Utrecht University
Faculty of Science
Department of Information and Computing Sciences

On the rank of Directed Hamiltonicity and beyond

Author:
Ioannis Katsikarelis

Supervisors:
Dr. Hans L. Bodlaender
Dr. Jesper Nederlof

Master Thesis: ICA-3869539

February 2014

Utrecht University

UTRECHT UNIVERSITY

Abstract

Faculty of Science
Department of Information and Computing Sciences

On the rank of Directed Hamiltonicity and beyond

by Ioannis KATSIKARELIS

Motivated by recent research making use of insights on the relationship between the optimal substructure of dynamic programming procedures and the rank of problem-specific matrices, we investigate the application of this approach in the context of Hamiltonicity.

In particular, after briefly introducing the setting, we determine the rank of the Directed Matching Connectivity matrix $\mathcal{D}_t$ that has rows/columns indexed by all perfect matchings on the complete digraph on t vertices, while an entry $\mathcal{D}_t[M_1, M_2]$ is 1, if $M_1 \cup M_2$ is a directed Hamiltonian cycle and 0 otherwise. We then provide a Monte Carlo algorithm that uses the row basis $\mathbf{X}_t$ for $\mathcal{D}_t$ to solve the DIRECTED HAMILTONIAN CYCLE problem on a given path decomposition of width $\mathtt{pw}$ in time $(2 + 2^{\frac{3}{2}})^{\mathtt{pw}}(n \cdot \mathtt{pw})^{\mathcal{O}(1)}$ via dynamic programming.

Subsequently, we study a generalized setting that considers partitions of elements into blocks of size k instead of perfect matchings only. After expanding the definitions of required concepts, we show how to construct sets of partitions $\mathbf{X}_{t,k}$ that generalize $\mathbf{X}_t$ for higher values of k and use them to obtain a non-trivial lower bound on the rank of the Partition Connectivity matrix $\mathcal{C}_{t,k}$, that itself extends previously considered matrices.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the most well-known problems in theoretical computer science and the mathematical field of graph theory is the Hamiltonian Cycle problem, that asks whether some given graph contains a Hamiltonian cycle, i.e. a cycle that visits each vertex exactly once. Both the cycles and the problem derive their name from W. R. Hamilton, the 19th-century Irish mathematician, who originally described the problem on dodecahedrons as a puzzle.

The Hamiltonian Cycle problem can be seen as a special case of the (likewise well-known) Traveling Salesman problem (TSP),[1] both being famously NP-complete. In fact, the Hamiltonian Cycle problem was included in the list of Karp's 21 NP-complete problems twice [25], formulated in regards to both directed and undirected graphs. In general, the number of possible vertex sequences that might provide solutions to the problem for some n-vertex input graph, could be as high as $n!$, making brute-force search algorithms inefficient. In addition, NP-completeness persists even if restrictive assumptions are made on the input graph, e.g. for undirected planar graphs of maximum degree three [17], or directed planar graphs with in/outdegree at most two [33].

The intractability of the Hamiltonian Cycle problem leads naturally to the application of alternative solution concepts, some of which are briefly discussed next. In the following chapters of this thesis, inspired by recent theoretical results introducing improved algorithms for the undirected formulation of the problem, we first present extensions of these results and an analogous algorithm for the directed version, and subsequently consider generalizations to the setting, in terms of connectivity capabilities between families of set partitions (akin to a hypergraph setting).

[1] Setting distances to 1 for every present edge and 2 between cities that are not adjacent, and verifying that the total distance is equal to the number of vertices gives the correspondence.

1.1 Previous work

Due to the importance of the Hamiltonian Cycle and Traveling Salesman problems (as well as their many variants), several algorithms and techniques have been devised to produce solutions for them, including the Lin-Kernighan heuristic [29], the Christofides approximation algorithm [11] and the polynomial time approximation scheme for Euclidean Traveling Salesman by Arora [1].

Important contributions making use of dynamic programming by Bellman [3], and also by Held and Karp [20], show that Traveling Salesman can be solved in $\mathcal{O}(n^2 2^n)$ time for any n-vertex input graph. This method considers subsets of the vertices and finds partial solutions (paths through these subsets) that visit each vertex in the subset once and avoid other vertices, using previously computed information to build concrete, global solutions. Their approach has been highly influential in later exact exponential time algorithms for intractable problems (see [16, Chapter 3]). Furthermore, this approach has even produced some improved algorithms in terms of space requirements, at the cost of (modestly) increased running times [2, 26].

One important question regarding the computational complexity of Hamiltonian Cycle and Traveling Salesman involves the identification of the smallest constant in the exponential part of the running time, or, in other words, whether either of these problems can be solved in $(2-\epsilon)^n n^{\mathcal{O}(1)}$ time, for some $\epsilon > 0$ [36]. The question was positively resolved for cubic graphs [15, 23], graphs of bounded degree (in broader terms) [6, 18], and claw-free graphs [10].

In [4], Björklund partially answered the question for the general case with a $1.657^n n^{\mathcal{O}(1)}$ time Monte Carlo algorithm, that also improves to a running time of $1.414^n n^{\mathcal{O}(1)}$ for bipartite graphs. The focus of his approach is on counting the number of Hamiltonian cycles and makes use of the inclusion-exclusion principle to reduce this problem to counting cycle covers through the computation of certain matrix determinants, subsequently answering the Hamiltonian Cycle problem. However, derandomizing this result, or generalizing it to consider edge weights (as in Traveling Salesman) without producing pseudo-polynomial dependencies on the weights seems like a demanding task.

Additionally, Cygan, Kratsch and Nederlof [14], using an alternative approach to decomposing Hamiltonian cycles and insights on the structure of certain matrices, presented a number of useful results, including $1.888^n n^{\mathcal{O}(1)}$ time, deterministic algorithms to compute the parity of the number of Hamiltonian cycles in undirected and directed bipartite graphs. This is an important problem in itself, and their algorithm was extended to construct Monte Carlo algorithms of the same running time for the decision version of

Hamiltonian Cycle for both undirected and directed bipartite graphs. As a corollary, a $1.1583^n n^{\mathcal{O}(1)}$ time Monte Carlo algorithm for the special case of cubic graphs is obtained as well, being (to our knowledge) the fastest algorithm for these graphs.

The algorithm of Björklund and Husfeldt for the computation of the parity of the number of Hamiltonian cycles in [5] has a running time of $1.619^n n^{\mathcal{O}(1)}$ and requires only polynomial space, without the bipartiteness restriction for the case of directed graphs. Contrary to that, the algorithm from [14] was instead fitting for the use of weights, allowing the application of the Isolation Lemma that led to the randomized algorithms for Hamiltonian Cycle. This approach is also utilized here and further explained in the following, while other results that stem from it (mostly relating to tree/path decompositions) are also discussed.

1.2 Our perspective

The classical dynamic programming algorithm of [3, 20] examines Hamiltonian cycles by concentrating on paths between certain vertices, keeping track of the set of vertices used in between, the essential information to characterize these paths. This leads to storing the endpoints and sets of visited vertices for all sub-paths, or $\mathcal{O}(n2^n)$ table entries. Alternatively, in [14], assuming an arbitrary ordering $e_1, \ldots, e_m$ of the set of edges, a Hamiltonian cycle H is decomposed in two parts $H_1 = \{e_1, \ldots, e_i\} \cap H$, and $H_2 = \{e_{i+1,\ldots,e_n}\} \cap H$, for every i. The required information to store for a set of partial solutions H_1 is then contained in the degrees of each vertex with respect to H_1 (being zero, one, or two) and the pairing in H_1 of vertices of degree one (being the endpoints of sub-paths). This characterization of partial solutions is implied by the use of dynamic programming on *tree/path decompositions*, the central method in [14].

Tree and path decompositions have become significant subjects of exploration in graph-theoretical research and the concepts of *treewidth/pathwidth* are two of the chief parameters in the analysis of related problems. The terms were introduced by Robertson and Seymour as part of their fundamental work on graph minors [35]. Informally, the treewidth/pathwidth of a graph is a measure of how tree/path-like the structure of the graph is. Interest in tree/path decompositions is justified by their enabling of efficient solution ideas for intractable problems (usually NP-hard), as applied to trees and paths, to be extended for the general case of input graphs. Given a tree/path decomposition of small width for some input graph, dynamic programming is applied on the decomposition to efficiently produce solutions to intractable problems, exploiting the decomposition's structure being in the form of a tree/path (see [7, 8]).

A significant theorem by Courcelle (from 1990) informally states that every graph property that can be defined in terms of monadic second-order logic, can be decided in linear time on graphs of bounded treewidth [12], in essence suggesting algorithms with running times in the $f(\mathtt{tw})n$ form, for any given n-vertex graph, assuming a tree decomposition of width $\mathtt{tw}$ for this graph is given as well.[2] For many such problems, effort to improve the actual functions $f(\mathtt{tw})$ led to several efficient algorithms with running times of the form $2^{\mathcal{O}(\mathtt{tw})}n^{\mathcal{O}(1)}$, especially for *local* problems: those whose solutions can be verified as such by some distributed algorithm separately operating on all neighborhoods of the input graph [19]. As an example, the well-known Vertex Cover and Independent Set problems can be solved in $2^{\mathcal{O}(\mathtt{tw})}n^{\mathcal{O}(1)}$ time [27, 32], via dynamic programming on a given tree decomposition.

Furthermore, assuming the Exponential Time Hypothesis (ETH), i.e. that CNF-SAT has no sub-exponential algorithm [21, 22], it can also be shown via Karp-reductions that there can be no sub-exponential algorithm for the above problems. Under the even stronger assumption of the Strong Exponential Time Hypothesis (SETH), the current algorithms would even be optimal with respect to polynomial jumps [30], meaning there can be no algorithms with running time $f(\mathtt{tw})^{1-\epsilon}n^{\mathcal{O}(1)}$ for $\epsilon > 0$, where $f(\mathtt{tw}) = 2^{\mathtt{tw}}$ for Independent Set and Max-Cut, while $f(\mathtt{tw}) = 3^{\mathtt{tw}}$ for Dominating Set.

On the other hand, Hamiltonian Cycle and Traveling Salesman do not belong in this category of local problems [19] and their exact complexity would thus require further investigation. Among results for other connectivity problems, Cygan et al. showed that Hamiltonian Cycle can be solved in $4^{\mathcal{O}(\mathtt{tw})}n^{\mathcal{O}(1)}$ time by a Monte Carlo algorithm [13], yet derandomizing these results and extending them to consider counting and weighted versions of the problem (without pseudo-polynomial dependencies on the weights) were still considered crucial issues.

This was essentially accomplished in [9], where certain matrix properties were studied in relation to the substructure of dynamic programming algorithms for connectivity problems. Entries for these matrices, whose rows and columns are indexed by partial solutions, are 1 if the corresponding partial solutions can be combined into a general solution for the problem and 0 otherwise. Keeping track of more partial solutions than the rank of each respective matrix would apparently imply redundancy, leading to deterministic $2^{\mathcal{O}(\mathtt{tw})}n^{\mathcal{O}(1)}$ time algorithms for several problems. Unfortunately, their approach did not provide a deterministic counterpart to the $4^{\mathtt{tw}}n^{\mathcal{O}(1)}$ time Monte Carlo algorithm for Hamiltonian Cycle.

[2] Due to the structural similarity between tree and path decompositions, these running times can also be expressed in terms of the pathwidth $\mathtt{pw}$, keeping the function on the number of vertices n linear.

These results directly influenced the work in [14], where a matrix of partial solutions for Hamiltonicity was studied, while gained insights on its structure were used to obtain many important results, some of which were previously discussed. The *Matching Connectivity Matrix* has rows and columns indexed by all perfect matchings on a specific number of vertices and after establishing its rank, a set of basis matchings was used in conjunction with their alternative way of decomposing a Hamiltonian cycle (as previously mentioned) to obtain a $(2+\sqrt{2})^{\mathtt{pw}}n^{\mathcal{O}(1)}$ time Monte Carlo algorithm for Hamiltonian Cycle. Moreover, this set of basis matchings was used as part of a reduction gadget to show that a substantial improvement to this running time would defy the Strong Exponential Time Hypothesis.

This alternative viewpoint concerning the anatomy of Hamiltonian cycles, as well as the rank-based approach to reducing the number of potentially useful partial solutions, towards application of dynamic programming on path decompositions is also adopted here. In particular, we look at two matrices related to the Matching Connectivity Matrix, and consequently by extension present a $(2+2^{\frac{3}{2}})^{\mathtt{pw}}(n\cdot\mathtt{pw})^{\mathcal{O}(1)}$ Monte Carlo algorithm for Directed Hamiltonian Cycle. We also look at matrices that generalize this method, with rows and columns indexed by partitions into larger blocks (instead of perfect matchings), with an outlook towards a broader understanding of the characteristics of connectivity properties.

This thesis is structured as follows: Chapter 2 provides formal definitions of terms and discusses techniques that will prove useful in later chapters. In Chapter 3, we investigate directed Hamiltonicity and present our dynamic programming algorithm on path decompositions, while Chapter 4 deals with generalizations of previously used concepts. Finally, Chapter 5 provides some conclusive remarks and discusses directions for future research.

Chapter 2

Preliminaries

In this chapter we give the background necessary for the results of the following chapters to be fully comprehensible, including definitions and statements of prior results.

2.1 Definitions

This section discusses notation and introduces the formal definitions of notions and terms used throughout this thesis. In general, standard graph notation is used: for a graph $G = (V, E)$, vertex and edge sets are denoted by $V(G)$ ($|V(G)| = n$) and $E(G)$, respectively.[1] For subsets $X, Y \subseteq V$, let $E(X, Y)$ denote the set of all edges with one endpoint in X and the other in Y. A *Hamiltonian cycle* is the edge set of a simple cycle that visits each vertex exactly once. A *cycle cover* is a set of edges $F \subseteq E$ such that every vertex of G is incident with exactly two edges of F, i.e. the edges form cycles, while in a *partial* cycle cover, every vertex is incident with at most two of the edges, i.e. the edges form paths and cycles.

A *perfect matching* of a graph G is a set of edges $W \subseteq E$ such that each vertex is incident with exactly one edge of W, while the union of any two perfect matchings in a graph forms a cycle cover of the graph (some cycles may only consist of two edges). In general, given some base set U (representing in our context a graph's vertices V), let $\Pi_2(U)$ denote the set of all perfect matchings of U. If U has no graph structure then $\Pi_2(U)$ includes all partitions into sets, or *blocks* of size two.[2] For two perfect matchings $M_1, M_2 \in \Pi_2(U)$, the *meet-operation* $\sqcap$ (borrowed from the partially ordered by refinement partition lattice) is used as in $M_1 \sqcap M_2 = \{U\}$ to express the fact that

[1] Definitions are given in terms of undirected graphs, but hold for the directed case as well (replacing *edges* with *arcs*), with any alterations explicitly noted, where required.

[2] More on generalizations like this can be found in Chapter 4.

the union of M_1 and M_2 is a Hamiltonian cycle (the meet-operation between M_1, M_2 results in getting the trivial partition $\{U\}$ into a single set).

Note that in directed settings, perfect matchings are similarly defined as partitions in blocks of size two, while the order of the two elements in each block determines the direction of each arc. In this case, for the union of two perfect matchings to produce a Hamiltonian cycle, additional requirements need to be met: each vertex has to appear first in the inner ordering of the block that the first matching assigns it to (the vertex is the origin of the arc), and second in the ordering of the block given by the other matching (the vertex is the endpoint of the other arc). In other words, in the graph resulting from the union of two perfect matchings every vertex is required to have exactly one incoming and one outgoing arc (one from each perfect matching).

Next follow the definitions of the central notions of pathwidth and path decompositions. Similar concepts were independently discovered in the 1970s and 1980s, but the dominant terms were both introduced by Robertson and Seymour in the early stages of their fundamental work on graph minors [34, 35].

Definition 2.1. A *path decomposition* of a graph $G = (V, E)$ is a path $\mathbb{P}$ in which every node x has an associated set of vertices $B_x \subseteq V$ (called a *bag*) such that $\bigcup B_x = V$ and the following properties hold:

1. For each edge $\{u, v\} \in E(G)$ there is a node x in $\mathbb{P}$ such that $u, v \in B_x$.
2. If $v \in B_x \cap B_y$ then $v \in B_z$ for all nodes z on the (unique) path from x to y in $\mathbb{P}$.

The *pathwidth* of $\mathbb{P}$ is the size of the largest bag minus one, and the pathwidth $\mathtt{pw}(G)$ of a graph G is the minimum pathwidth over all possible path decompositions of G.

Two related notions are those of *tree decompositions* and *treewidth*, where the nodes can form a tree instead of a single path. As these notions are not directly used here, their formal definitions are omitted.

Dynamic programming algorithms on path decompositions are commonly presented using the concept of *nice* path decompositions, introduced by Kloks [28]. The reason is their simplified structure, requiring bags of specific types only, while neighboring bags can only differ by at most one vertex, with no increase on the pathwidth. The extended version from [13] that utilizes *introduce edge bags* is used here.

Definition 2.2. A *nice path decomposition* is a path decomposition where the underlying path of nodes is ordered from left to right (the predecessor of any node is its left neighbor) and in which each bag is one of the following types:

- **First (leftmost) bag**: the bag associated with the leftmost node x is empty, i.e. $B_x = \emptyset$.
- **Introduce vertex bag**: an internal node x of $\mathbb{P}$ with predecessor y such that $B_x = B_y \cup \{v\}$ for some $v \notin B_y$. This bag is said to *introduce* v.
- **Introduce edge (arc) bag**: an internal node x of $\mathbb{P}$ labeled with an edge $\{u, v\} \in E(G)$ with one predecessor y for which $u, v \in B_x = B_y$. This bag is said to *introduce* $\{u, v\}$.
- **Forget bag**: an internal node x of $\mathbb{P}$ with one predecessor y for which $B_x = B_y \backslash \{v\}$ for some $v \in B_y$. This bag is said to *forget* v.
- **Last (rightmost) bag**: the bag associated with the rightmost node x is empty, i.e. $B_x = \emptyset$.

Note that any given path decomposition of pathwidth $\mathtt{pw}$ can be transformed into a nice path decomposition in time $n \cdot \mathtt{pw}^{\mathcal{O}(1)}$, without increasing its width.

2.2 The Matching Connectivity matrix

As previously mentioned, Cygan et al. introduced the *Matching Connectivity matrix* $\mathcal{H}_t$ and used its properties to obtain a variety of significant results in [14]. Due to the affinity of the concepts discussed in Chapter 3 with matters relating to the Matching Connectivity matrix and its characteristics, the latter are presented in this section.

Definition 2.3 (Matching Connectivity matrix [14])**.** For an even integer $t \geq 2$, the *Matching Connectivity matrix* $\mathcal{H}_t$ is a matrix that has rows and columns both labeled by all perfect matchings on the complete graph K_t on t vertices; an entry $\mathcal{H}_t[M_1, M_2]$ is 1, if $M_1 \cup M_2$ is a Hamiltonian cycle and 0 otherwise.

For $t = 2$ there is only one perfect matching and one trivial Hamiltonian cycle (for $t \geq 4$ no overlapping edges are allowed in a Hamiltonian cycle). For $t = 4$, there are 3 perfect matchings and $\mathcal{H}_3$ is the complement of the 3×3 identity matrix, while the 15×15 matrix $\mathcal{H}_6$ is shown in Table 2.1. In general, the dimension of $\mathcal{H}_t$ is given by:

$$\frac{t!}{(\frac{t}{2})! \cdot 2^{\frac{t}{2}}} \times \frac{t!}{(\frac{t}{2})! \cdot 2^{\frac{t}{2}}}$$

Despite the sizable dimension of $\mathcal{H}_t$, its rank was shown to be $2^{t/2-1}$ using a family $\mathbf{X}_t$ of perfect matchings whose corresponding rows (or columns) of $\mathcal{H}_t$ form a basis over the

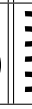
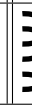
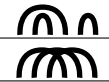
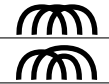
Nr.		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	LC
																	
1		0	0	0	0	1	1	0	1	1	1	1	0	1	1	0	1
2		0	0	0	1	0	1	1	1	0	0	1	1	1	0	1	2
3		0	0	0	1	1	0	1	0	1	1	0	1	0	1	1	1+2
4		0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	4
5		1	0	1	0	0	0	1	0	1	0	1	1	1	1	0	5
6		1	1	0	0	0	0	1	1	0	1	1	0	0	1	1	4+5
7		0	1	1	0	1	1	0	0	0	0	1	1	0	1	1	1+4
8		1	1	0	1	0	1	0	0	0	1	0	1	1	1	0	2+4+5
9		1	0	1	1	1	0	0	0	0	1	1	0	1	0	1	1+2+5
10		1	0	1	1	0	1	0	1	1	0	0	0	0	1	1	2+5
11		1	1	0	0	1	1	1	0	1	0	0	0	1	0	1	1+4+5
12		0	1	1	1	1	0	1	1	0	0	0	0	1	1	0	1+2+4
13		1	1	0	1	1	0	0	1	1	0	1	1	0	0	0	1+2+4+5
14		1	0	1	0	1	1	1	1	0	1	0	1	0	0	0	1+5
15		0	1	1	1	0	1	1	0	1	1	1	0	0	0	0	2+4

TABLE 2.1: The matrix $\mathcal{H}_6$. For $U = \{0, 1 \ldots, 5\}$, the first matching (first row/column) equals $\{\{0,1\},\{2,3\},\{4,5\}\}$. The last column depicts the linear combinations from the row basis set $\mathbf{X}_t = \{1,2,4,5\}$ (Definition 2.4). Table from [14].

field GF(2). Assuming an ordering on the vertices (considering $U_t := \{0, 1, \ldots, t-1\}$), all matchings in $\mathbf{X}_t$ only allow edges with endpoints at a distance at most 3 from each other, with respect to the ordering. More formally, for a perfect matching $M \in \Pi_2(U)$, let $\alpha_M : U \to U$ be a function that maps each element of U to its partner in M, i.e. $\alpha_M(i) = j$, if and only if $\{i,j\} \in M$. Then the set $\mathbf{X}_t$ is defined as follows:

Definition 2.4 (Basis set $\mathbf{X}_t$, [14])**.** Let ϵ denote the empty string, $X(2,\epsilon) := \{\{0,1\}\}$ and $\mathbf{X}_2 := \{X(2,\epsilon)\}$. Also let $t \geq 4$ be an even integer and a be a bit-string of length $\frac{t}{2} - 2$. The perfect matchings $X(t, a0)$ and $X(t, a1)$ of $U_t = \{0, \ldots, t-1\}$ are defined as follows:

$$X(t,a1) := X(t-2,a) \cup \{\{t-2, t-1\}\},$$
$$X(t,a0) := (X(t-2,a) \backslash \{\{t-3, a(t-3)\}\}) \cup \{\{t-2, a(t-3)\}, \{t-3, t-1\}\}.$$

Using the shorthand $X(a)$ for $X(2|a|+2, a)$, since the size t of the base set U_t is directly determined a, let $\mathbf{X}_t$ be the set of all perfect matchings $X := X(t,a)$ for any bit-string a of length $\frac{t}{2} - 1$.

The intuition behind the rather technical definition above is the following. Consider a grouping of the vertices as $0|1,2|3,4|\ldots|t-3,t-2|t-1$. The set $\mathbf{X}_t$ consists of all matchings whose edges only connect vertices from adjacent groups (crossing exactly one vertical line), corresponding to the possible bit-strings on $t/2-1$ bits: the first bit in the sequence (being 0 or 1) determines the connection of the first vertex (vertex 0) to either the first vertex of the second group (vertex 1), or the second (vertex 2). The second bit in the sequence then determines the connection of the other vertex of the second group (that the first bit did not connect to vertex 0) with either the first or the second vertex of the third group (vertex 3 or 4). This applies to the whole sequence, with every bit connecting the vertex that was not connected by the previous bit to one of the vertices of the next group. Since the last edge will always go to vertex $t-1$, only $t/2-1$ bits are required, fixing the size of $\mathbf{X}_t$ equal to $2^{t/2-1}$. In Table 2.1, matchings $1, 2, 4$ and 5 comply with the adjacent group restriction and can be seen to be exactly all such choices for connection, corresponding also to the 4 possible bit-strings on 2 bits.

The following theorem from [14, Corollary 3.5] gives the rank of the Matching Connectivity matrix as being equal to the size of $\mathbf{X}_t$, since the set of rows (or columns) labeled by matchings in $\mathbf{X}_t$ form a basis for $\mathcal{H}_t$. The proof is omitted here.

Theorem 2.5 ([14])**.** *The set $\mathbf{X}_t$ is a row basis for the Matching Connectivity matrix $\mathcal{H}_t$ and the rank of $\mathcal{H}_t$ over* GF(2) *is $2^{t/2-1}$ for all even integers $t \geq 2$.*

Since it will be of use in the next chapter, here follows one helpful theorem concerning the *block diagonal* form of a matrix and its rank.

Definition 2.6. A *block diagonal* matrix $\mathbf{A}$ is a square and block matrix, with square matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$ as its main diagonal blocks, such that the off-diagonal blocks are all zero matrices, i.e. it is the direct sum: $\mathbf{A} = \mathbf{A}_1 \oplus \mathbf{A}_2 \oplus \cdots \oplus \mathbf{A}_n$.

Theorem 2.7. *The rank of block diagonal matrix $\mathbf{A} = \mathbf{A}_1 \oplus \mathbf{A}_2 \oplus \cdots \oplus \mathbf{A}_n$ is equal to the sum of the ranks of its block sub-matrices:* $\operatorname{rk}(\mathbf{A}) = \sum_{i=1}^{n} \operatorname{rk}(\mathbf{A}_i)$.

2.3 Isolation Lemma

The Isolation Lemma [31] is an important result that has proven advantageous in the design of randomized algorithms as the one presented in Chapter 3, through the employment of random weights (in a particular range) leading to the identification of unique solutions with non-negligible probability.

First, for some function $\omega : U \to \{1, \ldots, N\}$ (where $N \in \mathbb{Z}^+$) assigning weights to members of a universe U in the range $\{1, \ldots, N\}$, let $\omega(S) = \sum_{e \in S} \omega(e)$ for subsets $S \subseteq U$. Also, let the probability of some event occurring be denoted by $\texttt{prob}[\cdot]$.

Definition 2.8. A function $\omega : U \to \mathbb{Z}$ *isolates* a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$.

Lemma 2.9 (Isolation Lemma, [31])**.** *Let $\mathcal{F} \subseteq 2^U$ be a set family over a universe U with $|\mathcal{F}| > 0$. For each $u \in U$, choose a weight $\omega(u) \in \{1, 2, \ldots, N\}$ uniformly and independently at random. Then $\texttt{prob}[\omega$ isolates $\mathcal{F}] \geq 1 - |U|/N$.*

Chapter 3

Directed Hamiltonicity

This chapter discusses the Directed Hamiltonian Cycle problem and introduces a new randomized algorithm that solves the problem on a graph G with a given path decomposition of width $\mathtt{pw}$ via dynamic programming in $(2+2^{\frac{3}{2}})^{\mathtt{pw}}(n\cdot\mathtt{pw})^{\mathcal{O}(1)}$ time. First, a matrix is defined and its rank investigated, a basis for which is subsequently utilized in the dynamic programming formulation.

3.1 The rank of Directed Hamiltonicity

In this section we investigate the rank of the *Directed* Matching Connectivity matrix $\mathcal{D}_t$, the matrix analogous to $\mathcal{H}_t$ for the case of directed graphs.

First, we introduce the *Bipartite* Matching Connectivity matrix $\mathcal{B}_t$ that will prove useful in analyzing the structure of $\mathcal{D}_t$.

Definition 3.1. For an even integer $t \geq 2$ the *Bipartite Matching Connectivity matrix* $\mathcal{B}_t$ is a matrix that has rows and columns both labeled by all perfect matchings on the complete balanced bipartite graph $K_{\frac{t}{2},\frac{t}{2}}$ on t vertices; an entry $\mathcal{B}_t[M_1, M_2]$ is 1, if $M_1 \cup M_2$ is a Hamiltonian cycle and 0 otherwise.

The dimension of $\mathcal{B}_t$ is $(\frac{t}{2})! \times (\frac{t}{2})!$ in general. As was the case with $\mathcal{H}_t$, for $t = 2$ there is only one perfect bipartite matching and one trivial Hamiltonian cycle, while in all cases where $t \geq 4$ no overlapping edges are allowed for an entry of $\mathcal{B}_t$ to be equal to 1. For $t = 4$, there are two perfect matchings and $\mathcal{B}_4$ is the complement of the 2×2 identity matrix. The 6×6 matrix $\mathcal{B}_6$ is shown in Table 3.1.

Next, we formally define the Directed Matching Connectivity matrix $\mathcal{D}_t$, followed by the formal resolution of its rank via Lemma 3.3 and Theorem 3.4.

Nr.		1	2	3	4	5	6
		$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$	$\rightleftarrows$
1	$\rightleftarrows$	0	0	0	1	1	0
2	$\rightleftarrows$	0	0	1	0	0	1
3	$\rightleftarrows$	0	1	0	0	0	1
4	$\rightleftarrows$	1	0	0	0	1	0
5	$\rightleftarrows$	1	0	0	1	0	0
6	$\rightleftarrows$	0	1	1	0	0	0

TABLE 3.1: The Bipartite Matching Connectivity matrix $\mathcal{B}_6$.

Definition 3.2. For an even integer $t \geq 2$ the *Directed Matching Connectivity matrix* $\mathcal{D}_t$ is a matrix that has rows and columns both labeled by all perfect matchings on the complete digraph on t vertices; an entry $\mathcal{D}_t[M_1, M_2]$ is 1, if $M_1 \cup M_2$ is a directed Hamiltonian cycle and 0 otherwise.

Lemma 3.3. *For all even integers $t \geq 2$, the rank of matrix $\mathcal{B}_t$ is $2^{\frac{t}{2}-1}$ over* GF(2).

Proof. Let $K_{\frac{t}{2},\frac{t}{2}} = (V = S \cup T, A)$ be the complete balanced bipartite graph on t vertices and assume an arbitrary ordering of the vertices: $S = \{s_1, s_2, \ldots, s_{\frac{t}{2}}\}$ and $T = \{t_1, t_2, \ldots, t_{\frac{t}{2}}\}$. Let $\pi := (s_1, t_1, t_2, s_2, s_3, t_3, t_4, s_4, s_5, \ldots)$, i.e. π is defined as the complete order on the vertices of both S and T that begins with the first vertex of S, followed by the first two vertices of T, alternatingly followed by subsequent couples of vertices from S and T.

Rearranging the vertices of $K_{\frac{t}{2},\frac{t}{2}}$ according to π makes it easy to see that $\mathcal{B}_t$ is in fact a sub-matrix of $\mathcal{H}_t$, with all matchings from $\mathbf{X}_t$ also present in $\mathcal{B}_t$ and by Theorem 2.5, forming a row basis here as well.[1] This gives an upper bound on the rank of $\mathcal{B}_t$ as being lower or equal to $2^{\frac{t}{2}-1}$, the rank of $\mathcal{H}_t$. Since $\mathcal{B}_t$ includes all matchings from $\mathbf{X}_t$, the corresponding rows/columns of which induce an identity sub-matrix in both $\mathcal{H}_t$ and $\mathcal{B}_t$, the rank of $\mathcal{B}_t$ can be no lower than $|\mathbf{X}_t| = 2^{\frac{t}{2}-1}$. Combining the upper and lower bound exactly determines the rank of $\mathcal{B}_t$ as $2^{\frac{t}{2}-1}$. ■

Theorem 3.4. *For all even integers $t \geq 2$, the rank of the Directed Matching Connectivity matrix $\mathcal{D}_t$ is $\binom{t}{\frac{t}{2}} \cdot 2^{\frac{t}{2}-1} < 2^{\frac{3}{2}t-1}$ over* GF(2).

Proof. Consider an arbitrary perfect matching M on the complete digraph on t vertices, i.e. a choice of $t/2$ directed arcs, such that each vertex is incident to exactly one of them. Irregardless of the specific choice of arcs, exactly half of the vertices will have an arc

[1] Matrix $\mathcal{B}_t$ is in fact made up of those matchings from $\mathcal{H}_t$ that do not violate bipartiteness with respect to π. As an example, matching 1 (first row/column, Table 2.1) from $\mathcal{H}_6$ is equivalent to matching 1 from $\mathcal{B}_6$ (Table 3.1), matching 2 from $\mathcal{H}_6$ is equivalent to matching 6 from $\mathcal{B}_6$ and matching 4 from $\mathcal{H}_6$ is equivalent to matching 2 from $\mathcal{B}_6$.

leading into them and the other half will have an arc originating from them, leading to a characterization of the vertices as *sources* or *sinks* and a partition (cut) of the set of all vertices U into two sets S and T.

For an entry $\mathcal{D}_t[M_1, M_2]$ to be 1, the following should hold: the implied sets of sources and sinks of M_1 and M_2 should be inverted ($S_1 = T_2, S_2 = T_1$) and no arcs should overlap between the two matchings. In all other cases, it is $\mathcal{D}_t[M_1, M_2] = 0$ by Definition 3.2.

Of the 2^t possible partitions of U into two sets S and T, the ones where $|S| = |T| = t/2$ define balanced bipartite graphs (note that also $S \cup T = U$ and $S \cap T = \emptyset$). The number of these partitions is given by:

$$\binom{t}{\frac{t}{2}} = \frac{t!}{(\frac{t}{2}!)^2}$$

For all even integers $t \geq 2$, it is $\binom{t}{\frac{t}{2}} < 2^t$. Retaining only the rows of $\mathcal{D}_t$ that are in agreement with any one of these partitions and the columns that correspond to the inverse partition (where sources are sinks and vice versa) will result in a matrix that is an exact copy of $\mathcal{B}_t$, as all perfect matchings are present in both cases and direction is subsumed by (implied) bipartiteness.

In fact, since all other entries are set to 0, the matrix $\mathcal{D}_t$ can be seen to be of the block-diagonal form (Definition 2.6) with $\binom{t}{\frac{t}{2}}$ copies of $\mathcal{B}_t$ as the blocks on the diagonal (by appropriately rearranging its rows/columns). Since, by Lemma 3.3 the rank of $\mathcal{B}_t$ is $2^{\frac{t}{2}-1}$, an application of Theorem 2.7 gives the claimed rank of $\mathcal{D}_t$. ■

3.2 Directed Hamiltonian Cycle on path decompositions

In this section, the dynamic programming algorithm for Directed Hamiltonian Cycle is introduced, inspired by the algorithm for the undirected version of the problem presented in [14].

Within our scope of dynamic programming on path decompositions, a Hamiltonian cycle H in a directed graph G can be analyzed in the following way: given some arbitrary ordering $a_1, a_2, \ldots, a_m$ on the set of arcs A, a Hamiltonian cycle H is decomposed for every i into $H_1 = \{a_1, \ldots, a_i\} \cap H$ and $H_2 = \{a_{i+1}, \ldots, a_m\} \cap H$. For some path decomposition (sequence of bags), partial solutions are sets of directed paths such that all vertices before the current bag are internal in some path and vertices in the current bag can be internal, unused, or endpoints of some path.

This implies that the essential information to store for a set of partial solutions H_1 are the degrees (0,1,2) of each vertex with respect to H_1 and the pairing in which vertices

of degree 1 are connected by H_1, for all partitions into endpoints, internal and unused vertices. Naturally, vertices of degree 2 are required to have one incoming and one outgoing arc for the formation of a directed path, while vertices of only indegree 1 can be paired with vertices of only the same outdegree. This pairing is in fact given by some perfect matching on the set of endpoints (bipartite).

The number of partial solutions allowed by this view does not lead to efficient computation, however, calling for an alternate approach: instead of storing all possible perfect matchings on the sets of endpoints for all partitions of the vertices into endpoints, internal and unused, only the aggregate connectivity information of all matchings (a "fingerprint") is stored. Fixing an ordering on the vertices, we only store for each perfect matching of the corresponding family $\mathbf{X}_t$ (rearranged according to the complete order π) the *number* of partial solutions that give a single cycle together with this matching modulo two, since the basis only works over the field GF(2). This allows us to compute the number of global solutions modulo two, and the existence of a unique Hamiltonian cycle of *minimum weight* is then guaranteed by an application of the Isolation Lemma.

In particular, the problem to be solved by dynamic programming on a path decomposition is the following: given a digraph $G = (V, A)$ along with a nice path decomposition of pathwidth `pw` and non-negative arc weights $\omega : A \to \{1, \ldots, \omega_{\max}\}$, the task is to compute for each $\omega^* \in \{1, \ldots, n \cdot \omega_{\max}\}$ the *parity* of the number of directed Hamiltonian cycles of G with weight exactly ω^*. The dynamic programming algorithm applied to the given path decomposition (viewed as a sequence of bags, ordered from left to right), for all partitions of the vertices in each bag into endpoints (further partitioned based on in/outdegree), internal and unused, takes a basis for the perfect matchings on the endpoints (bipartite) and computes the number of partial solutions that are consistent with each perfect matching (their union results in a single cycle), storing the parity of this number for each bag.

First, our dynamic programming algorithm requires one arc incident on a vertex v of degree at most `pw` to be "guessed" for a-priori inclusion in the Hamiltonian cycle. The vertex itself is chosen based on the structure of the given path decomposition, and "guessing" one of its arcs implies repetition of the algorithm's procedure with a different arc chosen each time.[2]

In a nice path decomposition, the introduce vertex bag that appears further to the right in the sequence can only introduce a vertex v of degree at most `pw`, since all neighbors of v need to appear in the same bag and no other possible neighbors can be introduced later in the sequence. The remaining bags on the right can then be freely rearranged as long as no vertex is forgotten before all its arcs have been introduced. Let (u, v) be the

[2]Due to the upper bound of `pw` on the degree of the vertex, the number of repetitions is also bounded.

selected arc incident on v (not necessarily incoming) and reorder the sequence such that the last bags are:

1. Introduce arc (u, v) bag, with current vertex set $\{u, v\}$
2. Forget vertex u bag
3. Forget vertex v bag
4. Last bag

The algorithm can then skip bags 2,3, and 4. and at bag 1. compute the parity of the number of partial cycle covers that would form a directed Hamiltonian cycle of weight ω^* (for some choice of such total weight) if arc (u, v) was included as well.

Given an arbitrary ordering of the vertices $V = \{v_1, \ldots, v_n\}$ of digraph $G = (V, A)$, let $\omega(v_i, v_j)$ denote the weight of any arc (v_i, v_j). Dynamic programming is applied to the given nice path decomposition, operating from left to right, until the introduce arc bag of the arc chosen at the start is reached, as explained above. For every bag, with vertex set B, table entries $d[B_0, B_1^+, B_1^-, B_2, \omega, M]$ (defined below) are computed for all partitions of the bag $B = B_0 \cup B_1 \cup B_2$, further partitions of the set of endpoints $B_1 = B_1^+ \cup B_1^-$ into sources B_1^+ and sinks B_1^- (where $|B_1^+| = |B_1^-| = \frac{|B_1|}{2}$), all integers $\omega \in \{0, \ldots, n \cdot \omega_{\max}\}$, and all perfect matchings M from a basis for B_1, according to Definition 2.4 (based on a standard arbitrarily fixed ordering induced from V) and rearranged according to order π to represent perfect bipartite matchings (as in the proof of Lemma 3.3).

Definition 3.5. Table entries $d[\cdot]$ contain the parity of the number of partial cycle covers C of the graph induced by all vertices that appear in the current and previous bags and all arcs introduced so far, such that:

a. $C \cup M$ is a single cycle.

b. The total weight of arcs in C is ω.

c. In C, vertices in B_0 have degree 0, vertices in B_2 have degree 2 with one incoming and one outgoing arc, vertices in B_1^+ have outdegree 1 and indegree 0, and vertices in B_1^- have indegree 1 and outdegree 0.

d. Vertices that only appear to the left of the current bag are denoted by B_l and have degree 2 with one incoming and one outgoing arc.

If C satisfies requirements b., c. and d. above, then C is called a $(B_0, B_1^+, B_1^-, B_2, B_l, \omega)$-*cycle cover*. If all requirements are met and the union of C with M is a single directed Hamiltonian cycle, then C is called a $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-*cycle cover*.

In the dynamic programming procedure described next, when a new arc is introduced in the path decomposition, a different representation of the same information regarding the basis set of perfect matchings for the set of endpoints B_1, based on the difference in orderings (giving rise to the basis set itself) is required to be efficiently computed. The following lemma from [14] that enables this change is applied by letting $\mathcal{C}$ be the set of all $(B_0, B_1^+, B_1^-, B_2, B_l, \omega)$-cycle covers and $S = B_1$. The proof is omitted here as it is directly obtained via the proof of [14, Lemma 4.1], since the basis set is still $\mathbf{X}_t$, with the implied rearrangement according to π causing no discrepancies.

Lemma 3.6 ([14])**.** *Let $\mathcal{C}$ denote an arbitrary set of partial cycle covers and $\mathbf{X}$ denote a family of basis matchings on some totally ordered set $S = \{v_0, \ldots, v_{t-1}\}$ of vertices of even size. Furthermore, for each $M \in \mathbf{X}$, let $T[M]$ denote the number of partial cycle covers $C \in \mathcal{C}$ such that $M \cup C$ is a single cycle. For any $\mathbf{X}'$ with respect to any other ordering of S, the corresponding values $T'[M']$ for $M' \in \mathbf{X}'$ can be computed in time $2^{t/2-1}t^{\mathcal{O}(1)}$.*

The algorithm operates on the sequence of bags of the path decomposition using the table $d'[\cdot]$ of the previous bag to compute the table $d[\cdot]$ of the current bag. Let d'_{uv} and d_{uv} denote the corresponding tables after the change of ordering and basis brings vertices u and v at the end of the total order. The dynamic programming follows.

First bag: The vertex set B of the first bag is empty and the only entry is trivial:

$$d[\emptyset, \emptyset, \emptyset, 0, \emptyset] \equiv 0.$$

Introduce vertex bag: The vertex set of the current bag is B, introducing vertex v, while that of the previous bag is $B\backslash\{v\}$, with any vertices occurring only on the left being the same for both bags. No arcs incident on v could be used in a partial cycle cover in the subgraph induced by the vertices of the current and previous bags, which gives:

$$d[B_0, B_1^+, B_1^-, B_2, \cdot, \cdot] \equiv 0,$$

for all partitions $B = B_0 \cup B_1 \cup B_2$, with $B_1 = B_1^+ \cup B_1^-$ as described above and $v \in B_1 \cup B_2$. For partitions where $v \in B_0$, irregardless of the weight ω and basis matching M, the required number is already computed in the previous bag's entry:

$$d[B_0, B_1^+, B_1^-, B_2, \omega, M] \equiv d'[B_0\backslash\{v\}, B_1^+, B_1^-, B_2, \omega, M].$$

Forget bag: The vertex set of the current bag is B, "forgetting" vertex v, while that of the previous bag is $B \cup \{v\}$. Since B_l denotes the set of vertices occurring only

left of the current bag, $B_l \backslash \{v\}$ denotes the set of vertices occurring only left of the previous bag. For any partition $B = B_0 \cup B_1 \cup B_2$, further partitioning $B_1 = B_1^+ \cup B_1^-$, matching M from the basis of B_1 (induced by a standard ordering) and weight ω, if C is a $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle cover, then v must have one incoming and one outgoing arc in C. Furthermore, the number of $(B_0, B_1^+, B_1^-, B_2 \cup \{v\}, B_l \backslash \{v\}, \omega, M)$-cycle covers C' is stored modulo two in the previous bag. These are also $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle covers. In both cases, it is:

$$d[B_0, B_1^+, B_1^-, B_2, \omega, M] \equiv d'[B_0, B_1^+, B_1^-, B_2 \cup \{v\}, \omega, M].$$

Introduce arc bag: The vertex set of the current bag is B, introducing arc (u, v),[3] where $u, v \in B$. The current and previous bags share the same vertex set B and set B_l of vertices occurring only to the left, yet partial solutions of the previous bag do not include the arc (u, v). As mentioned above, the table $d'_{uv}[\cdot]$ is computed from $d'[\cdot]$, using Lemma 3.6 for the change of basis so that u and v are the last two vertices in the ordering. The table $d_{uv}[\cdot]$ is subsequently computed from $d'_{uv}[\cdot]$ and afterwards transformed to $d[\cdot]$ by another application of Lemma 3.6.

In particular, for some arbitrary partition $B = B_0 \cup B_1 \cup B_2$, further partition $B_1 = B_1^+ \cup B_1^-$, integer weight ω and perfect matching M from a basis for B_1 with modified ordering so that u and v are the last two elements, the table $d_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M]$ needs to be computed from $d'_{uv}[\cdot]$. The possible cases are the following:

I. $u \in B_0$ **or** $v \in B_0$**:** Since no $(B_0, B_1^+, B_1^-, B_2, \omega)$-cycle cover, or perfect matching M on B_1 can contain the arc (u, v), it is for all basis matchings M:

$$d_{uv}[B_0, B_1^+, B_1^-, \omega, M] \equiv d'_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M]$$

II. $u \in B_1^+$ **and** $v \in B_1^-$**:** If the perfect matching M makes use of the arc (u, v), then no $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle cover can do the same, as that would result in a trivial cycle. The parity of the number of $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle covers that avoid the arc (u, v) is already stored in $d'_{uv}[\cdot]$.

If $(u, v) \notin M$, then $(p, u), (v, q) \in M$, for some vertices $p \in B_1^+$ and $q \in B_1^-$. Again, the parity of the number of $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle covers that avoid the arc (u, v) is already stored in $d'_{uv}[\cdot]$, letting us focus on $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle covers that do contain (u, v).

[3] This direction from u to v is actually an assumption, since after re-arranging according to π, direction from u to v or vice-versa depends on the parity of $\frac{|B_1|}{2}$, here assumed to be odd. The symmetrical nature of both cases leads to no loss of generality.

Let C be such a cycle cover, meaning that the union $C \cup M$ would contain a cycle $S = (x_1, \ldots, x_r, p, u, v, q, x_1)$. Since our computation of d_{uv} needs to come from some stored values for d'_{uv}, the intuition here is to modify the partial solutions to remove (u, v) from them, while keeping track of the modifications and mapping them to previously computed partial solutions for which the table entries are already stored. To that end, let $C' := C \backslash \{(u, v)\}$ and $M' := (M \backslash \{(p, u), (v, q)\}) \cup \{(p, q)\}$, i.e. C' has (u, v) missing and M' has the group $(p, u), (v, q)$ contracted into (p, q). One easily verifies that C' is a $(B_0 \cup \{u, v\}, B_1^+ \backslash \{u\}, B_1^- \backslash \{v\}, B_2, B_l, \omega - \omega(u, v))$-cycle cover and that $C' \cup M'$ will contain a cycle $S' = (x_1, \ldots, x_r, p, q, x_1)$, with any possible other cycles being the same as in $C \cup M$. This implies that C will be a $(B_0, B_1^+, B_1^-, B_2, B_l, \omega, M)$-cycle cover if and only if C' is a $(B_0 \cup \{u, v\}, B_1^+ \backslash \{u\}, B_1^- \backslash \{v\}, B_2, B_l, \omega - \omega(u, v), M')$-cycle cover.

In the new ordering of B_1, u and v are placed at the end and thus removing them will keep the ordering of the other vertices intact. Now, the existence of arcs $(p, u), (v, q) \in M$ and the structural restrictions on the basis set for B_1 give two possibilities for the latter part of the ordering of B_1: it either ends with 1) $\ldots, p, q, u, v$ or with 2) $\ldots, p, r, q, u, v$. For some bitstring a, if 1) is the case, then the ordering for $B_1 \backslash \{u, v\}$ ends with $\ldots, p, q$ and $M = X(|B_1|, a10)$, $M' = X(|B_1| - 2, a0)$. If 2) is the case, then the ordering for $B_1 \backslash \{u, v\}$ ends with $\ldots, p, r, q$ and $M = X(|B_1|, a00)$, $M' = X(|B_1| - 2, a0)$. The perfect matching M' is thus in the basis for $B_1 \backslash \{u, v\}$, its corresponding bitstring being the same as that for M with the last bit missing. In addition to the cycle covers that do not use (u, v), this gives:

$$d_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M] \equiv d'_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M] + d'_{uv}[B_0 \cup \{u, v\}, B_1^+ \backslash \{u\}, B_1^- \backslash \{v\}, B_2, \omega - \omega(u, v), M'].$$

III. $u \in B_1^+$ **and** $v \in B_2$**:** In this case it is $(u, v) \notin M$ and $(p, u) \in M$ for some $p \in B_1^-$. The contribution of $d'_{uv}[\cdot]$ will be used here as well, for all $(B_0, B_1^+, B_1^-, B_l, \omega, M)$-cycle covers that do not include (u, v). In case some $(B_0, B_1^+, B_1^-, B_2, B_l, \omega)$-cycle cover C makes use of the arc (u, v), then the union $C \cup M$ will contain a cycle $S = (x_1, \ldots, x_r, p, u, v, x_1)$. In a similar manner as for the previous case, let $C' := C \backslash \{(u, v)\}$ and $M' := (M \backslash \{(p, u)\}) \cup \{(p, v)\}$, with C' being a $(B_0 \cup \{u\}, (B_1^+ \backslash \{u\}) \cup \{v\}, B_1^-, B_2 \backslash \{v\}, B_l, \omega - \omega(u, v))$-cycle cover. The union $C' \cup M'$ contains a cycle $S' = (x_1, \ldots, x_r, p, v, x_1)$, with any other possible cycles being the same as in $C \cup M$. Again, $C \cup M$ will be a single cycle if and only if $C' \cup M'$ is a single cycle, leaving only the requirement that M' belongs in the basis for $(B_1^+ \backslash \{u\}) \cup \{v\}$.

Because of the modified ordering placing u and v last, they both come to occupy the same place at the end of the ordering for B_1 and $(B_1\backslash\{u\})\cup\{v\}$, respectively. Since the only difference is swapping vertex u with v, M' is in the basis for $(B_1\backslash\{u\})\cup\{v\}$ (even corresponding to the same bitstring). This gives in total:

$$d_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M] \equiv d'_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M] + d'_{uv}[B_0 \cup \{u\}, (B_1^+\backslash\{u\}) \cup \{v\}, B_1^-, B_2\backslash\{v\}, \omega - \omega(u,v), M'].$$

IV. $u \in B_2$ **and** $v \in B_1^-$**:** Due to their symmetry, this case is handled in the same way as the previous one.

V. $u, v \in B_2$**:** In this case, some $(B_0, B_1^+, B_1^-, B_2, B_l, \omega)$-cycle cover C may use the arc (u, v), yet no perfect matching M on B_1 can do the same. As before, the parity of the number of partial cycle covers consistent with M and not including (u, v) is already stored in $d'_{uv}[\cdot]$.

For $(B_0, B_1^+, B_1^-, B_2, B_l, \omega)$-cycle covers C that do use (u, v), let $C' := C\backslash\{(u, v)\}$ and $M' := M\cup\{(u, v)\}$, with C' being a $(B_0, B_1^+\cup\{u\}, B_1^-\cup\{v\}, B_2\backslash\{u, v\}, B_l, \omega - \omega(u, v))$-cycle cover. Here, the arc (u, v) was effectively "moved" into M and so $C \cup M = C' \cup M'$, making it clear that $C \cup M$ would be a single cycle if and only if $C' \cup M'$ is a single cycle.

Once more, M' has to be shown to belong in the basis for $B_1 \cup \{u, v\}$, based on the modified ordering: assuming that the ordering of B_1 ends with $\ldots, p, q, r$, then if $M = X(|B_1|, a0)$, it is $(p, r) \in M$, making $M' = X(|B_1| + 2, a01)$. Otherwise, if $M = X(|B_1|, a1)$, it is $(q, r) \in M$, making $M' = X(|B_1| + 2, a11)$ (it is only necessary to append a 1 to the bitstring for M). It is in total:

$$d_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M] \equiv d'_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M] + d'_{uv}[B_0, B_1^+ \cup \{u\}, B_1^- \cup \{v\}, B_2\backslash\{u, v\}, \omega - \omega(u, v), M \cup \{(u, v)\}].$$

This concludes the possibilities for computation of $d_{uv}[B_0, B_1^+, B_1^-, B_2, \omega, M]$. As mentioned, table $d[\cdot]$ that uses the representation based on the standard ordering is computed by yet another application of Lemma 3.6, concluding the handling of introduce arc bags and the dynamic programming description as well.

Time complexity: For every choice of weight $1 \leq \omega \leq n \cdot w_{\max}$ the number of possible tuples $(B_0, B_1^+, B_1^-, B_2, M)$, with B partitioned into B_0, B_1, B_2, further partitioning of B_1 into B_1^+ and B_1^-, and M being one of the $2^{\frac{|B_1|}{2}-1}$ basis matchings for B_1, is strictly smaller than $(2 + 2^{\frac{3}{2}})^{|B|}$, by the multinomial theorem and since $\binom{|B|}{\frac{|B|}{2}} \cdot 2^{\frac{|B|}{2}-1} < 2^{\frac{3}{2}|B|-1}$. Thus, we have the following:

Lemma 3.7. *Given a digraph G along with a path decomposition of width* $\mathtt{pw}$*, there exists an algorithm that computes the table entries $d[\cdot]$ that correspond to the last (furthest to the right) introduce arc bag of the path decomposition in $(2+2^{\frac{3}{2}})^{\mathtt{pw}} w_{\max}(n \cdot \mathtt{pw})^{\mathcal{O}(1)}$ time.*

The complete algorithm is stated in terms of the following theorem.

Theorem 3.8. *Given a digraph G along with a path decomposition of width* $\mathtt{pw}$*, there exists a Monte-Carlo algorithm that solves the* Directed Hamiltonian Cycle *problem in $(2+2^{\frac{3}{2}})^{\mathtt{pw}}(n \cdot \mathtt{pw})^{\mathcal{O}(1)}$ time. The algorithm does not give false positives and may give false negatives with probability at most $\frac{1}{2}$.*

Proof. First, for a vertex v introduced in the rightmost introduce vertex bag of the given path decomposition, an arc (u, v) is "guessed" for inclusion in the solution and the path is reordered as previously described, by repeating the following at most $\mathtt{pw}$ times.[4]

Then, for every arc $a \in A$, assign a weight $\omega(a) \in \{1, 2, \ldots, 2|A|\}$ uniformly and independently at random, and using Lemma 3.7 compute for every weight $w \leq n \cdot w_{\max} \leq n2|A|$ the table entry $d[\emptyset, \{u\}, \{v\}, \emptyset, \{(u, v)\}, w]$ of the rightmost introduce arc bag. Subsequently, output YES if one of these values is 1 and NO otherwise.

If the answer is positive, there must be a Hamiltonian cycle that contains the arc (u, v), by the definition of d. If the answer is negative, then supposing the existence of a Hamiltonian cycle, by Lemma 2.9 (Isolation Lemma) ω isolates the family of all Hamiltonian cycles of G with probability at least $1 - \frac{|A|}{2|A|} = \frac{1}{2}$. This means there exists a unique Hamiltonian cycle H of minimum weight $\omega(H)$, with probability at least $\frac{1}{2}$. Let the arc incident on v in H be (u, v).[5] As H is the only Hamiltonian cycle of weight $\omega(H)$, it must be $d[\emptyset, \{u\}, \{v\}, \emptyset, \{(u, v)\}, \omega(H) - \omega(u, v)] = 1$, i.e. with probability at most $\frac{1}{2}$ the algorithm will output NO, despite the given graph containing a Hamiltonian cycle. ■

[4] The direction of the chosen arc is not necessarily incoming, yet symmetry allows this assumption without any loss of generality.

[5] Again, without loss of generality.

Chapter 4

Generalizations to Hamiltonian Cycle

The focus of this chapter is on generalizations to the setting of the previous chapters, in particular relation to Hamiltonicity. In previous chapters, the structure and rank of the directed and undirected Matching Connectivity matrices were examined as a middle step towards efficient dynamic programming algorithms on path decompositions.

The crucial elements of this technique were the basis sets of matchings $\mathbf{X}_t$, to be used in the computations of the dynamic programming procedure instead of all possible matchings, avoiding redundant computation. Viewing perfect matchings as partitions on a set of elements U, where every block is constrained to only include two elements, questions naturally arise as to the structure and rank of corresponding matrices that include partitions that consist of larger blocks of elements.

4.1 Our extended setting

In this chapter we consider partitions of a base set U of t elements, such that every block consists of k elements. A pair of integers t, k is called *valid*, if the ratio $\frac{t}{k}$ is an integer ≥ 2. For any valid pair t, k, two partitions M_1 and M_2 *complete* each other, if $M_1 \sqcap M_2 = \{U\}$.

In the previous setting where only perfect matchings were considered, each block size was equal to $k = 2$ (representing an edge) and the number of elements t was always an even integer. Furthermore, the notion of completion between two matchings had a straightforward equivalence: the combination of two matchings M_1, M_2 that completed each other (i.e. $M_1 \sqcap M_2 = \{U\}$) was a Hamiltonian cycle.

In our current context, two partitions can be seen to complete each other if, intuitively, they share no identical blocks and there is a circular and exhaustive sequence of blocks alternating between the two partitions (without repetition), such that consecutive blocks in the sequence share some element(s). The following observation can help clarify the essence of this notion (see also the examples in Section 4.2).

Observation 4.1 (Premise for completion)**.** *For all valid t, k, if for any two partitions M and M' it is $M \sqcap M' = \{U\}$, then for any number and choice of blocks b_i from M, with $\bigcup_{\forall i} b_i = X \subset U$, there can be no choice of blocks b'_j from M', with $\bigcup_{\forall j} b'_j = X' \subseteq U$, such that $X = X'$, i.e. for any two partitions that* complete *each other, the union of any choice of blocks (but not all) from one partition can not result in the exact same elements as any choice of blocks from the other partition, and vice-versa.*

As an aid to visualizing the above observation, consider the case of $k = 2$: two perfect matchings do not result in a Hamiltonian cycle if any partial choice of edges from one matching results in the exact same set of vertices as any partial choice of edges from the other matching. This fact about completion directly generalizes for larger values of k.

In previous chapters, possible redundancy in the connectivity capabilities of perfect matchings was evaluated based on characterizations from linear algebra, namely the rank (and basis identification) of matrices indexed by all possible perfect matchings on sets of a given size. In parallel, we now investigate a related notion, termed *irreducibility* of a set of partitions, formally defined next.

Definition 4.2 (Irreducibility)**.** For some valid t, k, a set of partitions $\mathcal{R} = \{R, \bar{R}\}$, $|R| = |\bar{R}| = \frac{|\mathcal{R}|}{2}$ is *irreducible* for t, k, if the following conditions hold:

a. For every $M \in R$, there is a unique $\bar{M} \in \bar{R}$ (the *complement* of M), such that $M \sqcap \bar{M} = \{U\}$.

b. For any partition $M_* \notin \mathcal{R}$ (under t, k), there is at least one $M \in \mathcal{R}$, such that $M \sqcap M_* = \{U\}$.

The largest such set of partitions $\mathcal{R}$ for some valid t, k is called *maximum irreducible*. Maximum irreducible sets of partitions are of interest since they retain the property of capturing the essential (non-duplicate) capability for connection, in the same way our basis matchings did in previous chapters (being a set of perfect matchings whose size exactly matches the rank of $\mathcal{H}_t$).

As an example of these connectivity capabilities, consider a basis set $\mathbf{X}_t$ of perfect matchings: any other possible perfect matching $M' \notin \mathbf{X}_t$ is guaranteed to be completed

by at least one perfect matching $M \in \mathbf{X}_t$ from the basis set and furthermore, each perfect matching within the basis set only completes one other perfect matching from inside the basis set.

Any maximum irreducible set of partitions has the same property and additionally, the fact these partitions form a permutation sub-matrix in the equivalent matrices to $\mathcal{H}_t$ for higher values of k will enable us to show a lower bound on the rank of these matrices in the following sections (see Corollary 4.8).

Another useful observation concerning maximum irreducible sets is the following.

Observation 4.3. *For all valid t, k, if the size of any maximum irreducible set is l, then any other set of partitions $\mathcal{R}$, with $|\mathcal{R}| = l$, that satisfies only the first condition of Definition 4.2 is maximum irreducible as well.*

Intuitively, the above states that any set of size equal to some maximum irreducible set, that includes pairs of partitions that only complete each other within the set will include at least one partition that completes any other partition outside the set. To picture why this holds, consider the alternative: suppose some partition outside the given set is not completed by any partition within, then addition of this partition in the set (and some other completing it) would yield a larger set. This allows for characterization of some set of partitions as maximum irreducible, based solely on its size and inner organization.

4.2 An irreducible set of partitions

This section describes the construction of an irreducible set $\mathbf{X}_{t,k}$ of partitions for every choice of t, k. These sets can be seen to generalize the basis sets $\mathbf{X}_t$ of perfect matchings (Definition 2.4). In intuitive terms, the main idea of keeping connectivity restricted to elements that are relatively close together with respect to some ordering is retained here.

More formally, let the base set be $U_t := \{0, 1, \ldots, t-1\}$ and $\Pi_k(U_t)$ denote the set of all partitions on t elements, where every block is of size k. Also, let $S_{t,k}$ denote the set of all possible blocks in partitions from $\Pi_k(U_t)$. We define $f(M, \{u_1, \ldots, u_l\}, \{v_1, \ldots, v_l\}) : \Pi_k(U_t) \times S_{t,k} \times S_{t,k} \to \Pi_k(U_t)$, with $u_i, v_i \in U_t, \forall 1 \leq i \leq l$, to be a function that removes every u_i from the block that includes it in M, and replaces it with v_i in the same block.[1] The formal definition of $\mathbf{X}_{t,k}$ is the following:

[1] In fact, this definition of $f(\cdot, \cdot, \cdot)$ allows for partitions whose blocks may contain a *multiple* of k elements, instead of exactly k. Definitions and results stated in this section apply for both cases, yet as by Definition 4.4 our basis partitions only have blocks of size exactly k, this simplification was adopted here for clarity. See Section 4.3 for more details.

Definition 4.4. For $t = k$, let ϵ denote the empty string and $X_k(k, \epsilon) := \{0, 1, \ldots, k-1\}$, while $\mathbf{X}_{k,k} := \{X_k(k, \epsilon)\}$. For $\frac{t}{k} \geq 2$, let α be a bit-string of length $\frac{t}{k} - 2$. Partitions $X_k(t, a0)$ and $X_k(t, a1)$ on t, k are defined as follows:

$$X_k(t, a0) := X_k(t-k, a) \cup \{\{t-k, t-k+1, \ldots, t-1\}\}.$$

If $\frac{t}{k}$ is even:

$$X_k(t, a1) := f\left(X_k(t-k, a), \{t - \left\lceil \frac{3k}{2} \right\rceil, \ldots, t-k-1\}, \{t-k, \ldots, t - \left\lceil \frac{k}{2} \right\rceil\}\right) \cup \left\{\{t - \left\lceil \frac{3k}{2} \right\rceil, \ldots, t-k-1\}, \{\left\lceil \frac{k}{2} \right\rceil + 1, \ldots, t-1\}\right\}.$$

If $\frac{t}{k}$ is odd:

$$X_k(t, a1) := f\left(X_k(t-k, a), \{t - \left\lfloor \frac{3k}{2} \right\rfloor, \ldots, t-k-1\}, \{t-k, \ldots, t - \left\lfloor \frac{k}{2} \right\rfloor\}\right) \cup \left\{\{t - \left\lfloor \frac{3k}{2} \right\rfloor, \ldots, t-k-1\}, \{\left\lfloor \frac{k}{2} \right\rfloor + 1, \ldots, t-1\}\right\}.$$

Using the shorthand $X_k(a)$ for $X_k(k|a| + k, a)$, since a determines the size t of the base set U_t, let $\mathbf{X}_{t,k}$ be the set of all partitions $X := X_k(t, a)$ for any bit-string a of length $\frac{t}{k} - 1$.

As with Definition 2.4, the above is a rather too technical definition to provide intuition on the actual structure of these irreducible sets. Towards that end, we present the following two pairs of examples, shown in Figure 4.1 and Figure 4.2.

FIGURE 4.1: Example partitions from $\mathbf{X}_{12,3}$.

In the examples shown in Figure 4.1, the base set is $U_{12} = \{0, \ldots, 11\}$ and $k = 3$, i.e. we only consider partitions on 12 elements, such that each of their blocks contains 3 elements. The two partitions shown in Figure 4.1 are $X_3(010)$ and $X_3(101)$, on the left and right side of the figure, respectively. Elements can be seen as divided in successive groups (delimited by vertical lines in the figure) and each bit of the bit-string as assigned to every group, with the exception of the last two groups, for which no bit is required. Furthermore, the color of each element denotes the block it is placed in each partition.

Now, depending on the bit assigned to each group (being 0 or 1), all elements of that group that are not already included in some block by the partition, are placed in a block together with either the upper elements of the next group (bit equals 0), or with the

lower elements of the next group (bit equals 1): consider the partition shown on the left in Figure 4.1 ($X_3(010)$). The first bit is 0, so element 0 is included in a block together with elements 1 and 2. The second bit being 1, the remaining elements of the second group (3 and 4) are included in a block together with element 6, being the lower element of the next group. Finally, the last bit being 0, element 5 is included in a block with the upper elements of the next group (7 and 8), which only allows the last remaining elements (9,10 and 11) to be included in the same block. Conversely, observe that the partition shown on the right in Figure 4.1 ($X_3(101)$), due to its bit-string being the exact complement of $X_3(010)$, this partition assigns elements in an exactly opposite manner.

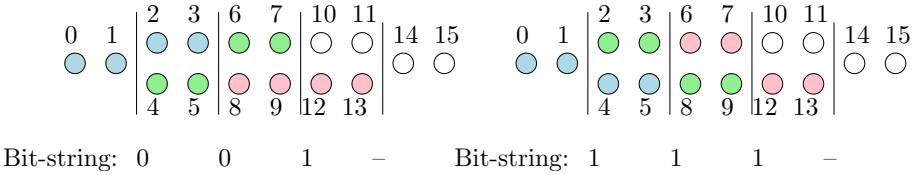


FIGURE 4.2: Example partitions from $\mathbf{X}_{16,4}$.

In the examples shown in Figure 4.2, the base set is U_{16} and $k = 4$, i.e. we only consider partitions on 16 elements, such that each of their blocks contains 4 elements. The two partitions shown in Figure 4.2 are $X_4(001)$ and $X_4(111)$, on the left and right side of the figure, respectively. As in the previous examples, elements can again be seen as divided in successive groups (delimited by vertical lines), each bit of the bit-string as assigned to every group, with the exception of the last two groups, and color again denotes the block each partition assigns every element in. The same ideas are applied to the construction of these partitions, as described in the previous example.

The general intuition behind this construction is the following: based on all bit-strings of length $\frac{t}{k} - 1$, the set $\mathbf{X}_{t,k}$ includes all partitions in which every block has exactly one vertical line dividing it and on either side of the line, the block includes all elements that are placed on the upper, or lower part of their group.

When recursively constructing further partitions following the addition of k extra elements to U_t, these elements will always either be blocked together in the new partition, or (almost) half the elements in the last block of the previous partition will be replaced by (almost) half the new elements and subsequently blocked together with the rest of the new elements, depending on whether the new last bit of the bit-string is 0, or 1, respectively.

The sets $\mathbf{X}_{t,k}$ can be easily seen to generalize the basis sets of perfect matchings $\mathbf{X}_t$, extensively used in the previous chapters. The following theorem states that these sets are in fact irreducible for all valid t, k.

Theorem 4.5. *The sets of partitions $\mathbf{X}_{t,k}$ are irreducible for all valid t, k.*

Proof. First, we will show that for all valid t, k and bit-strings a, b of length $\frac{t}{k} - 1$, two partitions $X_k(t, a), X_k(t, b) \in \mathbf{X}_{t,k}$ complete each other, i.e. it is $X_k(t, a) \sqcap X_k(t, b) = \{U_t\}$, if and only if it is $b = \bar{a}$, i.e. their generative bit-strings are complementary.

We assume that $b \neq \bar{a}$, meaning there is at least one position i (with $1 \leq i \leq \frac{t}{k} - 1$), such that $a[i] = b[i]$. We focus on the first such position to occur, meaning that for $1 \leq j \leq i$, it is $a[j] \neq b[j]$. By definition, partitions $X_k(t, a)$ and $X_k(t, b)$ then assign elements in exactly opposite manners, meaning there are no collections of blocks that only appear before group i, such that their unions give the same elements in both $X_k(t, a), X_k(t, b)$ (see Figure 4.2 for an example, with $j \leq 2, i = 3$).

Furthermore, since $a[i] = b[i]$, both partitions will block the same elements of group i (either upper or lower, depending on the value of i) with some elements from the previous group and the other elements will be included in the next block with some elements from the next group, in both partitions. But the unions of all blocks from each partition up to the elements of group i (those selected by the value of i) will then contain exactly the same elements and so $X_k(t, a) \sqcap X_k(t, b) \neq \{U_k\}$.[2] On the other hand, if $a[i] \neq b[i]$ for all $1 \leq i \leq \frac{t}{k} - 1$, then we have $X_k(t, a) \sqcap X_k(t, b) = \{U_t\}$, as only the unions of all blocks from each partition can contain exactly the same elements (see Figure 4.1 for an example).

Second, we will show that any partition $M_0 \notin \mathbf{X}_{t,k}$ is completed by at least two partitions from our set. It is easily verified that in any such partition, at least $2k$ elements will be in some configuration that is not assigned by any partition included in $\mathbf{X}_{t,k}$. Let $u_1, \ldots, u_{2k}$ be these elements and X_0 be the union of any given choice (and number) of block(s) from M_0.

If $u_i \notin X_0$ for every $i \in [1, 2k]$, then all given blocks from M_0 whose union is X_0 can be seen to have some implicit bit that would produce them in $\mathbf{X}_{t,k}$. All partitions M_j from $\mathbf{X}_{t,k}$ that assign complementary bits in the corresponding positions, give unions $X_j \neq X_0$ and since X_0 does not contain all elements, there are at least two disjoint such partitions in $\mathbf{X}_{t,k}$.

Otherwise, if X_0 contains at least one u_i, then it contains at least k of them: without loss of generality it will be $u_1, \ldots, u_k \in X_0$ (since X_0 is a union of blocks and includes some u_i, it must include at least one complete block of them). Again, for any block(s) included in X_0 that could be produced by the definition of $\mathbf{X}_{t,k}$, the partitions that assign complementary bits to the corresponding positions always avoid giving the same elements in unison, while any assignment based on the bit(s) that correspond to the groups in which these $u_1, \ldots, u_k$ belong to, will again produce some disjoint union.

[2] Alternatively, this can be seen as a "cut" in the result of the $\sqcap$ operation on these partitions.

Finally, if all $u_i \in X_0$, then there are at least two disjoint partitions in $\mathbf{X}_{t,k}$, such that no union of blocks from each partition can include exactly the same elements as X_0, due to symmetry with the first case, since $u_i \notin (U_t \setminus X_0)$ for every $i \in [1, 2k]$. ■

4.3 Further discussion

This section elaborates on our extended setting considering Hamiltonicity between partitions. First, some clarification is required on the form of the partitions we consider.

As previously mentioned, in our extended setting we consider partitions of a base set U_t of t elements, such that every block is of size exactly k. In this setting, we introduced an extension of the basis matchings $\mathbf{X}_t$ used in previous chapters, namely the irreducible set of partitions $\mathbf{X}_{t,k}$. For all valid t, k, the size of set $\mathbf{X}_{t,k}$ is $2^{\frac{t}{k}-1}$. Despite these sets only including partitions whose blocks are of size exactly k, their irreducibility can be seen to hold for a broader setting, in which we also consider partitions that group elements in larger blocks, as long as the size of every block is a *multiple* of k.

In fact, definitions and results stated in the previous sections already take this extended setting in consideration, yet their presentation was focused on the simplification that every block was of size exactly k, towards a more clear exposition. The analysis of the previous sections (Theorem 4.5 in particular) is still valid under this more general setup, allowing wider-ranging statements to be made. For instance, from Theorem 4.5 we know that all sets constructed according to Definition 4.4 are irreducible for any choice of valid t, k, yet as there also exist other potentially advantageous irreducible sets of partitions with blocks of size exactly k, there might also exist some useful irreducible sets of partitions with larger blocks.

One constraint on the organization of any maximum irreducible set, however, is exposed by the following observation. For all valid t, k, let the *trivial* partition be the one that groups all elements of U into a single block.

Observation 4.6. *For all valid t, k, no maximum irreducible set contains the trivial partition.*

Intuitively, any trivial partition consisting of just one block does not offer enough connectivity potential to be included in some maximum irreducible set, the purpose of which is precisely to capture the essential connectivity capabilities of all possible partitions.

The significance of maximum irreducible sets and their characteristics is underlined by the following statements, that place our discussion in a specific context. We consider the *Partition Connectivity matrix* $\mathcal{C}_{t,k}$, an extension of matrix $\mathcal{H}_t$ for higher values of k.

Definition 4.7. For all valid t, k, let the *Partition Connectivity matrix* $\mathcal{C}_{t,k}$ be the matrix that has rows and columns both labeled by all possible partitions of t elements, such that in every partition the size of any block is a multiple of k and an entry $\mathcal{C}_{t,k}[M_1, M_2]$ is 1, if $M_1 \sqcap M_2 = \{U_t\}$ and 0 otherwise.

From Theorem 4.5, we immediately get the following lower bound on the rank of $\mathcal{C}_{t,k}$, as the sub-matrix induced by all rows and columns of all partitions in $\mathbf{X}_{t,k}$ is a permutation matrix of size $2^{\frac{t}{k}-1} \times 2^{\frac{t}{k}-1}$.

Corollary 4.8. *For all valid t, k the rank of matrix $\mathcal{C}_{t,k}$ is at least $2^{\frac{t}{k}-1}$, over* GF(2).

Despite convincing hints providing strong indication towards it and some effort to provide a matching upper bound on the rank of $\mathcal{C}_{t,k}$, we were unable to prove the following:

Conjecture 4.9. *The sets of partitions $\mathbf{X}_{t,k}$ are* maximum *irreducible for all valid t, k.*

The above would imply there is no other possible set that defines a permutation sub-matrix within $\mathcal{C}_{t,k}$, a statement supported by the apparent structure of $\mathcal{C}_{t,k}$. Since for $k = 2$, the setting conforms to the one studied in [14] (and previous chapters), where maximum irreducible sets $\mathbf{X}_t$ were shown to be of size $2^{\frac{t}{2}-1}$, and due to other intermediate observations implying there is no implicit difference between settings defined by each particular choice of k, the validity of the above conjecture seems overt, yet rigorous proof remains elusive.

On the other hand, even if the size of any permutation sub-matrix within $\mathcal{C}_{t,k}$ is indeed bounded above by $2^{\frac{t}{2}-1} \times 2^{\frac{t}{2}-1}$, this would not be as strong a statement as the following:

Conjecture 4.10. *For all valid t, k the rank of matrix $\mathcal{C}_{t,k}$ is exactly $2^{\frac{t}{k}-1}$ over* GF(2).

If shown to hold, the above statement would provide a special case of Bollobás' well-known "Two Families" theorem (see [24, Theorem 8.7]). Due to the importance of this result in extremal set theory, the above conjecture could have a number of useful repercussions, yet the probable non-triviality of a potential proof needs be emphasized.

Chapter 5

Conclusion

In this thesis, we discussed a recent approach to solving the Hamiltonian Cycle problem on path decompositions and some of its subsequent consequences. This approach makes use of structural characterizations obtained from insights on connectivity potential offered by perfect matchings, via a viewpoint influenced by linear algebra.

After a brief outline of this novel perspective, we presented similar results for the directed version of the problem, i.e. we showed that a set of matchings $\mathbf{X}_t$ forms a basis of the Directed Matching Connectivity matrix $\mathcal{D}_t$, capturing the essential capability for Hamiltonicity. We showed explicit construction and application of this set and subsequently obtained a Monte Carlo algorithm that solves the decision version of Directed Hamiltonian Cycle on a given path decomposition of width $\mathtt{pw}$ in $(2+2^{\frac{3}{2}})^{\mathtt{pw}}(n \cdot \mathtt{pw})^{\mathcal{O}(1)}$ time via dynamic programming.

Moreover, we considered a generalized setting in which we investigated Hamiltonicity among partitions of elements, instead of only unions of perfect matchings. After extending our framework by introducing expanded definitions of the useful concepts, we were able to show how to construct sets of partitions $\mathbf{X}_{t,k}$ that retain the property of encapsulating non-redundant connectivity capability, while also providing a lower bound on the rank of the generalized Partition Connectivity matrix $\mathcal{C}_{t,k}$.

This investigation leads to a number of interesting open problems, such as finding formal proofs for Conjecture 4.9 that concerns an upper bound on the rank of $\mathcal{D}_{t,k}$, and Conjecture 4.10 that aims to determine it precisely, while also representing a special case of the well-known Bollobás' theorem.

Finally, other intriguing directions for future research would be to explore the range of problems to which this rank-based approach could be beneficially applied, or examine whether our own results could be easily derandomized and improved.

Bibliography

[1] Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[2] Eric T. Bax. Inclusion and exclusion algorithm for the Hamiltonian path problem. *Information Processing Letters*, 47(4):203–207, 1993.

[3] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, (1):61–63, 1962.

[4] Andreas Björklund. Determinant Sums for Undirected Hamiltonicity. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 173–182. IEEE Computer Society, 2010.

[5] Andreas Björklund and Thore Husfeldt. The Parity of Directed Hamiltonian Cycles. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pages 727–735.

[6] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Trimmed Moebius inversion and graphs of bounded degree. *Theory of Computing Systems*, 47(3):637–654, 2010.

[7] Hans L. Bodlaender. A Tourist Guide through Treewidth. *Acta Cybernetica*, 11: 1–23, 1993.

[8] Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Mathematical Foundations of Computer Science 1997*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin Heidelberg, 1997.

[9] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic Single Exponential Time Algorithms for Connectivity Problems Parameterized by Treewidth. *ICALP*, 40(1):196–207, 2013.

[10] Hajo Broersma, Fedor V. Fomin, Pim van t' Hof, and Daniël Paulusma. Fast Exact Algorithms for Hamiltonicity in Claw-Free Graphs. In *Graph-Theoretic Concepts*

in Computer Science, volume 5911 of *Lecture Notes in Computer Science*, pages 44–53. Springer Berlin Heidelberg, 2010.

[11] Nicos Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. *Technical Report, Graduate School of Industrial Administration, Carnegie Mellon University*, (388), 1976.

[12] Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.

[13] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan van Rooij, and Jakub Onufry Woltaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 150–159, 2011.

[14] Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity Checking via Bases of Perfect Matchings. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 301–310. ACM, 2013.

[15] David Eppstein. The Traveling Salesman Problem for Cubic Graphs. *Journal of Graph Algorithms and Applications.*, 11(1):61–81, 2007.

[16] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer-Verlag New York, Inc., 2010.

[17] Michael R. Garey, David S. Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, pages 237–267, 1976.

[18] Heidi Gebauer. On the Number of Hamilton Cycles in Bounded Degree Graphs. In *Proceedings of the Fifth Workshop on Analytic Algorithmics and Combinatorics*, ANALCO 2008, pages 241–248, 2008.

[19] Mika Göös and Jukka Suomela. Locally checkable proofs. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 159–168. ACM, 2011.

[20] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. In *Proceedings of the 1961 16th ACM National Meeting*, ACM '61, pages 71.201–71.204. ACM, 1961.

[21] Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[22] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63 (4):512–530, 2001.

[23] Kazuo Iwama and Takuya Nakashima. An Improved Exact Algorithm for Cubic Graph TSP. In *Computing and Combinatorics*, volume 4598 of *Lecture Notes in Computer Science*, pages 108–117. 2007.

[24] Stasys Jukna. *Extremal Combinatorics*. Texts in Theoretical Computer Science. Springer, Berlin Heidelberg, 2001.

[25] Richard M. Karp. Reducibility Among Combinatorial Problems. In Raymond E Miller and James W Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[26] Richard M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Operations Research Letters*, I(2):49–51, 1982.

[27] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005.

[28] Ton Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[29] Shu Lin and Brian W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21:498–516, 1973.

[30] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 777–789, 2011.

[31] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[32] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[33] Ján Plesník. The NP-Completeness of the Hamiltonian Cycle Problem in Planar Digraphs with Degree Bound Two. *Information Processing Letters*, 8(4):199–201, 1979.

[34] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.

[35] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 64:49–64, 1984.

[36] Gerhard J. Woeginger. Exact Algorithms for NP-hard Problems: A Survey. In *Combinatorial Optimization - Eureka, You Shrink!*, pages 185–207. Springer-Verlag New York, Inc., 2003.