Universität Bielefeld

Technische Fakultät
Abteilung Informationstechnik
Forschungsberichte

On Common Intervals with Errors

Cedric Chauve Yoan Diekmann Steffen Heber
Julia Mixtacki Sven Rahmann Jens Stoye

Report 2006-02

Impressum: Herausgeber:
Ellen Baake, Robert Giegerich, Ralf Hofestädt, Franz Kummert,
Peter Ladkin, Ralf Möller, Helge Ritter, Gerhard Sagerer,
Jens Stoye, Ipke Wachsmuth

Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31,
33501 Bielefeld, Germany

ISSN 0946-7831

On Common Intervals with Errors

Cedric Chauve [*] Yoan Diekmann [†] Steffen Heber [‡]
Julia Mixtacki [§] Sven Rahmann [¶] Jens Stoye [‖]

October 23, 2006

The information that groups of genes co-occur in several genomes provides a basis for further comparative genomic analysis. The task of finding such constellations, mostly referred to as *gene clusters*, has led to various models of increasing generality. A central feature to enhance the biological relevance of their definition when applied to real genomic data is to allow for slight differences in the gene content within a cluster, thus not only considering groups of exact equality. We contribute a model defining gene clusters as common intervals with errors and discuss different representations and the corresponding problems resulting for the search procedure.

1 Introduction

The last years have seen a steadily increasing number of available completely sequenced genomes, allowing access to information about gene locations of a large number of organisms. Based on this knowledge, the comparison of genomes on a higher level – that means, considering genes rather than nucleotides as smallest entities – provides insight into various problems emerging in genomic science.

One approach analysing the spatial organisation of genes is concerned with so called gene clusters, that are groups of genes co-localised in several genomes, which might result

[*]Comparative Genomics Laboratory and LaCIM, Université du Québec à Montréal, Canada, and Department of Mathematics, Simon Fraser University, Canada. Work funded by an NSERC discovery grant. chauve@lacim.uqam.ca

[†]Technische Fakultät, Universität Bielefeld, Germany. ydiekman@TechFak.Uni-Bielefeld.DE

[‡]Dept. of Computer Science, North Carolina State University, Raleigh, NC, USA. sheber@ncsu.edu

[§]International NRW Graduate School in Bioinformatics and Genome Research, Universität Bielefeld, Germany. julia.mixtacki@uni-bielefeld.de

[¶]Algorithmen und Statistik für Systembiologie, AG Genominformatik, Technische Fakultät, Universität Bielefeld, Germany. rahmann@CeBiTec.uni-bielefeld.de

[‖]AG Genominformatik, Technische Fakultät, and Institut fr Bioinformatik, Universität Bielefeld, Germany. stoye@TechFak.Uni-Bielefeld.DE

from many factors such as vertical inheritance or functional selection [32]. The fact that selective pressure has conserved the genes in each others' neighborhood during evolution indicates that gene order has functional implications [5], examples being operons of prokaryotic organisms [20] or metabolic relationships [25]. Thus finding gene clusters conserved through several genomes can amongst others be used for operon prediction, but also for functional inference of uncharacterised genes, if the function of a gene located in the same conserved cluster is known.

Other research fields use the information that the analysis of gene clusters provide, for example to reconstruct rearrangement events [9], with the aim of inferring ancestral genomes or elucidating phylogenetic relationships (see [3] for example).

To deal with areal organisation of genes, in comparative genomics gene order is often modelled by assigning a number to every gene, where in general each number labels a unique family of homologous genes. This represents a genome as a permutation if the numbers are unique, or as a sequence if genes exceeding a certain degree of similarity are assigned the same number. Though permutations are the stronger mathematical structure, they do not allow to model multiple occurrences of genes, namely paralogous genes that are results of a duplication of an ancestor gene. However, since this is a common event in evolution, sequences are the more natural way to model gene order, though at the price of apparently higher problem complexities.

Based on this model, one can define very generally a gene cluster as a set of gene family labels such that, in some genomes of the considered dataset, the genes belonging to these families are clustered together along the genome. These genome segments that contain the genes of a cluster, that are intervals of the integer sequences representing the genomes, are called the locations of the cluster. The main question in defining a gene cluster model is to describe (1) the kind of data (permutations or sequences) it applies to and (2) the link between clusters and their locations. We describe below the main gene cluster models developed so far.

The first concept subsequently applied to model gene clusters was defined by Uno and Yagiura [30], who devised the structure of common intervals on two permutations. Common intervals are pairs of intervals containing the same set of elements. For two permutations of n elements, the authors describe an $O(n + k)$ time algorithm, where k ($\leq n(n-1)/2$) is the number of common intervals. This algorithm is optimal in the sense that n and k correspond to the size of the input and output respectively.

This model was extended to more than two permutations by Heber and Stoye [17]. They defined a generating subset of i ($\leq n-1$) common intervals, called irreducible intervals, and devised an optimal $O(kn + K)$ time algorithm to find all K common intervals of k permutations of length n. A simpler algorithm achieving the same result was recently given in [2].

An important generalisation was made by Luc *et al.* [21] introducing the notion of gene teams. Gene teams on permutations are defined as common intervals but allow to have gaps between consecutive elements of the cluster not exceeding a specified threshold-number of genes. This way of dealing with errors, also referred to as *max-gap clusters* [18], completely disregards the "intruder"-genes as well as their total number, they are of

no further importance for the definition of a cluster. In [21], an $O(kn \log^2 n)$ algorithm is given detecting all gene teams in k permutations of length n. The algorithm finds gene teams that are present in all k permutations, though it could be modified by simple pairwise comparison such that gene teams are found that are present in at least $q \leq k$ of the input permutations. Such a threshold of the subset size is called a *quorum*.

A step towards modeling gene clusters on sequences was published by Eres *et al.* in [11]. They establish a definition of clusters on sequences, handling them as permutations of multisets of characters, therewith distinguishing between clusters with the same set of characters but different length.

Two algorithms reporting clusters defined as sets of characters on sequences were presented by Schmidt and Stoye [29], whose clusters are conceptually similar to common intervals with the advantage of allowing paralogs. They gave a simple $O(n^2)$ time and space algorithm for two sequences of length at most n whose time complexity increases to $O(k(1 + k - q)n^2)$ if applied to k sequences with a quorum q. A further result in [29], based on an earlier algorithm by Didier [7], has also quadratic time complexity, but requires only linear space. This algorithm was simplified in [8], where in addition another algorithm is presented that finds all common intervals in $O(n|\Sigma| \log |\Sigma|)$ time, based on the fingerprinting technique from [1].

The drawback of a rigid definition of clusters without considering errors can partially be compensated by a post-processing step [28], where clusters are fused that contain some non-matching genes according to a parameter p, called identity rate.

The conceptual union of gene teams and sequences is due to He and Goldwasser [16], who published a recursive algorithm detecting max-gap clusters, termed *COG teams*, on two sequences in $O(mn)$ time, m and n being the number of common genes in each sequence. The shortcoming of this approach is that it is not defined on and does not efficiently generalise to multiple genomes, as the algorithm's complexity would grow exponentially in the number of sequences. In fact, it was shown in [26] that the number of gene teams for more than two genomes represented by sequences can be exponential in the size of the genomes. This rules out any hope to design an efficient algorithm for computing all gene teams in more than two sequences.

As hinted in the description of gene teams, modelling clusters as subsequences with limited gap size has consequences in terms of cluster properties, which are not always favorable. All gapped cluster definitions ensure that the borders of any location of a cluster are no gaps, but in between no restriction bounds the number of genes not belonging to the cluster, as long as no gap is larger than the specified threshold. As a result, locations of max-gap clusters can grow arbitrarily in worst case without ensuring a minimum density, i.e. the ratio between the number of gene families that appear in this location and the number of genes actually belonging to the cluster can become arbitrarily small.

A biological interpretation of gene clusters conjectures that co-regulation processes due to the gene neighborhood on chromosomes is the evolutionary ground for the observed functional associations of the resulting proteins, which also speaks against a model not allowing to directly regulate the broadening of a cluster.

This report proposes an alternative model for gene clusters, basically differing in the way how errors are handled. The max-gap approach to introduce errors by allowing gaps between genes is modified. Instead, a distance measure between sets, inspired by the edit distance on sequences, is defined.

A similar approach is taken in [27], where additionally a quantitative quality measure of a cluster based on the set distance measure is introduced, and a general framework to formulate gene cluster detection as an integer linear program is presented.

The next section points out important gene cluster properties and introduces general definitions. Section 3 gives different representations of clusters and the corresponding search problems, which are discussed in Section 4 together with possible enhancements of the underlying model in future work. The last section concludes this report by summarizing its results.

2 General Definitions

Before defining a new model for gene clusters, it is helpful to first point out different properties that a new cluster model should combine, and thus achieve a high degree of flexibility. A more complete analysis of different gene cluster properties is contained in [18].

Permutation/sequence. As noted in the previous section, modelling genomes as sequences allows to consider paralogous genes and gene loss. Therefore this approach incorporates a higher degree of biological reality, whereas permutations generally allow to design more efficient algorithms.

Multiple genomes. It is a desirable property if gene cluster models extend to multiple genomes, without losing the ability to design a feasible algorithm to find them.

Quorum parameter. A quorum parameter enables to find clusters present only in a subset of all genomes, whereby the model gains flexibility and the number of potential clusters is augmented.

Allowing Errors. Relaxing the claim of equality between the gene set that defines a cluster and the gene contents of the intervals that define its locations improves the flexibility of the model as it allows to detect clusters that evolved through events of gene loss, insertion or fusion for example.

The notation used in the following sections is adapted from the one in [29].

Starting point is a finite alphabet $\Sigma = \{1, \ldots, \sigma\}$ of characters representing genes, which are concatenated leading to sequences representing genomes. For a sequence S, $S[a]$ refers to the a-th character of S, $S[a, b]$ to the substring $S[a]S[a + 1] \ldots S[b]$, and $|S|$ to the length of S. We assume that all sequences that occur are bounded in length by a finite number n.

Important for the further development of the model is the definition of character sets, which allows to consider intervals independently from multiple occurrences of characters.

Definition 1 (character set) *Given a string S over a finite alphabet Σ, the* character set *of S is defined by*

$$\mathit{CharSet}(S) := \{S[k] \mid 1 \leq k \leq |S|\} \subseteq \Sigma.$$

The set of all character sets of substrings of S is denoted by

$$\mathcal{C}^{all}(S) := \{\mathit{CharSet}(s) \mid s \text{ is a nonempty substring of } S\}.$$

Given a set of k distinct strings $\mathcal{S} = \{S_1, \ldots, S_k\}$, we extend the above definition of $\mathcal{C}^{all}$ to $\mathcal{S}$ in a natural way:

$$\mathcal{C}^{all} = \mathcal{C}^{all}(\mathcal{S}) := \bigcup_{i=1}^{k} \mathcal{C}^{all}(S_i).$$

Example. Consider the set of sequences $\mathcal{S} = \{S_1, S_2, S_3\}$ over the alphabet $\Sigma = \{1, 2, 3\}$.

	1	2	3	4	5	6
$S_1 =$	1	1	2	1		
$S_2 =$	1	2	3	3	2	1
$S_3 =$	3	3	2	1	1	

Then $\mathcal{C}^{all} = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$.

In a model making no allowance for errors, exact equality of character sets is the method of choice when defining a gene cluster. A way to relax this condition and detect more rearranged clusters is to introduce a notion of distance between sets, and consider resemblance within the bounds of a distance parameter instead of equality.

A possible distance measure is the following:

Definition 2 (symmetric set distance) *Given two character sets C and D, the* symmetric set distance *is defined as*

$$d_S(C, D) = |(C \cup D) \setminus (C \cap D)| = |C \setminus D| + |D \setminus C|.$$

A comparison with edit distances on sequences suggestes another distance measure as follows. In sequence alignment mostly three edit operations are considered: insertion, deletion and substitution, each of which can, in general, be assigned an individual cost. The Levenshtein or unit cost edit distance assigns a cost of one to all three operations.

Sequence alignments are often used in the comparison of biological sequences, where point mutations inevitably occur over time and modify sequences in the course of evolution. Assuming genomic dynamics – though on a wider time scale – leading to successive transformation of genes, an analogous consideration on sets leads to the following distance definition:

Definition 3 (set transformation distance) *Given two character sets C and D, the* set transformation distance *is defined as*

$$d_T(C, D) = \max\{|C \setminus D|, |D \setminus C|\}.$$

This distance between character sets alludes to the unit cost edit distance by counting mismatches just once.

Remark 1 It is easy to check that d_S and d_T are indeed proper distances. In particular, they satisfy the triangle inequality.

Given a parameter denoting a maximally allowed absolute or relative distance, it is possible to define the neighborhood of a character set, i.e., all character sets not exceeding the given distance from it.

Definition 4 ((absolute) d-neighborhood) *Given an integer $d \geq 0$ and a character set C over an alphabet Σ, the* (absolute) d-neighborhood $\mathcal{N}_d(C)$ *of C according to distance function dist is defined as*

$$\mathcal{N}_d(C) := \{C' \subseteq \Sigma \mid C' \neq \emptyset,\ dist(C, C') \leq d\}.$$

More generally, given a set of character sets $\mathcal{C} = \{C_1, \ldots, C_m\}$ the d-neighborhood of $\mathcal{C}$ is defined as

$$\mathcal{N}_d(\mathcal{C}) = \bigcup_{i=1}^{m} \mathcal{N}_d(C_i).$$

Finally, the d-neighborhood of a string S or of a set of strings $\mathcal{S}$ is defined as the d-neighborhood of the corresponding $\mathcal{C}^{all}$ sets:

$$\mathcal{N}_d(S) := \mathcal{N}_d(\mathcal{C}^{all}(S)), \qquad \mathcal{N}_d(\mathcal{S}) := \mathcal{N}_d(\mathcal{C}^{all}(\mathcal{S})).$$

Example. Given a character set $C = \{1, 2\}$ over the alphabet $\Sigma = \{1, 2, 3\}$, the 1-neighborhood of C according to the set transformation distance d_T is

$$\mathcal{N}_1(C) = \{\{1\}, \{2\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

The 1-neighborhood of the set of character sets $\mathcal{C}^{all}$ introduced in the last example is

$$\mathcal{N}_1(\mathcal{C}^{all}) = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\},$$

in this example the whole set of nonempty subsets of Σ.

Remark 2 Given an alphabet Σ, a string S and a parameter d, the d-neighborhood of S for the symmetric set distance d_S contains $\binom{n}{2} \cdot \sum_{i=0}^{d} \binom{|\Sigma|}{i}$ elements, which belongs to $O(n^2|\Sigma|^d)$. For the set transformation distance d_T, the d-neighborhood of S contains $\left(\binom{n}{2} \sum_{i=0}^{d} \binom{|S|_1}{i}\right) \cdot \left(\binom{n}{2} \sum_{j=0}^{d} \binom{|S|_0}{j}\right)$ elements, where $|S|_1$ is the number of elements of Σ appearing in S and $|S|_0 + |S|_1 = |\Sigma|$. This belongs to $O(n^4|\Sigma|^{2d})$ and is hence quadratic, in the worst case, with respect to the size of the d-neighborhood of S for the symmetric set distance.

In the definition of the d-neighborhood, the maximally allowed distance d is constant for all character sets, irrespective of their size. This can easily lead to problems: For any d, the neighborhood of all character sets of size $\leq d$ would contain the empty character set, which appears everywhere. Therefore it has been specifically excluded from the neighborhood definition. Even so, the neighborhoods of all character sets of size $\leq d+1$ contain $d+1$ singleton character sets, which are also expected to occur frequently. An alternative that avoids such problems is to define the neighborhood size relative to the character set size.

Definition 5 ((relative) ε-neighborhood) *Given a real number $\varepsilon \geq 0$ (typically for meaningful results, but not necessarily, $\varepsilon < 1$) and a character set $C \subseteq \Sigma$, the* (relative) ε-neighborhood $\mathcal{U}_\varepsilon(C)$ *of C according to an* integer-valued *distance function dist is defined as*

$$\mathcal{U}_\varepsilon(C) := \{C' \subseteq \Sigma \mid dist(C, C') \leq \lfloor \varepsilon \cdot |C| \rfloor\}.$$

For a set of characters sets $\mathcal{C} = \{C_1, \ldots, C_m\}$ and for a string S and a set $\mathcal{S}$ of strings, the ε-neighborhoods $\mathcal{U}_\varepsilon(\mathcal{C})$, $\mathcal{U}_\varepsilon(S)$, and $\mathcal{U}_\varepsilon(\mathcal{S})$, are defined similarly to the respective d-neighborhoods.

Remark 3 While the d-neighborhood has a simple symmetry property,

$$D \in \mathcal{N}_d(C) \iff C \in \mathcal{N}_d(D),$$

this is more complicated for the ε-neighborhood: Let C and D be two character sets with $|C| \geq |D|$ and let $\rho := |C|/|D| \geq 1$. Then

$$D \in \mathcal{U}_\varepsilon(C) \iff C \in \mathcal{U}_{\rho\varepsilon}(D).$$

Proof. Let $d := \lfloor \varepsilon |C| \rfloor = \lfloor \rho\varepsilon |D| \rfloor$.
Then $D \in \mathcal{U}_\varepsilon(C) \iff D \in \mathcal{N}_d(C) \iff C \in \mathcal{N}_d(D) \iff C \in \mathcal{U}_{\rho\varepsilon}(D)$. □

Based on the definition of distance between character sets, a cluster consists of a set of character sets not too different in their gene content, and the information where on the genomes intervals with these character sets can be found.

A first definition formalizes the notion of "not too different".

Definition 6 (d-occurrences, ε-occurrences, representative) *Let C_{repr} be a character set and $d \geq 0$ be an integer ($\varepsilon \geq 0$ a real number). The* d-occurrences *(ε-*occurrences*) of C_{repr} in a set of strings $\mathcal{S}$ are given by the set*

$$\mathcal{C} = \mathcal{C}^{all}(S) \cap \mathcal{N}_d(C_{repr}) \qquad \left(\mathcal{C} = \mathcal{C}^{all}(S) \cap \mathcal{U}_\varepsilon(C_{repr})\right).$$

C_{repr} is called a representative *of $\mathcal{C}$.*

Note that the representative itself does not need to be present in any of the sequences of $\mathcal{S}$; see also Figure 1.

The next definition allows to describe more precisely the occurrences of a character set or set of character sets in a set of sequences.

Definition 7 (location set) *Given k sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$ and a character set C, the* location set $\mathcal{L}_C$ *of C is the set of all triples (i, a, b) with*

$$CharSet(S_i[a, b]) = C$$

satisfying the length-maximality condition

$$(a = 1 \text{ or } S_i[a-1] \notin C) \text{ and } (b = |S_i| \text{ or } S_i[b+1] \notin C).$$

Given a set of character sets $\mathcal{C}$, $\mathcal{L}_\mathcal{C}$ designates the union of the location sets of all elements of $\mathcal{C}$.

A definition crucial to handle the quorum concerns the set of sequences covered by a location set.

Definition 8 (covering) *Given k sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$ and a character set C with its location set $\mathcal{L}_C = \{(i_1, a_1, b_1), \ldots, (i_l, a_l, b_l)\}$, the character set C or equivalently $\mathcal{L}_C$ is said to* cover *the set of sequences $\{S_{i_1}, \ldots, S_{i_l}\} \subseteq \mathcal{S}$.*

With these prerequisites a gene cluster on multiple sequences can be defined, allowing errors and a quorum.

Definition 9 (*d*-cluster, *ε*-cluster) *Given k sequences, an absolute distance threshold $d \geq 0$ (relative distance threshold $\varepsilon \geq 0$) and a quorum q, $2 \leq q \leq k$, a pair $(C_{repr}, \mathcal{L}_\mathcal{C})$ is called a* d-cluster *(ε-cluster) if and only if $\mathcal{L}_\mathcal{C}$ is the location set of the set $\mathcal{C}$ of all d-occurrences (ε-occurrences) of C_{repr}, and $\mathcal{L}_\mathcal{C}$ covers at least q sequences.*

We simply speak about a cluster *when we mean either a d-cluster or an ε-cluster.*

Example. Considering the sequences $\mathcal{S}$ defined in the first example and a quorum $q = 3$, the following pairs are 1-clusters according to the set transformation distance d_T:

$$\alpha = (\{1,3\}, \{(2,1,6),(3,1,5),(1,1,4),(2,1,2),(2,5,6),(3,3,5),(2,2,5),(3,1,3),\\ (1,1,2),(1,4,4),(2,1,1),(2,6,6),(3,4,5),(2,3,4),(3,1,2)\})$$

with d-occurrences $\mathcal{C}_\alpha = \{\{1,2,3\},\{1,2\},\{2,3\},\{1\},\{3\}\}$, and

$$\beta = (\{1,2,3\}, \{(2,1,6),(3,1,5),(1,1,4),(2,1,2),(2,5,6),(3,3,5),(2,2,5),(3,1,3)\})$$

with d-occurrences $\mathcal{C}_\beta = \{\{1,2,3\},\{1,2\},\{2,3\}\}$.

Remark 4 It follows immediately from the size of a d-neighborhood that the number of d-clusters can be exponential in d. More precisely, an upper bound on the number of d-clusters is the minimum of the cardinality of $\mathcal{N}_d(\mathcal{C}^{all})$ and of $2^{|\mathcal{C}^{all}|}$. However, if d is constant, then the number of d-clusters grows polynomially with the length of the sequences.

Definition 10 (maximality of a cluster) *Consider a fixed distance parameter d (ε). A d-(ε-)cluster $(C_{repr_1}, \mathcal{L}_{\mathcal{C}_1})$ is said to be* included *in a d-(ε-)cluster $(C_{repr_2}, \mathcal{L}_{\mathcal{C}_2})$ if $\mathcal{C}_1 \subset \mathcal{C}_2$, $\mathcal{C}_1 \neq \mathcal{C}_2$.*

A d-(ε-)cluster that is not included in any other d-(ε-)cluster is said to be maximal.

Example. Continuing the last example one can observe that 1-cluster β is included in 1-cluster α, since $\mathcal{C}_\beta \subseteq \mathcal{C}_\alpha$.

Definition 11 (equivalence of clusters) *Two clusters $(C_{repr_1}, \mathcal{L}_{\mathcal{C}_1})$ and $(C_{repr_2}, \mathcal{L}_{\mathcal{C}_2})$ are said to be* equivalent*, denoted by $(C_{repr_1}, \mathcal{L}_{\mathcal{C}_1}) \simeq (C_{repr_2}, \mathcal{L}_{\mathcal{C}_2})$ if $\mathcal{C}_1 = \mathcal{C}_2$. This relation $\simeq$ is an equivalence relation and we denote by $(\{C_{repr_1}, \ldots, C_{repr_k}\},\ \mathcal{L}_{\mathcal{C}})$ the equivalence class containing the clusters $(C_{repr_1},\ \mathcal{L}_{\mathcal{C}_1})$, …, $(C_{repr_k},\ \mathcal{L}_{\mathcal{C}_k})$.*

Example. Considering sequences $\mathcal{S}$ from the previous example, with $d_T = 1$ none of the present d-clusters are equivalent, since all have a different set of d-occurrences. However, with distance $d_T = 2$ all d-clusters become equivalent and one obtains the equivalence class $(\{\{1,2,3\}, \{1,2\}, \{2,3\}, \{1,3\}, \{1\}, \{2\}, \{3\}\}, \mathcal{L}_{\mathcal{C}})$ with d-occurrences $\mathcal{C} = \{\{1,2,3\}, \{1,2\}, \{2,3\}, \{1\}, \{2\}, \{3\}\}$.

Remark 5 Note that if a cluster of an equivalence class is maximal, then all clusters of this class are maximal, and we extend the notion of maximality to equivalence classes of clusters.

3 Finding Common Intervals with Errors

We present three algorithms for the gene cluster detection problem based on different formulations for common intervals with errors. As shown in the previous section, clusters can be defined in terms of character sets. We show that the gene cluster detection problem has equivalent formulations as a maximal clique enumeration problem, and, for d-neighborhoods, as a d-center decision problem. Using the different formulations, we gain more insight into the problem's structure and complexity.

We assume that $|\Sigma| = \Theta(n)$, i.e., alphabet size and sequence length have the same asymptotic behavior.

3.1 Common Intervals as Character Sets

We first give an algorithm to find all d-clusters in a given set of sequences that follows the definition of a d-cluster (Definition 9), as, basically, it generates all possible character sets that could be a representative, and then checks if each of these character sets is a representative for a set of approximate occurrences that cover at least q sequences (see Figure 1 for an illustration). The case of ε-clusters is discussed further below.

Algorithm 1 operates in two phases.

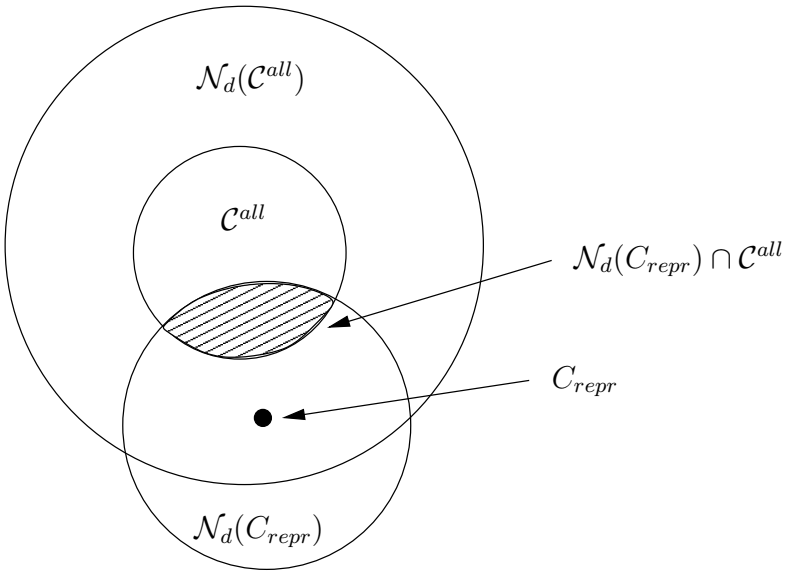

Figure 1: Graphical visualization of the set $\mathcal{C}^{all}$ of all character sets from substrings of the sequence set $\mathcal{S}$ and its d-neighborhood $\mathcal{N}_d(\mathcal{C}^{all})$ (which may be replaced by the ε-neighborhood $\mathcal{U}_\varepsilon(\mathcal{C}^{all})$). The representative $C_{repr} \in \mathcal{N}_d(\mathcal{C}^{all})$ need not occur in $\mathcal{C}^{all}$. The d-occurrences of $\mathcal{C}^{all}$ in $\mathcal{S}$ are given by the intersection of the d-neighborhood $\mathcal{N}_d(C_{repr})$ with $\mathcal{C}^{all}$, the shaded area.

1. First (lines 1–6), it builds $\mathcal{N}^{all} = \mathcal{N}_d(\mathcal{C}^{all})$, which consists of all possible candidates for representatives. This is done without explicitly generating the set $\mathcal{C}^{all}$ first. Instead, we iterate over each substring s in the sequence set $\mathcal{S}$ and add its d-neighborhood to $\mathcal{N}^{all}$. We leave open the details of the set data structure for $\mathcal{N}^{all}$, but suggest that it can be implemented by a trie.

   The running time of this phase is bounded as follows: There are $O(kn^2)$ iterations of the inner for loop (lines 5–6). For the symmetric set distance, each of them takes at most $O(n^{d+1})$ time, for a total running time of $O(kn^{d+3})$. For the set transformation distance, each iteration takes $O(n^{2d+1})$ time, for a total running time of $O(kn^{2d+3})$.

2. In the second phase the algorithm examines each candidate representative in turn (lines 7–24) to find out whether its d-occurrences cover at least q sequences (lines 9–15, the number of covered sequences is counted in the variable *hits*). If yes, we need to re-scan all sequences to find the exact location set of the d-occurrences of the representative (lines 17–23).

   The work for each candidate is $O(kn^3)$; and there are $O(kn^{d+2})$ candidates for symmetric set distance and $O(kn^{2d+2})$ candidates for set transformation distance. This gives a total running time of $O(k^2n^{d+5})$ for symmetric set distance and $O(k^2n^{2d+5})$ for set transformation distance.

Algorithm 1 Naive d-cluster detection

Input: k distinct sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$, a distance $d \geq 0$, a quorum $q \in \{2, \ldots, k\}$

1: ▷ *Build the set $\mathcal{N}^{all}$ of all candidates for C_{repr}:*
2: $\mathcal{N}^{all} \leftarrow \emptyset$
3: **for each** $S \in \mathcal{S}$ **do**
4: **for each** substring s in S **do**
5: $C \leftarrow CharSet(s)$
6: $\mathcal{N}^{all} \leftarrow \mathcal{N}^{all} \cup \mathcal{N}_d(C)$
7: ▷ *Try each candidate as a representative:*
8: **for each** $C_{repr} \in \mathcal{N}^{all}$ **do**
9: $hits \leftarrow 0$ ▷ *counts the number of sequences with at least one d-occurrence*
10: **for** $i = 1, \ldots, k$ **do**
11: **for each** substring s in S_i **do**
12: Compute $\delta \leftarrow dist(C_{repr}, CharSet(s))$
13: **if** $\delta \leq d$ **then**
14: $hits \leftarrow hits + 1$
15: **next** i ▷ *Move on after finding first d-occurrence*
16: **if** $hits \geq q$ **then**
17: ▷ *Compute the location set of C_{repr}'s d-occurrences*
18: $\mathcal{L} \leftarrow \emptyset$
19: **for** $i = 1, \ldots, k$ **do**
20: **for each** a, b with $1 \leq a \leq b \leq |S_i|$ **do**
21: $C \leftarrow CharSet(S_i[a, b])$
22: **if** $((a = 1$ **or** $S_i[a-1] \notin C)$ **and** $(b = n$ **or** $S_i[b+1] \notin C))$ **and** $dist(C_{repr}, C) \leq d$ **then**
23: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(i, a, b)\}$
24: **report** $(C_{repr}, \mathcal{L})$

Note that, to save memory, the set $\mathcal{N}^{all}$ may not need to be constructed explicitly. Then some C_{repr} might be tested several times, though, also generating the same output several times.

Clearly, the naive algorithm would benefit from several optimizations. For example, before line 11, there are still $k - i + 1$ sequences to check, and if this is lower than the missing number of hits ($q - hits$), we can abort checking this particular C_{repr}. Also, in the loops starting at lines 11 and 20, we may in fact not need to check every substring, but only those of certain constrained sizes. These optimizations are not the focus of this report, however.

We now discuss the necessary changes for ε-clusters: Instead of a distance parameter d, we now receive a relative distance threshold $0 < \varepsilon < 1$ as input.

The main difference is the generation of the candidate set $\mathcal{N}^{all}$, where we must ask in line 6: Which sets R have C in their ε-neighborhood? Recall that the neighborhood relation is not symmetric. We may replace line 6 in Algorithm 1 by:
$\mathcal{N}^{all} \leftarrow \mathcal{N}^{all} \cup \{R \subseteq \Sigma \mid C \in \mathcal{U}_\varepsilon(R)\}$.

We leave open the efficient construction of the above set and only give a size bound for R.

Proposition 1 *If $\varepsilon < 1$ and $C \in \mathcal{U}_\varepsilon(R)$, then*

$$\lceil |C|/(1+\varepsilon) \rceil \leq |R| \leq \lfloor |C|/(1-\varepsilon) \rfloor.$$

Proof. The condition for R is that $dist(R, C) \leq \varepsilon|R|$. For $\varepsilon < 1$, this bounds the size of R from above: If $|R| = |C| + \delta$, then $dist(R, C) \geq \delta$. Thus certainly, $C \notin \mathcal{U}_\varepsilon(R)$ for $\delta/|R| > \varepsilon$, or equivalently, $\varepsilon < \delta/(|C| + \delta)$, or $\delta > \varepsilon|C|/(1-\varepsilon)$, or $|R| > |C|/(1-\varepsilon)$. In other words, the size of admissible candidates R is bounded by $\lfloor |C|/(1-\varepsilon) \rfloor$.

With a similar argument for the ansatz $|R| = |C| - \delta$, we obtain that for $|R| < |C|/(1+\varepsilon)$ we also can never find C in the ε-neighborhood of R. □

The other (now obvious) changes to Algorithm 1 are to replace d in lines 13 and 22 by $\varepsilon \cdot |C_{repr}|$.

It is unclear how the complexity of the algorithm changes.

3.2 Common Intervals as Cliques in Graphs

In this section we provide an alternative formulation of clusters in terms of cliques in graphs. We first focus on d-clusters and discuss ε-clusters further below.

The intuition behind the clique representation is that character sets of the considered sequences that are members of a same d-cluster (as d-occurrences of a representative) are at most at a distance of $2d$, and can then be seen as a set of close nodes in a graph where character sets are vertices and edges are weighted by the distance between vertices.

More formally, we will consider a vertex-labeled bipartite graph $G = (U \cup V, E)$. The two vertex subsets are defined as follows. U contains one vertex u for each character set $C \in \mathcal{C}^{all}$, labeled with $\lambda(u) = C$. V contains one vertex v for each character set $C \in \mathcal{N}_d(\mathcal{C}^{all})$, labeled with $\lambda(v) = C$. Note that since $\mathcal{C}^{all} \subseteq \mathcal{N}_d(\mathcal{C}^{all})$, the label of each vertex in U also occurs as a label of some vertex in V. Two vertices $u \in U$ and $v \in V$ are connected by an edge if the distance between the represented character sets is not larger than d, i.e., $E = \{(u, v) \in U \times V \mid dist(\lambda(u), \lambda(v)) \leq d\}$. Finally, we say that a set of vertices $U' \subseteq U$ covers q sequences if the set of character sets $\lambda(U') = \cup_{u \in U'} \lambda(u)$ covers at least q sequences.

A schematic representation of such a bipartite graph is shown in Figure 2.

Proposition 2 *There is a one-to-one correspondence between maximal bipartite cliques of G that cover at least q sequences and equivalence classes of d-clusters that cover at least q sequences.*

Proof. Let $U' \subseteq U$ and $V' \subseteq V$ be two sets of vertices that define a maximal bipartite clique. By definition of G, for every $v \in V'$, the representative $\lambda(v)$ is at distance at most d of all the character sets in $\mathcal{C} = \lambda(U') := \cup_{u \in U'} \lambda(u)$. Moreover, the fact that the clique is maximal implies that $\mathcal{C}$ is the set of all d-occurrences of $\lambda(v)$, which is then a representative of $\mathcal{C}$. As U' covers at least q sequences, one can say that $(\lambda(v), \mathcal{L}_\mathcal{C})$ is a

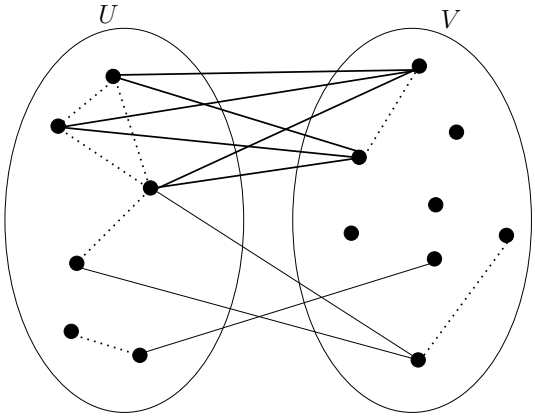

Figure 2: Bipartite graph G with vertex sets U and V such that $\lambda(U) = \mathcal{C}^{all}$ and $\lambda(V) = \mathcal{N}_d(\mathcal{C}^{all})$. Two vertices $u \in U$, $v \in V$ are connected by a solid edge if their distance is not greater than d. The upper bipartite clique represents an equivalence class with two representatives, the two lower ones represent d-clusters. The extended graph G' is obtained by adding the dotted edges between vertices within U (respectively V), if vertices have distance at most $2d$.

d-cluster that covers at least q sequences. Finally, by definition of a maximal bipartite clique, V' is the maximal set of vertices of V having such properties, and is then the complete set of representatives of $\mathcal{C}$, and then (U', V') defines the equivalence class containing $(\lambda(v), \mathcal{L}_\mathcal{C})$.

Conversely, let $(C_{repr}, \mathcal{L}_\mathcal{C})$ be a d-cluster covering q sequences. By definition of G, there is a subset $U' \subseteq U$ such that $\mathcal{C} = \lambda(U')$ and U' covers q sequences. There is also a vertex $v \in V$ such that $\lambda(v) = C_{repr}$, that is connected to all vertices of U' and no other vertex, as $\mathcal{C}$ is the set of all d-occurrences of C_{repr}. By extending these properties to all representatives of the clusters in the equivalence class of $(C_{repr}, \mathcal{L}_\mathcal{C})$, one gets a maximal bipartite clique in G. □

Remark 6 In the case of ε-clusters, we can proceed similarly, but define the vertex set V as the set $\mathcal{N}^{all}$ in the same way as in the ε-cluster version of Algorithm 1 (cf. Proposition 1). The edge set becomes $E := \{(u, v) \in U \times V \mid dist(\lambda(u), \lambda(v)) \leq \varepsilon|\lambda(v)|\}$.

Given an equivalence class of clusters, it is immediate to get all clusters in this class, by listing the vertices of V. Hence, the problem of detecting the set of all clusters reduces to the enumeration of all maximal bipartite cliques of G, which is a well studied algorithmic problem; see [22], for example.

The procedure is sketched in Algorithm 2. Its complexity depends on the construction of the graph G and finding its maximal cliques. The graph contains a number of vertices that is exponential in d (see also the previous section). In any case, if d is considered as a constant, the graph G has a number of vertices and of edges that is polynomial in n.

Algorithm 2 Maximal cliques as equivalence classes of clusters

Input: k sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$, a distance d, a quorum $q \in \{2, \ldots, k\}$

1: construct the vertex sets U and V, as explained in the text
2: construct the bipartite graph $G = (U \cup V, E)$, as explained in the text
3: **for each** maximal bipartite clique (U', V') in G **do**
4: **if** the vertices of U' cover at least q sequences **then**
5: **report** (U', V')

Unfortunately, a graph can have an exponential number of (maximal) cliques [23, 31], but there are efficient algorithms for the enumeration of all maximal cliques in bipartite graphs that have a time complexity that is polynomial in the number of maximal cliques [22].

Note that there are also efficient algorithms to enumerate all maximal cliques in non-bipartite graphs [4, 10]. In order to make use of these algorithms for d-clusters, one can define an extended version of G, denoted G', obtained by adding an edge between every pair of vertices of U (resp. V) that represent character sets at a distance of at most $2d$. It is easy to see that the (ordinary) maximal cliques in G' denote the same equivalence classes of d-clusters as the bipartite cliques in G, if they contain at least one vertex of each vertex set U and V.

3.3 Common Intervals as Binary Arrays

In this section, we only consider d-clusters, not ε-clusters, and we characterise them as binary arrays.

Recall that we assume that $\Sigma = \{1, \ldots, \sigma\}$. A character set $C \subseteq \Sigma$ can be represented as an array $A \in \{0, 1\}^\sigma$, where $A[i] = 1$, if character $i \in \Sigma$ is contained in C, and $A[i] = 0$, otherwise.

Definition 12 (d-center) *Given a set of binary arrays $\mathcal{A}$, a binary array $B \in \{0, 1\}^\sigma$ is called a* d-center *of $\mathcal{A}$ if and only if $dist(A, B) \leq d$ for all $A \in \mathcal{A}$.*

An illustration of this definition is given in Figure 3.

Observe that the number of d-centers for a given set of arrays can be exponential in d. For example, consider the arrays $A_1 = 0^\sigma$ and A_2 with $2d$ entries with ones. There exist $\binom{2d}{d}$ different d-centers.

Proposition 3 *Let $\mathcal{A}^{all}$ be the set of binary arrays representing the character sets from $\mathcal{C}^{all}$. If the character sets corresponding to a subset of $\mathcal{A}^{all}$ cover at least q sequences and this subset has a d-center, then there exists a d-cluster of the corresponding (and possibly some additional) character sets from $\mathcal{C}^{all}$.*

Proof. Consider a d-center B of the set $\{A_1, \ldots, A_k\} \subseteq \mathcal{A}^{all}$ whose corresponding character sets cover at least q sequences. Let C_{repr} be the character set corresponding to B. The d-occurrences of C_{repr} can be represented as

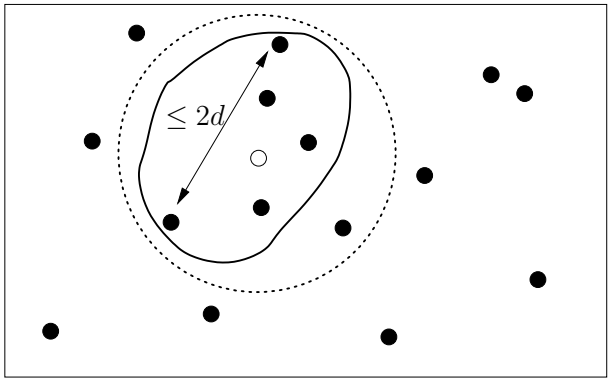

Figure 3: Space of bit-arrays with elements $\mathcal{A}^{all}$. The small open circle represents a d-center for a subset of elements of $\mathcal{A}^{all}$ encircled by the solid line. Note that such a subset is not necessarily a d-cluster, since further elements can be within distance d to the d-center (represented by the dotted circle). The pairwise distance between the elements within the dotted circle never exceeds $2d$.

$$\mathcal{C}^{all} \cap \mathcal{N}_d(C_{repr}) = \{C_1, \ldots, C_k, C_1^*, \ldots, C_l^*\},$$

where the character sets $\{C_1, \ldots, C_k\}$ correspond to the binary arrays $\{A_1, \ldots, A_k\}$. Thus, there exists a d-cluster $(C_{repr},\ \mathcal{L}_{\mathcal{C}})$ such that $\mathcal{C}$ consists of the character sets corresponding to $\{A_1, \ldots, A_k\}$ and possibly some more character sets. □

For the Hamming distance, defined as the number of positions in which two strings differ, there exist several results for the problem of finding an array B that minimizes the maximum distance between B and any other array of a given set of arrays. In the context of coding theory, the decision version of the problem has been shown to be NP complete [12]. A similar, more general result was given in [19]. Despite the NP hardness, the problem has been widely studied in the last few years due to the importance of its applications. Approximation algorithms were given in [13], and it was shown to be fixed-parameter tractable in [15].

The problem was also shown to be NP hard for the Levenshtein distance [6] and for the weighted edit distance [24] on sequences.

Like the edit distance, the set transformation distance allows for insertions, deletions, and substitutions, but on the character sets of substrings instead of the substrings itself. That makes it better comparable to the Hamming distance on binary arrays. More precisely, letting k_1 (respectively k_2) be the number of positions i with $A_1[i] = 1$ and $A_2[i] = 0$ (respectively $A_1[i] = 0$ and $A_2[i] = 1$), we have $d_T(A_1, A_2) = \max\{k_1, k_2\}$, while the Hamming distance is $d_H(A_1, A_2) = k_1 + k_2$.

The following proposition states a simple connection between the Hamming distance d_H and the set transformation distance d_T.

Proposition 4 *If there exists a d-center for $\{A_1, \ldots, A_k\}$ under d_H, then there exists*

a d-center for $\{A_1, \ldots, A_k\}$ *under* d_T*; if there exists a d-center for* $\{A_1, \ldots, A_k\}$ *under* d_T*, then there exists a* $2d$*-center for* $\{A_1, \ldots, A_k\}$ *under* d_H*.*

We leave it open for the moment if this can be used to show that finding a d_T-center is NP hard or not.

The following observation, shown for the Hamming distance in [14], also holds for the set transformation distance due to the triangle inequality.

Proposition 5 *Given* k *binary arrays* $A_1, \ldots, A_k$ *and an integer* $d \geq 0$*, if there are* $i, j \in \{1, \ldots, k\}$ *with* $d_T(A_i, A_j) > 2d$*, then there is no array* B *with* $\max_{i=1,\ldots,k} d_T(B, A_i) \leq d$*.*

Instead of testing all subsets of arrays whether they have a d-center, it is sufficient to test only those subsets whose maximal pairwise distance between any two arrays is not greater than $2d$. This heuristic is applied in Algorithm 3. In addition every set of character sets corresponding to a subset of arrays has to cover at least q sequences, which excludes more subsets and makes further tests dispensable.

A problem illustrated in Figure 3 is that a subset of $\mathcal{A}^{all}$ with a d-center forms not necessarily a d-cluster, since further elements of $\mathcal{A}^{all}$ can be within distance d to the d-center, depending on the choice of the d-center (see Proposition 3). To ensure that the output of Algorithm 3 consists only of representatives with all their d-occurrences, it is tested in line 7 whether the set of potential d-occurrences $\mathcal{A}^{pot}$ is already contained in a set stored before in an output set $\mathfrak{A}$. This way the returned $\mathfrak{A}$ consists only of pairs representing maximal d-clusters (see Definition 10), which does not lose any information and eliminates d-centers (representatives) with incomplete sets of d-occurrences.

Algorithm 3 Maximal subsets with d-center

Input: k sequences $\mathcal{S} = \{S_1, \ldots, S_k\}$, a distance d, a quorum $q \in \{2, \ldots, k\}$

1: let $\mathfrak{A} \leftarrow \emptyset$
2: let $\mathcal{A}^{all} \leftarrow \emptyset$
3: **for each** $S \in \mathcal{S}$ **do**
4: detect all character sets in S and add them as binary arrays to the set $\mathcal{A}^{all}$
5: **for each** $\mathcal{A}^{pot} = \{A_1, \ldots, A_k\} \subseteq \mathcal{A}^{all}$ with $dist(A_i, A_j) \leq 2d$ for all i, j **do**
6: **if** the character sets corresponding to $\mathcal{A}^{pot}$ cover at leat q sequences and there exists a d-center of $\mathcal{A}^{pot}$ **then**
7: **if** $\mathcal{A}^{pot}$ is no proper subset of some element of $\mathfrak{A}$ **then**
8: store $\mathcal{A}^{pot}$ and its d-center in $\mathfrak{A}$
9: return the elements of $\mathfrak{A}$

4 Discussion

A shortcoming of our three algorithms is the fact that cluster maximality is not efficiently taken into account. However, maximality is central to filter the enormous output produced by algorithms finding gene clusters allowing errors. Indeed, with our algorithms

it is possible to extract from the set of clusters the ones that are maximal, but it is still open to design methods that compute directly maximal gene clusters.

The redundancy of the output is a major problem with Algorithm 1, due to its approach that is centered on computing clusters from representatives. It is handled in a very natural way in the two other algorithms, in two different ways: in Algorithm 2, all possible representatives for a cluster are considered at the same time when extending a clique to get a maximal clique, while in Algorithm 3, clusters are detected from the set of all d-occurrences first, and only one representative is produced.

A disadvantage of the latter algorithms is that both require the enumeration of large combinatorial sets, namely maximal cliques of a graph or all subsets of the set of all character sets. On the other hand, the output of Algorithm 2 is an equivalence class of clusters which provides much more information than a single representative for a cluster. In order to produce such an output, Algorithm 1 does not seem to offer an efficient basis, and for Algorithm 3 it is not clear whether all d-centers for a given set of strings can be computed efficiently and represented in a compact way.

5 Conclusion

This report presents a model for gene clusters, defined on multiple sequences and allowing to handle errors in a way that overcomes the drawbacks of so called max-gap clusters. Previous gene cluster models are analysed and properties desirable to the design of the new model are worked out.

By devising three equivalent formulations of the presented approach, more insight to the complexity of finding clusters allowing for errors is gained.

Another formulation based on integer linear programs that incorporates all existing models and also offers a quantitative evaluation of cluster quality appears in [27].

The balance between a highest possible degree of flexibility and biologically relevant output in contrast to the feasibility of algorithms reporting those clusters remains the major challenge for new gene cluster models. The presented approach emphasizes the first aspect, though finding efficient algorithms would revaluate its usefulness.

References

[1] A. Amir, A. Apostolico, G. M. Landau, and G. Satta. Efficient text fingerprinting via Parikh mapping. *J. Discr. Alg.*, 1(5-6):409–421, 2003.

[2] A. Bergeron, C. Chauve, F. de Mongolfier, and M. Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. In *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, volume 3669 of *Lecture Notes in Computer Science*, pages 779–790. Springer Verlag, 2005.

[3] G. Blin, A. Chateau, C. Chauve, and Y. Gingras. Inferring positional homologs with common intervals of sequences. In *Comparative Genomics, RECOMB 2006 Inter-*

national Workshop, RCG 2006, volume 4205 of *Lecture Notes in Bioinformatics*, pages 24–38. Springer Verlag, 2006.

[4] F. Cazals and C. Karande. Reporting maximal cliques: new insights into an old problem. Technical Report 5615, INRIA, 2005.

[5] T. Dandekar, B. Snel, M. Huynen, and P. Bork. Conservation of gene order: a fingerprint for proteins that physically interact. *Trends in Biochemical Sciences*, 23(9):324–328, 1998.

[6] C. de la Higuera and F. Casacuberta. Topology of strings: Median string is NP-complete. *Theoretical Computer Science*, 230(1–2):39–48, 2000.

[7] G. Didier. Common intervals of two sequences. In G. Benson and R. Page, editors, *Proceedings of the Third International Workshop on Algorithms in Bioinformatics, WABI 2003*, volume 2812 of *LNBI*, pages 17–24, Berlin, 2003. Springer Verlag.

[8] G. Didier, T. Schmidt, J. Stoye, and D. Tsur. Character sets of strings. *J. Discr. Alg.*, To appear.

[9] Y. Diekmann, M.-F. Sagot, and E. Tannier. Evolution under reversals: parsimony and conservation of common intervals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, To appear.

[10] D. Eppstein. All maximal independent sets and dynamic dominance for sparse graphs. In *roceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 451–459. SIAM, 2005.

[11] R. Eres, G. M. Landau, and L. Parida. A combinatorial approach to automatic discovery of cluster-patterns. In *Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001 Jerusalem, Israel, July 1-4, 2001 Proceedings*, volume 2812 of *Lecture Notes in Computer Science*, pages 139–150. Springer Verlag, 2003.

[12] M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30:113–119, 1997.

[13] L. Gasieniec, J. Jansson, and A. Lingas. Efficient approximation algorithms for the hamming center problem. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms, 17-19 January 1999, Baltimore, Maryland*, pages 905–906. SIAM, 1999.

[14] J. Gramm, R. Niedermeier, and P. Rossmanith. Exact solutions for CLOSEST STRING and related problems. In *Algorithms and Computation, 12th International Symposium, ISAAC 2001, Christchurch, New Zealand, December 19-21, 2001, Proceedings*, volume 2223 of *Lecture Notes in Computer Science*, pages 441–453. Springer Verlag, 2001.

[15] J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

[16] X. He and M. H. Goldwasser. Identifying conserved gene clusters in the presence of homology families. *Journal of Computational Biology*, 12(6):638–656, 2005.

[17] S. Heber and J. Stoye. Finding all common intervals of k permutations. In *Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001, Jerusalem, Israel, July 1-4, 2001 Proceedings*, volume 2089 of *Lecture Notes in Computer Science*, pages 207–218. Springer Verlag, 2001.

[18] R. Hoberman and D. Durand. The incompatible desiderata of gene cluster properties. In *Comparative Genomics, RECOMB 2005 International Workshop, RCG 2005, Dublin, Ireland, September 18-20, 2005, Proceedings*, volume 3678 of *Lecture Notes in Bioinformatics*, pages 73–87. Springer Verlag, 2005.

[19] J. K. Lanctôt, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003.

[20] W. C. III Lathe, B. Snel, and P. Bork. Gene context of a higher order than operons. *Trends in Biochemical Sciences*, 25(10):474–479, 2000.

[21] N. Luc, J.-L. Risler, A. Bergeron, and M. Raffinot. Gene teams: a new formalization of gene clusters for comparative genomics. *Computational Biology and Chemistry*, 27(1):59–67, 2003.

[22] K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In *Algorithm Theory – SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, volume 3111 of *Lecture Notes in Computer Science*, pages 260–272. Springer Verlag, 2004.

[23] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.

[24] F. Nicolas and E. Rivals. Complexities of the centre and median string problems. In R. Baeza-Yates, E. Chavez, and M. Crochemore, editors, *Combinatorial Pattern Matching, 14th Annual Symposium, CPM 2003, Morelia, Michocn, Mexico, June 25-27, 2003, Proceedings*, volume 2676 of *Lecture Notes in Computer Science*, pages 315–327. Springer Verlag, 2003.

[25] R. Overbeek, M. Fonstein, M. D'Souza, G. D. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2896–2901, 1999.

[26] S. Pasek, A. Bergeron, J.-L. Rilser, A. Louis, E. Ollivier, and M. Raffinot. Identification of genomic features using microsyntenies of domains: domain teams. *Genome Research*, 15(6):867–874, 2005.

[27] S. Rahmann and G.W. Klau. Integer linear programs for discovering approximate gene clusters. In *Proceedings of WABI*, volume 4175 of *LNBI*, pages 298–309. Springer, 2006.

[28] T. Schmidt. *Efficient algorithms for gene cluster detection in prokaryotic genomes*. PhD thesis, Faculty of Technology, Bielefeld University, Germany, 2005.

[29] T. Schmidt and J. Stoye. Quadratic time algorithms for finding common intervals in two or more sequences. In *Combinatorial Pattern Matching, 15th Annual Symposium, CPM 2004, Istanbul,Turkey, July 5-7, 2004, Proceedings*, volume 3109 of *Lecture Notes in Computer Science*, pages 347–358. Springer Verlag, 2004.

[30] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.

[31] D.R. Wood. On the maximum number of cliques in a graph. Technical report, arXiv:math.CO/0602191, 2006.

[32] Y. Zheng, B. P. Anton, R. J. Roberts, and S. Kasif. Phylogenetic detection of conserved gene clusters in microbial genomes. *BMC Bioinformatics*, 6:243, 2005.

Bisher erschienene Reports an der Technischen Fakultät
Stand: 2006-06-09

94-01 Modular Properties of Composable Term Rewriting Systems
(Enno Ohlebusch)

94-02 Analysis and Applications of the Direct Cascade Architecture
(Enno Littmann, Helge Ritter)

94-03 From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction
(Robert Giegerich, Stefan Kurtz)

94-04 Die Verwendung unscharfer Maße zur Korrespondenzanalyse in Stereo Farbbildern
(André Wolfram, Alois Knoll)

94-05 Searching Correspondences in Colour Stereo Images – Recent Results Using the Fuzzy Integral
(André Wolfram, Alois Knoll)

94-06 A Basic Semantics for Computer Arithmetic
(Markus Freericks, A. Fauth, Alois Knoll)

94-07 Reverse Restructuring: Another Method of Solving Algebraic Equations
(Bernd Bütow, Stephan Thesing)

95-01 PaNaMa User Manual V1.3
(Bernd Bütow, Stephan Thesing)

95-02 Computer Based Training-Software: ein interaktiver Sequenzierkurs
(Frank Meier, Garrit Skrock, Robert Giegerich)

95-03 Fundamental Algorithms for a Declarative Pattern Matching System
(Stefan Kurtz)

95-04 On the Equivalence of E-Pattern Languages
(Enno Ohlebusch, Esko Ukkonen)

96-01 Static and Dynamic Filtering Methods for Approximate String Matching
(Robert Giegerich, Frank Hischke, Stefan Kurtz, Enno Ohlebusch)

96-02 Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language
(Alois Knoll, Bernd Hildebrand, Jianwei Zhang)

96-03 Correctness in System Engineering
(Peter Ladkin)

96-04 An Algebraic Approach to General Boolean Constraint Problems
(Hans-Werner Güsgen, Peter Ladkin)

96-05 Future University Computing Resources
(Peter Ladkin)

96-06 Lazy Cache Implements Complete Cache
(Peter Ladkin)

96-07 Formal but Lively Buffers in TLA+
(Peter Ladkin)

96-08 The X-31 and A320 Warsaw Crashes: Whodunnit?
(Peter Ladkin)

96-09 Reasons and Causes
(Peter Ladkin)

96-10 Comments on Confusing Conversation at Cali
(Dafydd Gibbon, Peter Ladkin)

96-11 On Needing Models
(Peter Ladkin)

96-12 Formalism Helps in Describing Accidents
(Peter Ladkin)

96-13 Explaining Failure with Tense Logic
(Peter Ladkin)

96-14 Some Dubious Theses in the Tense Logic of Accidents
(Peter Ladkin)

96-15 A Note on a Note on a Lemma of Ladkin
(Peter Ladkin)

96-16 News and Comment on the AeroPeru B757 Accident
(Peter Ladkin)

97-01 Analysing the Cali Accident With a WB-Graph
(Peter Ladkin)

97-02 Divide-and-Conquer Multiple Sequence Alignment
(Jens Stoye)

97-03 A System for the Content-Based Retrieval of Textual and Non-Textual Documents Based on Natural Language Queries
(Alois Knoll, Ingo Glöckner, Hermann Helbig, Sven Hartrumpf)

97-04 Rose: Generating Sequence Families
(Jens Stoye, Dirk Evers, Folker Meyer)

97-05 Fuzzy Quantifiers for Processing Natural Language Queries in Content-Based Multimedia Retrieval Systems
(Ingo Glöckner, Alois Knoll)

97-06 DFS – An Axiomatic Approach to Fuzzy Quantification
(Ingo Glöckner)

98-01 Kognitive Aspekte bei der Realisierung eines robusten Robotersystems für Konstruktionsaufgaben
(Alois Knoll, Bernd Hildebrandt)

98-02 A Declarative Approach to the Development of Dynamic Programming Algorithms, applied to RNA Folding
(Robert Giegerich)

98-03 Reducing the Space Requirement of Suffix Trees
(Stefan Kurtz)

99-01 Entscheidungskalküle
(Axel Saalbach, Christian Lange, Sascha Wendt, Mathias Katzer, Guillaume Dubois, Michael Höhl, Oliver Kuhn, Sven Wachsmuth, Gerhard Sagerer)

99-02 Transforming Conditional Rewrite Systems with Extra Variables into Unconditional Systems
(Enno Ohlebusch)

99-03 A Framework for Evaluating Approaches to Fuzzy Quantification
(Ingo Glöckner)

99-04 Towards Evaluation of Docking Hypotheses using elastic Matching
(Steffen Neumann, Stefan Posch, Gerhard Sagerer)

99-05 A Systematic Approach to Dynamic Programming in Bioinformatics. Part 1 and 2: Sequence Comparison and RNA Folding
(Robert Giegerich)

99-06 Autonomie für situierte Robotersysteme – Stand und Entwicklungslinien
(Alois Knoll)

2000-01 Advances in DFS Theory
(Ingo Glöckner)

2000-02 A Broad Class of DFS Models
(Ingo Glöckner)

2000-03 An Axiomatic Theory of Fuzzy Quantifiers in Natural Languages
(Ingo Glöckner)

2000-04 Affix Trees
(Jens Stoye)

2000-05 Computergestützte Auswertung von Spektren organischer Verbindungen
(Annika Büscher, Michaela Hohenner, Sascha Wendt, Markus Wiesecke, Frank Zöllner, Arne Wegener, Frank Bettenworth, Thorsten Twellmann, Jan Kleinlützum, Mathias Katzer, Sven Wachsmuth, Gerhard Sagerer)

2000-06 The Syntax and Semantics of a Language for Describing Complex Patterns in Biological Sequences
(Dirk Strothmann, Stefan Kurtz, Stefan Gräf, Gerhard Steger)

2000-07 Systematic Dynamic Programming in Bioinformatics (ISMB 2000 Tutorial Notes)
(Dirk J. Evers, Robert Giegerich)

2000-08 Difficulties when Aligning Structure Based RNAs with the Standard Edit Distance Method
(Christian Büschking)

2001-01 Standard Models of Fuzzy Quantification
(Ingo Glöckner)

2001-02 Causal System Analysis
(Peter B. Ladkin)

2001-03 A Rotamer Library for Protein-Protein Docking Using Energy Calculations and Statistics
(Kerstin Koch, Frank Zöllner, Gerhard Sagerer)

2001-04 Eine asynchrone Implementierung eines Microprozessors auf einem FPGA
(Marco Balke, Thomas Dettbarn, Robert Homann, Sebastian Jaenicke, Tim Köhler, Henning Mersch, Holger Weiss)

2001-05 Hierarchical Termination Revisited
(Enno Ohlebusch)

2002-01 Persistent Objects with O2DBI
(Jörn Clausen)

2002-02 Simulation von Phasenübergängen in Proteinmonoschichten
(Johanna Alichniewicz, Gabriele Holzschneider, Morris Michael, Ulf Schiller, Jan Stallkamp)

2002-03 Lecture Notes on Algebraic Dynamic Programming 2002
(Robert Giegerich)

2002-04 Side chain flexibility for 1:n protein-protein docking
(Kerstin Koch, Steffen Neumann, Frank Zöllner, Gerhard Sagerer)

2002-05 ElMaR: A Protein Docking System using Flexibility Information
(Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)

2002-06 Calculating Residue Flexibility Information from Statistics and Energy based Prediction
(Frank Zöllner, Steffen Neumann, Kerstin Koch, Franz Kummert, Gerhard Sagerer)

2002-07 Fundamentals of Fuzzy Quantification: Plausible Models, Constructive Principles, and Efficient Implementation
(Ingo Glöckner)

2002-08 Branching of Fuzzy Quantifiers and Multiple Variable Binding: An Extension of DFS Theory
(Ingo Glöckner)

2003-01 On the Similarity of Sets of Permutations and its Applications to Genome Comparison
(Anne Bergeron, Jens Stoye)

2003-02 SNP and mutation discovery using base-specific cleavage and MALDI-TOF mass spectrometry
(Sebastian Böcker)

2003-03 From RNA Folding to Thermodynamic Matching, including Pseudoknots
(Robert Giegerich, Jens Reeder)

2003-04 Sequencing from compomers: Using mass spectrometry for DNA de-novo sequencing of 200+ nt
(Sebastian Böcker)

2003-05 Systematic Investigation of Jumping Alignments
(Constantin Bannert)

2003-06 Suffix Tree Construction and Storage with Limited Main Memory
(Klaus-Bernd Schürmann, Jens Stoye)

2003-07 Sequencing from compomers in thepresence of false negative peaks
(Sebastian Böcker)

2003-08 Genalyzer: An Interactive Visualisation Tool for Large-Scale Sequence Matching – Biological Applications and User Manual
(Jomuna V. Choudhuri, Chris Schleiermacher)

2004-01 Sequencing From Compomers is NP-hard
(Sebastian Böcker)

2004-02 The Money Changing Problem revisited: Computing the Frobenius number in time $O(k\, a_1)$
(Sebastian Böcker, Zsuzsanna Lipták)

2004-03 Accelerating the Evaluation of Profile HMMs by Pruning Techniques
(Thomas Plötz, Gernot A. Fink)

2004-04 Optimal Group Testing Strategies with Interval Queries and Their Application to Splice Site Detection
(Ferdinando Cicalese, Peter Damaschke, Ugo Vaccaro)

2004-05 Compressed Representation of Sequences and Full-Text Indexes
(Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, Gonzalo Navarro)

2005-01 Overlaps Help: Improved Bounds for Group Testing with Interval Queries
(Ferdinando Cicalese, Peter Damaschke, Libertad Tansini, Sören Werth)

2005-02 Two batch Fault-tolerant search with error cost constraints: An application to learning
(Ferdinando Cicalese)

2005-03 Searching for the Shortest Common Supersequence
(Sergio A. de Carvalho Jr., Sven Rahmann)

2005-04 Counting Suffix Arrays and Strings
(Klaus-Bernd Schürmann, Jens Stoye)

2005-05 Alignment of Tandem Repeats with Excision, Duplication, Substitution and Indels (EDSI)
(Michael Sammeth, Jens Stoye)

2005-06 Statistics of Cleavage Fragments in Random Weighted Strings
(Hans-Michael Kaltenbach, Henner Sudek, Sebastian Böcker, Sven Rahmann)

2006-01 Decomposing metabolomic isotope patterns
(Sebastian Böcker, Zsuzsanna Lipták, Anton Pervukhin)