electronics

MDPI

Article

Genetic Algorithms to Maximize the Relevant Mutual Information in Communication Receivers †

Jan Lewandowsky [1,*], Sumedh Jitendra Dongare [2], Rocío Martín Lima [1], Marc Adrat [1], Matthias Schrammen [3] and Peter Jax [3]

[1] Fraunhofer Institute for Communication, Information Processing and Ergonomics, Fraunhoferstraße 20, 53343 Wachtberg, Germany; rocio.martin.lima@fkie.fraunhofer.de (R.M.L.); marc.adrat@fkie.fraunhofer.de (M.A.)
[2] Communications Engineering Lab, Technical University of Darmstadt, Landgraf-Georg-Straße 4, 64283 Darmstadt, Germany; s.dongare@nt.tu-darmstadt.de
[3] Institute of Communication Systems, RWTH Aachen University, Muffeter Weg 3a, 52074 Aachen, Germany; schrammen@iks.rwth-aachen.de (M.S.); jax@iks.rwth-aachen.de (P.J.)
* Correspondence: jan.lewandowsky@fkie.fraunhofer.de; Tel.: +49-228-9435-731

† This article is an extended and improved version of our paper published in: Lewandowsky, J.; Dongare, S.J.; Adrat, M.; Schrammen, M.; Jax, P. Optimizing parametrized information bottleneck compression mappings with genetic algorithms. In Proceedings of the 14th International Conference on Signal Processing and Communication Systems (ICSPCS'2020), Adelaide, Australia, 14–16 December 2020; pp. 1–8.

Abstract: The preservation of relevant mutual information under compression is the fundamental challenge of the information bottleneck method. It has many applications in machine learning and in communications. The recent literature describes successful applications of this concept in quantized detection and channel decoding schemes. The focal idea is to build receiver algorithms intended to preserve the maximum possible amount of relevant information, despite very coarse quantization. The existent literature shows that the resulting quantized receiver algorithms can achieve performance very close to that of conventional high-precision systems. Moreover, all demanding signal processing operations get replaced with lookup operations in the considered system design. In this paper, we develop the idea of maximizing the preserved relevant information in communication receivers further by considering parametrized systems. Such systems can help overcome the need of lookup tables in cases where their huge sizes make them impractical. We propose to apply genetic algorithms which are inspired from the natural evolution of the species for the problem of parameter optimization. We exemplarily investigate receiver-sided channel output quantization and demodulation to illustrate the notable performance and the flexibility of the proposed concept.

Keywords: information bottleneck; mutual information; genetic algorithms; machine learning

Citation: Lewandowsky, J.; Dongare, S.J.; Martín Lima, R.; Adrat, M.; Schrammen, M.; Jax, P. Genetic Algorithms to Maximize the Relevant Mutual Information in Communication Receivers. *Electronics* **2021**, *10*, 1434. https://doi.org/10.3390/electronics10121434

Academic Editor: Tadeusz A. Wysocki

Received: 20 May 2021
Accepted: 10 June 2021
Published: 15 June 2021

1. Introduction

The information bottleneck method is a powerful framework from the machine learning field [1]. Its fundamental idea is to compress an observed random variable Y to some compressed representation T according to a compression rule. This rule is designed to preserve so-called relevant mutual information $\mathrm{I}(\mathsf{X};\mathsf{T}) \leq \mathrm{I}(\mathsf{X};\mathsf{Y})$, where X is a properly chosen relevant random variable of interest. The method is very generic and has numerous applications, for example, in image and speech processing, in astronomy and in neuroscience [2–4].

In the past few years, the method has also attracted considerable attention in the communications community. It was revealed to be useful in the design of strongly quantized baseband signal processing algorithms for detection and channel decoding with low complexity, but performance close to that of non-quantized conventional reference algorithms [5–7]. The communications-related applications of the method lead from the design of channel output quantizers over the decoding of low-density parity-check codes

and polar codes to entire baseband receiver chains that include channel estimation and detection [5–13]. Fundamentally, the idea of most aforementioned applications of the method in communications is to design deterministic compression mappings $t = f(y)$ that replace the classical arithmetical operations in the baseband signal processing algorithms. These mappings are typically considered as lookup tables that store the respective t for each possible y. The lookup table approach sketched above is well-suited for many of the baseband signal processing problems already studied in communications. In some other applications, however, it is desirable to have an arithmetical rule or a sequence of processing steps in an algorithm which maps an observed realization y onto the compressed t. This is the case, for example, when the cardinality of Y and, therefore, the resulting lookup table implementing $t = f(y)$ becomes fairly large. As a result, it is meaningful to consider parametrized compression mappings $t = f_{\boldsymbol{\theta}}(y)$ with M parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_{M-1}]$ that preserve a desired large amount of mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$.

In this article, we develop parametrized mappings for communication receivers that only need few parameters and simple signal processing operations to preserve significant amounts of relevant information. The mappings investigated use exact or approximate nearest neighbor search algorithms [14,15]. Other approaches to designing parametrized systems exist in the literature. Some of the most popular use neural networks [16–18]. Our motivation to study the proposed nearest neighbor search-based systems instead is that they offer a very simple implementation with a small number of mathematical operations to determine the system output t. This is an important aspect for their practical use in communication receivers.

Finding optimum parameters $\boldsymbol{\theta}$, however, is cumbersome for the proposed parametrized mappings, especially if approximate nearest neighbor search algorithms are used. Therefore, we use genetic algorithms for the required optimization of the parameters $\boldsymbol{\theta}$. Genetic algorithms are very generic and powerful optimization algorithms that are inspired by the natural evolution of the species [19,20]. Their general idea is to create a population of candidate solutions to an optimization problem. Then, a so-called fitness of each individual in the population is evaluated with respect to the target function. The members of the population breed novel generations by combining their genetic information using simple crossover operators. In this process, the Darwinistic principle of promoting solutions with higher fitness is applied and also mutations happen. Fascinatingly, like this genetic algorithms can in fact find very good solutions to very complicated optimization tasks [19–22].

The above motivates us to apply genetic algorithms to optimize parametrized compression mappings that aim for maximum preservation of relevant information. Such mappings have numerous applications in learning and also in the baseband signal processing of communication receivers. This article investigates the receiver-sided channel output quantization in communication receivers based on nearest neighbor search algorithms, similar to the original conference version of this article [23]. As novel contributions, we introduce and optimize parametrized mappings that involve K-dimensional trees [24,25]. We propose and investigate the design of a novel demodulation scheme for data transmission using non-binary low-density parity-check codes with binary phase-shift keying (BPSK) modulation which is based on nearest neighbor search in the K-dimensional trees as an entirely new contribution of this article.

In summary, the contributions of this article are:

- We develop and investigate the idea of applying genetic algorithms to maximize mutual information in a parametrized information bottleneck setup for communication receivers.
- We design very powerful parametrized compression mappings that preserve large amounts of relevant information with very few parameters. These mappings are based on exact and approximate nearest neighbor search algorithms.
- We illustrate enormous flexibility and generality of the considered approach.

- We present results on channel output quantization and demodulation in communication receivers.

The article is structured as follows. The next section provides a brief overview of the required preliminaries. In Section 3, we propose different classes of parametrized mappings that can preserve significant amounts of relevant information. Moreover, we motivate and explain their genetic optimization. Section 4 then provides practical results on the proposed communication receiver design with maximum preservation of relevant information. Finally, Section 5 concludes the article.

2. Preliminaries

This section introduces fundamentals on the information bottleneck method and genetic algorithms. At the end of the section, two important distance metrics for vectors are briefly recalled that will be required in the remainder of the article.

2.1. The Information Bottleneck Method

The information bottleneck method is an information theoretical framework introduced by N. Tishby et al. in [1]. It originates from machine learning and considers three discrete random variables X, Y and T which form a Markov chain $\mathsf{X} \to \mathsf{Y} \to \mathsf{T}$. X is termed the relevant random variable. The idea is that Y is observed and shall be compressed to a more compact representation T. It is well-known from the famous rate-distortion theory that in this context a compression corresponds to minimizing the so-called compression information $\mathrm{I}(\mathsf{Y};\mathsf{T})$. However, it shall be guaranteed that also the mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ is maximized. As a result, one can conclude that X defines which features of Y are considered to be relevant and shall be preserved under compression. The compression rule that maps a realization $y \in \mathcal{Y}$ onto its compressed representation $t \in \mathcal{T}$ is typically considered as a conditional probability distribution $p(t|y)$. This allows us to cover probabilistic and also deterministic mappings of y onto t. In this article, however, we will restrict ourselves to deterministic mappings $p(t|y) \in \{0,1\}\ \forall (y,t)$ that, of course, fulfill the law of total probability. In this situation, t is a determinstic function of y, i.e., $t = f(y)$.

There exist many information bottleneck algorithms [26–30] that can construct the desired compression mapping $t = f(y)$ for a given cardinality of $\mathcal{T}$. A popular information bottleneck algorithm in communications is the KL-means algorithm from [26,27]. Due to the fact that Y is discrete, it is possible to store the mapping $t = f(y)$ in a lookup table with size $|\mathcal{Y}|$ by just storing each t for the respective y. The mapping $t = f(y)$ then clusters the event space of Y into several clusters $\mathcal{Y}_t$ which, mathematically, are the preimages of $t = f(y)$.

2.2. Genetic Algorithms

Genetic algorithms are very powerful and generic optimization algorithms that have various applications in many fields of engineering [19–22]. They aim to mimic the natural evolution of the species to solve multi-parameter optimization problems. Consider the problem of finding parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_{M-1}]$ that maximize a function $g : \mathbb{R}^M \longrightarrow \mathbb{R}_0^+$.

In order to find optimum parameters $\boldsymbol{\theta}$, a genetic algorithm works on a population $\mathcal{P} = \left\{\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(n_{\mathrm{pop}}-1)}\right\}$ of n_{pop} candidate solutions. Initially, this population is often drawn randomly. The real world parameter description $\boldsymbol{\theta}^{(l)}$ is typically termed the phenotype of an individual in the population. Each member $\boldsymbol{\theta}^{(l)}$, $l \in \{0, 1, \ldots, n_{\mathrm{pop}} - 1\}$ of the population implies a certain value of the target function $g\left(\boldsymbol{\theta}^{(l)}\right)$ which is readily termed the fitness of this individual.

In addition to the phenotype description of every individual, a genotype description can be introduced. The idea is to encode the numerical values of the parameters $\theta_m^{(l)}$ using so-called alleles into a long genetic string. In the simplest form, the alleles are just binary zeros or ones and the genotype of an individual is a long sequence of these numbers,

accordingly. For a given phenotype, one can determine the genotype by considering uniform discretization of the search spaces $[\theta_m^{\min}, \theta_m^{\max}]$ for the parameters θ_m into 2^{r_m} regions, respectively. Like this the values of the parameters $\theta_m^{(l)}$ can be interpreted as bit sequences of length r_m which encode the corresponding index of the region in binary form. A simple method to obtain the respective bit sequences is determining the region indices $z_m^{(l)}$ for all the appearing $\theta_m^{(l)}$ in $\boldsymbol{\theta}^{(l)}$ as

$$z_m^{(l)} = \left\lfloor \frac{\theta_m^{(l)} - \theta_m^{\min}}{\theta_m^{\max} - \theta_m^{\min}} (2^{r_m} - 1) \right\rfloor. \tag{1}$$

The $z_m^{(l)}$ are integers and can be converted into their binary representations easily. Then, one just concatenates all the obtained binary numbers to a long binary string to obtain the genotype. As an example, consider $\boldsymbol{\theta}^{(l)} = \left[\theta_0^{(l)}, \theta_1^{(l)}\right] = [1.326, -0.839]$, $\theta_0, \theta_1 \in [-2, 2]$, $r_0 = r_1 = 8$. One obtains $z_0^{(l)} = 212$, $z_1^{(l)} = 74$ and the corresponding genotype $[11010100\,01001010]$. Of course, the reverse genotype to phenotype conversion can be done similarly.

An instance of the population $\mathcal{P}$ exists in a generation of the genetic algorithm. In every generation, parent solutions are randomly selected from $\mathcal{P}$ and their genetic information is combined using simple genetic crossover operators on the genotypes to create children which form the population of the following generation. Such a crossover operation with $n_{\text{cross}} = 2$ crossover positions is illustrated in Figure 1.

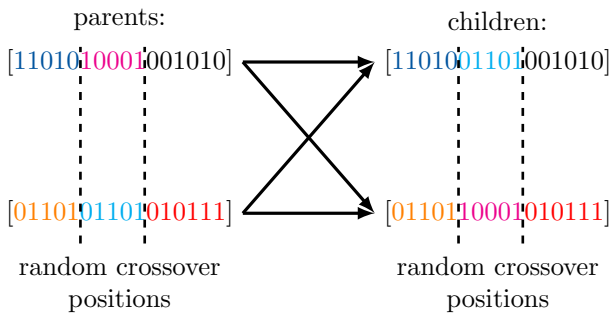


Figure 1. Illustration of a two point crossover in the processing of a genetic algorithm. The genotypes of the children are formed by combining the genotypes of both parents.

It is key that in the described processing, the individuals with higher fitness are more likely to become parents of the next generation than the weaker individuals with lower fitness. This is realized using simple inversion sampling to draw the parents. Moreover, the concept of elitism promotes the fittest individuals and guarantees them propagating their genetic material into the next generation. Finally, mutations of alleles in the genotypes of the children are performed with a certain mutation probability p_{mut} to assure some diversity.

Fascinatingly, when the processing is executed for several generations, genetic algorithms can find very good solutions to enormously complicated optimization tasks [19]. A particular strength of genetic algorithms is their generality. They need no other assumptions on the target function than that it allows to measure the fitness of an individual in the population. This motivates us to investigate the possibility of maximizing the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ under compression in information bottleneck settings with genetic algorithms.

2.3. Distance Metrics

In this section, we want to briefly recall two elementary distance metrics for vectors $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]$ and $\boldsymbol{\theta}_t = [\theta_{t,0}, \theta_{t,1}, \ldots, \theta_{t,N-1}]$ that will be used frequently in the

remainder of the article. A well-known distance measure is the Euclidean distance between $\mathbf{y}$ and $\boldsymbol{\theta}_t$, i.e.,

$$d_{\mathrm{E}}(\mathbf{y}, \boldsymbol{\theta}_t) = \sqrt{\sum_{n=0}^{N-1} (y_n - \theta_{t,n})^2}. \tag{2}$$

When it comes to implementation, the Euclidean distance has some disadvantages. In particular, taking the square under the root in Equation (2) requires costly multiplications in digital hardware. In addition, the square root is also costly on some signal processing platforms. As a result, in some applications a more favorable distance is the Manhattan distance [31] given by

$$d_{\mathrm{M}}(\mathbf{y}, \boldsymbol{\theta}_t) = \sum_{n=0}^{N-1} |y_n - \theta_{t,n}|. \tag{3}$$

This distance measure only requires sign inversions and additions which are fairly low-cost operations.

3. Design of Parametrized Compression Mappings That Maximize Relevant Information for Communication Receivers

The general system setup that we consider in this article is sketched in Figure 2.

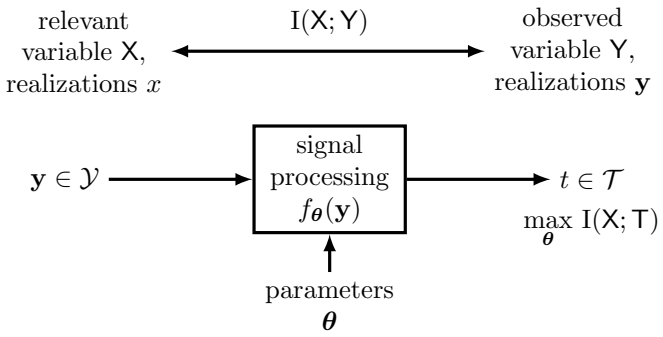


Figure 2. Optimizing parameters in a parametrized information bottleneck like setup. The parameters $\boldsymbol{\theta}$ shall be tuned to maximize $\mathrm{I}(\mathsf{X};\mathsf{T})$ for a given parametrized function $f_{\boldsymbol{\theta}}(\mathbf{y})$.

As shown there, we consider a generic receiver-sided signal processing scheme that inputs an observed random variable Y. The observed random variable Y is a random vector with realizations $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]$ because many signal processing components in communications process more than one scalar input variable. The system has M tunable parameters $\theta_m \in \mathbb{R}$, $m \in \{0, 1, \ldots, M-1\}$. The design idea for tuning the parameters θ_m is choosing them, such that the mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ is maximized. Like this, the system output T shares a desired huge amount of information with the relevant random variable X. We consider the system output $t \in \mathcal{T}$ to be from some finite set $\mathcal{T}$ with cardinality $|\mathcal{T}|$. Only this cardinality $|\mathcal{T}|$, the mapping rule of $\mathbf{y}$ onto t implied by $t = f_{\boldsymbol{\theta}}(\mathbf{y})$ and the joint probability distribution $p(x, \mathbf{y})$ determine $\mathrm{I}(\mathsf{X};\mathsf{T})$. In contrast, $\mathrm{I}(\mathsf{X};\mathsf{T})$ does not depend on the particular elements of $\mathcal{T}$. The reason is that $\mathrm{I}(\mathsf{X};\mathsf{T})$ is determined only by the probability distributions $p(x, t)$, $p(x)$ and $p(t)$, as this mutual information is given by

$$\mathrm{I}(\mathsf{X};\mathsf{T}) = \sum_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} p(x, t) \log \frac{p(x, t)}{p(x)\, p(t)}. \tag{4}$$

After all, the considered system design can be understood as an instance of the information bottleneck method described in Section 2.1. In contrast to the classical information bottleneck approach from [1], however, a parametrized system design for the mapping of realizations $\mathbf{y}$ onto t by $t = f_{\boldsymbol{\theta}}(\mathbf{y})$ is considered here. In addition, the choice of the output cardinality $|\mathcal{T}|$ allows us to adjust an inherent compression level achieved by the system, as this cardinality determines the number of bits required to represent the system output.

The system design approach introduced in Figure 2 has very intuitive applications in the communications context. Consider, for example, the data transmission scheme sketched in Figure 3. In this example, a phase shift keying (PSK) modulation scheme is used to transmit data over a complex additive white Gaussian noise (AWGN) channel. The transmission of the complex symbol $x = x^{\mathrm{re}} + jx^{\mathrm{im}}$ yields the channel observation $y = y^{\mathrm{re}} + jy^{\mathrm{im}}$ at the receiving end. Obviously, the system fed with the samples $\mathbf{y} = [y^{\mathrm{re}}, y^{\mathrm{im}}]$ in vector notation should preserve information on the transmitted modulation symbol x in this example. Considering outputs $t \in \mathcal{T}$ to be from a discrete set of integers $\mathcal{T} = \{0, 1, \ldots, 2^q - 1\}$, the system conducts a q bit quantization of the continuous received samples y with a minimum loss of relevant information on the transmitted modulation symbol x. In addition, each $t \in \mathcal{T}$ implies a conditional probability distribution $p(x|t)$. Therefore, the considered system can also be used straightforwardly for demodulation of the transmitted symbol x. The considered system will be investigated further in Section 4.

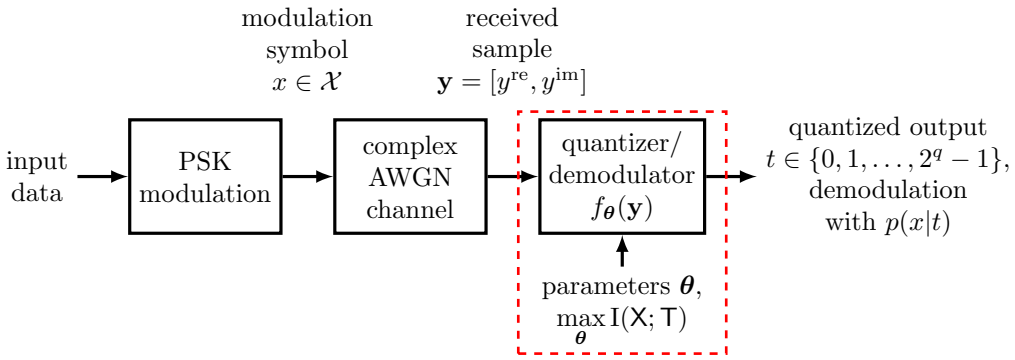


Figure 3. Exemplary application of a parametrized mapping $t = f_{\boldsymbol{\theta}}(\mathbf{y})$ for the quantization and demodulation of an AWGN channel output under PSK modulation. The system output $t \in \mathcal{T}$ shall be highly informative about the transmitted modulation symbol $x \in \mathcal{X}$.

3.1. Flexible Parametrized Mappings

Independent of the techniques used for the parameter optimization that we will describe later, the system design sketched above needs flexible classes of parametrized functions $f_{\boldsymbol{\theta}}(\mathbf{y})$ which allow to preserve significant amounts of relevant information $\mathrm{I}(\mathsf{X};\mathsf{T}) \leq \mathrm{I}(\mathsf{X};\mathsf{Y})$ for properly tuned parameters $\boldsymbol{\theta}$. We propose different ideas to implement the mapping of $\mathbf{y}$ onto $t \in \mathcal{T}$ in the considered systems which are described in the following. The considered mappings are all instances of nearest neighbor search algorithms [14] which need the definition of a distance metric like the ones from Section 2.3.

3.1.1. Clustering by Simple Exact Nearest Neighbor Search

The first class of parametrized mappings of $\mathbf{y}$ onto $t \in \mathcal{T} = \{0, 1, \ldots, |\mathcal{T}| - 1\}$ that we consider determines the outgoing t for an incoming $\mathbf{y}$ as

$$t = f_{\boldsymbol{\theta}}(\mathbf{y}) = \arg\min_{t' \in \mathcal{T}} \{ d(\mathbf{y}, \boldsymbol{\theta}_{t'}) \}, \tag{5}$$

where $d(\mathbf{y}, \boldsymbol{\theta}_t)$ is some properly defined, but at the same time, arbitrary distance measure between an incoming vector $\mathbf{y}$ and an optimized parameter vector $\boldsymbol{\theta}_t$ of the same dimension N as $\mathbf{y}$. In this article, we will consider the Euclidean distance $d_{\mathrm{E}}(\mathbf{y}, \boldsymbol{\theta}_t)$ and the Manhattan distance $d_{\mathrm{M}}(\mathbf{y}, \boldsymbol{\theta}_t)$ from Section 2.3, but we want to stress that the proposed method can deal with arbitrary distances. This mapping is characterized by $|\mathcal{T}|$ such parameter vectors $\boldsymbol{\theta}_t = [\theta_{t,0}, \theta_{t,1}, \ldots, \theta_{t,N-1}]$ which we compactly gather in a long vector $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|\mathcal{T}|-1}]$. As each vector $\boldsymbol{\theta}_t$ has length N, there are $N \cdot |\mathcal{T}|$ parameters θ_m in $\boldsymbol{\theta}$. Clearly, the approach is very much inspired by a vector quantizer which we aim to design with a genetic algorithm such that it maximizes the mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$.

In its simplest form, the considered mapping can be implemented by calculating all possible distances $d(\mathbf{y}, \boldsymbol{\theta}_t)\ \forall t \in \mathcal{T}$ and choosing the vector $\boldsymbol{\theta}_t$ with the smallest distance. This approach is sometimes also termed the naive nearest neighbor search [14], but for small values of $|\mathcal{T}|$ it offers a quite practical solution to identifying the nearest neighbor. The integer index t of the closest found vector then is the output of the system.

3.1.2. Exact and Approximate Nearest Neighbor Clustering Using *K*-Dimensional Trees

The simple nearest neighbor search approach from above has the apparent disadvantage that its complexity grows linearly with $|\mathcal{T}|$. As a result, the simple nearest neighbor search is limited to moderate cardinalities $|\mathcal{T}|$ in practice. Aiming for $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx \mathrm{I}(\mathsf{X};\mathsf{Y})$, however, often requires quite large cardinalities $|\mathcal{T}|$.

Fortunately, so-called *K*-dimensional tree data structures [24,25] can help to reduce the complexity of the simple nearest neighbor search algorithm for large $|\mathcal{T}|$. These data structures can often determine the nearest neighbor of $\mathbf{y}$ without explicitly calculating all possible distances $d(\mathbf{y}, \boldsymbol{\theta}_t)\ \forall t \in \mathcal{T}$. The resulting average query complexity of a *K*-dimensional tree scales logarithmically with the number $|\mathcal{T}|$ of vectors $\boldsymbol{\theta}_t$, hence typically resulting in a drastic reduction of required distance calculations in comparison to the simple nearest neighbor search. It shall be mentioned, however, that the worst case complexity of a search still is $\mathcal{O}(|\mathcal{T}|)$. K denotes the dimensionality of the data. In our case, K corresponds to the number N of inputs processed by the system from Figure 2.

Figure 4 shows an exemplary *K*-dimensional tree which can be used to conduct nearest neighbor search in an exemplary set of $|\mathcal{T}| = 7$ vectors $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_6$ with length $N = 3$ that we have chosen randomly for illustration purposes. We consider the task of finding the node $\boldsymbol{\theta}_t$ with the smallest Euclidean distance to an exemplary query vector $\mathbf{y}$ that is also provided in Figure 4. The true nearest neighbor of $\mathbf{y}$ is $\boldsymbol{\theta}_5$ with Euclidean distance $d_{\mathrm{E}}(\mathbf{y}, \boldsymbol{\theta}_t) \approx 1.52$.

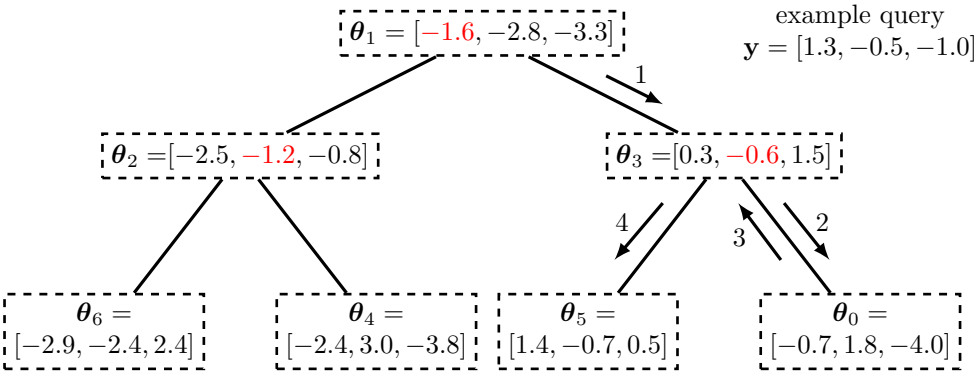


Figure 4. Exemplary *K*-dimensional tree with $|\mathcal{T}| = 7$ length $N = 3$ random vectors $\boldsymbol{\theta_0}, \boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_6}$. The vectors are arranged according to decision thresholds given by the axis coordinate highlighted in red in the subsequent levels of the tree. Arrows indicate the processing of querying the nearest neighbor of the vector $\mathbf{y}$.

The general principle of the search in the *K*-dimensional tree is that most of the explicit distance calculations are avoided and replaced by very simple threshold decisions along the axis of the data points. As it is highlighted in red in Figure 4 in the root node, the first axis considered corresponds to the first coordinate $\theta_{t,0}$. It is easy to see that all points in the left half of the tree underneath the root node fulfill $\theta_{t,0} \leq -1.6$ and all the points in the right half have $\theta_{t,0} > -1.6$.

As a result, for querying the first coordinate of $\mathbf{y}$ is compared with the first coordinate of the root node. Due to the fact that $y_0 > \theta_{1,0}$, the query goes to the right child $\boldsymbol{\theta}_3$ of the root node which is indicated using the arrow labeled 1. This processing is now repeated, but in the next reached node, the axis to split is the second, i.e., $\boldsymbol{\theta}_{t,1}$, as indicated in red again. The change of the considered axis in the subsequent levels of the tree is fundamental.

In each level, only the distances $d(\mathbf{y}, \boldsymbol{\theta}_t)$ to the visited nodes are calculated and only their minimum is stored and tracked. At node $\boldsymbol{\theta}_3$ we have the distance $d_E(\mathbf{y}, \boldsymbol{\theta}_3) \approx 2.69$ in our example.

Obviously, the example query follows the path labeled 2, as $y_1 = -0.5 > -0.6$ and the query reaches the leaf node $\boldsymbol{\theta}_0$. The distance to this node is $d_E(\mathbf{y}, \boldsymbol{\theta}_0) \approx 4.28$, so $\boldsymbol{\theta}_3$ stays closer.

The described processing does not guarantee finding the true nearest neighbor of $\mathbf{y}$ which is given by $\boldsymbol{\theta}_5$ so far. Fascinatingly, however, it is very easy to find out, whether the decision for a certain axis made so far went into the direction of the true nearest neighbor. In order to do that, backtracing the path taken is required. In each visited node now the distance of $\mathbf{y}$ along the split axis of the data in that node has to be considered only. If this distance is smaller, than the minimum distance obtained so far, it follows that following the other branch could be better.

In our example, when $\boldsymbol{\theta}_3$ is visited again, it is easy to find that the distance along axis $\theta_{t,1}$ in the node $\boldsymbol{\theta}_3$ is $\sqrt{(-0.6 + 0.5)^2} = 0.1 < d_E(\mathbf{y}, \boldsymbol{\theta}_3) \approx 2.69$, so the other branch labeled by arrow 4 is taken into account and the true nearest neighbor is found. The backtracing now can reach the root node and the processing is over.

Interestingly, the described processing can be implemented very elegantly using the programming method of recursion. The recursion for the backtracing, however also adds a significant amount of complexity. It is, therefore, mentionable that a very simple approximate nearest neighbor search algorithm with much lower complexity can be implemented in the K-dimensional tree by dismissing the backtracing. Like this, the search complexity can be fixed to $\mathcal{O}(\log_2(|\mathcal{T}|))$. The results presented in Section 4.2 show that in the considered application no practically relevant disadvantage of using approximate instead of exact nearest neighbor search exists.

Exactly as in Section 3.1.1, the (approximate) nearest neighbor seach algorithm outputs the index t of the closest found point which is the system output from Figure 2.

3.1.3. Approximate Nearest Neighbor Clustering Using Neighborhood Graphs

Another reduced complexity approximate algorithm for the problem of finding an approximate nearest neighbor of a query point $\mathbf{y}$ exists in the literature [14,15]. This algorithm is based on a proximity neighborhood graph of nodes that correspond to the candidate points $\boldsymbol{\theta}_t$. For simplicity, we consider neighborhood graphs, where all nodes have n_{neighbor} neighbors which correspond to the n_{neighbor} closest points under the considered distance.

The neighborhood graph-based approximate nearest neighbor search algorithm is depicted in Figure 5. When a new query point $\mathbf{y}$ shall be located, one enters the graph from any entry node and checks whether or not there are points in the neighborhood of the entry node which are closer to the query than the entry node itself. If this is the case, the closest found neighbor becomes the novel entry node and the processing starts over. In the shown example, the processing will stop after the neighbors of the entry node have been processed. Of course, this procedure can be executed for several initial entry nodes n_{entry} to improve the accuracy. It is also very easy to add a complexity constraint on the maximum number of allowed distance calculations by only allowing a certain path length $l_{\text{path}}^{\max}$ while jumping through the neighborhood graph. In order to achieve a desired minimum of distance calculations in the design, we define a set of n_{entry} entry nodes and first choose to determine the closest entry node to the query from that set. Then we only run the approximate nearest neighbor search described above from the closest found entry node. In the considered design, the worst case number of distance calculations to determine t for a given $\mathbf{y}$ is given by

$$n_{\text{dist}}^{\max} = n_{\text{entry}} + n_{\text{neighbor}} \cdot l_{\text{path}}^{\max}. \tag{6}$$

Please note that this number is independent of $|\mathcal{T}|$. As a result, one can allow for a very large number of candidate vectors $\boldsymbol{\theta}_t$ without a proportional increase in the number of required distance calculations.

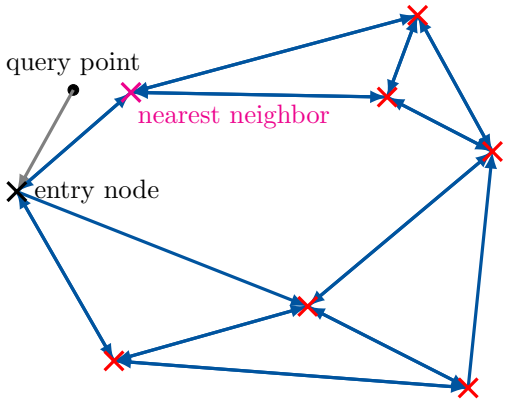


Figure 5. Visualization of the approximate nearest neighbor search in a neighborhood graph. The algorithm traverses through the neighborhood graph greedily. $\times$ markers correspond to the parameters $\boldsymbol{\theta}_t$ to be optimized, blue arrows indicate neighborhood relations.

Again the approximate nearest neighbor search algorithm then just outputs the integer index t of the approximate closest point $\boldsymbol{\theta}_t$ to $\mathbf{y}$. Clearly, the possible performance of the algorithm in terms of the preservation of $\mathrm{I}(\mathsf{X};\mathsf{T})$ and its complexity depend on the parameters n_{entry}, $l_{\text{path}}^{\max}$ and especially on n_{neighbor} which defines the sparsity of the neighborhood graph. Moreover, the particular set of entry nodes has an impact on the preserved relevant information. We will see in the practical results in Section 4, that quite sparse graphs with few entry nodes and small path length have the ability to preserve very significant amounts of $\mathrm{I}(\mathsf{X};\mathsf{T})$.

3.2. Genetic Algorithm Optimization

In Sections 3.1.1–3.1.3, different approaches to the problem of finding the (approximate) nearest neighbor $\boldsymbol{\theta}_t$ of the system input $\mathbf{y}$ from Figure 2 were proposed and described. Our intention is using the described approaches to implement the mapping $t = f_{\boldsymbol{\theta}}(\mathbf{y})$. In doing so, the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|\mathcal{T}|-1}]$ shall be tuned, such that the mutual information $\mathrm{I}(\mathsf{X};\mathsf{T})$ is maximized for a given $|\mathcal{T}|$. This naturally raises the question of how we can determine optimum parameters $\boldsymbol{\theta}$. We propose to perform the optimization of the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|\mathcal{T}|-1}]$ for the considered mappings and irrespective of the used distance function $d(\mathbf{y}, \boldsymbol{\theta}_t)$ with a genetic algorithm for various reasons explained in the following. Afterwards, we describe how to perform the parameter optimization with a genetic algorithm.

3.2.1. Why Genetic Algorithms?

Standard parameter optimization problems are often tackled by the application of gradient-based methods. A very famous example for this is the parameter optimization required to train neural networks in machine learning [16].

Considering Equation (5) again, however, reveals that using a gradient-based approach is cumbersome in our context. This equation involves a min operation which causes differentiability issues. A typical way to overcome them would be to use a smooth approximation [31], e.g., the softmin operation instead of the min during optimization, but we note that like this, we would in fact not optimize the deterministic mapping rule that we aim for in Equation (5), but only some non-deterministic approximation. Genetic algorithms, however, can directly optimize the deterministic mapping rule, as will be explained soon.

Moreover, depending on the distance metric used, more issues can arise. If the Manhattan distance from Equation (3) shall be used, the non-differentiability of the absolute magnitude $|.|$ involved adds to the min from Equation (5) which makes a gradient approach for the optimization of the parameters $\theta_{t,n}$ very cumbersome and would require mathematical approximations and workarounds [31]. Genetic algorithms, in contrast, can easily deal with this matter.

Finally and most importantly, in Sections 3.1.2 and 3.1.3, we have also studied approximate solutions to the nearest neighbor problem. These have drastically reduced complexity in terms of the number of distance calculations required. If such heuristic algorithms are applied, one can imagine the min operation from Equation (5) to be replaced with an approximate min. This operation is extremely hard, if not impossible, to describe analytically. Considering the greedy processing of the approximate nearest neighbor search algorithms from Sections 3.1.2 and 3.1.3, it is intuitively clear that for both, there is no mathematical expression to adequately describe the mapping of $\mathbf{y}$ onto t, even though it is deterministic. The mapping rules are rather given by subsequent processing steps in greedy algorithms.

As a result, the parameter optimization to maximize $\mathrm{I}(\mathsf{X};\mathsf{T})$ with standard gradient methods is not possible in these cases. Genetic algorithms, however, can be applied easily as discussed in the following.

3.2.2. Using Genetic Algorithms to Maximize the Preserved Relevant Information

We propose to perform the optimization of the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{|\mathcal{T}|-1}]$ for all considered mappings and irrespective of the actually used distance function $d(\mathbf{y}, \boldsymbol{\theta}_t)$ with a genetic algorithm. For that purpose, we initially draw a population of individuals $\mathcal{P} = \left\{\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(n_{\mathrm{pop}}-1)}\right\}$. As it is typically assumed in the information bottleneck setup, we assume that the joint probability distribution $p(x, \mathbf{y})$ is known.

For any population member $\boldsymbol{\theta}^{(l)}$ it is then straightforward to determine the joint probability distribution $p(x, t)$ for this particular individual as

$$p^{(l)}(x,t) = p^{(l)}(x|t)p^{(l)}(t) = \sum_{\substack{\mathbf{y}\in\mathcal{Y}:\\ t=f_{\boldsymbol{\theta}^{(l)}}(\mathbf{y})}} p(x,\mathbf{y}), \tag{7}$$

and

$$p^{(l)}(t) = \sum_{x\in\mathcal{X}} p^{(l)}(x,t). \tag{8}$$

These distributions directly allow us to calculate the respective preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ for this population member according to Equation (4), that is,

$$\mathrm{I}^{(l)}(\mathsf{X};\mathsf{T}) = \sum_{x\in\mathcal{X}}\sum_{t\in\mathcal{T}} p^{(l)}(x,t)\log\frac{p^{(l)}(x,t)}{p(x)p^{(l)}(t)}. \tag{9}$$

Note that $0 \leq \mathrm{I}^{(l)}(\mathsf{X};\mathsf{T}) \leq \mathrm{I}(\mathsf{X};\mathsf{Y})$ by definition. This allows us to use the mutual information $\mathrm{I}^{(l)}(\mathsf{X};\mathsf{T})$ directly as fitness $g\left(\boldsymbol{\theta}^{(l)}\right)$ of the population members $\boldsymbol{\theta}^{(l)}$ in the generations of the genetic algorithm. The rest of the processing then just follows the standard processing of genetic algorithms using selection, genetic crossovers and mutations over the generations as described, for example, in [19,20].

It is very important to note that all the involved equations can be evaluated totally irrespective of the actual operations performed in the signal processing block $f_{\boldsymbol{\theta}}(\mathbf{y})$ from Figure 2. The presented equations in fact work for all possible deterministic mappings of $\mathbf{y}$ onto $t \in \mathcal{T}$. The genetic algorithm just treats $f_{\boldsymbol{\theta}}(\mathbf{y})$ as a black box. Therefore, we can just use either the exact or the approximate nearest neighbor search approaches from Sections 3.1.1–3.1.3. We can also freely decide what distance measure $d(\mathbf{y}, \boldsymbol{\theta}_t)$ we want to use. As a result, the presented approach is very generic.

4. Results and Discussion

This section presents results on the application of the proposed parametrized compression mappings for quantizing the output of a communications channel and demodulation with the developed system design approach. It shall be mentioned that the applications studied serve to illustrate the method and the performance of the designed mappings. They allow us a very vivid illustration that reveals insights into the working of the proposed

method. However, numerous other applications can be investigated in future work, for example, in channel decoding, detection and other receiver-sided baseband signal processing tasks [5–13].

4.1. Quantization of the Channel Output with Minimum Loss of Relevant Information

In the following, we first consider KL-means quantization as proposed in [26]. KL-means quantization shall serve as a benchmark for the designed parametrized compression mappings. The most important figure of merit that we consider is the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ for a given output cardinality of the designed quantizers.

4.1.1. Information Bottleneck Quantizer Design with the KL-Means Algorithm

An intuitive application of the information bottleneck method in communications is the design of a channel output quantizer that maximizes the relevant information on the transmitted modulation symbols X. As already discussed and shown in Figure 3, in this context, Y corresponds to the received channel output. If Y is continuous, for example, for an AWGN channel, it has to be very finely discretized to $|\mathcal{Y}|$ uniformly spaced samples on some interval of interest. T is the quantized output variable of the quantizer. A q bit quantizer designed with the Information Bottleneck method maps realizations y onto quantization indices $t \in \mathcal{T} = \{0, 1, \ldots, 2^q - 1\}$, such that $|\mathcal{T}| = 2^q$ and $\mathrm{I}(\mathsf{X};\mathsf{T}) \to \max$. $\mathrm{I}(\mathsf{X};\mathsf{T})$ is independent of the elements in $\mathcal{T}$. We consider integer quantization indices that need q bits in the hardware.

As in [26], we consider complex AWGN channels and complex modulation alphabets, such that the continuous received sample at a certain time instance is

$$y = y^{\mathrm{re}} + jy^{\mathrm{im}} = (x^{\mathrm{re}} + n^{\mathrm{re}}) + j(x^{\mathrm{im}} + n^{\mathrm{im}}), \tag{10}$$

where $n^{\mathrm{re}} + jn^{\mathrm{im}}$ is a realization of a complex valued, circularly symmetric Gaussian process with variance σ_n^2 and mean 0 and $x = x^{\mathrm{re}} + jx^{\mathrm{im}}$ is a complex modulation symbol. For a simple notation, we assume that y is already finely discretized using a large number of $|\mathcal{Y}|$ uniformly spaced samples in a grid on the complex plane with $\sqrt{|\mathcal{Y}|}$ points for y^{re} and y^{im}, respectively. In addition, we define the vector representation $\mathbf{y} = [y^{\mathrm{re}}, y^{\mathrm{im}}]$ of the received sample.

In this situation, we want to quantize $\mathbf{y}$ to a number of $|\mathcal{T}| << |\mathcal{Y}|$ quantization regions. The considered quantizers are particularly useful for phase-shift keying (PSK) signals [26]. An example for 8-PSK under AWGN with noise variance $\sigma_n^2 = 0.5$ is provided in Figure 6. This figure shows the quantization regions obtained with the KL-means algorithm in the complex plane.

For this example, y^{re} and y^{im} were both finely discretized into $\sqrt{|\mathcal{Y}|} = 256$ uniformly spaced samples on the interval $[-1.5, +1.5]$ with properly paying attention to clipping effects. Like this, one obtains a grid with cardinality $|\mathcal{Y}| = 256^2 = 65,536$ in the complex plane. This grid was quantized to $|\mathcal{T}| = 16$ different quantization regions. This implies strong compression.

A typical application of the designed quantizer could be in a radio, where the analog-to-digital converter has a resolution of 8 bits for the real and the in-phase component of the received signal, but the signal shall be quantized to be processed further using just 4 bits per sample with minimum relevant information loss. In this example, $\mathrm{I}(\mathsf{X};\mathsf{Y}) \approx 1.49533$ bit and $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 1.38887$ bit. This indicates that despite the very coarse quantization a significant amount of relevant information on the transmitted modulation symbols (that is, around 92.8%) is preserved. Hence, it illustrates that the KL-means algorithm preserves relevant information.

Note that, due to the very complicated shape of the optimized quantization regions obtained using the KL-means algorithm from Figure 6, this quantizer cannot be characterized by simple thresholds for y^{re} and y^{im}. The KL-means algorithm instead delivers a table which holds the respective $t \in \mathcal{T}$ for all of the possible vectors $\mathbf{y} = [y^{\mathrm{re}}, y^{\mathrm{im}}]$, such

that, effectively one ends up with a lookup table of size $|\mathcal{Y}| = 65{,}536$ that characterizes the quantizer.

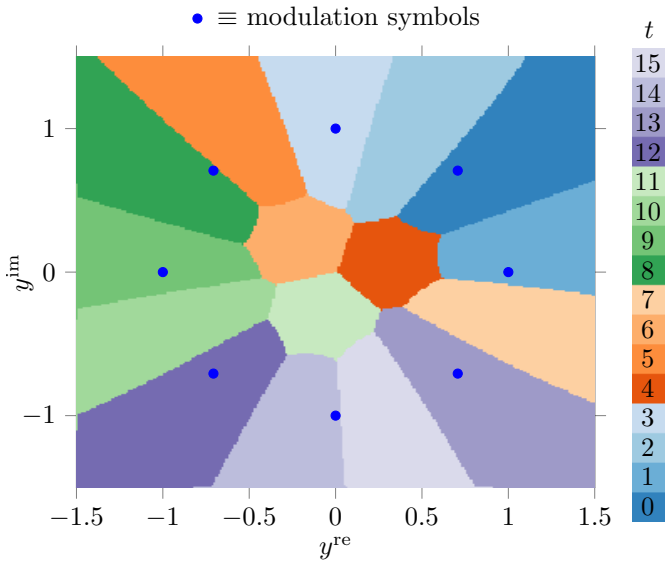
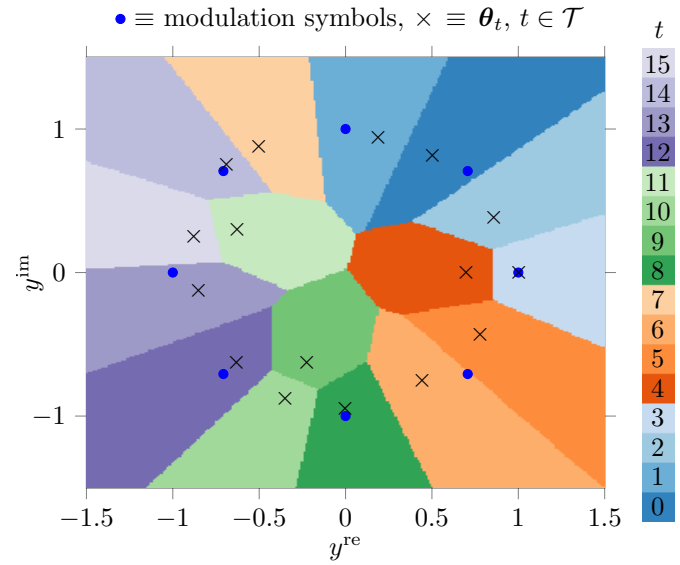


Figure 6. Quantization regions for the output of an AWGN channel with 8-PSK modulation in the complex plane for $\sigma_n^2 = 0.5$, $|\mathcal{T}| = 16$ constructed with the KL-means algorithm. $\mathrm{I}(\mathsf{X};\mathsf{Y}) \approx 1.49533$ bit, $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 1.38887$ bit.

4.1.2. Genetic Algorithm Quantizer Design Using Exact Nearest Neighbor Search

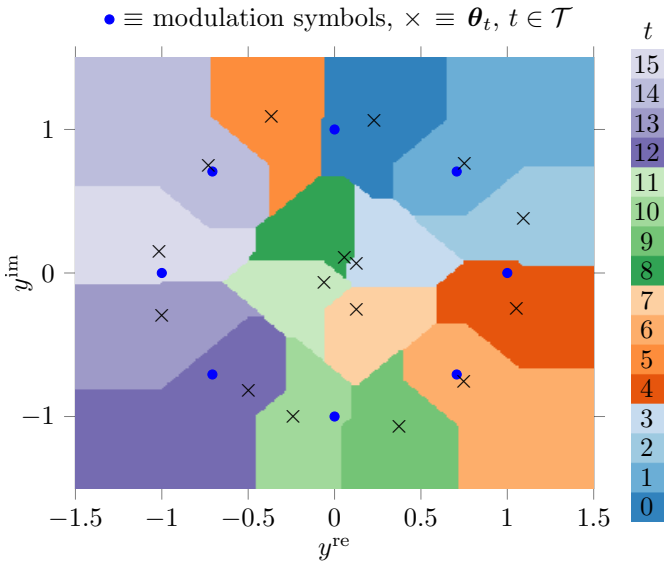
Figure 7 shows the quantization regions obtained for a simple exact nearest neighbor search approach described in Section 3.1.1.

Figure 7a uses the Euclidean distance and Figure 7b uses the Manhattan distance in Equation (5). The parameters $\boldsymbol{\theta}_t$ were tuned using the genetic algorithm based method from Section 3.2.2.



(**a**) Euclidean distance, $\mathrm{I}(\mathsf{X};\mathsf{Y}) \approx 1.49533$ bit, $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 1.38286$ bit.

Figure 7. *Cont.*



(**b**) Manhattan distance, $\mathrm{I}(\mathsf{X};\mathsf{Y}) \approx 1.49533$ bit, $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 1.37313$ bit.

Figure 7. Quantization regions for the output of an AWGN channel with 8-PSK modulation in the complex plane for $\sigma_n^2 = 0.5$, $|\mathcal{T}| = 16$ constructed with the genetic algorithm.

For the genetic algorithm optimization, we have used the configuration consolidated in Table 1. This configuration was determined experimentally and found to yield good results.

Table 1. Overview of parameters of the genetic algorithm.

Parameter	Description	Value
n_{pop}	population size	200
n_{cross}	number of crossovers for genetic combination	5
p_{mut}	mutation probability of each bit in the genotype	10^{-4}
r_m	number of bits to represent a parameter in the genotype	10
n_{elite}	number of elite population members (guaranteed to breed)	20
n_{gen}	number of evolved generations	2000

The phenotypes $\boldsymbol{\theta}^{(l)}$ in this scenario hold $2 \cdot |\mathcal{T}| = 2 \cdot 16 = 32$ real valued parameters that represent the real and the imaginary parts of 16 complex numbers. The optimized parameters $\boldsymbol{\theta}_t$ are denoted using $\times$ markers in Figure 7 in the complex plane. As it can be seen, the genetic algorithm automatically learns favorable positions $\boldsymbol{\theta}_t$ in terms of the maximum preservation of $\mathrm{I}(\mathsf{X};\mathsf{T})$ under the respective distance $d(\mathbf{y}, \boldsymbol{\theta}_t)$. The quantizers from Figure 7a,b both can be described with $32 << 65,536$ parameters, but have quantization regions with very complicated shapes that allow us to preserve large amounts of relevant information. The preserved relevant mutual information is $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 1.38286$ bit (i.e., 92.5% of $\mathrm{I}(\mathsf{X};\mathsf{Y})$) for the Euclidean distance and $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx 1.37313$ bit (i.e., 91.8% of $\mathrm{I}(\mathsf{X};\mathsf{Y})$) for the Manhattan distance.

The conference version of this article [23] also holds a quantitative comparison for different signal-to-noise ratios (SNRs) that we skip here for brevity.

4.1.3. Genetic Algorithm Quantizer Design with Approximate Nearest Neighbor Search

The numbers presented in the prior section illustrate that for $|\mathcal{T}| = 16$, there is still a mentionable gap between $\mathrm{I}(\mathsf{X};\mathsf{Y})$ and $\mathrm{I}(\mathsf{X};\mathsf{T})$ for all considered quantizers. In order to close that gap, one has to increase the output cardinality $|\mathcal{T}|$ of the quantizer to further decrease the quantization loss. This, however, proportionally increases the number of distance calculations for the method from Section 3.1.1. Here we use the approximate

nearest neighbor search algorithm from Section 3.1.3 to overcome that issue. Figure 8 compares the preserved relevant information $\mathrm{I}(\mathsf{X};\mathsf{T})$ of the KL-means algorithm and the proposed genetic algorithm optimized compression mappings for an output cardinality of $|\mathcal{T}| = 256$ as a function of the SNR $1/\sigma_n^2$ of the AWGN channel. Due to its simpler distance calculation, we only consider the Manhattan distance $d_{\mathrm{M}}(\mathbf{y}, \boldsymbol{\theta}_t)$ here. For this investigation, we were forced to decrease the cardinality of the grid that finely discretizes the complex plane to $\sqrt{|\mathcal{Y}|} = 128$ points for $y^{\mathrm{re}}, y^{\mathrm{im}} \in [-1.5, 1.5]$, respectively. The reason is that the time complexity of the KL-means algorithm from [26] is proportional to the product $|\mathcal{T}| \cdot |\mathcal{Y}|$ and with $|\mathcal{Y}| = 65,536$ as used in the previous investigation and $|\mathcal{T}| = 256$ used here, it was just not possible to create the KL-means quantizers in a reasonable time, even though we have used a highly-parallel implementation of that algorithm which parallelizes the algorithm on a graphics card [32]. Please note that using a coarser grid slightly degrades $\mathrm{I}(\mathsf{X};\mathsf{Y})$. This indicates that using the KL-means algorithm for very large cardinalities $|\mathcal{T}|$ is challenging. The method proposed here, however, easily allows using such a large $|\mathcal{T}|$.

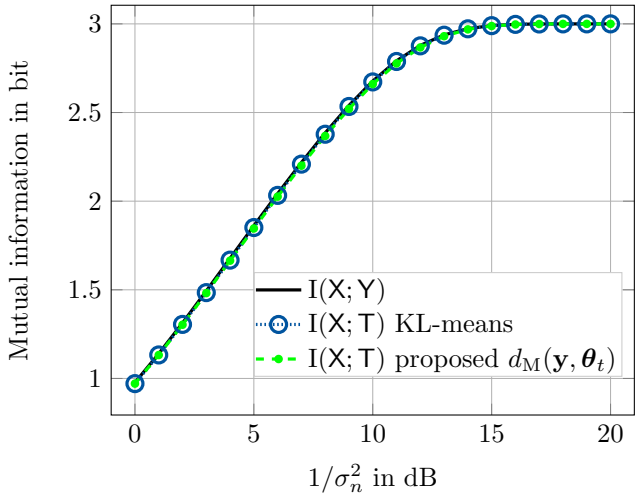


Figure 8. Comparison of KL-means and genetic algorithm optimized quantization using approximate nearest neighbor search from Section 3.1.3 for $|\mathcal{T}| = 256$ in terms of preserved $\mathrm{I}(\mathsf{X};\mathsf{T})$. $\mathrm{I}(\mathsf{X};\mathsf{Y})$ serves as an ultimate upper bound.

The approximate nearest neighbor search algorithm from Section 3.1.3 used the parameters $n_{\mathrm{neighbor}} = 6$ neighbors, $n_{\mathrm{entry}} = 6$ entry nodes and a maximum path length of $l_{\mathrm{path}}^{\mathrm{max}} = 5$. These parameters were found to offer a good tradeoff between sparsity of the neighborhood graph and performance. The worst case number of distance calculations to determine t for a given $\mathbf{y}$ in this setting is $n_{\mathrm{dist}}^{\mathrm{max}} = 6 + 6 \cdot 5 = 36$ according to Equation (6) which is significantly less than $|\mathcal{T}| = 256$. During our experiments we found out that it is even possible to reduce the maximum number of distance calculations further by decreasing $n_{\mathrm{entry}}, n_{\mathrm{neighbor}}$ or $l_{\mathrm{path}}^{\mathrm{max}}$ at the expense of very slight losses in $\mathrm{I}(\mathsf{X};\mathsf{T})$. Moreover, we have added the choice of the first entry node as a parameter to the genetic algorithm such that it is included in the optimization process. The rest of the entry nodes is chosen, such that all resulting entry nodes have possibly large distances among each other. The shown results indicate that the proposed compression mappings based on the approximate nearest neighbor search algorithm from Section 3.1.3 with parameters $\boldsymbol{\theta}$ optimized using genetic algorithms can deal with very huge cardinalities $|\mathcal{T}|$. Such large cardinalities $|\mathcal{T}|$ are required to minimize the remaining quantization loss, such that $\mathrm{I}(\mathsf{X};\mathsf{T}) \approx \mathrm{I}(\mathsf{X};\mathsf{Y})$, as it can clearly be seen in Figure 8. Moreover, the performance is virtually identical to the KL-means quantizers.

4.2. Genetic Algorithm Designed Demodulation Using K-Dimensional Trees

Next, we want to investigate an application of the proposed baseband signal processing approach illustrated in Figure 2 in a data transmission system that employs a non-binary low-density parity-check code over the Galois field 2^N $\left(\mathrm{GF}\left(2^N\right)\right)$ with $N > 1$ for forward error correction, but uses BPSK for signalling over an AWGN channel. A data transmission scheme similar to the one studied here was investigated for a lookup table-based information Bottleneck approach in [33]. For a deep introduction to non-binary low-density parity-check codes we kindly refer the reader to [34].

Pairing a non-binary channel code with BPSK offers a particularly interesting use case of the system illustrated in Figure 2. As it will be explained in the following, in the considered setup N received samples have to be processed for the demodulation of a $\mathrm{GF}\left(2^N\right)$ symbol at the receiving end. Hence, this problem perfectly matches the architecture of the considered system.

For completeness, it shall be mentioned that it is also common to pair non-binary channel codes with 2^N-ary modulation schemes, for example, 2^N-PSK. For such a coding and modulation scheme, the demodulation problem can be conducted using the systems investigated in Section 4.1. To do so, one has to use $p(x|t)$ after the quantization, as it has already been mentioned in Section 3.

The data transmission system that includes a non-binary channel code and BPSK modulation studied in this section is sketched in Figure 9.

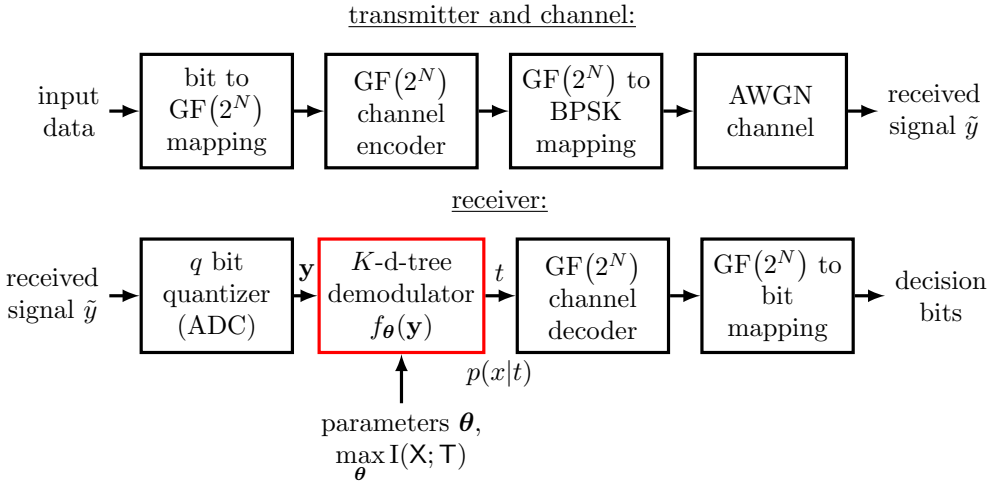


Figure 9. Illustration of the considered data transmission scheme which employs a non-binary low-density parity-check code over $\mathrm{GF}\left(2^N\right)$. At the receiver, a demodulator using the nearest neighbor search from Section 3.1.2 in K-dimensional trees is employed.

The upper part of the figure shows the considered transmitter and the channel. The lower part illustrates the receiver processing including the demodulator designed with a genetic algorithm.

In the transmitter, random data bits are mapped onto $\mathrm{GF}\left(2^N\right)$ symbols and then encoded using a non-binary low-density parity-check encoder with code rate R. In order to transmit the data over an AWGN channel using BPSK modulation, each output symbol of the encoder is mapped onto N consecutive BPSK symbols which are transmitted over the channel.

At the receiving end, first a coarse analog-to-digital conversion is performed using a q bit quantizer. N outputs $y_k \in \{0, 1, \ldots, 2^q - 1\}$, $k \in \{0, 1, \ldots, N-1\}$ from this quantizer correspond to the received samples for the transmitted BPSK symbols for a single $\mathrm{GF}\left(2^N\right)$ output symbol of the channel encoder in this setup. The scalar channel output quantizer is designed as explained in [11].

The next crucial task of the communication receiver is to provide symbol probabilities for $x \in \mathrm{GF}(2^N)$ to the channel decoder, such that it can perform soft channel decoding. The applied channel decoder performs the iterative sum-product algorithm, also known as belief-propagation decoding, to decode the non-binary low-density parity-check code with a maximum of $i_{\max}$ decoding iterations.

As a result, an output symbol $x \in \mathrm{GF}(2^N)$ of the channel encoder forms the relevant random variable X for our proposed demodulator. We use a genetic algorithm optimized demodulator which conducts either approximate or exact nearest neighbor search in a K-dimensional tree. The demodulator determines the index t of the nearest neighbor $\boldsymbol{\theta}_t$ as explained in Section 3.1.2 and delivers the symbol probability $p(x|t)$ to the channel decoder. Please note that the distribution $p(x|t)$ is obtained as a side product of the genetic algorithm optimization, as it is inherently determined to compute $\mathrm{I}(\mathsf{X};\mathsf{T})$ (cf. Equations (7) and (8)).

After decoding, the decoded information symbols are transformed into the decision bits by reversing the transmitter-sided bit-to-symbol mapping. For brevity, we provide the parameters that characterize the data transmission scheme used in this section further in Table 2.

Table 2. Overview of parameters of data transmission system.

Parameter	Description	Value
q	bit width for channel output quantization	4
$i_{\max}$	number of belief propagation decoding iterations	20
2^N	field order of non-binary low-density parity-check code	8
K_{cw}	uncoded information per codeword (in GF(8) symbols)	130
N_{cw}	codeword length of non-binary code (in GF(8) symbols)	260
R	code rate of non-binary low-density parity-check code	0.5
d_{c}	check node degree of applied regular code	4
d_{v}	variable node degree of applied regular code	2
$d(\mathbf{y}, \boldsymbol{\theta}_t)$	distance type used in K-dimenstional tree demodulator	Euclidean

We compare the bit error rate performances of the considered data transmission scheme including the proposed demodulation technique with state-of-the-art methods in a bit error rate simulation. Due to the fact that the optimum parameters $\boldsymbol{\theta}$ depend on the channel E_b/N_0, we have designed the proposed tree-based demodulators for different E_b/N_0 offline, stored them together with the corresponding distributions $p(x|t)$ and used them in the simulation. The space complexity of storing the K-dimensional tree is linear in $|\mathcal{T}|$, i.e., $\mathcal{O}(|\mathcal{T}|)$. Therefore, storing the obtained demodulators for the different E_b/N_0 is technically not challenging and only needs a few kilobytes of memory. As a result, the construction costs of the tree were one-time costs that only affected the genetic algorithm optimization, but not the demodulator implementation.

We have used the same optimization settings for the genetic algorithm as in Sections 4.1.2 and 4.1.3 (cf. Table 1). As a result, the design of the demodulators could be conducted offline, such that no on-the-fly generation was required. Conducting the genetic algorithm optimization only needed a few minutes on a standard computer.

As the toughest reference system, we consider a demodulator which has access to the continuous received samples $\tilde{\mathbf{y}} = [\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_{N-1}]$ in double floating point precision, i.e., no quantizer is involved. In this case, the a posteriori distribution $p(x|\tilde{\mathbf{y}})$ is determined for each symbol in the transmitted codeword and delivered to the channel decoder for decoding. Assuming equally likely symbols $x \in \mathrm{GF}(2^N)$, it is given by

$$p(x|\tilde{\mathbf{y}}) \propto \prod_{k=0}^{N-1} \exp\left(-\frac{|\tilde{y}_k - [\mathbf{x}]_k|^2}{2\,\sigma_n^2}\right) = \exp\left(-\frac{\sum_{k=0}^{N-1} |\tilde{y}_k - [\mathbf{x}]_k|^2}{2\,\sigma_n^2}\right) \quad \forall x \in \mathrm{GF}\left(2^N\right), \tag{11}$$

where $\mathbf{x}$ is a vector with the BPSK symbols transmitted over the channel for symbol $x \in \mathrm{GF}(2^N)$ and $[\mathbf{x}]_k$ denotes the k-th element of this vector. Please note that this demodulator also requires calculating 2^N squared Euclidean distances in the argument of the exponential (one for each Galois field symbol). In addition, it needs several divisions and the evaluation of the exponential function. Especially the latter is costly in digital hardware. Our aim is to approach the performance of this non-quantized reference system with the proposed demodulation techniques as closely as possible while circumventing most of the costly signal processing operations.

For reference, we also consider a very simple demodulator. This demodulator performs a hard decision on the BPSK symbols on the channel and maps this hard decision onto the corresponding $\mathrm{GF}(2^N)$ symbol directly. The decoder then is fed with a distribution that mimics $p(x|\tilde{\mathbf{y}})$ with probability 1 for the hard decision symbol and 0 for all others. This system, of course, cannot profit from soft information from the demodulation process. We use it to illustrate the gains of using soft demodulation in the data transmission system.

Figure 10 shows bit error rate performances of the considered data transmission system over E_b/N_0 for the different applied demodulation techniques. Of course, the non-quantized soft-decision reference system (◇-markers) has the best possible performance, as it suffers from no quantization loss at all. Comparing it to the system with hard demodulation (⊗-markers) shows that at an exemplary bit error rate of 10^{-4} a soft demodulation gain of more than 3 dB over E_b/N_0 exists for this data transmission system with a non-binary low-density parity-check code over $\mathrm{GF}(8)$.

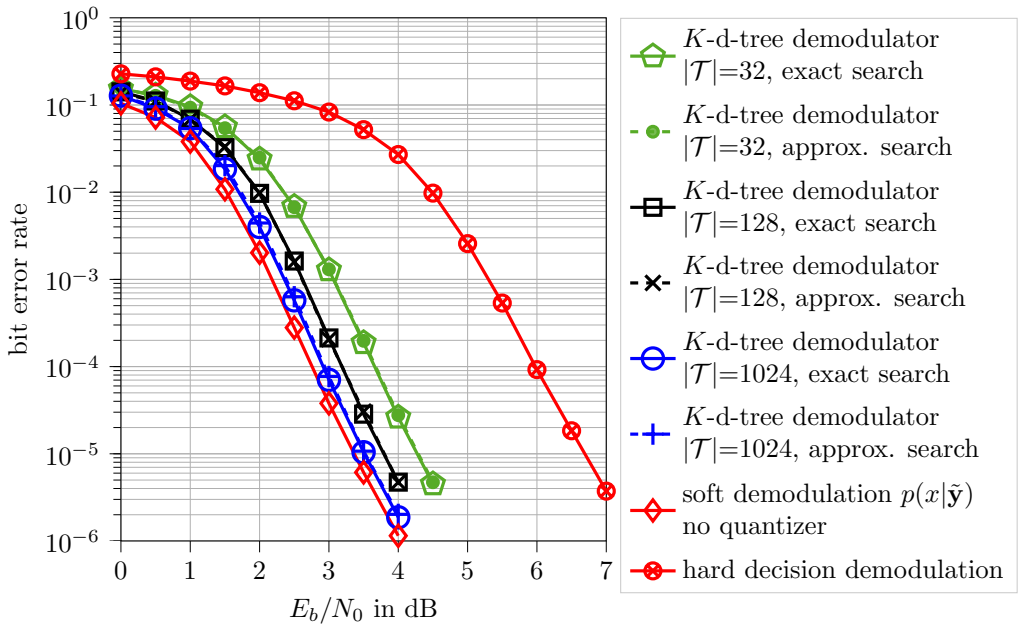


Figure 10. Bit error rate results for non-binary $\mathrm{GF}(8)$ low-density parity-check encoded data transmission over an AWGN channel with the parameters mentioned in Table 2. The proposed K-dimensional tree demodulators can achieve performance very close to the considered optimum reference system.

Interestingly, for the proposed K-dimensional tree demodulators with different cardinalities $|\mathcal{T}|$ the shown results indicate that with proposed genetic algorithm optimization of the vectors $\boldsymbol{\theta}_t$, one can learn very powerful demodulators which can approach the performance of the optimum considered reference scheme up to a very slight loss over E_b/N_0. For the demodulator with exact nearest neighbor search and $|\mathcal{T}| = 1024$ (○-markers), almost the full soft processing gain, i.e., more than 3 dB over E_b/N_0 can be realized, even though a coarse $q = 4$ bit channel output quantizer is in place. The remaining gap to the non-quantized reference demodulator is just 0.2 dB at a bit error rate of 10^{-4}. At the same time, most of the signal processing operations inside the demodulator degenerate to simple

threshold decisions along the axis of the vectors in the processing of the search in the K-dimensional tree described in Section 3.1.2. Even if the absolute number of vectors $\boldsymbol{\theta}_t$ is very large, only very few distance calculations need to be performed. This goes back to the logarithmic average search complexity in the K-dimensional tree described in Section 3.1.2. As a result, using large output cardinalities like $|\mathcal{T}| = 1024$ which are required to achieve performance so close to the optimum reference scheme is easily possible here. With the simple nearest neighbor search approach from Section 3.1.1, in contrast, using such a large cardinality $|\mathcal{T}|$ is practically infeasible.

Another very interesting observation from Figure 10 is that the bit error rates obtained with exact and approximate nearest-neighbor search for the same output cardinalities $|\mathcal{T}|$ superimpose (cf. ($\circ$, $+$)-markers ($\square$, $\times$)-markers, ($\pentagon$, $\bullet$)-markers). This finding is in fact very important because it highlights that the genetic algorithm automatically learns the different mapping rule applied inside the demodulator and tunes the parameters $\boldsymbol{\theta}$ accordingly.

As it has been explained in Section 3.1.3 switching to approximate nearest neighbor search yields a fixed search complexity $\mathcal{O}(\log_2(|\mathcal{T}|))$. In the considered case for $|\mathcal{T}| = 1024$ using approximate nearest neighbor search, typically $n_{\text{dist}} = 10$ distances have to be calculated to achieve performance enormously close to the soft demodulation reference system. Please note that for a number of $\boldsymbol{\theta}_t$ vectors which is a power of 2 there is the possibility that $\log_2(|\mathcal{T}|)$+1 distance calculations are needed. This, however only affects one of all possible paths in the tree and happens very rarely, especially if the tree is large. Therefore, the single additional distance calculation may be neglected. Anyway, we will mention it as the worst-case to be precise in the following. For the GF(8) code used here, according to Equation (11) the soft demodulator has to determine $n_{\text{dist}} = 8$ distances to obtain the probabilities $p(x|\tilde{\mathbf{y}})\ \forall x \in \text{GF}(8)$. However, it is important to note that the soft symbol demodulator reference system has a significantly higher complexity anyway.

Most importantly, the non-quantized soft demodulator uses 64 bit double floating-point values from the channel. The proposed demodulator circumvents the need of representing and processing the received samples from the channel with high precision, as it directly works on the $q = 4$ bit output integers $y_k \in \{0, 1, \ldots, 15\}$ from the quantizer. There is no need to represent the quantized received values using real numbers as representation values, as the genetic algorithm learns to directly process the q bit quantization indices. This alone yields a significant complexity reduction of the receiver because the resolution used for the analogue-to-digital conversion of the receiver can be reduced significantly. In addition, the soft demodulator reference system requires divisions by $2\,\sigma_n^2$ and $2^N = 2^3 = 8$ evaluations of the very costly exponential function in the considered case of a GF(8) code. Once the right hand side of Equation (11) has been evaluated for all $x \in \text{GF}(8)$, one needs a normalization step to obtain a valid probability distribution $p(x|\tilde{\mathbf{y}})$ which needs seven summation and eight division operations for the used GF(8) code. All these add on top of the required eight distance calculations.

The proposed system with $|\mathcal{T}| = 1024$ and approximate nearest neighbor search trades the required high precision of the analog-to-digital conversion and the numerous mentioned costly operations for typically two (worst case: three) additional distance calculations and very simple thresholding operations during the search in the K-dimensional tree. Despite this, it achieves almost identical performance as the optimum non-quantized reference scheme.

Finally, the curves for $|\mathcal{T}| = 128$ and $|\mathcal{T}| = 32$ for approximate nearest neighbor search in Figure 10 ($\times$-markers, $\bullet$-markers) reveal that even the demodulators with fewer distance calculations than the optimum soft demodulator can already realize very significant soft processing gains in comparison to the hard decision demodulator. At a bit error rate of 10^{-4}, the demodulator with $|\mathcal{T}| = 32$, i.e., typically just five (worst case: six) distance calculations achieves more than 2 dB soft processing gain over E_b/N_0 in comparison to the hard decision demodulator. The one for $|\mathcal{T}| = 128$ with typically seven (worst case: eight) distance calculations achieves 2.5 dB and has a remaining gap of approximately 0.5 dB to

the non-quantized reference scheme. This illustrates that the proposed method allows to flexibly tune the trade-off between complexity and performance.

5. Conclusions

In this article, genetic algorithms were successfully applied for the optimization of parametrized compression mappings that shall preserve a maximum possible amount of relevant information. These mappings were used to build subsystems of communication receivers, i.e., channel output quantizers and demodulators. To the best of our knowledge, our conference version of this article [23] described the first application of genetic algorithms for the maximization of mutual information in this context. It investigated potential applications of this principle for distance-based channel output quantization. The results were also included in this article. The resulting distance-based quantizers can compete with quantizers designed with the KL-means information bottleneck algorithm while requiring significantly fewer parameters for their description. The graph-based approximate nearest neighbor search algorithm used in this application allows for a tunable complexity and only needs a small number of distance calculations.

As a novelty, we have developed the idea of maximizing the relevant mutual information in communication receivers with genetic algorithms further and also presented entirely new results. We have introduced the idea to apply either approximate or exact nearest neighbor search in K-dimensional trees in the receiver-sided signal processing to build signal processing blocks that aim for maximum preservation of relevant information. That technique was exemplarily used to build a novel demodulation technique for a data transmission scheme using non-binary low-density parity-check codes. The resulting demodulators can achieve the performance of a non-quantized optimum reference scheme up to a small fraction of a decibel over E_b/N_0, even though all costly signal processing breaks down to a simple and very efficient search in a K-dimensional tree. We have also shown that using an approximate nearest neighbor search instead of an exact one does not cause significant performance degradation, but further reduces the complexity of the considered mappings based on K-dimensional trees.

The proposed method is very generic and can also be applied to other signal processing problems. A possible future application of the proposed method could be the reduced complexity decoding of non-binary low-density parity-check codes.

Author Contributions: Conceptualization, J.L., S.J.D., R.M.L.; methodology, J.L., S.J.D., R.M.L.; software, J.L., S.J.D., R.M.L.; validation, J.L., S.J.D., R.M.L.; formal analysis, J.L., S.J.D., R.M.L., M.A., M.S., P.J.; investigation, J.L., S.J.D., R.M.L., M.A., M.S., P.J.; resources, M.A.; writing—original draft preparation, J.L.; writing—review and editing, J.L., S.J.D., R.M.L., M.A., M.S., P.J.; visualization, J.L., S.J.D., R.M.L.; supervision, M.A., P.J.; project administration, M.A., M.S., P.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tishby, N.; Pereira, F.C.; Bialek, W. The information bottleneck method. In Proceedings of the 37th Allerton Conference on Communication and Computation, Monticello, IL, USA, 22–24 September 1999; pp. 368–377.
2. Slonim, N.; Somerville, R.; Tishby, N.; Lahav, O. Objective classification of galaxy spectra using the information bottleneck method. *Mon. Not. R. Astron. Soc.* **2001**, *323*, 270–284. [CrossRef]
3. Bardera, A.; Rigau, J.; Boada, I.; Feixas, M.; Sbert, M. Image segmentation using information bottleneck method. *IEEE Trans. Image Process.* **2009**, *18*, 1601–1612. [CrossRef] [PubMed]
4. Buddha, S.; So, K.; Carmena, J.; Gastpar, M. Function identification in neuron populations via information bottleneck. *Entropy* **2013**, *15*, 1587–1608. [CrossRef]
5. Romero, F.J.C.; Kurkoski, B.M. LDPC decoding mappings that maximize mutual information. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 2391–2401. [CrossRef]
6. Lewandowsky, J.; Bauch, G.; Tschauner, M.; Oppermann, P. Design and evaluation of information bottleneck LDPC decoders for digital signal processors. *IEICE Trans. Commun.* **2019**, *E102-B*, 1363–1370. [CrossRef]

7. Lewandowsky, J.; Stark, M.; Bauch, G. A discrete information bottleneck receiver with iterative decision feedback channel estimation. In Proceedings of the 2018 IEEE 10th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Hong Kong, China, 25–29 November 2018.
8. Hassanpour, S.; Monsees, T.; Wübben, D.; Dekorsy, A. Forward-aware information bottleneck-based vector quantization for noisy channels. *IEEE Trans. Commun.* **2020**, *68*, 7911–7926. [CrossRef]
9. Kern, D.; Kühn, V. On information bottleneck graphs to design compress and forward quantizers with side information for multi-carrier transmission. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017.
10. Steiner, S.; Kühn, V. Optimization of distributed quantizers using an alternating information bottleneck approach. In Proceedings of the 2019 23rd International ITG Workshop on Smart Antennas (WSA), Vienna, Austria, 24–26 April 2019; pp. 1–6.
11. Lewandowsky, J.; Bauch, G. Information-optimum LDPC decoders based on the information bottleneck method. *IEEE Access* **2018**, *6*, 4054–4071. [CrossRef]
12. Shah, S.A.A.; Stark, M.; Bauch, G. Design of quantized decoders for polar codes using the information bottleneck method. In Proceedings of the 12th International ITG Conference on Systems, Communications and Coding 2019 (SCC'2019), Rostock, Germany, 11–14 February 2019; pp. 1–6.
13. Shah, S.A.A.; Stark, M.; Bauch, G. Coarsely quantized decoding and construction of polar codes using the information bottleneck method. *Algorithms* **2019**, *12*, 192. [CrossRef]
14. Malkov, Y.A.; Yashunin, D.A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 824–836. [CrossRef] [PubMed]
15. Prokhorenkova, L.; Shekhovtsov, A. Graph-based nearest neighbor search: From practice to theory. In Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 12–18 July 2020; pp. 7803–7813.
16. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
17. Tishby, N.; Zaslavsky, N. Deep learning and the information bottleneck principle. In Proceedings of the 2015 IEEE Information Theory Workshop (ITW), Jerusalem, Israel, 26 April–1 May 2015; pp. 1–5.
18. Alemi, A.; Fischer, I.; Dillon, J.; Murphy, K. Deep variational information bottleneck. In Proceedings of the 5th International Conference on Learning Representations (ICLR) 2017, Toulon, France, 24–26 April 2017.
19. Coley, D.A. *An Introduction to Genetic Algorithms for Scientists and Engineers*; World Scientific Publishing Co., Inc.: Singapore, 1998.
20. Yu, X.; Gen, M. *Introduction to Evolutionary Algorithms*; Springer: London, UK, 2010.
21. Elkelesh, A.; Ebada, M.; Cammerer, S.; ten Brink, S. Decoder-tailored polar code design using the genetic algorithm. *IEEE Trans. Commun.* **2019**, *67*, 4521–4534. [CrossRef]
22. Wang, Y.; Li , L.; Chen, L. An advanced genetic algorithm for traveling salesman problem. In Proceedings of the 2009 Third International Conference on Genetic and Evolutionary Computing, Guilin, China, 14–17 October 2009; pp. 101–104.
23. Lewandowsky, J.; Dongare, S.J.; Adrat, M.; Schrammen, M.; Jax, P. Optimizing parametrized information bottleneck compression mappings with genetic algorithms. In Proceedings of the 2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS), Adelaide, Australia, 14–16 December 2020; pp. 1–8.
24. Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
25. Brown, R.A. Building a Balanced k-d Tree in $O(kn \log n)$ Time. *J. Comput. Graph. Tech. (JCGT)* **2015**, *4*, 50–68.
26. Zhang, J.A.; Kurkoski, B.M. Low-complexity quantization of discrete memoryless channels. In Proceedings of the 2016 International Symposium on Information Theory and Its Applications (ISITA), Monterey, CA, USA, 30 October–2 November 2016; pp. 448–452.
27. Kurkoski, B.M. On the relationship between the KL means algorithm and the information bottleneck method. In Proceedings of the 2017 11th International ITG Conference on Systems, Communications and Coding (SCC), Hamburg, Germany, 6–9 February 2017; pp. 1–6.
28. Hassanpour, S.; Wübben, D.; Dekorsy, A. A graph-based message passing approach for joint source-channel coding via information bottleneck principle. In Proceedings of the 2018 IEEE 10th International Symposium on Turbo Codes and Iterative Information Processing (ISTC), Hong Kong, China, 25–29 November 2018.
29. Slonim, N. The Information Bottleneck: Theory and Applications. Ph.D. Dissertation, Hebrew University of Jerusalem, Jerusalem, Israel, 2002.
30. Hassanpour, S.; Wübben, D.; Dekorsy, A.; Kurkoski, B. On the relation between the asymptotic performance of different algorithms for information bottleneck framework. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017.
31. Lange, M.; Zühlke, D.; Holz, O.; Villmann, T. Applications of lp-norms and their smooth approximations for gradient based learning vector quantization. In Proceedings of the 22nd European Symposium on Artificial Neural Networks, Bruges, Belgium, 23–25 April 2014; pp. 271–276.
32. Information Bottleneck Algorithms for Relevant-Information-Preserving Signal Processing in Python. Available online: https://collaborating.tuhh.de/cip3725/ib_base (accessed on 19 August 2020).

33. Stark, M.; Bauch, G.; Lewandowsky, J.; Saha, S. Decoding of non-binary LDPC codes using the information bottleneck method. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
34. Carrasco, R.A.; Johnston, M. *Non-Binary Error Control Coding for Wireless Communication and Data Storage*; Wiley: Chichester, UK, 2008.