



ELSEVIER

Available online at www.sciencedirect.com

 ScienceDirect

Discrete Optimization 4 (2007) 154–162

DISCRETE
OPTIMIZATION

www.elsevier.com/locate/disopt

A polynomial time equivalence between DNA sequencing and the exact perfect matching problem[☆]

Jacek Błażewicz^{a,b}, Piotr Formanowicz^{a,b,*}, Marta Kasprzak^{a,b}, Petra Schuurman^c,
Gerhard J. Woeginger^c

^a *Institute of Computing Science, Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland*

^b *Institute of Bioorganic Chemistry, Polish Academy of Sciences, Noskowskiego 12/14, 61-704 Poznań, Poland*

^c *Department of Mathematics and Computer Science, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands*

Received 21 October 2004; received in revised form 11 July 2006; accepted 20 July 2006

Available online 22 December 2006

Abstract

We investigate the computational complexity of a combinatorial problem that arises in DNA sequencing by hybridization: The input consists of an integer ℓ together with a set S of words of length k over the four symbols A, C, G, T . The problem is to decide whether there exists a word of length ℓ that contains every word in S at least once as a subword, and does not contain any other subword of length k .

The computational complexity of this problem has been open for some time, and it remains open. What we prove is that this problem is polynomial time equivalent to the exact perfect matching problem in bipartite graphs, which is another infamous combinatorial optimization problem of unknown computational complexity.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Graph theory; Computational complexity; Computational biology; DNA sequencing

1. Introduction

This paper is centered around two algorithmic problems. The first problem is the *Exact Perfect Matching* problem that asks whether a given edge weighted graph possesses a perfect matching with weight exactly equal to a given bound. This problem is of great practical importance, and it has applications in bus-driver scheduling, in biomedical image analysis, and in the Ising model in theoretical physics; see Leclerc [10]. In 1982, Papadimitriou and Yannakakis [14] observed that this problem is NP-hard when the weights are encoded in binary, and they asked about its complexity when the weights are encoded in unary. Barahona and Pulleyblank [2] and Leclerc [11] show that

[☆] Part of this work has been presented at 28th International Workshop on Graph Theoretical Concepts in Computer Science (WG 2002) and has been published as a conference paper in the LNCS series of Springer-Verlag. The research has been partially supported by the Polish Ministry of Science and Higher Education grant No. 3T11F00227.

* Corresponding author at: Institute of Computing Science, Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland. Tel.: +48 61 8528503; fax: +48 61 8771525.

E-mail address: piotr.formanowicz@cs.put.poznan.pl (P. Formanowicz).

some special cases (like the case of planar graphs) are polynomially solvable for unary encoded weights. Mulmuley et al. [13] show that the Exact Perfect Matching problem with unary encoded weights lies in the complexity class RP, and consequently has a *randomized* polynomial time solution algorithm. However, determining the exact deterministic complexity of this problem still remains unsettled after twenty years, and by now is recognized as an outstanding open problem. A slightly simpler looking, but equally open variant is the following restriction of the exact perfect matching problem to bipartite graphs:

Problem: Exact bipartite matching (EX-MATCH)

Input: A bipartite multi-graph $(X \cup Y, E)$ where E is a multi-subset of $X \times Y$. Non-negative integer weights $w(e)$ on the edges $e \in E$ that are encoded in unary. An integer α .

Question: Does this graph have a perfect matching of weight exactly α ?

This bipartite variant is also discussed by Barahona and Pulleyblank [2]. Karzanov [9] gives a polynomial time algorithm for the special case where the input graph is a complete bipartite graph, and where the edge weights are restricted to 0 and 1.

Now let us turn to the second main problem that will be investigated in this paper. Deoxyribonucleic acid (DNA, for short) exists in the form of a double helix that consists of two twisted strings of *nucleotides*. These nucleotides only differ in their nitrogenous bases adenine (A), cytosine (C), guanine (G), and thymine (T). Their order codes genetic information that is symbolically written as a sequence over the four letters A, C, G, and T. The first stage of discovering genetic information is to analyze and to determine this sequence of bases for a given DNA sequence (see for instance Bains and Smith [1]). The length of the sequence can be determined by so-called gel electrophoresis. One method of sequence structure analysis is based on *hybridization* experiments that compare a DNA chain against a library of all possible single-stranded DNA fragments of length k . The outcome of the hybridization is the *k-spectrum* of the sequence, i.e., the set of all fragments of length k of this sequence.

In the ideal case no errors occur in the hybridization experiment, and the k -spectrum of a DNA sequence of length ℓ consists of exactly $\ell - k + 1$ pairwise distinct words of length k . This ideal case with a complete k -spectrum is well-understood (see Theorem 1), but never occurs in the real world. In the real world, the hybridization experiments suffer from *negative* errors (if the measured spectrum is incomplete) and from *positive* errors (if the measured spectrum contains additional elements that do not show up in the sequence). Especially, a fragment of length k may appear *many* times in the sequence, whereas the experiments only detect that it occurs at least *once*. DNA sequencing under negative and positive errors is discussed by Błażewicz et al. [4]. In a paper, Błażewicz and Kasprzak [6] investigate the computational complexity of several variants of DNA sequencing under negative and positive errors; all these variants turn out to be unary NP-hard in their search versions. However, the complexity of the following fairly innocent looking variant remained open.

Problem: DNA sequencing with unknown multiplicities (DNA-SEQ)

Input: Integers ℓ and k . A set S of words of length k over the alphabet $\{A, C, G, T\}$.

Question: Does there exist a word of length ℓ with k -spectrum equal to S ?

In this paper we will prove that the two problems EX-MATCH and DNA-SEQ are polynomially reducible to each other and hence are polynomial time equivalent. If one of them is polynomially solvable, then the other one is polynomially solvable, too. If one of them is NP-hard, then the other one is NP-hard, too. More generally, if one of these two problems is complete for some complexity class under polynomial time reductions, then so is the other problem.

Organization of this paper. The paper is a long (cyclic!) chain of polynomial time reductions that traverses a number of intermediate problems. In Section 2 we reduce problem DNA-SEQ to the problem of deciding the existence of certain Eulerian multi-graphs. Then in Section 3, this Eulerian problem EX-EULERCYCLE is reduced to problem EX-MATCH. Next, in Section 4 problem EX-MATCH is reduced to a highly restricted special case that we call EX-MATCH⁻. Finally, in Section 5 this highly restricted special case EX-MATCH⁻ will be reduced to problem DNA-SEQ. By doing this, we return to our starting point, and the chain of reductions is closed to a cycle.

Notation and definitions. Throughout the paper, we stick to the standard graph terminology as given in the book of Berge [3]. However, we want to clarify the usage of the following terms: A graph $G = (V, E)$ is *cubic*, if each vertex in V is incident to exactly three edges. For a multi-graph $G = (V, E)$, its *underlying simple graph* $G' = (V, E')$ is

the (unique) simple graph that results from G by keeping exactly one edge from every bundle of multiple edges. A multi-graph G is a *super-multi-graph* of a simple graph G' , if G' is the underlying simple graph of G . In other words, a super-multi-graph of the simple graph G' contains every edge in G' at least once, and does not contain any other edges.

A *path* is an alternating sequence $v_0, e_1, v_1, \dots, e_n, v_n$ of vertices and edges such that edge e_i always connects vertex v_{i-1} to v_i . A path is called *simple* if its vertices are pairwise distinct (except possibly the pair v_0 and v_n). A *cycle* is a path with $v_0 = v_n$. The *length* of a path (cycle) is the number of edges in this path (cycle). An *Eulerian path* (Eulerian cycle) in a graph $G = (V, E)$ is a path (cycle) in G that uses every edge in E exactly once. It is well known that a directed multi-graph possesses an Eulerian cycle if and only if all vertices have in-degree equal to their out-degree and if its underlying undirected graph is connected.

2. From DNA sequencing to exact Eulerian cycles

In this section we recapitulate the relationship between DNA sequencing and the existence of Eulerian paths in certain directed graphs. This relationship has been observed by Pevzner [15], and later on has been discussed in detail by Błażewicz et al. [5]. The underlying combinatorial ideas go (at least) back to de Bruijn [7].

For a word u of length k over the alphabet $\{A, C, G, T\}$, we denote by $\text{prefix}_{k-1}(u)$ the word that consists of the first $k-1$ letters in u , and by $\text{suffix}_{k-1}(u)$ the word that consists of the last $k-1$ letters in u . For a set S of words of length k over the alphabet $\{A, C, G, T\}$, we introduce the following directed graph $G_S = (V_S, A_S)$: The vertex set is defined by

$$V_S = \{\text{prefix}_{k-1}(u), \text{suffix}_{k-1}(u) : u \in S\}.$$

The arc set A_S contains $|S|$ arcs that are defined as follows: For every word $u \in S$, there is a corresponding arc $a(u)$ in A_S that goes from the vertex $\text{prefix}_{k-1}(u)$ to the vertex $\text{suffix}_{k-1}(u)$.

Theorem 1 (Pevzner [15]). *Let S be a set of words of length k over the alphabet $\{A, C, G, T\}$, and let $G_S = (V_S, A_S)$ be the corresponding directed graph. Then there exists a word w of length $|S| + k - 1$ with k -spectrum equal to S , if and only if G_S possesses an Eulerian path. ■*

The statement of this theorem is actually quite easy to see: For $i = 1, \dots, |S|$ let w_i denote the subword of w of length k that starts with the i th letter and ends with the $(i+k-1)$ th letter. Then $S = \{w_1, \dots, w_{|S|}\}$ and the arcs $a(w_1), \dots, a(w_{|S|})$ in this ordering form an Eulerian path for G_S . Vice versa, every Eulerian path in G_S gives a sequence $w_1, \dots, w_{|S|}$ of words of length k that can be combined into a word of length $|S| + k - 1$. To summarize, **Theorem 1** yields a polynomial time algorithm for the special case of problem DNA-SEQ where every word in S occurs *exactly once* as a subword in the word w . By considering the exact number of occurrences of every subword from S , the statement in **Theorem 1** can be generalized to our sequencing problem DNA-SEQ.

Theorem 2. *Let S be a set of words of length k over the alphabet $\{A, C, G, T\}$, let $G_S = (V_S, A_S)$ be the corresponding directed graph, and let $\ell \geq |S| + k - 1$ be an integer.*

There exists a word w of length ℓ with k -spectrum S , if and only if there exists a super-multi-graph of G_S that contains exactly $\ell - k + 1$ arcs and that possesses an Eulerian path. ■

As an immediate consequence of **Theorem 2**, problem DNA-SEQ is polynomial time reducible to the problem EX-EULERPATH defined below.

Problem: Exact Eulerian path (EX-EULERPATH)

Input: A simple directed graph $G = (V, A)$. Two vertices $s, t \in V$. An integer β .

Question: Does there exist a super-multi-graph of G that contains exactly β arcs and that possesses an Eulerian path that starts in s and ends in t ?

Indeed, by trying all possibilities for a start vertex s and an end vertex t in the graph G_S and by setting $\beta = \ell - k + 1$, one may solve an instance of DNA-SEQ by solving $O(|V_S|^2) = O(|S|^2)$ instances of EX-EULERPATH.

In the special case of problem EX-EULERPATH where $s = t$ holds, we are looking for an exact Eulerian cycle in some super-multi-graph of G (EX-EULERCYCLE). The general problem EX-EULERPATH is polynomial time

reducible to EX-EULERCYCLE: Take an instance G of EX-EULERPATH, and create a new directed path with β arcs that goes from t to s . Consider this new graph G^+ together with the value $\gamma = 2\beta$ as an instance of EX-EULERCYCLE. It can be easily verified that the original graph G has a super-multi-graph with an Eulerian path of length β going from s to t , if and only if the new graph G^+ has a super-multi-graph with an Eulerian cycle of length $\gamma = 2\beta$.

Problem: Exact Eulerian cycle (EX-EULERCYCLE)

Input: A simple directed graph $G = (V, A)$. An integer γ .

Question: Does there exist a super-multi-graph of G that contains exactly γ arcs and that possesses an Eulerian cycle?

To summarize, in this section we have shown that DNA-SEQ is polynomial time reducible to EX-EULERPATH, and that EX-EULERPATH in turn is polynomial time reducible to EX-EULERCYCLE.

3. From exact Eulerian cycles to exact bipartite matchings

In this section we will show that problem EX-EULERCYCLE is polynomial time reducible to the exact bipartite matching problem EX-MATCH. Hence, consider an arbitrary instance $(G; \gamma)$ of problem EX-EULERCYCLE. We denote by $\deg^+(v)$ and $\deg^-(v)$ the in-degree and the out-degree of the vertex $v \in V$. A vertex v is called *positive* if $\deg^+(v) > \deg^-(v)$, *neutral* if $\deg^+(v) = \deg^-(v)$, and *negative* if $\deg^+(v) < \deg^-(v)$. Furthermore, we denote by Δ the sum of $\deg^+(v) - \deg^-(v)$ taken over all positive vertices v . Clearly, this value Δ also equals the sum of $\deg^-(v) - \deg^+(v)$ taken over all negative vertices v . Note furthermore that $|\Delta| \leq |V|^2$.

For vertices $x, y \in V$ and for integers k with $1 \leq k \leq |V|$, we introduce the Boolean predicate $P[x, y; k]$ that is true if and only if in the graph G there exists a (not necessarily simple) directed path of length k that goes from x to y . Moreover, for integers k with $1 \leq k \leq |V|$ we introduce the Boolean predicate $C[k]$ that is true if and only if in the graph G there exists a (not necessarily simple) directed cycle of length k . The truth values of all these predicates can be determined in polynomial time by standard dynamic programming approaches.

Now assume that the instance $(G; \gamma)$ of EX-EULERCYCLE has answer YES, and let $G^* = (V, A^*)$ be a super-multi-graph of G with γ arcs that certifies this answer. That means that the graph G^* is strongly connected (or equivalently: that the graph G is strongly connected), and that in G^* every vertex $v \in V$ has its in-degree equal to its out-degree.

Lemma 3. *The arc set A^* of the graph G^* can be partitioned in the following way.*

- (A1) *The set A of arcs in the underlying simple graph G .*
- (A2) *A set of Δ (not necessarily simple) directed paths (a path considered as an ordered set of arcs) of length less or equal to $|V|$. Each such path starts in a positive vertex and ends in a negative vertex. In every positive vertex v , there start exactly $\deg^+(v) - \deg^-(v)$ of these paths. In every negative vertex v , there end exactly $\deg^-(v) - \deg^+(v)$ of these paths.*
- (A3) *A set of (not necessarily simple) directed cycles of length less than or equal to $|V|$. ■*

We denote by $m(A1)$, $m(A2)$, $m(A3)$ the number of arcs of type (A1), (A2), (A3) in such a partition. Now if we want to solve EX-EULERCYCLE, we must determine whether there exist arc sets of type (A1), (A2), (A3) with $m(A1) + m(A2) + m(A3) = \gamma$. Trivially, $m(A1) = |A|$. Moreover, since $|\Delta| \leq |V|^2$ we get $m(A2) \leq |V|^3$. But what are the exact candidate values that $m(A2)$ and $m(A3)$ can take?

The candidate values for $m(A2)$ can be determined via the exact bipartite matching problem (see the introductory section for an exact definition of this problem). This is done as follows. For every positive vertex x in G , we introduce $\deg^+(x) - \deg^-(x)$ independent copies in the set X of the bipartition. For every negative vertex y in G , we introduce $\deg^-(y) - \deg^+(y)$ independent copies in the set Y of the bipartition. Whenever the predicate $P[x, y; k]$ is true for some positive vertex x , some negative vertex y , and some integer k with $1 \leq k \leq |V|$, we introduce an edge of weight k in E from every copy of x to every copy of y . Since the edge weights are polynomially bounded by $|V|$, we may as well encode them in unary. It is easily verified that α_2 is a possible value for $m(A2)$ if and only if the resulting bipartite graph has a perfect matching of weight exactly α_2 . Hence, by solving a polynomial number of exact bipartite matching instances we can determine for every α_2 with $1 \leq \alpha_2 \leq |V|^3$ whether it is a candidate value for $m(A2)$.

The candidate values for $m(A3)$ can be expressed in terms of the predicates $C[k]$ with $1 \leq k \leq |V|$. We define $C = \{k : C[k], 1 \leq k \leq |V|\}$ as the set of possible cycle lengths. In determining whether an integer α_3 is a candidate

for $m(A_3)$ we distinguish the two cases $\alpha_3 \leq |V|^2$ and $\alpha_3 > |V|^2$. The first case $\alpha_3 \leq |V|^2$ boils down to an unbounded subset sum problem in which the item sizes in C and the goal value α_3 all are polynomially bounded in $|V|$. It is well-known that such a polynomially bounded special case of the subset sum problem is polynomially solvable by dynamic programming (see for instance the book by Martello and Toth [12]). For the second case $\alpha_3 > |V|^2$ we apply a famous result on the Frobenius problem.

Theorem 4 (Erdős and Graham [8]). *Let $0 < z_1 < z_2 < \dots < z_n \leq Z$ be integers with $\gcd(z_1, z_2, \dots, z_n) = 1$. Then every integer above $2Z^2/n$ can be expressed in the form $\sum_{i=1}^n x_i z_i$ with non-negative integers x_1, \dots, x_n . ■*

In our case, all the values in the set C are bounded by $|V|$; therefore we have $Z = |V|$ and $n = |C|$. If $|C| \geq 2$ and the greatest common divisor of the numbers in C is one, then every $\alpha_3 > |V|^2$ is a candidate for $m(A_3)$. If $|C| \geq 2$ and the greatest common divisor of the numbers in C equals d , then $\alpha_3 > |V|^2$ is a candidate for $m(A_3)$ if and only if it is divisible by d . The case $|C| = 0$ is straightforward. In the case $|C| = 1$, a number $\alpha_3 > |V|^2$ is a candidate value for $m(A_3)$ if and only if it is divisible by the unique element d in C .

Now let us put everything together. In order to solve an instance of EX-EULERCYCLE, we first determine the candidate values for $m(A_2)$ (by solving a polynomial number of exact bipartite matching instances) and for $m(A_3)$ (by solving a polynomially bounded subset sum instance). The instance has a solution if and only if G is strongly connected and $\gamma = m(A_1) + m(A_2) + m(A_3)$ has a solution over the candidate values for $m(A_1), m(A_2), m(A_3)$. We know that $m(A_1) = |A|$ and that all candidate values for $m(A_2)$ are between 1 and $|V|^3$. Therefore, we can simply search through all $O(|V|^3)$ possibilities for $m(A_2)$ and check whether $\gamma - |A| - m(A_2)$ is a feasible candidate for $m(A_3)$. Summarizing, this yields that problem EX-EULERCYCLE indeed is polynomial time reducible to problem EX-MATCH.

4. From exact matching to restricted exact matching

In this section we will prove that the exact bipartite matching problem EX-MATCH is polynomial time reducible to the following highly restricted special case of it.

Problem: Restricted exact bipartite matching (EX-MATCH⁻)

Input: A bipartite graph $B = (X \cup Y, E)$ with $E \subseteq X \times Y$ that is connected and cubic. For every edge $e \in E$ a weight $w(e) \in \{0, 1, \alpha + 1\}$ where α is a given integer.

Question: Does this bipartite graph have a perfect matching of weight exactly α ?

Hence, consider an integer α and a bipartite multi-graph $B = (X \cup Y, E)$ as instance for the general problem EX-MATCH. The reduction to EX-MATCH⁻ will be done in four steps: The first step is a simple preprocessing step that makes the graph connected, and that also gets rid of vertices of degree one. In the second step, we will get rid of the vertices of degree four and more; simultaneously, the graph will become simple. In the third step, we will bring the edge weights down to 0–1. Finally in the fourth step, we will make the graph cubic.

Making the graph connected and getting rid of vertices of degree one. Let C_1, \dots, C_r denote the connected components of graph B . If one of these components C_i consists of an odd number of vertices, then B does not possess any perfect matching. In this case the answer to EX-MATCH is trivially NO. If all r components contain an even number of vertices, then we connect them to each other by $r - 1$ bridges of weight 1. The resulting multi-graph is connected and still bipartite. It is easy to see from parity considerations that none of the introduced bridges can be used in a perfect matching.

For every vertex of degree one, we duplicate the incident edge. This increases the vertex degree to two, but does not change the set of possible weights of perfect matchings in the graph. The resulting multi-graph with these $r - 1$ bridges and all these duplicated edges is denoted by $B_1 = (X_1 \cup Y_1, E_1)$. This graph B_1 is bipartite, connected, and does not contain vertices of degree smaller than two. Moreover, B_1 has a perfect matching of weight α if and only if the original graph B has a perfect matching of weight α .

Getting rid of vertices of degree at least four. From the bipartite multi-graph B_1 , we now create a simple bipartite graph B_2 : We replace every vertex u in B_1 with degree d by a simple path $P(u)$ of $2d - 1$ new vertices $u_1 - u'_1 - u_2 - u'_2 - \dots - u_{d-1} - u'_{d-1} - u_d$. The $d - 1$ primed vertices u'_1, \dots, u'_{d-1} in $P(u)$ are of degree two and have no further connections to the rest of the graph. Each of the d unprimed vertices u_1, \dots, u_d has exactly

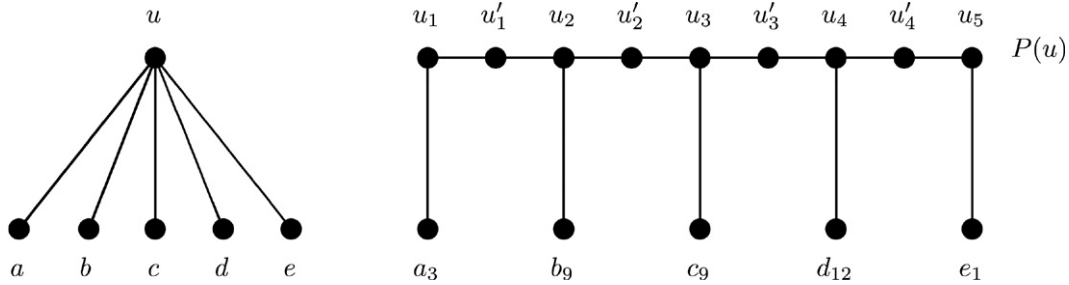


Fig. 1. How to replace a vertex u of degree 5 by a path $P(u)$ with 9 vertices.

one further edge into the rest of the graph: For every edge $[u, v]$ in E_1 , there is a corresponding edge in E_2 that goes from an unprimed vertex in the path $P(u)$ to an unprimed vertex in the path $P(v)$. See Fig. 1 for an illustration. All $2d - 2$ edges in the path $P(u)$ receive weight 0. An edge $[u_i, v_j]$ that connects $P(u)$ to $P(v)$ in B_2 receives the same weight as the corresponding edge $[u, v]$ in B_1 . This completes the (polynomial time) construction of graph B_2 . Clearly, B_2 is a simple, bipartite, connected graph in which all vertices have degree two or three.

It is straightforward to see that B_2 has a perfect matching of weight α if and only if the graph B_1 has a perfect matching of weight α .

Bringing the edge weights down to 0–1. Let W denote the maximum edge weight in the graph B_2 . Every edge e in E_2 is subdivided by $2W$ new vertices. This replaces the edge e by a path $P(e)$ of length $2W + 1$, and the resulting graph B_3 is still simple and bipartite. For an edge e with weight $w(e)$, the path $P(e)$ consists of $w(e)$ edges of weight 1 and $2W - w(e) + 1$ edges of weight 0: All edges in even positions (the second, fourth, sixth, etc. edge on the path) receive weight 0. Exactly $w(e)$ of the edges in odd positions receive weight 1, and the remaining edges in odd positions receive weight 0. This completes the description of the bipartite graph $B_3 = (X_3 \cup Y_3, E_3)$. Since all edge weights are encoded in unary, B_3 can be computed in polynomial time. B_3 is a simple, bipartite, connected graph in which all vertices have degree two or three, and in which all edges have weight 0 or 1. The graph B_3 has a perfect matching of weight α if and only if B_2 has a perfect matching of weight α .

Making the graph cubic. If $|X_3| \neq |Y_3|$ holds in the graph $B_3 = (X_3 \cup Y_3, E_3)$, then B_3 does not possess any perfect matching, and the answer to EX-MATCH is trivially NO. Therefore, we will assume that $|X_3| = |Y_3|$ holds. Let $X'_3 \subseteq X_3$ and $Y'_3 \subseteq Y_3$ contain the vertices of degree two. Since in B_3 all vertices are of degree two or three, we get that $|X'_3| = |Y'_3|$. We create a perfect matching between X'_3 and Y'_3 with all edges of weight $\alpha + 1$, and we add it to B_3 . The resulting graph B_4 is cubic and connected, and the weights of all edges are in $\{0, 1, \alpha + 1\}$. Moreover, B_4 has a perfect matching of weight α if and only if B_3 has a perfect matching of weight α .

The final graph B_4 together with α constitutes a feasible input for the problem EX-MATCH⁻. It is a YES instance for EX-MATCH⁻, if and only if the original graph B together with α is a YES instance for EX-MATCH. Hence, we have reached the goal of this section and established that EX-MATCH is polynomial time reducible to its special case EX-MATCH⁻. The following corollary will not be needed in our further arguments, but we feel that it might be of separate interest. Its proof follows by terminating the above reduction after the third step with the graph B_3 .

Corollary 5. *Problem EX-MATCH is polynomial time equivalent to its special case where the bipartite input graph is simple and connected, and has only vertices of degree two and three, and has only edge weights zero and one.*

5. From restricted exact matching back to DNA sequencing

In this section we will prove that the restricted exact bipartite matching problem EX-MATCH⁻ is polynomial time reducible to problem DNA-SEQ. This will be done by moving through an auxiliary instance of problem EX-EULERCYCLE (see Section 2 for the definition of this problem).

Indeed, consider a connected, cubic, bipartite graph $B = (X \cup Y, E)$ as instance for EX-MATCH⁻. If $|X| \neq |Y|$ holds, then the graph B does not possess any perfect matching, and the answer to EX-MATCH⁻ is trivially NO. Therefore, we will assume that $|X| = |Y| = q$ holds. Without loss of generality, we assume furthermore that q is sufficiently large to satisfy $2^q \geq 100q^3$. Note that any perfect matching in B has weight at most q (in the case it does not use edges of weight $\alpha + 1$) or weight at least $\alpha + 1$ (in any other case). Therefore, we will assume from now

on that $\alpha \leq q$. We denote by m_0, m_1 , and $m_{\alpha+1}$ the number of edges of weight 0, 1, and $\alpha + 1$, respectively. Note that $m_0 + m_1 + m_{\alpha+1} = |E| = 3q$. Finally, let f be an arbitrary bijection between the two sets X and Y , and let $Q = 2q + 8$.

From the bipartite graph B we will now construct a directed graph $G = (V, A)$ as an instance of EX-EULERCYCLE. This directed graph G consists of $2q$ primary vertices and of many secondary vertices. The primary vertices are grouped into the two sets $X' = \{x' | x \in X\}$ and $Y' = \{y' | y \in Y\}$. The arcs in G are defined as follows.

- (P1) For every edge $e = [x, y]$ in B with $x \in X$ and $y \in Y$, there is a corresponding directed path $P(e)$ in G that goes from x' to y' . If the edge e has weight $w(e)$, then this path has length $(w(e) + 1)Q$.
- (P2) For every $y \in Y$ and $x = f(y)$ in B , there are four directed paths of length $8(q^2 + q)Q$ in G that all connect y' to x' .

All these introduced directed paths are internally pairwise vertex-disjoint. Their internal vertices (that all have in-degree and out-degree one) form the secondary vertices of the graph G . Every vertex in X' has in-degree four and out-degree three. Every vertex in Y' has in-degree three and out-degree four. Since the bipartite graph B is connected, also the constructed graph G is connected. A rough estimation shows that G has less than $50q^3Q$ vertices.

Lemma 6. *The instance B of EX-MATCH⁻ has a perfect matching of weight exactly α , if and only if the constructed instance G of EX-EULERCYCLE has a super-multi-graph with an Eulerian cycle of length exactly*

$$\gamma = (m_0Q + 2m_1Q + m_{\alpha+1}(\alpha + 2)Q) + 32q(q^2 + q)Q + (\alpha + q)Q.$$

Proof. (Only if). Assume that B has a perfect matching M of weight α . We construct a super-multi-graph G^* of G , in which all vertices in $X' \cup Y'$ have in-degree and out-degree four. We take all arcs from G into G^* . Moreover, for each of the q edges $e \in M$, we take one additional copy of the directed path $P(e)$ into G^* . It is easily verified that in G^* every vertex has in-degree equal to out-degree. Since the graph G^* is also connected, it has an Eulerian cycle.

Now let us determine the number of arcs in G^* . First, G^* contains all paths of type (P1). There are m_0 of those paths that have length Q , m_1 of length $2Q$, and $m_{\alpha+1}$ of length $(\alpha + 2)Q$. Secondly, G^* contains all paths of type (P2). There are $4q$ of them, and each has length $8(q^2 + q)Q$. Thirdly, there are the arcs in the additional q paths $P(e)$ with $e \in M$. These paths contribute

$$\sum_{e \in M} (w(e) + 1)Q = Q \sum_{e \in M} w(e) + Q|M| = (\alpha + q)Q$$

arcs. Altogether, this yields exactly γ arcs in G^* .

(If). Assume that G has a super-multi-graph G^* with an Eulerian cycle of length γ . It can be seen that this multi-graph G^* must consist of one or more copies of every path introduced in (P1) and (P2). We claim that G^* cannot contain two copies of any path of type (P2). Otherwise, the total number of arcs in the copies of paths of type (P2) is at least

$$(4q + 1) \cdot 8(q^2 + q)Q = 6(q^2 + q)Q + 32q(q^2 + q)Q + 2(q^2 + q)Q > \gamma.$$

Since G^* has only γ arcs, we have arrived at the desired contradiction. We conclude that every path of type (P2) shows up exactly once in G^* , and hence these paths altogether contribute $32q(q^2 + q)Q$ arcs. As a consequence, in G^* every vertex in X' has in-degree four, and every vertex in Y' has out-degree four.

Since G^* has an Eulerian cycle, every vertex in X' must have out-degree four, and every vertex in Y' must have in-degree four. These degrees can only result from copies of paths of type (P1). Each path $P(e)$ with $e \in E$ must show up at least once in G^* . This yields $(m_0Q + m_12Q + m_{\alpha+1}(\alpha + 2)Q)$ arcs, out-degrees three for vertices in X' , and in-degrees three for vertices in Y' . The remaining $(\alpha + q)Q$ arcs in G^* must come from a system of q paths of type (P1) that connect every vertex in X' to exactly one vertex in Y' . In the bipartite graph B , the edge set M that contains those edges e for which $P(e)$ is in this system of paths forms a perfect matching of weight α . ■

Our next goal is to establish that for an appropriate value k , there exists a k -spectrum S of words over the alphabet $\{A, C, G, T\}$ such that the above constructed directed graph $G = (V, A)$ equals $G_S = (V_S, A_S)$; see Section 2 for definitions. The main difficulty is to ensure that distinct vertices in V are associated with distinct DNA words of length $k - 1$.

We partition the vertices in V into a set C of so-called *crucial* vertices and a set $V - C$ of *non-crucial* vertices. Every vertex in the set $X' \cup Y'$ is crucial. Moreover, every vertex on the paths of type (P1) and (P2) whose distance to $X' \cup Y'$ is divisible by Q is a crucial vertex. Note that the length of every path of type (P1) and (P2) is a multiple of Q , and that the crucial vertices divide these paths into connected chunks of Q arcs and $Q - 1$ non-crucial vertices. If there is a directed path of length Q from a crucial vertex u to another crucial vertex v in G , then we say that u is a *predecessor* of v , and that v is a *successor* of u . It can be seen that every vertex in X' has four predecessors and three successors, that every vertex in Y' has three predecessors and four successors, and that all remaining crucial vertices have one predecessor and one successor. Moreover, it can be checked that $|C| \leq 50q^3$ holds.

For every crucial vertex $v \in C$, we fix two labels $LLABEL(v)$ and $RLABEL(v)$. These labels are $2|C|$ pairwise distinct words of length q over the alphabet $\{A, G\}$. We need $2|C| \leq 100q^3$ distinct labels that we can choose from 2^q words. Since we assumed $100q^3 \leq 2^q$, such pairwise distinct labels indeed exist and can be found easily. With the help of these labels, we will now define for every vertex $z \in V$ a corresponding word $WORD(z)$ of length $Q = 2q + 8$.

- If z is a crucial vertex, then $WORD(z)$ starts with one of the letters A, C, G, T , followed by the label $LLABEL(z)$, followed by a string of six T 's, then followed by the label $RLABEL(z)$, and ending with one of the letters A, C, G, T . For crucial vertices, these words fulfill the following conditions: (W1) If v_1 and v_2 are successors of the same vertex u , then $WORD(v_1)$ and $WORD(v_2)$ end with different letters. (W2) If u_1 and u_2 are predecessors of the same vertex v , then $WORD(u_1)$ and $WORD(u_2)$ start with different letters. Since every crucial vertex has at most four successors and at most four predecessors, these conditions can indeed be met with the alphabet $\{A, C, G, T\}$ of size four.
- If z is a non-crucial vertex, then it lies on a uniquely defined directed path of length Q that connects some crucial vertex u to another crucial vertex v . Let $u = z_0, z_1, \dots, z_Q = v$ denote the sequence of vertices along such a path. Then $WORD(z_i)$ consists of the last $Q - i$ letters of $WORD(u)$ followed by the first i letters of $WORD(v)$.

Lemma 7. For $u, v \in V$ with $u \neq v$, we always have $WORD(u) \neq WORD(v)$.

Proof. Consider a word $WORD(z)$ of length $2q + 8$ over $\{A, C, G, T\}$. We show that from $WORD(z)$, we can uniquely localize the corresponding vertex z in G .

By our construction, $WORD(z)$ either contains a subword of six consecutive T 's, or it starts with i consecutive T 's and ends with $6 - i$ consecutive T 's for some $1 \leq i \leq 5$. In either case, the q letters preceding or succeeding these T 's in $WORD(z)$ will form one of the labels $LLABEL(v)$ or $RLABEL(v)$ for some crucial vertex v . If vertex v is not contained in $X' \cup Y'$, then the relative position of this label in $WORD(z)$ uniquely determines the vertex z . If v is contained in $X' \cup Y'$, then either (i) $WORD(z)$ contains the first letter of the word of a successor of v , or (ii) it contains the last letter of the word of a predecessor of v , or (iii) it contains none of these letters. In the cases (i) and (ii), by conditions (W1) and (W2) these letters uniquely identify the corresponding successor or predecessor. Then we can again use the relative position of the label $LLABEL(v)$ or $RLABEL(v)$ in $WORD(z)$ to uniquely determine the vertex z . Finally, in case (iii) the vertex z must coincide with v . ■

Now we are almost done: We choose $k = Q + 1 = 2q + 9$ as the word length for the k -spectrum S . Every vertex $v \in V$ is labeled by the word $WORD(v)$ of length $k - 1$, and thus becomes a member of V_S . By Lemma 7, different vertices are labeled by different words. For every arc $a = (u, v) \in A$, the last $k - 2$ letters in $WORD(u)$ agree with the first $k - 2$ letters in $WORD(v)$. Hence, this arc a can be correctly encoded by the word $WORD(a)$ of length k that starts with the first letter of $WORD(u)$, followed by these $k - 2$ agreeing letters, and ending with the last letter of $WORD(v)$. We define $S = \{WORD(a) | a \in A\}$ as our k -spectrum, and we thus derive $G_S = G$. By the above discussion and by Lemma 6, the graph G_S has a super-multi-graph with an Eulerian cycle of length γ , if and only if the bipartite graph B has a perfect matching of weight α .

In the final step, we move from problem EX-EULERCYCLE to problem EX-EULERPATH: Let $a = (u, v)$ be an arc on one of the paths of type (P2) such that u and v both are non-crucial vertices. We remove this arc a from the graph G , and we simultaneously define $S_{\text{final}} = S - \{WORD(a)\}$. This leads to a vertex u of out-degree 0 and to a vertex v of in-degree 0. It can be verified that the resulting graph has a super-multi-graph with an Eulerian path of length $\gamma - 1$ (that of course starts in vertex v and ends in vertex u), if and only if the bipartite graph B has a perfect matching of weight α . Finally, we get from Theorem 2 that there exists a word of length $\gamma - 1$ with k -spectrum S_{final} , if and only if the bipartite graph B has a perfect matching of weight α . This means that problem EX-MATCH⁻ is polynomially reducible to problem DNA-SEQ.

By combining the reductions from Section 2 through 5, we arrive at the main result of this paper.

Theorem 8. *The two problems EX-MATCH and DNA-SEQ are polynomial time reducible to each other.* ■

References

- [1] W. Bains, G.C. Smith, A novel method for nucleic acid sequence determination, *Journal of Theoretical Biology* 135 (1988) 303–307.
- [2] F. Barahona, W.R. Pulleyblank, Exact arborescences, matchings, and cycles, *Discrete Applied Mathematics* 16 (1987) 91–99.
- [3] C. Berge, *Graphs and Hypergraphs*, North Holland, 1973.
- [4] J. Błażewicz, P. Formanowicz, M. Kasprzak, W.T. Markiewicz, J. Węglarz, DNA sequencing with positive and negative errors, *Journal of Computational Biology* 6 (1999) 113–123.
- [5] J. Błażewicz, A. Hertz, D. Kobler, D. de Werra, On some properties of DNA graphs, *Discrete Applied Mathematics* 98 (1999) 1–19.
- [6] J. Błażewicz, M. Kasprzak, Complexity of DNA sequencing by hybridization, *Theoretical Computer Science* 290 (2001) 1459–1473.
- [7] N.G. de Bruijn, A combinatorial problem, *Koninklijke Nederlandse Akademie van Wetenschappen te Amsterdam. Proceedings* 49 (1946) 758–764.
- [8] P. Erdős, R.L. Graham, On a linear diophantine problem of Frobenius, *Acta Arithmetica* 21 (1972) 399–408.
- [9] A.V. Karzanov, Maximum matching of given weight in complete and complete bipartite graphs, *Cybernetics* 23 (1987) 8–13. Translation from *Kibernetika* 1 (1987) 7–11.
- [10] M. Leclerc, Polynomial time algorithms for exact matching problems, Master’s Thesis, University of Waterloo, Waterloo, 1986.
- [11] M. Leclerc, Optimizing over a slice of the bipartite matching polytope, *Discrete Mathematics* 73 (1988/89) 159–162.
- [12] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, 1990.
- [13] K. Mulmuley, U. Vazirani, V.V. Vazirani, Matching is as easy as matrix inversion, *Combinatorica* 7 (1987) 105–113.
- [14] C.H. Papadimitriou, M. Yannakakis, The complexity of restricted spanning tree problems, *Journal of the ACM* 29 (1982) 285–309.
- [15] P.A. Pevzner, ℓ -tuple DNA sequencing: Computer analysis, *Journal of Biomolecular Structure and Dynamics* 7 (1989) 63–73.