Algorithms for NP-Hard Problems via Rank-Related Parameters of Matrices

Jesper Nederlof(✉)

Eindhoven University of Technology, Utrecht University, Utrecht, The Netherlands
j.nederlof@uu.nl

Abstract. We survey a number of recent results that relate the fine-grained complexity of several NP-Hard problems with the rank of certain matrices. The main technical theme is that for a wide variety of Divide & Conquer algorithms, structural insights on associated *partial solutions matrices* may directly lead to speedups.

1 Introduction

Rank is a fundamental concept in linear algebra to express algebraic dependence in relations described by matrices. It has numerous applications in theoretical computer science and mathematics, ranging from algebraic complexity [BCS97], communication complexity [LS88], to extremal combinatorics [Mat10].

A common phenomenon in these areas is that low rank often helps in proving combinatorial upper bounds or in designing algorithms, e.g., through representative sets [BCKN15,FLPS16,KW14] or the polynomial method [Wil14].

In particular, rank has recently found applications in *fine-grained complexity* and the closely related area of *parameterized complexity*. For example, influential results such as algorithms for kernelization [KW14], the longest path problem [Mon85], and connectivity problems parameterized by treewidth [CNP+11, CKN13,BCKN15], rely crucially on certain low-rank factorizations.

Low-rank factorizations especially arise very naturally when applying the general Divide & Conquer and the closely related Dynamic Programming technique. Recall that these techniques (conceptually) partition a solution into *partial solutions*. Typically, lists of candidates for these partial solutions are maintained by an algorithm that gradually filters and extends these partial solutions to a complete solution.

The dominating term in the runtime of such an algorithm is the *number* of such partial solutions. But sometimes, there is no need to keep track of all partial solutions because of *group domination*: For example, suppose that partial solutions $s_0, s_1, \ldots, s_l$ are such that for any partial solution t that forms a complete solution with s_0 there is also an $i > 0$ such s_i forms a complete solution with t. Then of course, s_0 can be safely disregarded as partial solution.

Supported by the Netherlands Organization for Scientific Research under project no. 024.002.003 and the European Research Council under project no. 617951.

F. V. Fomin et al. (Eds.): Bodlaender Festschrift, LNCS 12160, pp. 145–164, 2020.
https://doi.org/10.1007/978-3-030-42071-0_11

In this survey we study several standard Divide & Conquer algorithms from the field of fine-grained complexity for NP-hard problems and explore how group domination helps to improve them. A crucial tool in these are *partial solution* matrices. Given two groups of partial solutions R and C a partial solution matrix $\mathbf{A} \in \{0,1\}^{R \times C}$ is a matrix such that $\mathbf{A}[p,q] = 1$ if and only if partial solutions p and q combine to a complete solution. We will see that insights on rank-related parameters of the partial solution matrices can be used to speed up the associated Divide & Conquer algorithms in a variety of settings.

Organization. This survey is organized as follows: In Sect. 2 we introduce used notation In Sect. 3 we introduce some matrices along with their various parameters, which will be used in Sect. 4 to provide algorithms for various NP-complete problems. Finally, in Sect. 5 we mention some other directions that we do not fully touch.

2 Preliminaries

We let A^B denote the set of vectors or functions indexed by B with values in A. The symbol ε denotes the empty string, vector or partition. If b is a Boolean, we denote $[b]$ for 1 if b if true and 0 otherwise. On the other hand, if $[b]$ is an integer we use $[b]$ to denote $\{1, \dots, b\}$.

In this survey all matrices will be written in bold font. If $\mathbf{M} \in \{0,1\}^{R \times C}$ is a matrix and $X \subseteq R$ and $Y \subseteq C$ we denote $\mathbf{M}[X,Y]$ for the matrix induced by rows X and columns Y. If either X or Y is replaced with a $\cdot$ this means no restriction is placed on the rows or columns, respectively. We let $\equiv_2$ denote equivalence modulo 2. If $Y, Y' \subseteq R$, we denote $\mathbf{M}[X, Y \circ Y']$ for the matrix obtained by horizontally concatenating the matrices $\mathbf{M}[X,Y]$ and $\mathbf{M}[X,Y']$.

Partitions and the Partition Lattice. Given a set U, we use $\Pi(U)$ for the set of all partitions of U, i.e. a family of subsets of U that are pairwise disjoint and whose union equals U. It is known that, together with the coarsening relation $\sqsubseteq$, $\Pi(U)$ gives a lattice, with the maximum element being $\{U\}$ and the minimum element being the partition into singletons. We denote $\sqcup$ for the join operation and $\sqcap$ for the meet operation in this lattice; these operators are associative and commutative. I.e., for two partitions p and q, $p \sqcup q$ is obtained as follows: let $\sim$ be the relation on the elements with $v \sim w$, if and only if v and w belong to the same set in p or v and w belong to the same set in q. Now, $p \sqcup q$ is the partition of U into the equivalence classes of the transitive closure of $\sim$. (In simple graph terms: build a graph H with a vertex set U, by turning each set in p and each set in q into a clique. Now, $p \sqcup q$ is the partition of U into the connected components of H.) $p \sqcap q$ precisely consists of all sets that are the nonempty intersection of a set from p and a set from q. We use $\Pi_m(U) \subset \Pi(U)$ to denote the set of all partitions of U in blocks of size 2, or equivalently, the set of perfect matchings over U. Moreover, $\Pi_2(U)$ denotes the set of all partitions with two blocks, i.e.

cuts. Thus there are partitions $\{X, Y\}$ where $X \cap Y = \emptyset$, $X \cup Y = U$ and X or Y may equal the empty set. Given $p \in \Pi(U)$ we let $\#\texttt{blocks}(p)$ denote the number of blocks of p. We sometimes formally interpret a partition as a family of disjoint subsets in the natural way. If $p = \{P_1, \ldots, P_l\} \in \Pi(U)$ and $X \subseteq U$ we define $p_{|X} = \{P_1 \setminus X, \ldots, P_l \setminus X\}$ as the restriction of p onto X. Also, if $A \subseteq U$, we let $\{A\}$ denote the partition with the single non-trivial block A.

3 Some Matrices and Their Rank-Related Parameters

In this section we introduce and study a number of families of matrices that will serve as partial solution matrices in the next section. In order to use them as such the following terminology will be useful:

Definition 3.1. *A family of matrices $\{\mathbf{A}_t\}_t$ is* explicit *if the following holds for every t: If $\mathbf{A}_t$ is an $n \times n$ matrix, then given t and $1 \leq r, c \leq n$, the entry $\mathbf{A}_t[i, j]$ can be computed in* $\mathrm{polylog}(n)$ *time. A factorization $\mathbf{A}_t = \mathbf{L}_t \mathbf{R}_t$ is* explicit *if $\{\mathbf{L}_t\}_t$ is explicit.*

This section is organized as follows: In Subsect. 3.1 we define a number of rank-related parameters. In the subsequent subsections we present case studies of matrices where the different rank-related parameters are useful: In Subsect. 3.2 we study the field rank, in Subsect. 3.3 the Boolean rank, and in Subsect. 3.4 the support rank.

3.1 Some Rank-Related Parameters of Matrices

We study several parameters that express various sorts of (algebraic) dependence between rows of a matrix. Let $\mathbf{A}_t$ be a binary matrix, for a field $\mathbb{F}$, we denote $\mathrm{rk}_{\mathbb{F}}(\mathbf{A}_t)$ for the rank of $\mathbf{A}_t$ over $\mathbb{F}$. We define the *field rank* of $\mathbf{A}_t$ as the minimum of $\mathrm{rk}_{\mathbb{F}}(\mathbf{A}_t)$ over all reasonable[1] fields $\mathbb{F}$.

We define the *support rank* $\texttt{supRank}(\mathbf{A}_t)$ of a matrix $\mathbf{A}_t$ to be the minimum rank of a matrix $\mathbf{A}'_t$ over a finite field $\mathbb{F}$ with the property that $\mathbf{A}_t[p, q]$ is non-zero if and only if $\mathbf{A}'_t[p, q]$ is non-zero for every p, q. This parameter goes by several names, such as the 'non-deterministic rank' [Wol03], and its computation has received significant attention by researchers working on linear algebra.[2]

We let $\texttt{boolRank}(\mathbf{A}_t)$ denote the *Boolean rank* of matrix $\mathbf{A}_t$. This is the minimum size of a family $\mathcal{F}$ of submatrices of $\mathbf{A}$ with value 1 in each cell with the following property: every matrix cell with of $\mathbf{A}$ with value 1 is contained in at least one submatrix in $\mathcal{F}$. Such a family $\mathcal{F}$ is often called a *rectangle cover*. Boolean rank can also be defined as the rank of $\mathbf{A}_t$ over the Boolean semiring $(\{0, 1\}, \wedge, \vee)$: A matrix $\mathbf{A}_t$ has Boolean rank at most r if there exist Boolean matrices $\mathbf{L}_t$ and $\mathbf{R}_t$ such that $\mathbf{A}_t[p, q] = \vee_{i=1}^{r} (\mathbf{L}_t[p, i] \wedge \mathbf{R}_t[i, q])$.

[1] Since things get a bit tricky formally here, let's just say we restrict attention to the fields $\mathbb{R}$ and $\mathbb{F}_p$ for finite p.

[2] https://aimath.org/pastworkshops/matrixspectrum.html.

The Boolean rank can also be interpreted as the minimum 'biclique cover' of the bipartite graph of which $\mathbf{A}$ is the incidence matrix. It is worthwhile noticing that $\texttt{boolRank}(\mathbf{A}_t)$ is equal to the logarithm of the non-deterministic communication complexity [KN97].

We let $\texttt{indMatch}(\mathbf{A}_t)$ denote the maximum size of a permutation matrix (i.e. exactly one cell with value 1 per row and column) that is a submatrix of $\mathbf{A}_t$. We use this notation since $\texttt{indMatch}(\mathbf{A}_t)$ can be seen to be equal to the largest *induced matching* of the bipartite graph that has $\mathbf{A}_t$ as its incidence matrix.

Definition 3.2. *Given a matrix $\mathbf{A}_t \in \{0,1\}^{R \times C}$ and a subset $X \subseteq R$, a subset $X' \subseteq X$ is a* representative set of X with respect to $\mathbf{A}_t$ *if for every $c \in C$, there exists an $r \in X$ such that $\mathbf{A}_t[r,c] = 1$ only if there exists $r' \in X'$ such that $\mathbf{A}_t[r',c] = 1$.*

It is easy to see that representation is *transitive*: If X is a representative set of Y and Y is a representative set of Z, then X is a representative set of Z.

We observe that a set of rows X has no representative set of X as a strict subset if and only if every element $r \in X$ has a 'reason' to be included, i.e. a column c such that $\mathbf{A}_t[r,c] = 1$ and $\mathbf{A}_t[r',c] = 0$ for every $r' \in X_t \setminus r$.

Observation 3.1. *Let $\mathbf{A}_t \in \{0,1\}^{R \times C}$. A set $X \subseteq R$, is an inclusion-wise minimal representative set of itself if and only if there exists $Y \subseteq C$ such that $\mathbf{A}_t[X,Y]$ is a permutation matrix.*

We are interested in computing small representative sets for any (worst-case) set of rows. Observation 3.1 implies that the minimum size representative set of any set of rows is at most $\texttt{indMatch}(\mathbf{A}_t)$, and that there exists some set of rows for which the minimum size of a representative set equals $\texttt{indMatch}(\mathbf{A}_t)$. Thus to understand the exact efficiency of computing representative sets, the quantity $\texttt{indMatch}(\mathbf{A}_t)$ is of relevance.

Unfortunately it is NP-complete to compute $\texttt{indMatch}(\mathbf{A}_t)$ even in special cases such as matrices with at most 3 non-zero values per row and column [Loz02]. Moreover, even if there would be a polynomial time algorithm, in many cases we would like to avoid to construct the matrix $\mathbf{A}_t$ explicitly. Fortunately, the following lemma shows that often we can compute representative sets in time *sublinear* in terms of the dimensions of the matrix if it has a small factorization.

Lemma 3.1. *Suppose $\mathbf{A}_t \in \{0,1\}^{R \times C}$ has field, support or Boolean rank r and the associated factorization is explicit. Then, a representative set of a given subset of rows $X \subseteq R$ can be found in $|X|r^{\omega-1}\mathrm{polylog}(r)$ time, where $\omega < 2.371$ is the smallest number such that two $(n \times n)$-matrices can be multiplied in $n^{\omega+o(1)}$ time. Moreover, if $\mathbf{A}_t$ has Boolean rank r, the runtime can be reduced to $|X|r \cdot \mathrm{polylog}(r)$ time.*

Proof. Let $\mathbf{A}'_t = \mathbf{L}_t\mathbf{R}_t$ be the explicit factorization of rank r, where $\mathbf{A}'_t$ is a matrix such that $\mathbf{A}'_t \neq 0$ if and only if $\mathbf{A}_t \neq 0$. So $\mathbf{L}_t$ is an $(|R| \times r)$-matrix and $\mathbf{R}_t$ is an $(r \times |C|)$-matrix. We first focus on the field and support rank.

Construct the matrix $\mathbf{L}_t[X,\cdot]$ explicitly. Note this is possible within $|X|\cdot r\cdot \text{polylog}(r)$ time: This matrix has $|X|\times r$ entries, and that each entry of it can be computed in $\text{polylog}(r)$ time since the factorization $\mathbf{A}'_t=\mathbf{L}_t\mathbf{R}_t$ is assumed to be explicit. Now use fast Gaussian elimination algorithm based on fast matrix multiplication algorithms [BCKN15, Lemma 3.15] to compute a row basis X' of $\mathbf{L}_t[X,\cdot]$ in time $|X|r^{\omega-1}$, where $\omega<2.373$ is a number such that two $n\times n$ matrices can be multiplied within $n^{\omega+o(1)}$ time.

It remains to show that X' is a representative set of X. Let c be a column and let r be a row such that $\mathbf{A}[r,c]\neq 0$. This implies that $\mathbf{A}'[r,c]\neq 0$. Since X' is a row-basis of $\mathbf{L}'_t$, there exist $r_1,\dots,r_\ell\in R$ and $\lambda_1,\dots,\lambda_l\in\mathbb{F}$ such that

$$\sum_{i=1}^{\ell}\lambda_i\mathbf{L}'_t[r_i,\cdot]=\mathbf{L}_t[r,\cdot],\qquad\text{which implies}\qquad\sum_{i=1}^{\ell}\lambda_i\mathbf{A}'_t[r_i,\cdot]=\mathbf{A}'_t[r,\cdot].$$

Note that the implication follows from post-multiplying both sides of the first equation with $\mathbf{R}_t$. In particular, the latter implies that $\sum_{i=1}^{\ell}\lambda_i\mathbf{A}'_t[r_i,c]\neq 0$. Thus $\mathbf{A}'_t[r_i,c]\neq 0$ for some i, as required.

For the Boolean rank factorization, let $\mathbf{L}_t$ and $\mathbf{R}_t$ be the matrices of the explicit factorization (note that now the factorization is over the $\wedge-\vee$ semi-ring). Construct the matrix $\mathbf{L}_t[X,\cdot]$ explicitly and let $X'\subseteq X$ be all elements $r\in X$ for which there is a c such that $\mathbf{L}_t[r,c]=1$ and $\mathbf{L}_t[r',c]=0$ for every $r'\in X\setminus r$. It is clear that X' is a representative set of X since the set of columns with a cell with value 1 is by construction the same in $\mathbf{L}_t[X,\cdot]$ and in $\mathbf{L}_t[X',\cdot]$. This computes a representative set in $|X|\cdot r\cdot\text{polylog}(r)$ time. $\square$

3.2 Field Rank: Partitions and Matchings

We now introduce two matrices that express connectivity of subgraphs.

Partitions Connectivity Matrix. The following matrix was instrumental for the derandomization of the Cut&Count approach [CNP+11] from [BCKN15].

Definition 3.3. *For $t\geq 0$, define matrix $\mathbf{P}_t\in\{0,1\}^{\Pi([t])\times\Pi([t])}$ as*

$$\mathbf{P}_t[p,q]=\begin{cases}1, & \textit{if } p\sqcup q=\{[t]\},\\ 0, & \textit{otherwise.}\end{cases}$$

Suppose that t is odd and let $P,Q\subseteq\Pi([t])$ be all partitions with one block of size $(t-1)/2+1$ that contains the element 1 and all other blocks singleton. It is easy to see that for $p\in P$ and $q\in Q$ we have $p\sqcup q=\{[t]\}$ if and only if the non-singleton blocks of p and q are X and $([t]\setminus X)\cup 1$, for some $X\subseteq[t]\setminus 1$. This shows that $\texttt{indMatch}(\mathbf{P}_t)$ roughly 2^t. We continue with showing that the rank of $\mathbf{P}_t$ over $\mathbb{F}_2$ is only slightly higher. To do so we first define the factorizing matrices:

Definition 3.4. *For $t \geq 0$, define matrix $\mathbf{F}_t \in \{0,1\}^{\Pi([t]) \times \Pi_2([t]))}$ that has rows index by partitions and columns indexed by cuts as*

$$\mathbf{F}_t[p, \{X, Y\}] = \begin{cases} 1, & \text{if } p \text{ refines } \{X, Y\}, \\ 0, & \text{otherwise.} \end{cases}$$

Since there are at most 2^t cuts the following implies the promised rank upper bound:

Lemma 3.2 (Cut&Count factorization). $\mathbf{P}_t \equiv_2 \mathbf{F}_t \cdot \mathbf{F}_t^T$.

Proof. Let $p, q \in \Pi([t])$. By expanding the definition of matrix multiplication, we have that

$$(\mathbf{F}_t \cdot \mathbf{F}_t^T)[p, q] = \sum_{\{X,Y\} \in \Pi_2([t])} [p \sqsubseteq \{X, Y\}] \cdot [q \sqsubseteq \{X, Y\}].$$

Since $(\Pi([t]), \sqsubseteq)$ is a lattice, $p, q \sqsubseteq \{X, Y\}$ is equivalent with $p \sqcup q \sqsubseteq \{X, Y\}$ and we can rewrite into

$$= \sum_{\{X,Y\} \in \Pi_2([t])} [p \sqcup q \sqsubseteq \{X, Y\}]$$

The number of cuts that coarsen a partition is exactly its number of blocks minus 1 since for each component we can choose a side and divide by 2 because of a cut is an unordered pair.

$$\begin{aligned} &= 2^{\#\texttt{blocks}(p \sqcup q) - 1} \\ &\equiv_2 [\#\texttt{blocks}(p \sqcup q) = 1] \quad = \quad [p \sqcup q = \{[t]\}] \quad = \quad \mathbf{P}_t[p, q]. \end{aligned}$$ □

It can also be shown that $\mathrm{rk}_{\mathbb{R}}(\mathbf{P}_t) \leq 4^t$ using the 'squared determinant approach' from [BCKN15].

Matchings Connectivity Matrix. Note that the aforementioned construction of an induced matching of $\mathbf{P}_t$ crucially relies on partitions with many singleton blocks. A natural question is how large induced matchings exist in the submatrix of $\mathbf{P}_t$ induced by all partitions without singleton blocks. While the answer to this question is not known,[3] significant progress was made on the following even smaller submatrix of $\mathbf{P}_t$:

Definition 3.5. *For $t \geq 0$, define matrix $\mathbf{H}_t \in \{0,1\}^{\Pi_m([t]) \times \Pi_m([t])}$ as*

$$\mathbf{H}_t[P, Q] = \begin{cases} 1, & \text{if } P \cup Q \text{ is a Hamiltonian Cycle}, \\ 0, & \text{otherwise.} \end{cases}$$

We now define a family of matchings of $\mathbf{H}_t$ that are crucial to understand the structure of $\mathbf{H}_t$. See Fig. 1 for an illustration.

[3] At least, to the author.

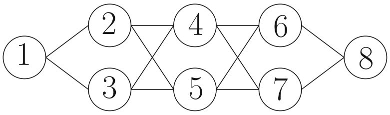

Fig. 1. The graph Z_8.

Definition 3.6 (Basis matchings). *Let $t \geq 2$ be an even integer, and let $Z_t = ([t], E)$ be a graph with vertices $[t]$ and edges $E = \{\{i,j\} : \lfloor j/2 \rfloor = \lfloor i/2 \rfloor + 1\}$. Define $\mathcal{X}_t$ to be the set of perfect matchings of Z_t.*

It can be shown that for every perfect matching M of Z_t there is a unique different perfect matching $\overline{M}$ such that $M \cup \overline{M}$ is a Hamiltonian cycle of Z_t. This proves that $\texttt{indMatch}_t(\mathbf{H}_t) \geq |\mathcal{X}_t|$. This bound turns out to be tight. Even stronger, it turns out that $\mathcal{X}_t$ is a row-basis of $\mathbf{H}_t$, and thus $\mathrm{rk}_{\mathbb{F}_2}(\mathbf{H}_t) = |\mathcal{X}_t| = 2^{t/2-1}$, by virtue of the following factorization:

Lemma 3.3 ([CKN13]). *If P, Q are two perfect matchings of K_t, then*

$$\mathbf{H}_t[P,Q] \equiv_2 \sum_{M \in \mathcal{X}_t} [P \cup M \textit{ is an Ham. Cycle }] \cdot [Q \cup \overline{M} \textit{ is an Ham. Cycle}].$$

Let us remark that other variants of the rank of $\mathbf{H}_t$ over the reals also have been studied. In [RS95], the authors showed that if $\mathbf{H}_t$ is restricted to all perfect matchings on the complete balanced bipartite graph on t vertices, then its rank is $\binom{t-2}{t/2-1}$. Their motivation was to disprove the original formulation of the 'log-rank conjecture' in communication complexity. They achieved this by relating their rank bound to a second bound: The non-deterministic communication complexity of the same submatrix of $\mathbf{H}_t$ is $\Omega(n \log\log n)$. In [CLN18] the authors showed that $\mathrm{rk}_{\mathbb{R}}(\mathbf{A}_t)$ equals 4^t, modulo some poly(t) factors.

3.3 Boolean Rank: Disjointness Matrix

We now define one of the most well-studied families of matrices in the field of communication complexity:

Definition 3.7. *For $t \geq p, q \geq 1$, define matrix $\mathbf{D}_{t,p,q} \in \{0,1\}^{\binom{[t]}{p} \times \binom{[t]}{q}}$ as*

$$\mathbf{D}_{t,p,q}[P,Q] = \begin{cases} 1, & \textit{if } P \cap Q = \emptyset, \\ 0, & \textit{otherwise.} \end{cases}$$

This time, we focus on the Boolean rank:

Lemma 3.4. *For even k,* $\texttt{boolRank}(\mathbf{D}_{t,k/2,k/2}) = O(2^k k \log t)$.

Proof. We use the probabilistic method. Note that if $P \cap Q = \emptyset$, then $\Pr[P \subseteq S \wedge S \cap Q = \emptyset] = 2^{-k}$. Pick $S_1, \ldots, S_l$, where $l = 2^k \cdot k \log 20t$. If $P \cap Q = \emptyset$, the probability that there is no i such that $P \subseteq S_i$ and $S_i \cap Q = \emptyset$ is

$$(1 - 2^{-k})^l \leq \exp(-l/2^k) = \exp(-20k \log t) \leq 1/t^k.$$

By a union bound, with positive probability there exists an i such that $P \subseteq S_i$ and $S_i \cap Q = \emptyset$ for each of the $\binom{t}{k/2}^2 \leq t^k$ possible disjoint pairs P, Q. In particular, a family $S_1, \ldots, S_l$ with this property exists, and this can be used as the rectangle cover. $\square$

Note the above proof is standard in Communication Complexity[4]. The above argument can be generalized to upper bounds on the Boolean rank of $\mathbf{D}_{t,p,q}$ by choosing a different distribution of the S_i's, and also can be made explicit by employing techniques reminiscent to [AYZ95] to get the following result:

Lemma 3.5 ([FLPS16]). $\texttt{boolRank}(\mathbf{D}_{t,p,q}[P, Q]) = O(\binom{p+q}{p} 2^{o(p+q)} \log t)$*, and the associated factorization is semi-explicit, in the sense that it could be computed in* $O(\binom{p+q}{p} 2^{o(p+q)} t \log t)$ *time.*

3.4 Support Rank: Linear Independence and Bipartite Colorings

Sometimes, in order to compute small representative sets quickly, it may be needed to consider the rank of different matrices with the same support. Consider the following example: Let $\mathbf{A}$ be the complement of a $t \times t$ identity matrix. It is easily seen that $\texttt{indMatch}(\mathbf{A}) = 2$, but there is a large gap with the rank of $\mathbf{A}$ which typically is t or $t - 1$. We resolve this gap by studying the rank of a different matrix with same support.

Linear Independence. The following matrix expresses when two linear independent sets again form an linear independent set. It arises frequently especially due to connections with matroid theory.

Definition 3.8. *Let $\mathbb{F}$ be a field and let $\mathbf{M} \in \mathbb{F}^{R \times C}$ be a matrix. Define a matrix $\mathbf{L_M} \in \{0,1\}^{\binom{C}{p} \times \binom{C}{q}}$ as*

$$\mathbf{L_M}[P, Q] = \begin{cases} 1, & \textit{if } \mathrm{rk}_{\mathbb{F}}(\mathbf{M}[\cdot, P \cup Q]) = p + q, \\ 0, & \textit{otherwise}. \end{cases}$$

Note that, even if $p = q = 1$ and $\mathbf{M}$ is an identity matrix, then the matrix $\mathbf{L_M}$ is the complement of the $|C| \times |C|$ identity matrix which has high rank (as mentioned above). Therefore, indeed resorting to support rank is needed here to get a low rank factorization. Define $\bar{I} := [p + q] \setminus I$ to be the complement of I, and define $\Sigma I = \sum_{i \in I} i$.

[4] See e.g. http://www.tcs.tifr.res.in/~prahladh/teaching/2011-12/comm/lectures/l03.pdf.

Lemma 3.6 (Generalized Laplace Expansion, Lemma [CFK+15]). *Let* $\mathbf{M} \in \{0,1\}^{(p+q)\times(p+q)}$ *and let* $P, Q \subseteq [p+q]$ *with* $|P| = p$ *and* $|Q| = q$. *Then*

$$\det(\mathbf{M}) = (-1)^{\lceil p/2 \rceil} \sum_{I \subseteq [p+q], |I|=p} (-1)^{\Sigma I} \det(\mathbf{M}[I,P]) \cdot \det(\mathbf{M}[\bar{I},Q])$$

We start with employing generalized Laplace expansions to factorize $\mathbf{L}$ in a natural special case:

Lemma 3.7. *If* $p+q = |R|$, $\texttt{supRank}(\mathbf{L}_{\mathbf{M}}[P,Q]) = \binom{p+q}{p}$ *and the associated factorization is explicit.*

Proof. Define $\mathbf{L}'_{\mathbf{M}}[P,Q] = \det(\mathbf{M}[\cdot, P \circ Q])$, where the $\circ$ operator denotes concatenation (see Sect. 2). We will show that $\mathbf{L}'_{\mathbf{M}}$ has the same support as $\mathbf{L}'_{\mathbf{M}}$ and low rank over $\mathbb{F}$. As the determinant of a square matrix is non-zero if and only if it is of full rank, we have that $\det(\mathbf{M}[\cdot, P \circ Q])$ is non-zero if and only if $\mathrm{rk}_{\mathbb{F}}(\mathbf{M}[\cdot, P \cup Q]) = p+q$, as required. The lemma now is a consequence of the following factorization implied by Lemma 3.6.

$$\mathbf{L}'_{\mathbf{M}}[P,Q] = (-1)^{\lceil p/2 \rceil} \sum_{I \subseteq [p+q], |I|=p} (-1)^{\Sigma I} \det(\mathbf{M}[I,P]) \cdot \det(\mathbf{M}[\bar{I},Q]). \qquad \square$$

We continue with focusing on the case $p+q \ll |R|$. A natural idea is to pre-multiply $\mathbf{M}$ with a random $(p+q) \times n$ matrix. This indeed works if we allow for randomized algorithms, but with constant probability the sought independent set may become dependent. A derandomized version of this 'truncation' operation was presented in [LMPS18], leading to the following result:

Lemma 3.8 ([LMPS18]). $\texttt{supRank}(\mathbf{L}_{\mathbf{M}}) = \binom{p+q}{p}$, *and the associated factorization is explicit.*

This bound has quite diverse applications: For example, it generalizes and refines the rank bound from Lemma 3.2, and it even strengthens this bound in the special case that the partitions are 'unbalanced'. See [LMPS18] for more details.

Colorings Matrix. We now introduce a matrix that naturally arises in graph coloring problems. It was defined for this purpose in [JN18], but somewhat surprisingly also found an application in the area of online algorithms [BEKN18].

Definition 3.9. *For an integer* $c \geq 1$ *and bipartite graph* H *with parts* $X = \{x_1, \ldots, x_t\}$ *and* $Y = \{y_1, \ldots, y_{t'}\}$ *and ordered edges in* $X \times Y$, *define matrix* $\mathbf{C}_{c,H} \in \{0,1\}^{[c]^X \times [c]^Y}$ *as*

$$\mathbf{C}_{c,H}[p,q] = \begin{cases} 1, & \textit{if } p_i \neq q_j \textit{ for every } (i,j) \in E(H)), \\ 0, & \textit{otherwise.} \end{cases}$$

Note that even if H is a single edge, $\mathbf{C}_{c,H}$ is the complement of the $(c \times c)$ identity matrix. Therefore, indeed resorting to support rank is needed here to get a low rank factorization.

Lemma 3.9. $\texttt{supRank}(\mathbf{C}_{c,H}) = 2^t$, *and the associated factorization is explicit.*

Proof. Define a matrix $\mathbf{C}'_{c,H}$ as follows

$$\mathbf{C}'_{c,H}[p,q] = \prod_{(i,j)\in E(H)} (p_i - q_j).$$

Since the product vanishes whenever $p_i = q_j$ for some $(i,j) \in E(H)$ and it is the product of positive numbers otherwise, we see that indeed $\mathbf{C}'_{c,H}[p,q] \neq 0$ if and only if $\mathbf{C}_{c,H}[p,q] \neq 0$. Moreover, this matrix has a low rank factorization that follows directly from expanding the parentheses to state the polynomial in its standard form: In particular, we have that $\mathbf{C}'_{c,H}[p,q]$ equals

$$\begin{aligned}
&\prod_{(i,j)\in E(H)} (p_i - q_j) \\
&= \sum_{W\subseteq E(H)} \left(\prod_{i\in X} p_i^{d_W(i)}\right) \left(\prod_{j\in Y} (-q_j)^{d_{E(H)\setminus W}(j)}\right) \\
&= \sum_{(d_i\in\{0,\dots,d_{E(H)}(i)\})_{i\in X}} \left(\prod_{i\in X} p_i^{d_i}\right) \left(\sum_{\substack{W\subseteq E(H)\\ \forall i\in X: d_W(i)=d_i}} \prod_{j\in Y} (-q_j)^{d_{E(H)\setminus W}(j)}\right), \qquad (1)
\end{aligned}$$

where the second equality follows by expanding the product and the third equality follows by grouping the summands on the number of edges incident to vertices in W included in X. It is easily seen that (1) gives a factorization of $\mathbf{C}'_{c,H}$ of rank at most the maximum number of the possibilities for d, since the inner dimension of the implied factorization is indexed by the possible vectors d. These are vectors d with $|X|$ coordinates where each $d_i \in \{0,\dots,d_{|E(H)|}(i)\}$. The vector $d_{E(H)}$ that maximizes the number of such possible vectors while satisfying $\sum_{i\in X} d_{E(H)}(i) = k$ is the vector with k coordinates being equal to 1 by convexity (i.e., H is a matching) in which case the number of possibilities for $d_i \in \{0,1\}$ for all k vertices in X. $\square$

4 Using Low Rank Matrix Factorizations to Speed up Dynamic Programming

In this section we will use the insights from the previous section to speed up several natural dynamic programming algorithms. In Subsect. 4.1 this is a natural $O^*(q^k)$ time algorithm to decide whether a given graph with given permutation of cutwidth k has a proper q-coloring (see the section for definitions). We improve the runtime to $O^*(c^k)$ time where c is a constant independent of q.

In Subsect. 4.2 we study two connectivity problems parameterized by pathwidth and show they can be solved in $O^*(c^{pw})$ time by building on natural $O^*(pw^{pw})$ time dynamic programming algorithms.

Finally, in Subsect. 4.3 we present one of the first uses of representative sets to solve k-path in $O^*(c^k)$ by speeding up a natural $n^{O(k)}$ time algorithm.

4.1 Cutwidth

In this subsection we demonstrate the methods based on low rank factorizations on the graph coloring problem. Recall that in the graph coloring problem one is given an undirected graph $G = (V, E)$ and an integer q, and one is asked whether there exists a *proper coloring*, which is a vector $x \in [q]^V$ such that $x_v \neq x_w$ for every $\{v, w\} \in E$. Let $\{v_1, \ldots, v_n\} = V(G)$ be a linear ordering of its vertices.

We denote all edges as directed pairs (v_i, v_j) with $i < j$. For $i = 1, \ldots, n$, define V_i as the i'th prefix of this ordering, C_i as the i'th cut in this ordering, and X_i and Y_i as the left and respectively right endpoints of the edges in this cut, i.e.

$$\begin{aligned}
V_i &= \{v_1, \ldots, v_i\}, \\
C_i &= \{(v_l, v_r) \in E(G) : l \leq i < r\}, \\
X_i &= \{v_l \in V(G) : \exists (v_l, v_r) \in C_i \wedge l < r\}, \\
Y_i &= \{v_r \in V(G) : \exists (v_l, v_r) \in C_i \wedge l < r\}.
\end{aligned}$$

Note that $X_i \subseteq X_{i-1} \cup \{v_i\}$ and $Y_{i-1} \subseteq Y_i \cup \{v_i\}$. We let H_i denote the bipartite graph with parts X_i, Y_i and edge set C_i. We study the graph coloring problem in the setting where one is given a permutation of low cutwidth, which is defined as follows:

Definition 4.1. *The* cutwidth *of the linear order* $\{v_1, \ldots, v_n\}$ *is the maximum value of* $|C_i|$ *taken over all* i.

We use the following notation: A vector $x \in V^I$ is *an extension* of a vector $x' \in V^{I'}$ if $I' \subseteq I$ and $x'_i = x_i$ for every $i \in I'$. If $x \in V^I$ and $P \subseteq I$ then the projection $x_{|P}$ is defined as the unique vector in V^P of which x is an extension. For $i = 1, \ldots, n$, we define $T[i] \subseteq [q]^{X_i}$ to be the set of all q-colorings of the vertices in X_i that can be extended to a proper q-coloring of $G[V_i]$. The following lemma allows to compute representative sets of $T[i]$.

Lemma 4.1 ([JN18]). *If* $T'[i-1]$ *is a representative set of* $T[i-1]$ *with respect to* $\mathbf{C}_{q,H_{i-1}}$, *then* $T'[i]$ *is a representative set of* $T[i]$ *with respect to* $\mathbf{C}_{q,H_i}$, *where*

$$T'[i] = \Big\{ (x \cup (v_i, c))_{|X_i} : x \in T'[i-1], c \in [q], \big(\forall v \in N(v_i) \cap X_{i-1} \colon x_v \neq c\big) \Big\}.$$

We remark that the lemma is very similar to the recurrence underlying a standard $O^*(q^k)$ time dynamic programming algorithm for the task at hand, but it is formulated in the language of this survey in order to allow for a speed up via representative sets as we now outline:

Theorem 4.1 ([JN18]). *The graph coloring problem can be solved in $O^*(2^{\omega \cdot k})$ time, assuming a linear order of cutwidth at most k is given.*

Proof. Compute $T'[0] = T[0]$ to be the singleton set with only the unique zero-dimensional vector. The for each $i = 1, \ldots, n$ to the following: First use Lemma 4.1 to compute $T'[i]$ from $T'[i-1]$. After each such step, use Lemma 3.1 with the explicit factorization of Lemma 3.9 to compute a subset $T''[i]$ of $T'[i]$ that represents $T'[i]$ with respect to $\mathbf{C}_{q,H_i}$. By transitivity it also represents $T[i]$ and we can set $T'[i] := T''[i]$ and continue with computing $T'[i+1]$. In the end we can check whether a proper q-coloring exists since it does if and only if $T[n]$ (and thus $T'[n]$) is non-empty by definition of $T[n]$. The run time follows since the number of partial solutions is at most 2^k at every step and the bottleneck is due to the application of Lemma 3.1. $\square$

4.2 Pathwidth

A *path decomposition* of a graph $G = (V, E)$ is a path $\mathbb{P}$ in which each 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 of a graph G is the minimum pathwidth over all possible path decompositions of G. We define *nice path decompositions* as follows.

Definition 4.2 (Nice Path Decomposition). *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 of one of the following types:*

First bag: *the bag associated with the leftmost node x is empty, $B_x = \emptyset$.*
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 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 $\{u, v\}$, *and every edge is introduced by exactly one bag.*
Forget bag: *an internal node x of $\mathbb{P}$ with one predecessor y for which $B_x = B_y \setminus \{v\}$ for some $v \in B_y$. This bag is said to* forget v.
Last bag: *the bag associated with the rightmost node x is empty, $B_x = \emptyset$.*

It is easy to verify that any given path decomposition of pathwidth pw can be transformed in time $|V(G)|pw^{O(1)}$ into a nice path decomposition without increasing the width. For a bag B_i, we define the $G_i = (\cup_{j=1}^{i} B_j, E_i)$ where E_i are all edges introduced in bags $B_1, \ldots, B_i$.

Steiner Tree. In the Steiner Tree problem[5] one is given an undirected graph G, a vertex subset $K \subseteq V(G)$, and an integer s. The goal is to determine if there exists a subset $K \subseteq Y \subseteq V(G)$ such that $|Y| \leq s$ and $G[Y]$ is connected. As in the previous case studies, we first present a recurrence that allows to gradually build partial solutions. To facilitate this we use the following notation:

Definition 4.3. *Given a graph G', a subset $Y \subseteq V(G')$ and a partition $p \in \Pi(X)$ where $X \subseteq Y$, we say that Y* connects p in G' *if for every two vertices $u, v \in X$ the following holds: u and v are connected in $G'[Y]$ if and only if u and v are in the same block in p.*

For a bag B_i, a subset X and an integer s we define $T[i, X, s]$ to be the set of partitions $p \in \Pi(X)$ such that there exists a subset $Y \subseteq V(G_i)$ that connects p in G_i and satisfies $K \subseteq Y$, $|Y| \leq s$ and

$$\forall u \in Y \ \exists v \in X \ : u \text{ and } v \text{ are connected in } G_i[Y].$$

We now show how to compute entries $T[i, \cdot, \cdot]$ given the appropriate entries $T[i-1, \cdot, \cdot]$, by distinguishing on what kind of bag X_i is:

First Bag. If i is the first bag, $T[i, X, s] = \{\varepsilon\}$, where ε is the empty partition.

Introduce Vertex Bag. If B_i introduces a vertex v, note that G_i contains v as an isolated vertex (as we did not introduce any of its incident edges). If $v \in K$ it needs to be included in X. Hence, if $v \notin X$ we have that

$$T[i, X, s] = \begin{cases} \emptyset & \text{if } v \in K \text{ and } v \notin X, \\ T[i-1, X, s], & \text{if } v \notin K \text{ and } v \notin X. \end{cases}$$

Moreover if v is included in the solution, it should also be included in the partitions as a singleton:

$$T[i, X \cup \{v\}, s] = \{p \cup \{\{v\}\} \ | \ p \in T[i-1, X, s-1]\}.$$

Introduce Edge Bag. If an edge $\{u, v\}$ is introduced in B_i we have that $T[i, X, s] = T[i-1, X, s]$ if $\{u, v\} \not\subseteq X$, and otherwise

$$T[i, X, s] = \{p \sqcup \{\{u, v\}\} \ | \ p \in T[i-1, X, s]\}.$$

Note that here $\{u, v\}$ denoted the partition of X with $\{u, v\}$ as only non-trivial block.

Forget Vertex Bag. If a vertex v is forgotten in B_i, all partitions in $T[i-1, X, s]$ remain in $T[i, X, s]$ and all partitions in

$$T[i-1, X \cup \{v\}, s]$$

remain in $T[i, X, s]$ if they do not include v as a singleton:

$$T[i, X, s] = T[i-1, X, s] \cup \left\{p_{|X} \ | \ p \in T[i, X \cup \{v\}], \{v\} \notin p\right\}.$$

[5] For ease of exposition, we discuss a less general variant of the Steiner tree problem. The same methods can also solve more general versions within time that only depends linearly on the number of vertices, see [BCKN15] or the exposition in [CFK+15].

With all recurrences in place, we are ready to sketch the algorithm for Steiner tree:

Theorem 4.2. *Given a graph G and a path decomposition of G of width pw, any instance of Steiner tree on G can be solved in $O^*((1+2^\omega)^{pw})$ time.*

Proof Sketch. Let $\{B_i\}_{i=1}^{l}$ be the path decomposition. For every X and s, we compute a family of partitions $T'[i,X,s]$ that is a representative set for $T[i,X,s]$ with respect to $\mathbf{P}_{|X|}$, based on representative sets $T'[i-1,X',s]$ of $T[i-1,X',s]$ with respect to $\mathbf{P}_{|X'|}$. By following the above recurrence (but with all occurrences of $T[\cdot,\cdot,\cdot]$ with $T'[\cdot,\cdot,\cdot]$). It can be shown that in all cases indeed the resulting set $T'[i,X,s]$ is representative of $T[i,X,s]$. Alternating this computation with the table reduction procedure from Lemma 3.1 ensures $|T'[i,X,s]| \leq 2^{|X|}\mathrm{poly}(n)$ and it runs in $2^{\omega|X|}$ time. Summing over all possibilities for X per bag, the run time becomes $n^{O(1)}\sum_{X\subseteq B_i} 2^{\omega|X|} = n^{O(1)}(1+2^\omega)^{pw}$ time. □

For completeness, we remark that the Steiner tree problem can be solved in $O^*(3^{pw})$ time by a randomized algorithm [CNP+11].

Hamiltonian Cycle. In the Hamiltonian cycle problem one is given an undirected graph G, and is asked whether there exists a simple cycle $C \subseteq E(G)$ with $|C| = n$.

Definition 4.4. *Given a graph G', a subset $Y \subseteq E(G')$ and a partition $p \in \Pi(X)$ where $X \subseteq V(G')$, we say that Y* connects *p if for every two vertices $u,v \in X$ the following holds: u and v are connected in $(\cup_{e\in Y} e, Y)$ if and only if u and v are in the same block in p.*

For a bag B_i, a vector $d \in \{0,1,2\}^{B_i}$ we define $T[i,d]$ to be

$$\Big\{ M \in \Pi_m(d^{-1}(1)) \Big| \exists Y \subseteq E(G_i) : Y \text{ connects } p \wedge \forall v \in B_i : d_Y(v) = d_v \\ \wedge \forall v \in V(G_i)\setminus B_i : d_Y(v) = 2 \Big\},$$

where we let $d_Y(v)$ denote the number of edges in Y that is incident to v.

Similar to the algorithm for Steiner tree, a recurrence for $T[i,d]$ in terms of $T[i-1,d]$ can be formulated, and the same recurrence can be used to compute a set $T'[i,d]$ that is a representative set of $T[i,d]$ with respect to $\mathbf{H}_{|d^{-1}(1)|}$ from entries of the type $T'[i-1,d]$ that are representative sets of $T[i-1,d]$ with respect to $\mathbf{H}_{|d^{-1}(1)|}$. We refer to [BCKN15] for details.

By interleaving these computations with an algorithm implied by Lemma 3.1 with the matrix $\mathbf{H}_t$ and its factorization from Lemma 3.3 we can obtain the following theorem in a way similar to the previous sections:

Theorem 4.3 ([BCKN15]). *Given a graph G with path decomposition of width pw, it can be determined in $O^*((2+2^{\omega/2})^{pw})$ time whether G has a Hamiltonian cycle.*

In fact, the same running time can be obtained for the weighted version of the problem (the Traveling Salesman Problem). We would also like to mention that the problem can be solved in $O^*((2+\sqrt{2})^{pw})$ time with a randomized algorithm [CKN13].

4.3 *k*-Path

In the k-path problem one is given a graph $G = ([n], E)$ and an integer k. The task is to determine whether G has a path on at least k vertices. Recall a path is a sequence of vertices such that consecutive vertices are adjacent and each vertex occurs at most once in the sequence. We outline an approach that was originally described in the paper that introduced representative sets [Mon85][6] For every $i = 1, \dots, k$ and $v \in V$ we define

$$T[i,v] = \left\{ X \in \binom{[n]}{i} \middle| \exists \text{ path that ends at } v \text{ and visits } X \right\}.$$

By trying all possibilities for the penultimate vertex v' in the path the following recurrence can be obtained:

$$T[i,v] = \{X \cup \{v\} : X \in T[i-1,v'], v \in N(v')\}.$$

Similarly we have that

Lemma 4.2. *If $T'[i-1,v']$ is a representative set of $T[i-1,v']$ with respect to $\mathbf{D}_{n,i-1,k-(i-1)}$, then $T'[i,v]$ is a representative set of $T[i,v]$ with respect to $\mathbf{D}_{n,i,k-i}$ where*

$$T'[i,v] = \{X \cup \{v\} : X \in T'[i-1,v'], v \in N(v)\}.$$

Similarly as before, we use Lemma 3.1 in combination with Lemma 4.2 to obtain the following result:

Theorem 4.4. *Given a graph G and an integer k, it can be determined in $O^*(4^k)$ time whether G has a path on at least k vertices.*

Proof. Compute $T'[1,\{v\}] = T[1,\{v\}] = \{\{v\}\}$. For $i = 2, \dots, k$ do the following: Compute $T'[i,v]$ as defined in Lemma 4.2 for every $v \in V$. Afterwards use Lemma 3.1 to compute a set $T''[i,v]$ that is a representative set of $T'[i,v]$ with respect to $\mathbf{D}_{n,i,k-i}$. By Lemma 3.1, $|T''[i,v]| \leq \binom{k}{i}$ and the time required to compute the set is at most 4^k. By transitivity, it will also be a representative set of $T[i,v]$ and we can set $T'[i,v] = T''[i,v]$ and use it in the next iteration to compute a family that is a representative for $T[i+1,v]$.

Afterwards, we can return whether G has a path on at least k vertices since it does if and only if $T'[k,v]$ is non-empty for some vertex v. ◻

[6] Indeed, the idea of representing partial solutions with a strict subset is natural, but to the author's knowledge [Mon85] was the first paper (in parameterized complexity) to use a generalization of this concept beyond equivalence classes.

Let us remark for completeness that the currently fastest deterministic algorithm for k-path refines the above approach and solves the problem in $O^*(2.597^k)$ time [Zeh15]. In the randomized setting, a beautiful algorithm from [BHKK17] solves the problem in $O^*(1.66^k)$ time.

5 Other Relevant Directions

This survey focused on only few applications in the area of parameterized complexity. We list a few of the most relevant directions not yet discussed.

5.1 Pair Problems

For a fixed family of explicit matrices $\{\mathbf{A}_t\}_t$, we may study the following problem PAIR($\mathbf{A}$): Given an integer t and sets P, Q such that $\mathbf{A}_t \in \{0,1\}^{R \times C}$ and $P \subseteq R$ and $Q \subseteq C$, the goal is to detect whether there exists $p \in P$ and $q \in Q$ such that $\mathbf{A}_t[p,q] = 1$. Freivalds [Fre77] famous matrix multiplication algorithm can be used to obtain the following result by computing $\mathbf{A}_t = (r\mathbf{L}_t)(\mathbf{R}_t r')$ with random vectors $r \in \{0,1\}^P$ and $r' \in \{0,1\}^Q$:

Observation 5.1. *If $\mathbf{A}_t$ has an explicit[7] field, support or Boolean rank r_t factorization, then an instance (t, P, Q) of* PAIR($\mathbf{A}$) *can be solved with a randomized algorithm in $r_t(|P| + |Q|) \cdot \mathrm{polylog}(t))$ time.*

An interesting special case is PAIR($\mathbf{D}_{t,p,q}$), also known as the *orthogonal vectors* problem. Several algorithms for this problem have been developed that rely on interesting rank parameters of $\mathbf{D}_{t,p,q}$ such as the rank over the reals [BHKK09] and an intriguing variant of 'probabilistic rank' [AWY15, AW17].

An especially interesting theme is that of *sparse factorizations*. That is, a factorization $\mathbf{A}_t = \mathbf{L}_t\mathbf{R}_t$ such that both $\mathbf{L}$ and $\mathbf{R}$ are relatively sparse.

Sparse factorizations for PAIR($\mathbf{D}_{t,p,q}$) are used in for example in [FLPS16]. In an unpublished note [Ned17], the author observed that if a natural algorithm that relies on a sparse Boolean rank decomposition of Lemma 3.5 can be improved slightly, a classic algorithm for the Subset Sum problem can be improved.

In a very recent work [Ned19] on improving the Bellman-Held-Karp algorithm for the Traveling Salesman Problem, the author studied the problem PAIR($\mathbf{H}$). That is, given two families of perfect matchings $P, Q \subseteq \Pi_m([t])$, determine whether there exist perfect matchings $p \in P$ and $q \in Q$ that form a Hamiltonian cycle of the complete graph K_t on t vertices. By combining the rank bound $\mathrm{rk}_{\mathbb{F}_2}(\mathbf{H}_t) = 2^{t/2-1}$ with Observation 5.1 this problem can be solved in $O((|P| + |Q|)2^{t/2}t^{O(1)})$ time with a randomized algorithm. In [Ned19] an $O(((|P| + |Q|)2^{3t/10} + 3^{t/2})t^{O(1)})$ time randomized algorithm was given that was instrumental to obtain a new result on TSP. Curiously this faster algorithm for PAIR($\mathbf{H}$) uses a factorization of $\mathbf{H}$ of *higher* rank, but since it is much *sparser* it is nevertheless more useful to solve PAIR($\mathbf{H}$). We refer to [Ned19] for more discussion and details.

[7] As a minor technical caveat, both $\mathbf{L}_t$ and $\mathbf{R}_t$ need to be explicit.

5.2 Matrix Multiplication

It should be noted that often the use of Lemma 3.1 as described in Sect. 4 does not yield the fastest algorithms, as these rely on more algebraic ideas such as Observation 5.1. At a high level, these algorithms are obtained by applying the low rank factorization at a more general level. Slightly more formal, one could see many dynamic programming algorithms as evaluating a chain of matrix multiplications $\mathbf{A}_1\mathbf{A}_2 \cdot \mathbf{A}_l$ where $\mathbf{A}_1$ is a row vector and $\mathbf{A}_l$ is a column vector. Given low-rank factorization of these matrices, their product can be evaluated quickly if their products are evaluated in a clever order. If the factorizations are over finite fields or in terms of support rank, typically standard tools in complexity theory such as polynomial identity testing or the isolation lemma can be used to solve the (unweighted variants) of the decision problems by introducing randomization.

Notably, the algorithms obtained via this method are often known to be optimal under the Strong Exponential Time Hypothesis[8] (SETH). For example, this gives rise to an $O^*(3^{pw})$ time algorithm for Steiner Tree, an $O^*((2+\sqrt{2})^{pw})$ algorithm for Hamiltonian cycle, and an $O^*(2^k)$ time algorithm for graph coloring (where k is the cutwidth of a given permutation). Furthermore, these algorithms cannot be improved to $O^*((3-\varepsilon)^{pw})$ time, $O^*((2+\sqrt{2}-\varepsilon)^{pw})$ time, and $O^*((2-\varepsilon)^k)$ time for any $\varepsilon > 0$, unless the SETH fails.

5.3 Counting Algorithms

If the number solutions needs to be counted instead of detecting only one, only the rank over the reals can be applied in general. Instead, if one needs to count the number of solutions modulo a prime p, the rank over $\mathbb{Z}_p$ can be used.

A particularly non-trivial case is that of counting Hamiltonian cycles parameterized by the pathwidth. In [CLN18] a general connection between the complexity of the problem and the rank of $\mathbf{H}$ was shown:

Theorem 5.1. *Let $r = \lim_{t\to\infty} \log_2(\mathrm{rk}(\mathbf{H}_t))/t$. Assuming SETH, there is no $\varepsilon > 0$ such that the number of Hamiltonian cycles can be computed in $O^*((2+r-\epsilon)^{pw})$ time on graphs with a given path decomposition of width pw. For prime numbers p, the same applies to counting Hamiltonian cycles modulo p when replacing r by r_p, which is defined analogously to r by taking the rank over $\mathbb{Z}_p$.*

Determining the rank of $\mathbf{H}_t$ over various fields turns out a challenging job. Over the reals, it was shown in [CLN18] that the rank of $\mathbf{H}$ is (up to factors polynomial in t) equal to 4^t. Thus, by Theorem 5.1 the existing $O^*(6^{pw})$ algorithm from [BCKN15] cannot be significantly improved, assuming SETH.

---

[8] This hypothesis postulates that for every $\varepsilon > 0$ there is an integer k such k-CNF satisfiability on n variables cannot be solved in $O^*((2-\varepsilon)^n)$ time.

5.4 Further Results

This survey is far from exhaustive and biased towards the familiarity of the author. Other interesting connections between fine-grained complexity can be found in papers on the *probabilistic rank* [AW17], Waring rank [Pra18]. Since many algorithms on fine-grained complexity of hard problems rely on *fast matrix multiplication*, the rich theory underlying these fast algorithms that features a plethora of variants (tensor) rank can also be considered to be in the same category.

Let us conclude by remarking that studying problem specific tensors arising from divide and conquer algorithms that merge triples of partial solutions into a complete solution may be a good source of further research opportunities.

Acknowledgements. The author would like to thank Johan van Rooij and Stefan Kratsch for their valuable feedback on a previous version of this survey.

References

[AW17] Alman, J., Williams, R.R.: Probabilistic rank and matrix rigidity. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, 19–23 June 2017, pp. 641–652 (2017)

[AWY15] Abboud, A., Williams, R.R., Yu, H.: More applications of the polynomial method to algorithm design. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, 4–6 January 2015, pp. 218–230 (2015)

[AYZ95] Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM **42**(4), 844–856 (1995)

[BCKN15] Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015)

[BCS97] Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Grundlehren der mathematischen Wissenschaften, vol. 315. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-662-03338-8

[BEKN18] Bansal, N., Eliás, M., Koumoutsos, G., Nederlof, J.: Competitive algorithms for generalized k-server in uniform metrics. In: Czumaj, A., (ed.), Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, 7–10 January 2018, pp. 992–1001. SIAM (2018)

[BHKK09] Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Counting paths and packings in halves. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 578–586. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04128-0_52

[BHKK17] Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Narrow sieves for parameterized paths and packings. J. Comput. Syst. Sci. **87**, 119–139 (2017)

[CFK+15] Cygan, M., et al.: Parameterized Algorithms. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-21275-3

[CKN13] Cygan, M., Kratsch, S., Nederlof, J.: Fast Hamiltonicity checking via bases of perfect matchings. In: Symposium on Theory of Computing Conference, STOC 2013, Palo Alto, CA, USA, 1–4 June 2013, pp. 301–310 (2013)

[CLN18] Curticapean, R., Lindzey, N., Nederlof, J.: A tight lower bound for counting Hamiltonian cycles via matrix rank. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, 7–10 January 2018, pp. 1080–1099 (2018)

[CNP+11] Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: Ostrovsky, R. (ed.), IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 150–159. IEEE Computer Society (2011)

[FLPS16] Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 29:1–29:60 (2016)

[Fre77] Freivalds, R.: Probabilistic machines can use less running time. In: Information Processing, Proceedings of the 7th IFIP Congress 1977, Toronto, Canada, 8–12 August 1977, pp. 839–842 (1977)

[JN18] Jansen, B.M.P., Nederlof, J.: Computing the chromatic number using graph decompositions via matrix rank. In: Azar, Y., Bast, H., Herman, G. (eds.) 26th Annual European Symposium on Algorithms, ESA 2018, 20–22 August 2018, Helsinki, Finland. LIPIcs, vol. 112, pp. 47:1–47:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)

[KN97] Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)

[KW14] Kratsch, S., Wahlström, M.: Compression via matroids: a randomized polynomial kernel for odd cycle transversal. ACM Trans. Algorithms **10**(4), 20:1–20:15 (2014)

[LMPS18] Lokshtanov, D., Misra, P., Panolan, F., Saurabh, S.: Deterministic truncation of linear matroids. ACM Trans. Algorithms **14**(2), 14:1–14:20 (2018)

[Loz02] Lozin, V.V.: On maximum induced matchings in bipartite graphs. Inf. Process. Lett. **81**(1), 7–11 (2002)

[LS88] Lovász, L., Saks, M.E.: Lattices, Möbius functions and communication complexity. In: 29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24–26 October 1988, pp. 81–90. IEEE Computer Society (1988)

[Mat10] Matoušek, J.: Thirty-Three Miniatures: Mathematical and Algorithmic Applications of Linear Algebra, vol. 53. American Mathematical Society, Providence (2010)

[Mon85] Monien, B.: How to find long paths efficiently. North-Holland Math. Stud. **109**, 239–254 (1985). Analysis and Design of Algorithms for Combinatorial Problems

[Ned17] Nederlof, J.: Faster subset sum via improved orthogonal vectors (2017). http://www.win.tue.nl/~jnederlo/problem.pdf

[Ned19] Nederlof, J.: Bipartite TSP in $O(1.9999^n)$ time, assuming quadratic time matrix multiplication (2019). Unpublished

[Pra18] Pratt, K.: Faster algorithms via waring decompositions. In: The Proceedings of FOCS 2018 (2018, to appear)

[RS95] Raz, R., Spieker, B.: On the "log rank"-conjecture in communication complexity. Combinatorica **15**(4), 567–588 (1995)

[Wil14] Williams, R.: The polynomial method in circuit complexity applied to algorithm design (invited talk). In: Raman, V., Suresh, S.P. (eds.), 34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, 15–17 December 2014, New Delhi, India. LIPIcs, vol. 29, pp. 47–60. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2014)

[Wol03] de Wolf, R.: Nondeterministic quantum query and communication complexities. SIAM J. Comput. **32**(3), 681–699 (2003)

[Zeh15] Zehavi, M.: Mixing color coding-related techniques. In: Bansal, N., Finocchi, I. (eds.) ESA 2015. LNCS, vol. 9294, pp. 1037–1049. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48350-3_86