{\mathbf{q}}}$. Using these observations $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$ can be computed as:

$$\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l) = \begin{cases} \left(\begin{array}{l} \gamma(\bar{w}, i, j+1, \overline{\mathbf{f}}, \overline{\mathbf{q}} - \mathrm{P}(\bar{w}_{j+1}), t, l) + \\ \sum\limits_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{f}})} X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}') \cdot \gamma(\bar{w}, i-j-1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}} - \overline{\mathbf{q}}', t, l) \end{array} \right) & i > j \\ \sum\limits_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{f}})} \left(\binom{F}{\overline{\mathbf{q}}'} - X(\bar{w}_{t+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}') \right) \cdot \binom{F \cdot l}{\overline{\mathbf{q}} - \overline{\mathbf{q}}'} & i = j = 0 \\ 0 & i = j, i > 0 \end{cases}$$

Using $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, l)$, the size of $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ can be computed in the same manner as the size of $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}})$. More precisely, two cases are considered based on the value of g_d and j.

Case 1: $g_d + j \leq f_d$. In this case every word $\bar{v} \in \mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ can be written as $\bar{a} : \langle \bar{w}_{[1,j]} : \bar{b} \rangle_\theta : \bar{c}$ where:

- $\bar{a}$ is a $(f_1, f_2, \dots, f_{d-1}, g_d)$ dimensional word for which there exists no translation $r \in Z_{(f_1, f_2, \dots, g_d)}$ such that $(\langle \bar{a} \rangle_r)_{[1, g_d - r_d]} < \bar{w}_{[1, g_d - r_d]}$.
- $\bar{b}$ is some word of dimensions $(f_1, f_2, \dots, f_{d-1})$ that is smaller than $\bar{w}_{j+1}$.
- θ is some translation in $Z_{(f_1, f_2, \dots, f_{d-1})}$.
- $\bar{c}$ is an unrestricted word of dimensions $(f_1, f_2, \dots, f_{d-1}, f_d - (g_d + j + 1))$.

Note that $\gamma(\bar{w}, g_d, 0, (f_1, f_2, \dots, f_{d-1}), \overline{\mathbf{q}}, j, n - g_d - j - 1)$ counts the number of possible values of $\bar{a}, \bar{b}$ and $\bar{c}$. The number of possible values of θ is equal to the size of the set $\mathbf{\Theta} = \{r \in Z_{\overline{\mathbf{f}}} : \nexists s \in Z_{\overline{\mathbf{f}}}$ where $s < r$ and $\langle \bar{w} \rangle_r = \langle \bar{w} \rangle_s\}$. Therefore the size of $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ when $g_d + j < f_d$ is given by:

$$\gamma(\bar{w}, g_d, 0, (f_1, f_2, \dots, f_{d-1}), \overline{\mathbf{q}}, j, n - g_d - j - 1) \cdot |\mathbf{\Theta}|$$

Case 2: $g_d + j > f_d$. In this case every word $\bar{v} \in \mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ can be written as $\langle \bar{w}_{[j+g_d-f_d, j]} : \bar{b} \rangle_\theta : \bar{a} : \langle \bar{w}_{[1, j+g_d-f_d]} \rangle_\theta$ where:

- $\bar{a}$ is a $(f_1, f_2, \dots, f_{d-1}, f_d - (j+1))$ dimensional word for which there exists no translation $r \in Z_{(f_1, f_2, \dots, f_{d-1}, g_d)}$ such that $\langle \bar{a} \rangle_r < \bar{w}_{[1, g_d]}$.

- $\bar{b}$ is some word of dimensions $(n_1, n_2, \ldots, n_{d-1})$ that is smaller than $\bar{w}_{j+1}$.
- θ is a translation in the set $\mathbf{\Theta} = \{r \in Z_{(f_1,f_2,\ldots f_{d-1})} : \nexists s \in Z_{(f_1,f_2,\ldots f_{d-1})}$ where $s < r$ and $\langle \bar{w}_{[1,j]} \rangle_r = \langle \bar{w}_{[1,j]} \rangle_s\}$.

The number of possible values of θ is equal to the size of the set $\mathbf{\Theta}$ as in Case 1. The number of possible values of $\bar{b}$ in this case is somewhat more complicated than in Case 1. Let t be the length of the longest suffix of $\bar{w}_{[j+g_d-n_d,j]}$ such that $\bar{w}_{[j-t,j]} = \bar{w}_{[1,t]}$. To avoid $\langle \bar{v} \rangle_\psi$, for some $\psi \in Z_{(f_1,f_2,\ldots,f_{d-1},n_d-g_d)}$, being smaller than $\bar{w}$, $\bar{b}$ must be greater than or equal to $\bar{w}_{t+1}$. Let $\gamma'(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ return only the number of words with Parikh vector $\overline{\mathbf{q}}$ that are greater than $\bar{w}$ for any rotation of the suffix, defined as:

$$\gamma'(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}) = \begin{cases} \begin{pmatrix} \gamma'(\bar{w}, i, j+1, \overline{\mathbf{f}}, \overline{\mathbf{q}} - \mathrm{P}(\bar{w}_{j+1})) + \\ \displaystyle\sum_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{h}[\mathbf{i}]})} X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}') \cdot \gamma'(\bar{w}, i-j-1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}} - \overline{\mathbf{q}}') \end{pmatrix} & i > j \\ 1 & i = j = 0 \\ 0 & \textit{otherwise.} \end{cases}$$

Using $\gamma'(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$, the number of words greater than $\bar{w}_{t+1}$ is given by

$$\sum_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{f}})} \gamma'(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}}').$$

This gives the number of possible values of $\bar{a}$ and $\bar{b}$ where $\bar{a} > \bar{w}_{t+1}$ as

$$\sum_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{f}})} \big(\gamma'(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}}') - \gamma'(\bar{w}_{j+1}, 1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}}')\big) \cdot \gamma'(\bar{w}, i, f_d - (j+1), \overline{\mathbf{f}}, \overline{\mathbf{q}} - \overline{\mathbf{q}}').$$

Accounting for the case where $\bar{a} = \bar{w}_{t+1}$, the size of $\mathbf{B}(\bar{w}, g_d, j, \overline{\mathbf{f}}, \overline{\mathbf{q}})$ when $g_d + j > f_d$ is given by:

$$|\mathbf{\Theta}| \cdot \begin{pmatrix} \gamma'(\bar{w}, f_d - j, t+1, \overline{\mathbf{f}}, \overline{\mathbf{q}} - \mathrm{P}(\bar{w}_{t+1})) + \\ \displaystyle\sum_{\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}}, \overline{\mathbf{f}})} \big(\gamma'(\bar{w}_{t+1}, 1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}}') - \gamma'(\bar{w}_{j+1}, 1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}}')\big) \cdot \gamma'(\bar{w}, i, f_d - (j+1), \overline{\mathbf{f}}, \overline{\mathbf{q}} - \overline{\mathbf{q}}') \end{pmatrix}$$

Theorem 27. *The rank of a word among the set of multidimensional necklaces of dimensions $\overline{\mathbf{n}} = (n_1, n_2, \dots, n_d)$ over the alphabet Σ with the Parikh vector $\overline{\mathbf{q}}$ can be computed in $O(N^{6+q})$ time, where $q = |\Sigma|$ and $N = n_1 \cdot n_2 \cdot \dots \cdot n_d)$.*

Proof. Following the same arguments from Theorem 26, the complexity cost of this problem comes from computing $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$. In order to compute $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$, a dynamic programming approach is used. Observe that $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$ can be computed in $|V(\overline{\mathbf{q}})|$ steps if $X(\bar{w}_{j+1}, \overline{\mathbf{f}}, \overline{\mathbf{q}}')$ and $\gamma(\bar{w}, i-j-1, 0, \overline{\mathbf{f}}, \overline{\mathbf{q}} - \overline{\mathbf{q}}', t, l)$ have been computed for every $\overline{\mathbf{q}}' \in V(\overline{\mathbf{q}})$. Further, $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}, \overline{\mathbf{q}}, t, l)$ can be computed in $O(1)$ time when $i = j$ if $X(\bar{w}_j, \overline{\mathbf{f}}, \overline{\mathbf{q}}')$ has been precomputed for every value of $\bar{w}_j, \overline{\mathbf{f}}$ and $\overline{\mathbf{q}}'$.

In order to compute $X(\bar{w}_j, \overline{\mathbf{f}}, \overline{\mathbf{q}}')$, it is necessary to compute the rank of $\bar{w}_j$ among the set of $d-1$ atranslational words, in turn requiring $\gamma(\bar{w}, i, j, \overline{\mathbf{f}}', \overline{\mathbf{q}}, t, l)$ to be computed for every $\overline{\mathbf{f}}' \in \{(m_1, m_2, \dots, m_{d-2}) : n_i \bmod m_i \equiv 0\}$. By repeating the same arguments from Theorem 26, the problem of ranking fixed content necklaces can be done in an additional factor of $O(N^{q+1})$, accounting for the number of possible Parikh vectors $\overline{\mathbf{q}}$, and possible values of l. Therefore, the total complexity is $O(N^{6+q})$. □

8.3 Generating and Unranking Multidimensional Necklaces

To complete the generalisation of results on one dimensional necklace to the multidimensional setting, there are two major problems left. These are the problems of generating the set of necklaces in order efficiently and of *unranking* a necklace. The unranking problem asks, given an alphabet Σ of size q, and dimension vector $\overline{\mathbf{n}}$, what is the i^{th} necklace in $\mathcal{N}_q^{\overline{\mathbf{n}}}$. In many ways the relationship between generation and unranking can be seen as analogous to the problems of counting the number of necklaces and ranking a given necklace. In both cases one problem relates to the complete set, while the other focuses on a single necklace.

8.3.1 Generating Necklaces

Starting with the problem of generation, the idea presented here is based on generation of lower dimensional necklaces, generalising the 1D techniques to the higher dimensional setting. For the 1D setting, there have been several approaches for the generation of necklaces in constant amortised time, notably those of Cattell et. al. [17] and of Fredricksen and Maiorana [31]. A tempting approach would be to make an alphabet of size equal to the number of necklaces with dimensions $(n_1, \dots, n_{d-1})$ and to generate the 1D necklaces

$$\begin{array}{lllllll}
\begin{bmatrix} A & A \\ A & A \end{bmatrix} & \rightarrow \begin{bmatrix} A & A \\ A & B \end{bmatrix} & \rightarrow \begin{bmatrix} A & A \\ B & B \end{bmatrix} & \rightarrow \begin{bmatrix} A & B \\ A & B \end{bmatrix} & \rightarrow \begin{bmatrix} A & B \\ B & A \end{bmatrix} & \rightarrow \begin{bmatrix} A & B \\ B & B \end{bmatrix} & \rightarrow \begin{bmatrix} B & B \\ B & B \end{bmatrix} \\
\begin{bmatrix} 1 \\ 1 \end{bmatrix} & \rightarrow \begin{bmatrix} 1 \\ 2 \end{bmatrix} & \rightarrow \begin{bmatrix} 1 \\ 3 \end{bmatrix} & \rightarrow \begin{bmatrix} 2 \\ 2 \end{bmatrix} & \rightarrow \begin{bmatrix} 2 \\ translated(2) \end{bmatrix} & \rightarrow \begin{bmatrix} 2 \\ 3 \end{bmatrix} & \rightarrow \begin{bmatrix} 3 \\ 3 \end{bmatrix}
\end{array}$$

Figure 8.1: An example of generation of $(2, 2)$ necklaces, over the alphabet (A, B). The following mapping from necklace to code has been used: $AA \rightarrow 1$, $AB \rightarrow 2$, $BB \rightarrow 3$.

from that. While this approach would generate a set of necklaces, as each d-dimensional necklace is comprised of a set of $d-1$-dimensional necklaces, it would also miss any in which one or more slices are translated by any degree. Similarly, representing every slice under each translation would generate words that are not necklaces. Let us illustrate it for a set of necklaces over a binary alphabet with dimensions $(2, 2)$. The complete set of necklaces is given in Figure 8.1. Of particular interest is the necklace represented by $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$. While the first row, AB, is the canonical form of a 1D necklace, BA is not as it is equal to AB after a cyclic shift. Despite AB occurring as the necklace representation multiple times prior to this, BA only occurs at this point. As such, the situations where some slice may or may not be translated need to be understood and taken into account in order to generate the set of necklaces.

Before generating the set of necklace, the idea of a *multidimensional prenecklace* must be established. A prenecklace is a word $\bar{w}$ of dimensions $(n_1, n_2, \dots, n_d)$ such that there exists some necklace of dimensions $(n_1, n_2, \dots, n_{d-1}, n_d + m)$ represented by a word $\bar{u}$ such that $\bar{u}_{[1,n_d]} = \bar{w}$. Note that every necklace is a prenecklace.

Lemma 54. *Given* $\tilde{\mathbf{w}}, \tilde{\mathbf{u}} \in |\mathcal{N}_q^{\mathbf{n}}|$ *such that* $rank(\tilde{\mathbf{u}}) = rank(\tilde{\mathbf{w}}) + 1$, *let* $Pre(\bar{w}, \bar{u}) = \{\bar{v} \in \Sigma^{\mathbf{n}} : \bar{u} > \bar{v} > \bar{w}, \bar{v}$ *is a prenecklace*$\}$. *The size of* $Pre(\bar{w}, \bar{u})$ *is at most* n_d.

Proof. This is statement is proven constructively. Let $NextPrenecklace(\bar{u})$ return the smallest prenecklace greater than $\bar{u}$. Given some word $\bar{u}$, let p be the length of the longest prefix of $\bar{u}$ that is a necklace. If $p < n_d$, the word $\bar{u}'$ is defined $\bar{u}'_i = \bar{u}_{i \bmod p}$. If $\bar{u}' \neq \bar{u}$, then $\bar{u}'$ is the smallest prenecklace that is greater than $\bar{u}$. Otherwise, let i be the last slice of $\bar{u}$ such that $\bar{u}_i \neq \bar{Q}$. Note that $\bar{u}_{[1,i-1]}\bar{Q}^{n_d - i}$ is a necklace. The auxiliary function $NextSlice(\bar{v})$ is introduced as returning the subsequent word in the ordering defined in Section 2.

$$NextSlice(\bar{v}) = \begin{cases} translate(\bar{v}) & TR(\bar{v}) < TP(\bar{v}) \\ NextNecklace(\langle \bar{v} \rangle) & otherwise. \end{cases}$$

Here $NextNecklace$ is treated as a black box that returns the next necklace in the ordering. Note that $\bar{u}_{[1,i-1]} : NextSlice(\bar{u}_i))_{j \bmod i}$ must be a necklace as any suffix of $\bar{u}_{[1,i-1]} : NextSlice(\bar{u}_i))_{j \bmod i}$ must be greater than $\bar{u}_{[1,i-1]}$. The word $\bar{u}'$ is redefined as $\bar{u}'_j = (\bar{u}_{[1,i-1]} : NextSlice(\bar{u}_i))_{j \bmod i}$. As $\bar{u}_{[1,i-1]} : NextSlice(\bar{u}_i))_{j \bmod i}$ is a necklace, $\bar{u}'$ is a prenecklace. Therefore $\bar{u}'$ is returned. To determine the size of $Pre(\bar{w}, \bar{u})$, note that the slice at position $i + 1$ must be smaller than $\bar{Q}$, therefore by repeating this process at most n_d times, the necklace of rank $rank(\bar{w}) + 1$ is found, and hence the size of $Pre(\bar{w}, \bar{u})$ is at most n_d. □

Theorem 28. *Let $\bar{w}$ be a word of dimensions $\mathbf{\overline{n}}$. $NextNecklace(\bar{w})$ returns the smallest word $\bar{u} > \bar{w}$ such that $\bar{u} = \langle \bar{u} \rangle$ in $O(N)$ time and requiring no more than $O(N)$ space.*

Proof. Following Lemma 54, note that by applying the function $NextPrenecklace$ at most n_d times, the smallest necklace greater than $\bar{w}$ can be determined. As each call to $NextPrenecklace$ requires $NextNecklace$ as a subroutine, to determine the next prenecklace of dimensions $d-1$, n_{d-1} prenecklaces of dimensions $d-2$ must be determined. Following this logic, to determine the next prenecklace of dimensions d at most $\frac{N}{n_d \cdot n_{d-1} \cdot \ldots \cdot n_{d-i+1}}$ prenecklaces of dimensions i must be considered. Therefore a total of $O(N)$ time is needed to compute all n_d prenecklaces. As it takes at most $O(N)$ time to determine if a word is a necklace, this process takes at most $O(N)$ time. □

8.3.2 Unranking Necklaces

Lemma 55. *The number of necklaces in $|\mathcal{N}_q^{\mathbf{\overline{n}}}|$ with a given prefix $\bar{w}$ can be determined in $O(N^5)$ time.*

Proof. Let $\bar{w}$ be a word of dimensions $(n_1, n_2, \ldots, n_{d-1}, a)$, where $a \leq n_d$. To determine the number of necklaces with a prefix $\bar{w}$, two new words $\bar{u}$ and $\bar{v}$ are defined such that $\bar{u}$ is the smallest necklace reference with the prefix $\bar{w}$, and $\bar{v}$ the greatest. The value of $\bar{u}$ is determined by first constructing the word $\bar{u}'$ where $\bar{u}'_i = \bar{w}_{i \bmod a}$. If $\bar{u}'$ is a necklace representative, then $\bar{u} = \bar{u}'$. Otherwise using Theorem 28, the value of $\bar{u}$ is computed from $\bar{u}'$ in at most $O(N)$ operations. Let $\bar{Q} = q^{(n_1, n_2, \ldots, n_{d-1})}$. The word $\bar{v}$ is defined as being equal to $\bar{w} : \bar{Q}^{n_d - a}$. If $\bar{v}$ is not a necklace representative then there exists no necklace with $\bar{w}$ as a prefix. Otherwise, the number of necklaces with $\bar{w}$ as a prefix equals $RN(\bar{v}) - RN(\bar{u}) + 1$. $\square$

Using Lemma 55, a recursive unranking algorithm can be built by iteratively building the prefix of the i^{th} necklace in $|\mathcal{N}_q^{\mathbf{n}}|$.

Theorem 29. *The i^{th} necklace in $\mathcal{N}_q^{\mathbf{n}}$ can be generated (unranked) in $O\left(N^{6(d+1)} \cdot \log^d(q)\right)$ time with no more than $O(N^5)$ space complexity.*

Proof. The unranking procedure is done in a similar manner to the one dimensional case as presented by Sawada and Williams [92]. At a high level, the idea is to iteratively generate the necklace by generating prefixes of increasing length. Let $\bar{w}$ be the canonical representative of the i^{th} necklace. Further let $\bar{Q} = q^{(n_1, n_2, \dots, n_{d-1})}$, the word of dimensions $(n_1, n_2, \dots, n_{d-1})$ where every position is occupied by the symbol k. The first slice of $\bar{w}$ is determined through a binary search. Let $\bar{u}$ be the canonical representation of j^{th} necklace of dimensions $(n_1, n_2, \dots, n_{d-1})$. Note that if $\bar{u}$ is the first slice of $\bar{w}$, then the rank of $\bar{w}$ must be between the rank of the smallest necklace starting with $\bar{u}$ and the greatest. These necklaces are determined using the same process as laid out in Lemma 55. Let $\bar{a}$ be the smallest such word and $\bar{b}$ the greatest. Therefore $\bar{u}$ is the fist slice of $\bar{w}$ if and only if $RN(\bar{a}) \leq i \leq RN(\bar{b}$. Otherwise, depending on the value of i relative to $RN(\bar{a})$ and $RN(\bar{b})$ the next value of $\bar{u}$ is checked, with $\bar{u}$ determined by a binary search. Note that there are at most q^{N/n_d} necklaces of size $(n_1, n_2, \dots, n_{d-1})$, the binary search requires at most $\log(q^{N/n_d}) = \frac{N}{n_d} \log k$ necklaces to be checked.

For the t^{th} slice, where $t \geq 2$, the process is slightly more complicated. As in the first case, to determine if the $\langle \bar{w}_t \rangle = \bar{u}$, the smallest and largest such words are determined and ranked. To that end, let $\bar{a}$ be the smallest possible word that is the canonical form of a necklace and has the prefix $\bar{w}_{[1,t-1]} : \langle \bar{u} \rangle_g$, and let $\bar{b}$ be the greatest. The value of $\bar{a}$ is computed in $O(N)$ time following the techniques outlined in Theorem 28. The word $\bar{b} = \bar{w}_{[1,t-1]} : \langle \bar{u} \rangle_g : \bar{Q}^{n_d - t}$ where g is the largest translation such that $\bar{u} \neq \langle \bar{u} \rangle_g$. Using these words, $\langle \bar{w}_t \rangle = \bar{u}$ if and only if $RN(\bar{a}) \leq i \leq RN(\bar{b})$.

The time complexity of this process comes from the recursive nature of algorithm. In dimension d, n_d slices need to be computed, each requiring at most $\frac{N}{n_d} \cdot \log(q)$ necklaces to be ranked, the ranking having a complexity of N^5. Note that while determining the necklace that needs to be ranked has a complexity of N^2, this is not multiplicative with the complexity of ranking as each step is done independently. To determine each of these necklaces, a necklace of dimensions $(n_1, n_2, \dots, n_{d-1})$ must be unranked, adding an additional complexity of $n_{d-1} \cdot \frac{N}{n_d \cdot n_{d-1}} \cdot \frac{N^5}{n_d^5} \cdot \log(q)$. As each dimension requires necklaces of the dimension one lower to be computed, the total complexity is $O\left(\prod_{i=0}^{d} \frac{N^6 \cdot \log(q)}{\prod_{j \in [1,i]} n_{d-j}^6}\right)$. In the worst case, where $n_1 = N$ and $n_i = 1$ for $i \in [2, d]$, this is simplified to $O\left(N^{6(d+1)} \cdot \log^d(q)\right)$.

Regarding space complexity, observe that at each step it is necessary to rank the two words sharing some prefix of length j, requiring $O(N^5)$ space. As no further information needs to be stored beyond the necklaces being ranked and a trivial amount of information regarding the current process of the binary search, the total space complexity is no more than $O(N^5)$. □

Lemma 56. *The number of necklaces in the set $\mathcal{N}_{\mathbf{p}}^{\mathbf{n}}$ sharing a given prefix $\bar{a}$ can be computed in $O(n^{6+q})$ time.*

Proof. Note that the ranking process outline in Theorem 27 allows the rank of the canonical form of any necklace to be computed within the set $\mathcal{N}_{\mathbf{p}}^{\mathbf{n}}$ in $O(n^{6+q})$ time. Therefore by comparing the ranks of the smallest and largest necklaces sharing $\bar{a}$ as a prefix, the number of necklaces in $\mathcal{N}_{\mathbf{p}}^{\mathbf{n}}$ sharing the prefix can be computed. Following Theorem 28, the smallest and largest necklaces can be found in $O(N)$ time. As the ranking process requires at most $O(n^{6+q})$ time, the total complexity of determining the number of necklaces sharing a given prefix is $O(n^{6+q})$. □

Corollary 9. *The i^{th} necklace in $\mathcal{N}_{\mathbf{p}}^{\mathbf{n}}$ can be unranked in $O(N^{(q+7)(d+1)}) \log^d(q)$ time.*

Proof. Fixed content multidimensional necklaces can be unranked in the same manner as unconstrained necklaces, presented in Theorem 15. As in that theorem, a binary search is used over the alphabet Σ to determine the i^{th} necklace iteratively. Following Lemma 56, the number of necklaces sharing a given prefix can be computed in $O(n^{6+q})$ time. The complexity of this process is given by the same arguments as in Theorem 15, with the additional cost due to the added complexity of ranking fixed content necklaces compared to unconstrained necklaces, being $O(N^{6+q})$ and $O(N^5)$ respectively. □

Chapter 9

Conclusions

In this thesis we have provided a large collection of results within the theme of Crystal Structure Prediction. These lay the foundation for future work in this theme, while leaving open many potential directions for future research.

In Chapters 3 and 4 we looked at the hardness of this problem in a verity of settings. Chapter 3 showed that abstract version of CSP is undecidable for the $2-\mathcal{CMV}$ class of energy functions including the Buckingham-Coulomb and Ising models. Chapter 4 strengths the results of Chapter 3 showing that the problem remains NP-hard even when the size of the unit cell is restricted. Additionally Theorem 2 shows that, for the Buckingham-Coulomb potential and a given size of unit cell, CSP remains hard to approximate within any positive factor. Finally, Theorem 3 provides a parameterised algorithm for the fixed-size unit cell in one dimension for any function with a cut-off distance, including both the Buckingham-Coulomb and Ising interactions.

This leaves open the question of the hardness of alternative forms of CSP. One direction would be to look at more restricted energy functions. The goal of this direction would be to show either undecidability or NP-hardness for our models of CSP under these constraints. One further direction to consider is moving to the continuous setting. The goal in the continuous setting would be both to provide a formal mathematical model, and to provide a proof of either undecidability or NP-hardness. Both of these setting would strengthen the claims regarding the complexity of this problem in more realistic settings.

Chapter 5 provides the underlying results for the k-centre problem on cyclic words. Theorem 10 shows that the problem of verifying a solution to the k-centre problem on cyclic words is itself NP-hard. Theorem 11 uses de-Bruijn sequences to show that the

k-centre problem for necklaces of length n can be solved within a factor of $1 + \frac{\log_q(kn)}{n - \log_q(kn)} - \frac{\log_q^2(kn)}{2n(n - \log_q(kn))}$. Similarly, Theorem 12 uses de-Bruijn hyper tori to show that the k-centre problem for d-dimensional necklaces of dimensions $\overline{\mathbf{n}}$ can be solved within a factor of $1 + \frac{\log_q(kN)}{N - \log_q(kN)} - \frac{\log_q^2(kN)}{2N(N - \log_q(kN))}$. Theorem 13 provides a prefix based algorithm for approximating the solution to the k-centre problem for d-dimensional necklaces within a factor of $1 + \frac{\log_q(kN)}{N - \log_q(kN)} - \frac{\log_q^2(k)}{2N(N - \log_q(kN))}$.

The main open problem left by these results is on the computational complexity of the the k-centre problem on cyclic words. While Theorem 10 shows that the problem is NP-hard to verify, it does not show that the problem itself is NP-hard to solve. Indeed it is possible that there exists either a better algorithm for the k-centre problem or a stronger bound showing that our algorithm is optimal. We would make two conjectures regarding Problem 11. First, we conject that it is like that the problem is at least NP-hard. This conjecture is equivalent to saying that the k-sampling problem on the graphs corresponding to the set of necklaces over the overlap distance can not be solved in faster than linear time relative to the size of the graph. Secondly, we conject that it is likely that there is no better polynomial time approximation algorithm for Problem 11. The gap between the lower and upper bound is likely to be resolved by improving the lower bound through more explicit analysis.

Chapters 6, 7, and 8 use the prefix based algorithm from Chapter 5 as motivation to study classes of cyclic words. Chapter 6 studies the class of bracelets. Theorem 14 shows how to rank bracelets in lexicographic order. Theorem 14 answers the question posed by Sawada and Williams [92] as to weather or not bracelets may be ranked in polynomial time in the affirmative. However, we leave open the second part of that open question, namely if there exists a cubic time algorithm for ranking bracelets.

Chapter 7 covers three further classes of cyclic words: the class of direct solutions to Diophantine equations; the class of necklace solutions to Diophantine equations; and the class of necklaces not containing any subword from a given set $\mathcal{F}$ of forbidden subwords. Theorem 19 provides an algorithm to rank a word amongst the set of direct solutions to a linear equation $A \cdot \overline{\mathbf{x}} = \overline{\mathbf{C}}$ in lexicographic order in $O(C \cdot n \cdot q^2)$ time, where C refers to the product of the entries of the vector $\overline{\mathbf{C}}$, i.e. $C = \overline{\mathbf{C}}_1 \cdot \overline{\mathbf{C}}_2 \cdot \ldots \cdot \overline{\mathbf{C}}_m$. Theorem 24 provides an algorithm for for ranking necklaces with forbidden subwords in $O(\log^2(n) \cdot n^8 \cdot |\mathcal{F}|^2)$ time. Section 8.1 provides equations for counting multidimensional necklaces in the general case. Section 8.1.1 extends these algorithms to the fixed content setting. Theorem

26 provides an $O(N^5)$ time algorithm to rank multidimensional necklaces. Theorem 27 compliments Theorem 26 by providing an $O(N^{6+q})$ time algorithm to rank fixed content multidimensional necklaces. Finally Theorems 28 and 29 provide algorithms to generate and unrank multidimensional necklaces.

While these results provide a large diversity in terms of ranking cyclic words, there are several directions that would be useful to extend our ranking and by extensions k-centre problem for these classes. The most natural extensions would be in generalising concepts such as forbidden subwords and necklaces with Parikh vectors satisfying systems of linear equations to the multidimensional case.

One particularly interesting direction would be in generalising bracelets to multiple dimensions. There are several possible ways to do so, determined by which forms of symmetry we wish to capture. The most natural direction would be to allow for reflections along each dimension, in effect generalising the Dihedral group to multiple dimensions analogously to our generalisation of the cyclic group. An alternative but very appealing direction would be to account for *rotational* symmetry. This would account for rotating each word against some two dimensional plane. While the number of potential operations in this case could make full generalisation to the multidimensional setting difficult, even a formal description of these words in dimensions two and three could provide a significant reduction in the space of potential crystal structures. As such, work in this direction could prove to be invaluable for the motivation of sampling crystal structures.

A further direction that could provide strong result would be in the field of lower bounds. This is a large open area, as not only are there no lower bounds for ranking bracelets, fixed content necklace or multidimensional necklaces, there are in fact no lower bounds even for ranking necklaces. We would conject that the current $O(n^2)$ algorithm is optimal for the necklace ranking problem, however it remains to either prove this conjecture, or to strengthen it building on the techniques provided by both Abboud et al. [1] and Bringmann [13].

Perhaps the main direction this thesis opens are those regarding the k-centre problem on implicitly defined graphs. To the best of our knowledge these are the first results concerning the k-centre problem for such objects. The diversity of our results regarding the approximation ratio suggests that a similar prefix based approach may be used for alternative structures. Some natural settings to look at would be unlabelled necklaces, chord diagrams and partitions.

Further, by providing new means to sample diverse structures we provide a new means for crystal structure prediction. Not only is this a powerful tool for solving crystal structure prediction both in combination with other techniques and as a technique own its own merits. While both further theoretical and experimental results are needed to better understand the utilisation of these techniques, our initial theoretical analysis show great promise as a novel tool for solving crystal structure prediction.

Bibliography

[1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. *SIAM Journal on Computing*, 47(6):2527–2555, 2018.

[2] D. Adamson, A. Deligkas, V. V. Gusev, and I. Potapov. On the hardness of energy minimisation for crystal structure prediction. In *SOFSEM 2020*, volume 12011 of *Lecture Notes in Computer Science*, pages 587–596, 2020.

[3] Duncan Adamson, Argyrios Deligkas, Vladimir V. Gusev, and Igor Potapov. The k-centre selection problem for multidimensional necklaces, 2021. `arXiv:2108.01990`.

[4] Duncan Adamson, Vladimir V. Gusev, Igor Potapov, and Argyrios Deligkas. Ranking Bracelets in Polynomial Time. In Paweł Gawrychowski and Tatiana Starikovskaya, editors, *32nd Annual Symposium on Combinatorial Pattern Matching (CPM 2021)*, volume 191 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/13955`, `doi:10.4230/LIPIcs.CPM.2021.4`.

[5] Cyril Allauzen and Bruno Durand. Tiling problems. In The Classical Decision Problems*, Perspectives in Mathematical Logic*, pages 407–420. Springer, 2001.

[6] M. Anselmo, M. Madonia, and C. Selmi. Toroidal Codes and Conjugate Pictures. In *LATA 2019*, volume 11417 of *Lecture Notes in Computer Science*, pages 288–301, 2019.

[7] D. Antypov, A. Deligkas, V.V. Gusev, M. J. Rosseinsky, P. G. Spirakis, and M. Theofilatos. Crystal Structure Prediction via Oblivious Local Search. In *SEA 2020*, volume 160 of *LIPIcs*, pages 21:1–21:14, 2020.

[8] Kyriakos Axiotis and Christos Tzamos. Capacitated dynamic programming: Faster knapsack and graph algorithms, 2018. `arXiv:1802.06440`.

[9] F Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241–3253, oct 1982.

[10] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.

[11] Robert Berger. *The undecidability of the domino problem.* Number 66 in memoirs of the american mathematical society. American Mathematical Soc., 1966.

[12] Ulrich Berger and Faron Moller. Report on bctcs & algouk 2020. *Bulletin of EATCS*, 2(130), 2020.

[13] Karl Bringmann. Fine-grained complexity theory (tutorial). In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[14] Richard A Buckingham. The classical equation of state of gaseous helium, neon and argon. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 1938.

[15] Seth T Call, Dmitry Yu Zubarev, and Alexander I Boldyrev. Global minimum structure searches via particle swarm optimization. *Journal of computational chemistry*, 28(7):1177–1186, 2007.

[16] Richard Catlow and Scott M Woodley. Crystal structure prediction from first principles. *Nature materials*, 7(12):937, 2008.

[17] K. Cattell, F. Ruskey, J. Sawada, M. Serra, and C.R. Miers. Fast Algorithms to Generate Necklaces, Unlabeled Necklaces, and Irreducible Polynomials over GF(2). *Journal of Algorithms*, 37(2):267–282, 2000.

[18] Marcia R. Cerioli, Luerbio Faria, Talita O. Ferreira, and Fábio Protti. A note on maximum independent sets and minimum clique partitions in unit disk graphs and penny graphs: complexity and approximation. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 45(3):331–346, 2011.

[19] F. Chung, P. Diaconis, and R. Graham. Universal cycles for combinatorial structures. *Discrete Mathematics*, 110(1-3):43–59, 1992.

[20] Barry A Cipra. The ising model is np-complete. *SIAM News*, 33(6):1–3, 2000.

[21] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, pages 73–78, 2003.

[22] C. Collins, G. R. Darling, and M. J. Rosseinsky. The Flexible Unit Structure Engine (FUSE) for probe structure-based composition prediction. *Faraday Discussions*, 211(0):117–131, oct 2018.

[23] C Collins, MS Dyer, MJ Pitcher, GFS Whitehead, M Zanella, P Mandal, JB Claridge, GR Darling, and MJ Rosseinsky. Accelerated discovery of two crystal structure types in a complex inorganic phase field. *Nature*, 546(7657):280, 2017.

[24] N.G. de Bruijn. *Acknowledgement of priority to C. Flye Sainte-Marie on the counting of circular arrangements of 2^n zeros and ones that show each n-letter word exactly once.* EUT report. WSK, Dept. of Mathematics and Computing Science. Technische Hogeschool Eindhoven, 1975.

[25] Nicolaas Govert De Bruijn. A combinatorial problem. In *Proc. Koninklijke Nederlandse Academie van Wetenschappen*, volume 49, pages 758–764, 1946.

[26] David M Deaven and Kai-Ming Ho. Molecular geometry optimization with a genetic algorithm. *Physical review letters*, 75(2):288, 1995.

[27] M. S. Dyer, C. Collins, D. Hodgeman, P. A. Chater, A. Demont, S. Romani, R. Sayers, M. F. Thomas, J. B. Claridge, G. R. Darling, and M. J. Rosseinsky. Computationally assisted identification of functional inorganic materials. *Science*, 340(6134):847–852, 2013.

[28] James W Evans, V Fournée, C Ghosh, Cynthia J Jenks, Thomas A Lograsso, AR Ross, KJ Schnitzenbaumer, Patricia A Thiel, and B Unal. Nucleation and growth

of ag islands on fivefold al-pd-mn quasicrystal surfaces: Dependence of island density on temperature and flux. *Physical Review B*, 75(6):064205, 2007.

[29] A. E. Feldmann and D. Marx. The parameterized hardness of the k-center problem in transportation networks. *Algorithmica*, pages 1989–2005, 2020.

[30] Moti Frances and Ami Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

[31] H. Fredricksen and J. Maiorana. Necklaces of beads in k colors and k-ary de Bruijn sequences. *Discrete Mathematics*, 23(3):207–210, 1978.

[32] Harold Fredricksen and Irving J Kessler. An algorithm for generating necklaces of beads in two colors. *Discrete mathematics*, 61(2-3):181–188, 1986.

[33] CM Freeman, JM Newsam, SM Levine, and CRA Catlow. Inorganic crystal structure prediction using simplified potentials and experimental unit cells: Application to the polymorphs of titanium dioxide. *Journal of Materials Chemistry*, 3(5):531–535, 1993.

[34] Guilhem Gamard, Gwenaël Richomme, Jeffrey Shallit, and Taylor J. Smith. Periodicity in rectangular arrays. *Information Processing Letters*, 118:58–63, 2017. URL: `https://www.sciencedirect.com/science/article/pii/S0020019016301387`, `doi:https://doi.org/10.1016/j.ipl.2016.09.011`.

[35] M. Garey and D. Johnson. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[36] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD explorations newsletter*, 5(1):49–58, 2003.

[37] L. Gasieniec, J. Jansson, and A. Lingas. Efficient approximation algorithms for the Hamming Center Problem. In *SODA 1999*, pages 905–906, 1999.

[38] EN Gilbert and John Riordan. Symmetry types of periodic sequences. *Illinois Journal of Mathematics*, 5(4):657–665, 1961.

[39] Stefan Goedecker. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *The Journal of chemical physics*, 120(21):9911–9917, 2004.

[40] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[41] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[42] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics : a foundation for computer science*. Addison-Wesley, 1994.

[43] Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

[44] U. I. Gupta, D. T. Lee, and C. K. Wong. Ranking and unranking of B-trees. *Journal of Algorithms*, 4(1):51–60, mar 1983. doi:10.1016/0196-6774(83)90034-2.

[45] P. Hartman and J. Sawada. Ranking and unranking fixed-density necklaces and Lyndon words. *Theoretical Computer Science*, 791:36–47, 2019.

[46] Johan Håstad. Clique is hard to approximate within n-e. *Acta Mathematica*, 182(1):105–142, 1999.

[47] Dorit S Hochbaum. Various notions of approximations: Good, better, best and more. *Approximation algorithms for NP-hard problems*, 1997.

[48] Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, 33(3):533–550, 1986.

[49] V. Horan and B. Stevens. Locating patterns in the de Bruijn torus. *Discrete Mathematics*, 339(4):1274–1282, 2016.

[50] Wen-Lian Hsu and George L Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.

[51] G. Hurlbert and G. Isaak. On the de Bruijn Torus problem. *Journal of Combinatorial Theory, Series A*, 64(1):50–62, 1993.

[52] G. Hurlbert and G. Isaak. New constructions for De Bruijn tori. *Designs, Codes and Cryptography*, 6(1):47–56, 1995.

[53] G. H. Hurlbert, C. J. Mitchell, and K. G. Paterson. On the existence of de Bruijn Tori with two by two windows. *Journal of Combinatorial Theory. Series A*, 76(2):213–230, 1996.

[54] Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925.

[55] Yishan Jiao, Jingyi Xu, and Ming Li. On the k-closest substring and k-consensus pattern problems. In *Combinatorial Pattern Matching*, pages 130–144, 2004.

[56] Matthew Johnson, Barnaby Martin, George B Mertzios, and Daniël Paulusma. Report on bctcs & algouk 2019. *Bulletin of EATCS*, 2(128), 2019.

[57] S. Karim, J. Sawada, Z. Alamgir, and S. M. Husnine. Generating bracelets with fixed content. *Theoretical Computer Science*, 475:103–112, mar 2013. `doi:10.1016/j.tcs.2012.11.024`.

[58] Saira Karim, Joe Sawada, Zareen Alamgir, and SM Husnine. Generating bracelets with fixed content. *Theoretical Computer Science*, 475:103–112, 2013.

[59] Donald E. Knuth. *The Art of Computer Programming: Combinatorial Algorithms, Part 1.* Addison-Wesley Professional, 1st edition, 2011.

[60] T. Kociumaka, J. Radoszewski, and W. Rytter. Computing k-th Lyndon word and decoding lexicographically minimal de Bruijn sequence. In *Symposium on Combinatorial Pattern Matching*, pages 202–211. Springer International Publishing, 2014.

[61] S. Kopparty, M. Kumar, and M. Saks. Efficient indexing of necklaces and irreducible polynomials over finite fields. *Theory of Computing*, 12(1):1–27, 2016.

[62] J. K. Lanctot, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003.

[63] J Kevin Lanctot, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Information and Computation*, 185(1):41–55, 2003.

[64] Harry R. Lewis. Complexity of solvable cases of the decision problem for the predicate calculus. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 35–47, 1978. `doi:10.1109/SFCS.1978.9`.

[65] Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002.

[66] David C Lonie and Eva Zurek. Xtalopt: An open-source evolutionary algorithm for crystal structure prediction. *Computer Physics Communications*, 182(2):372–387, 2011.

[67] Andriy O. Lyakhov, Artem R. Oganov, and Mario Valle. How to predict very large and complex crystal structures. *Computer Physics Communications*, 181(9):1623 – 1632, 2010.

[68] Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. In *Annual International Conference on Research in Computational Molecular Biology*, pages 396–409. Springer, 2008.

[69] Alan L Mackay. Crystallography and the penrose pattern. *Physica A: Statistical Mechanics and its Applications*, 114(1-3):609–613, 1982.

[70] John Maddox. Crystals from first principles. *Nature*, 335(6187):201–201, 1988. `doi:10.1038/335201a0`.

[71] Maria Madonia. Two-dimensional codes. In *Conference on Computability in Europe*, pages 301–305. Springer, 2020.

[72] Martin Mareš and Milan Straka. Linear-time ranking of permutations. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *Algorithms – ESA 2007*, pages 187–193, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[73] Caroline Mellot-Draznieks, Stéphanie Girard, Gérard Férey, J. Christian Schön, Zeljko Cancarevic, and Martin Jansen. Computational design and prediction of interesting not-yet-synthesized structures of inorganic materials by using building unit concepts. *Chemistry – A European Journal*, 8(18):4102–4113, 2002.

[74] Wendy Myrvold and Frank Ruskey. Ranking and unranking permutations in linear time. *Information Processing Letters*, 79(6):281 – 284, 2001. URL: `http://www.sciencedirect.com/science/article/pii/S0020019001001417`, `doi:https://doi.org/10.1016/S0020-0190(01)00141-7`.

[75] A.R. Oganov. Crystal structure prediction: reflections on present status and challenges. *Faraday Discuss.*, 211(0):643–660, 2018.

[76] Artem R Oganov and Colin W Glass. Crystal structure prediction using ab initio evolutionary techniques: Principles and applications. *The Journal of chemical physics*, 124(24), 2006.

[77] Artem R Oganov, Chris J Pickard, Qiang Zhu, and Richard J Needs. Structure prediction drives materials discovery. *Nature Reviews Materials*, 4(5):331–348, 2019.

[78] J. M. Pallo. Enumerating, Ranking and Unranking Binary Trees. *The Computer Journal*, 29(2):171–175, feb 1986. URL: `https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/29.2.171`, `doi:10.1093/comjnl/29.2.171`.

[79] Jean Pannetier, J Bassas-Alsina, Juan Rodriguez-Carvajal, and Vincent Caignaert. Prediction of crystal structures from crystal chemistry rules by simulated annealing. *Nature*, 346(6282):343–345, 1990.

[80] D. Perrin. *Words*. Cambridge University Press, 2 edition, 1997.

[81] Chris J Pickard and RJ Needs. High-pressure phases of silane. *Physical review letters*, 97(4):045504, 2006.

[82] Chris J Pickard and RJ Needs. Ab initio random structure searching. *Journal of Physics: Condensed Matter*, 23(5):053201, 2011.

[83] David Pisinger. Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, 33(1):1–14, 1999.

[84] J. Piskorski, M. Sydow, and K. Wieloch. Comparison of string distance metrics for lemmatisation of named entities in polish. In *Language and Technology Conference*, pages 413–427, 2007.

[85] Ján Plesník. On the computational complexity of centers locating in a graph. *Aplikace matematiky*, 25(6):445–452, 1980.

[86] G. Recchia and M. M. Louwerse. A comparison of string similarity measures for toponym matching. In *SIGSPATIAL 2013*, pages 54–61, 2013.

[87] Z. Rotman and E. Eisenberg. Finite-temperature liquid-quasicrystal transition in a lattice model. *Phys. Rev. E*, 83:011123, Jan 2011. URL: `https://link.aps.org/doi/10.1103/PhysRevE.83.011123`, `doi:10.1103/PhysRevE.83.011123`.

[88] F. Ruskey, C. Savage, and T. Min Yih Wang. Generating necklaces. *Journal of Algorithms*, 13(3):414–430, 1992.

[89] F. Ruskey and J. Sawada. Efficient algorithm for generating necklaces with fixed density. *SIAM Journal on Computing*, 29(2):671–684, 1999.

[90] F. Ruskey and J. Sawada. Generating necklaces and strings with forbidden substrings. In *COCOON 2000*, volume 1858 of *Lecture Notes in Computer Science*, pages 330–339, 2000.

[91] C Flye Saint-Marie. Solution to question. *l'Intermediaire des Mathematiciens*, 48, 1894.

[92] J. Sawada and A. Williams. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *Journal of Discrete Algorithms*, 43:95–110, 2017.

[93] Joe Sawada. Generating bracelets in constant amortized time. *SIAM Journal on Computing*, 31(1):259–268, jan 2001. URL: `http://epubs.siam.org/doi/10.1137/S0097539700377037`, `doi:10.1137/S0097539700377037`.

[94] Joe Sawada. A fast algorithm to generate necklaces with fixed content. *Theoretical Computer Science*, 301(1-3):477–489, may 2003. `doi:10.1016/S0304-3975(03)00049-5`.

[95] Martin U Schmidt and Ulli Englert. Prediction of crystal structures. *Journal of the Chemical Society, Dalton Transactions*, 10:2077–2082, 1996.

[96] J Christian Schön and Martin Jansen. First step towards planning of syntheses in solid-state chemistry: determination of promising structure candidates by global optimization. *Angewandte Chemie International Edition in English*, 35(12):1286–1304, 1996.

[97] Toshihiro Shimizu, Takuro Fukunaga, and Hiroshi Nagamochi. Unranking of small combinations from large sets. *Journal of Discrete Algorithms*, 29:8

– 20, 2014. URL: `http://www.sciencedirect.com/science/article/pii/S1570866714000483`, `doi:https://doi.org/10.1016/j.jda.2014.07.004`.

[98] H. E. Stanley. Dependence of critical properties on dimensionality of spins. *Phys. Rev. Lett.*, 20:589–592, Mar 1968.

[99] Patrick Totzke and Michele Zito. Report on bctcs 2021. *Bulletin of EATCS*, 2(134), 2021.

[100] David J Wales and Jonathan PK Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[101] Yanchao Wang, Jian Lv, Li Zhu, Shaohua Lu, Ketao Yin, Quan Li, Hui Wang, Lijun Zhang, and Yassie ma. Materials discovery via calypso methodology. *Journal of physics. Condensed matter : an Institute of Physics journal*, 27:203203, 04 2015.

[102] L T Wille and J Vennik. Computational complexity of the ground-state determination of atomic clusters. *Journal of Physics A: Mathematical and General*, 18(8):L419–L422, jun 1985.

[103] S. G. Williamson. Ranking algorithms for lists of partitions. *SIAM Journal on Computing*, 5(4):602–617, 1976. `arXiv:https://doi.org/10.1137/0205039`, `doi:10.1137/0205039`.