# A Memetic Algorithm for Deinterleaving Pulse Trains

Jean Pinsolle[1,2], Olivier Goudet[0000−0001−7040−5052]2, Cyrille Enderli[1], and Jin-Kao Hao[0000−0001−8813−4377]2

[1] Thales DMS France SAS, 2 Av. Jean d'Alembert, 78190 Trappes
jean.pinsolle@fr.thalesgroup.com
[2] LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
{olivier.goudet, jin-kao.hao}@univ-angers.fr

**Abstract.** This paper deals with the problem of deinterleaving a sequence of signals received from different emitters at different time steps. It is assumed that this pulse sequence can be modeled by a collection of processes over disjoint finite sub-alphabets, which have been randomly interleaved by a switch process. A known method to solve this problem is to maximize the likelihood of the model which involves a partitioning problem of the whole alphabet. This work presents a new memetic algorithm using a dedicated likelihood-based crossover to efficiently explore the space of possible partitions. The algorithm is first evaluated on synthetic data generated with Markov processes, then its performance is assessed on electronic warfare datasets.

**Keywords:** Memetic algorithm, Markov process, partitioning problem, deinterleaving pulse trains, electronic warfare

## 1 Introduction

This paper presents an optimization algorithm for deinterleaving data streams that can be described by interleaved Markov processes. Even though such a method can be applied to many fields, the original motivation of this paper is related to radar warning receivers, which are passive sensors performing among other tasks the deinterleaving of pulse trains received from multiple emitters over a common channel.

In this context, pulses are emitted by different radars present in the environment and are intercepted by a single receiver. Each pulse is described by several characteristics called Pulse Description Words (PDW). Some of these features are called *primary* because they are measured in the early stages of the radio frequency signal reception chain, such as time of arrival (ToA), carrier frequency (CF), pulse duration (PD), signal amplitude or angle of arrival (AoA), while others are called *secondary*, such as the time interval between two consecutive pulses (Pulse Repetition Interval, PRI) because they characterize a pulse train. In the case of conventional radars with simple interpulse modulation, basic PRI clustering methods may be sufficient to solve the problem [10,11,13]. For more

complex data, multivariate methods such as [3], leveraging on different pulse features (CF, AoA, and PD), have been proposed.

However, modern radars can create much more complex patterns, which may result in a loss of performance of basic clustering methods, and produce errors in the deinterleaving process such as transmitter track proliferation or radar misses. To overcome these limitations, new methods based on inferring mixtures of Markov chains [1] have been proposed in the radar pulse train deinterleaving literature [5]. These methods aim to address complex scenarios by reducing the surplus of clusters found by classical methods. In these interleaved Markov process (IMP) methods, a clustering algorithm is first applied to group the different pulses into different clusters (or letters). Then, in a second step, a partition of these letters into different groups is performed in order to identify the different emitters which could have generated the observed sequence of symbols. This partition of the different symbols is typically done by maximizing a penalized likelihood score, which has been proven consistent under mild conditions on the switch and component processes [14].

In general, such methods for deinterleaving finite memory processes via penalized maximum likelihood raise a challenging combinatorial problem, because finding the optimal partition may require evaluating all the possible partitions of the observed symbols into different groups. Since this search space of all partitions grows exponentially with the number of symbols, an exhaustive search is in general not feasible in a reasonable amount of time. Therefore, heuristics based on greedy criteria have been proposed in [5,14] to provide an approximate solution to this problem in a limited amount of time. However, such greedy searches are prone to easily get stuck in local optima, especially when the search space becomes huge.

In this paper, we propose a new heuristic to solve this deinterleaving partition problem (DPP), by noticing that this problem can be seen as a particular grouping problem. The main contribution of this work is a new memetic algorithm for alphabet partitioning called MAAP, inspired by the memetic framework HEAD [12], which obtains state-of-the-art results for another grouping problem namely the graph coloring problem. The MAAP algorithm takes into account specific features related to penalized entropy estimates in order to speed up the search in the space of all partitions. In addition, it introduces a new likelihood-based crossover capable of sharing low entropy sub-alphabets that will be transmitted to the next generations.

The rest of the paper is organized as follows: Section 2 presents the formal background for the deinterleaving of Markov processes. Section 3 presents the settings of the optimization problem. Section 4 describes the proposed memetic algorithm. Section 5 reports the results on synthetic datasets generated with Markov processes, while Section 6 provides illustrative examples of radar interceptions in a realistic context. Section 7 discusses the contribution and presents some perspectives for future work.

## 2 Deinterleaving Markov Processes : Formal Background

In this section, we summarize the formal background of deinterleaving a set of finite memory processes on disjoint subsets and the penalized maximum likelihood method introduced in [14] to solve this problem.

### 2.1 Interleaved Markov Generative Process

Let $z^n = z_1, \ldots, z_n$ be an observed sequence of $n$ symbols ordered by their time of arrival. Each symbol is drawn from a finite set $\mathcal{A}$ (alphabet).

The underlying generative model of this sequence is assumed to be an *interleaved Markov process* $P = \mathcal{I}_{\Pi}(P_1, \ldots, P_m; P_w)$, where $m > 0$ is the number of different emitters, $P_i$ is an independent component random process for emitter $i$, generating symbols in the sub-alphabet $A_i \subset \mathcal{A}$, $P_w$ is a random switch process over the emitters, and $\Pi = \{A_1, \ldots, A_m\}$ is the partition of $\mathcal{A}$ into the sub-alphabets $A_i$, for $i = 1, \ldots, m$, which are assumed to be non-empty and disjoint.

It is further assumed that all Markov processes are time-homogeneous, independent, ergodic, and with finite memory. Let $k_i$ be the order of $P_i$, and $\mathbf{k} = (k_1, \ldots, k_m; k_w)$ denote the vector containing the orders of the corresponding processes $(P_1, ..., P_m; P_w)$. All states are assumed to be reachable and recurrent, and it is assumed that all symbols $a \in \mathcal{A}$ occur infinitely and their stationary marginal probabilities are positive. There is no assumption on the initial state of the processes.

According to this IMP $P$, at each time step, $t = 1, \ldots, n$ and given the prefix $z^{t-1} = z_1, \ldots, z_{t-1}$ of the sequence already generated at time $t-1$, a process $P_i$ is selected by the switch process $P_w$, then $P_i$ selects a letter $z_t$ from $A_i$ and adds it to the prefix sequence $z^{t-1}$ to form the sequence $z^t$.

Formally, this generative process can be written as

$$P(z_t|z^{t-1}) = P_w(i|\sigma_{\Pi}(z^{t-1}))P_i(z_t|z^{t-1}[A_i]), \tag{1}$$

where $\sigma_{\Pi}(z^{t-1})$ is the sequence of integers $i \in \{1, ..., m\}$ derived from the switch selection of the processes $P_i$ to generate the sequence $z^{t-1}$ and $z^{t-1}[A_i]$ is the sub-string of the sequence $z^{t-1}$ obtained by deleting all symbols not in $A_i$, note that we do not write a sum on $i$ since $P_i(z_t|z^{t-1}[A_i])$ is null for another alphabet than $A_i$.

By recursive application of Equation (1), the probability of occurrence of a sequence $z^t$ is then (with a slight abuse of notation)

$$P(z^t) = P_w(\sigma_{\Pi}(z^t)) \prod_{i=1}^{m} P_i(z^t[A_i]). \tag{2}$$

### 2.2 Penalized Maximum Likelihood Score

For a Markov process $P$ of order $k$ which generates a sequence $u^t$ of letters drawn from $\mathcal{A}$, the maximum likelihood (ML) of $u^t$ is given by

$$P_k^{ML}(u^t) = \prod_{a^{k+1} \in u^t} P(a^{k+1}|a^k) = \prod_{a^{k+1} \in u^t} \left(\frac{N_{u^t}(a^{k+1})}{N_{u^{t-1}}(a^k)}\right)^{N_{u^t}(a^{k+1})}, \tag{3}$$

with $a^k$ a pattern of $k$ letters in $u^t$ of length $k$, $P(a^{k+1}|a^k)$ the transition probability from $a^k$ to $a^{k+1}$ and $N_{u^t}(a^{k+1})$ the number of patterns $a^{k+1}$ in $u^t$. We denote $\hat{H}_k(u^t) = -\log P_k^{ML}(u^t)$ the corresponding ML entropy.

Knowing that the processes are independent and according to Equation (2), the global ML entropy $\hat{H}_{\Pi,\mathbf{k}}(z^n)$ of a sequence $z^n$ under an IMP model induced by the partition $\Pi$ and the vector order $\mathbf{k}$ is given by the addition of the ML entropy of each process:

$$\hat{H}_{\Pi,\mathbf{k}}(z^n) = \sum_{i=1}^{m} \hat{H}_{k_i}(z^n[A_i]) + \hat{H}_{k_w}(\sigma_\Pi(z^n)). \tag{4}$$

A global penalized entropy is further defined by adding a penalty term:

$$C_{(\Pi,\mathbf{k})}(z^n) = \hat{H}_{\Pi,\mathbf{k}}(z^n) + \beta\kappa \log n, \tag{5}$$

with $\beta$ a constant and $\kappa$ the number of free parameters in the model, which corresponds to the number of free parameters in the different processes:

$$\kappa = \sum_{i=1}^{m} |A_i|^{k_i}(|A_i| - 1) + m^{k_w}(m-1). \tag{6}$$

Finally, the IMP estimate, i.e., the deinterleaving scheme, is given by minimizing the previous cost function:

$$(\hat{\Pi}, \hat{\mathbf{k}}) = \underset{(\Pi,\mathbf{k})}{argmin}\ C_{(\Pi,\mathbf{k})}(z^n). \tag{7}$$

It is known that the scheme almost surely converges to an equivalent IMP representation as the sequence $n$ approaches infinity [14].

## 3 Problem Settings and Motivation for this Work

Given an observed sequence $z^n$ of length $n$, assumed to have been generated from an IMP $P$ defined in the previous section, with unknown number of emitters $m$ and unknown processes $P_w$ and $P_i$ for $i = 1, \ldots, m$, the problem that we address in this paper is to retrieve the partition $\Pi = \{A_1, ..., A_m\}$. This deinterleaving process problem is denoted as DPP in the following. Note that we do not address the problem of retrieving exactly the processes $P_w$ and $P_i$, which is a more difficult estimation problem.

We assume in this work that each process has a maximum order $k_{max}$. Therefore, we search for the couple of order vector $\hat{\mathbf{k}} \in \Omega_{k_{max}}$ and partition $\hat{\Pi} \in \Omega_{\Pi}$ minimizing the global ML entropy $C_{\hat{\Pi},\hat{\mathbf{k}}}$ given by Equation (5) with $\Omega_{k_{max}}$ the set of possible order vectors given by

$$\Omega_{k_{max}} = \{(k_1, \ldots, k_m; k_w), 1 \leq k_i \leq k_{max}, i = 1, \ldots, m, w\}, \tag{8}$$

and $\Omega_{\Pi}$ the search space of the alphabet partitions given by

$$\Omega_{\Pi} = \{\{A_1, \ldots, A_m\}, \mathcal{A} = \bigcup_{i=1}^{m} A_i, A_i \cap A_j = \emptyset, 1 \leq i, j \leq m, 1 \leq m \leq |\mathcal{A}|\}. \tag{9}$$

Therefore, solving the DPP is a double problem combining an estimation problem consisting in finding the optimal order vector $\mathbf{k}$ for each evaluated partition $\Pi$ and a combinatorial optimization problem on the space of all partitions $\Pi$ of the symbol alphabet $\mathcal{A}$.

### 3.1 Decomposable Score for Estimating Processes Optimal Order

Given a candidate partition $\hat{\Pi} = \cup_{i=1}^{m} \hat{A}_i$ and order vector $\hat{\mathbf{k}} = (\hat{k}_1, \ldots, \hat{k}_m; \hat{k}_w)$, we first observe that Equation (5) can be rewritten as

$$C_{\hat{\Pi},\hat{\mathbf{k}}}(z^n) = \sum_{i=1}^{m} \hat{H}_{\hat{k}_i}(z^n[\hat{A}_i]) + \hat{H}_{\hat{k}_w}(\sigma_{\hat{\Pi}}(z^n)) \tag{10}$$

$$+ \beta \log n \sum_{i=1}^{m} |\hat{A}_i|^{\hat{k}_i}(|\hat{A}_i| - 1) + \beta \log n \; m^{\hat{k}_w}(m-1) \tag{11}$$

$$= \sum_{i=1}^{m} C_{\hat{A}_i,\hat{k}_i}(z^n) + C_{\sigma_{\hat{\Pi}},\hat{k}_w}(z^n), \tag{12}$$

with $C_{\hat{A}_i,\hat{k}_i}(z^n) = \hat{H}_{\hat{k}_i}(z^n[\hat{A}_i]) + \beta \log n \; |\hat{A}_i|^{\hat{k}_i}(|\hat{A}_i| - 1)$, the penalized entropy of the estimated process $\hat{P}_i$ of order $\hat{k}_i$ generating sub-alphabet $\hat{A}_i$, and $C_{\sigma_{\hat{\Pi}},\hat{k}_w}(z^n) = \hat{H}_{\hat{k}_w}(\sigma_{\hat{\Pi}}(z^n)) + \beta \log n \; m^{\hat{k}_w}(m-1)$, the penalized entropy of the switch process related to partition $\hat{\Pi}$.

With this decomposition, we observe that given an estimated sub-alphabet $\hat{A}_i$, finding the optimal order $k_i$ of the process of each penalized entropy term $C_{\hat{A}_i,\hat{k}_i}(z^n)$ can be done *independently* of the global partition $\hat{\Pi}$ being evaluated. We denote as $C_{\hat{A}_i,k_i^*}(z^n)$, the optimal penalized entropy obtained for the observed sequence $z^n$ and the optimal order $1 \leq k_i^* \leq k_{max}$.

### 3.2 A Combinatorial Problem in the Space of Partitions

Since the space search grows exponentially with the number of letters, an exhaustive search is in general not feasible in a reasonable amount of time for the DPP. Greedy searches have recently been proposed by [5,14] to solve this combinatorial problem. However, such greedy local searches are prone to get stuck in local optima.

In this paper, we propose an improved heuristic to find the best partition $\hat{\Pi}$ in the huge search space $\Omega_\Pi$, by noticing that the studied problem is a particular grouping problem [15]. Given a set $S$ of elements, a grouping problem involves partitioning set $S$ into a number of disjoint groups $S_i$ optimizing a given objective function and possibly satisfying some given constraints.

In the DPP, the alphabet $\mathcal{A}$ corresponds to the set of elements $S$, and each sub-alphabet $A_i$ corresponds to a group $S_i$. The task is to find a partition $\hat{\Pi} \in \Omega_\Pi$ such that the global score $f(\hat{\Pi}) = C_{\hat{\Pi},\mathbf{k}^*}(z^n)$ is minimized (with $C_{\hat{\Pi},\mathbf{k}^*}(z^n)$ the penalized entropy evaluated for the observed sequence $z^n$, the partition $\hat{\Pi}$ and the optimal order vector $\mathbf{k}^* \in \Omega_{k_{max}}$ associated).

## 4 A Memetic Algorithm for Alphabet Partitioning

In this section, we present a new memetic algorithm for alphabet partitioning called MAAP to solve the DPP seen as a grouping problem. Following the main ideas of the HEAD algorithm [12], the proposed algorithm relies on a reduced population of only two individuals and uses a dedicated crossover operator. For local optimization, it employs a tabu search procedure.

### 4.1 General Framework

The general algorithm architecture of the proposed MAAP algorithm is described in Algorithm 1.

The population is initialized with two random partitions in $\Omega_\Pi$ (see Section 4.2). Then at each generation, the algorithm alternates two steps:

1. an intensification phase, where the two individuals of the population $\Pi_1$, $\Pi_2$ are improved by a tabu search procedure (called TabuAP) during $nb_{iter}$ iterations (see Section 4.3). This step produces two individuals $\Pi'_1$, $\Pi'_2$.
2. a diversification procedure, where two different children $\Pi_1$, $\Pi_2$ are generated from the two best partitions $\Pi'_1$, $\Pi'_2$ obtained from the tabuAP local search procedure. The crossover used to generate the two offspring partitions is a dedicated likelihood score-based crossover for the DPP (called GLPX) inspired by the well-known GPX crossover [6] for the graph coloring problem, it is explained in detail in section 4.4.

The following subsections describe each step of the MAAP algorithm.

**Algorithm 1** MAAP - Memetic algorithm for alphabet partitioning

**Input: Observed sequence $z^n$ of $n$ letters drawn from the alphabet $\mathcal{A}$.**
**Output: The best partition $\Pi_{best}$ found so far**
$\Pi_1,\ \Pi_2,\ \Pi_{best} \leftarrow random\_initialization$ ▷ **Section 4.2**
**while stop condition is not met do**
    $\Pi_1' \leftarrow TabuAP(\Pi_1, z^n)$ ▷ **Local tabu searches (see Section 4.3)**
    $\Pi_2' \leftarrow TabuAP(\Pi_2, z^n)$
    **if** $f(\Pi_1') < f(\Pi_{best})$ **then**
        $\Pi_{best} \leftarrow \Pi_1'$
    **end if**
    **if** $f(\Pi_2') < f(\Pi_{best})$ **then**
        $\Pi_{best} \leftarrow \Pi_2'$
    **end if**
    $\Pi_1 \leftarrow GLPX(\Pi_1', \Pi_2', z^n)$ ▷ **Crossover operators (see Section 4.4)**
    $\Pi_2 \leftarrow GLPX(\Pi_2', \Pi_1', z^n)$
**end while**
**return** $\Pi_{best}$

### 4.2 Initialisation

During the initialization procedure, the partitions $\Pi_1,\ \Pi_2,\ \Pi_{best} \in \Omega_\Pi$ are randomly built. In order to build a random partition, the letters in $\mathcal{A}$ are considered in alphabetical order. Then at each step, if the partition being constructed has already $m$ groups, the incoming letter $a$ has a probability equal to $\frac{1}{m+1}$ to be placed in each existing group of letters $A_i$ with $i = 1, \ldots, m$, and a probability $\frac{1}{m+1}$ to be placed in a new group $A_{m+1}$. This process is repeated until all letters are assigned to a sub-alphabet $A_i$. This procedure allows the creation of a partition randomly and uniformly in the search space $\Omega_\Pi$.

In order to ensure that the two individuals $\Pi_1$ and $\Pi_2$ are different in the population at the beginning, this initialization procedure is repeated until the set-theoretic partition distance between $\Pi_1$ and $\Pi_2$ is greater than 0. The set-theoretic partition distance between two partitions $\Pi_1 = \cup_{i=1}^{m} A_i$ and $\Pi_2 = \cup_{j=1}^{l} B_j$ is defined as the minimum number of one-move steps needed to transform $\Pi_1$ into $\Pi_2$ (up to a group permutation). This distance can be computed by solving a maximum weight bipartite matching problem if we consider each sub-alphabet $A_i$ of $\Pi_1$ and $B_j$ of $\Pi_2$ as nodes of a bipartite graph connected by edges $e_{ij} = \{A_i, B_j\}$. Each edge $e_{ij}$ has a weight $w_{ij}$ corresponding to the number of letters shared by the two corresponding sub-alphabets $A_i$ and $B_j$. This matching problem can be solved by the Hungarian algorithm [8] with a time complexity of $O(p^3)$ with $p = max(m, l)$. It produces a matching of maximum cardinality $0 \leq q \leq |\mathcal{A}|$ and the set-theoretic partition distance $D(\Pi_1, \Pi_2)$ is then defined as $|\mathcal{A}| - q$. Note that this distance will also be useful for the experiments. It is indeed a relevant scoring metric that can be used to evaluate the quality of an alphabet partition with respect to a known ground truth when working with simulated data (see Sections 5.2 and 5.3).

### 4.3 Tabu Search Procedure

The tabu search procedure for alphabet partitioning (called TabuAP) used during the intensification phase is inspired from the popular TabuCol algorithm for the graph coloring problem [7]. Some adjustments are made to adapt this tabu search to our partitioning problem.

**Neighborhood of a Partition** TabuAP explores the search space $\Omega_{\Pi}$ of all possible partitions that can be formed with the alphabet $\mathcal{A}$, by making transitions from the current solution to one neighboring solution.

A neighboring solution is generated by using the *one-move* operator. For a partition $\Pi = \cup_{i=1}^{m} A_i \in \Omega_{\Pi}$, the *one-move* operator displaces a letter $a \in A_i$ to a different sub-alphabet $A_j, j \neq i$. Let $\Pi \oplus < a, A_i, A_j >$ be the resulting neighboring partition. We then define the *one-move* neighborhood by

$$N(\Pi) = \{\Pi \oplus < a, A_i, A_j >: a \in A_i, 1 \leq i \leq m,\ 1 \leq j \leq m+1, m+1 \leq |\mathcal{A}|\}. \tag{13}$$

Notice that with this neighborhood, a letter $a \in A_i,\ i \neq m$ is allowed to be transferred to an existing group $A_j$ for $j = 1, \ldots, m$, with $j \neq i$, or to be placed in a new group $A_{m+1}$, which increases the total number of groups by one.

**Tabu search** The tabu search procedure iteratively replaces the current solution $\Pi$ by a neighboring solution $\Pi'$ taken from the one-move neighborhood $N(\Pi)$ until it reaches a maximum of $nb_{iter}$ iterations of tabu search or the cutoff time for the MAAP algorithm is reached.

At each iteration, TabuAP examines the neighborhood and selects the best admissible neighboring solution $\Pi'$ to replace $\Pi$. A neighboring solution $\Pi \oplus < a, A_i, A_j >$ built from $\Pi$ is said to be admissible if the associated one-move $< a, A_i, A_j >$ was not registered in a tabu list. Each time such one-move is performed, it is added to the tabu list and forbidden during the $t = r(3) + \alpha|\mathcal{A}|$ next iterations (tabu tenure) where $r$ is a random number uniformly drawn in $1, ..., 3$ and $\alpha$ is a hyperparameter of the algorithm set to the value of 0.6.

In order to compute the best admissible partition in the neighborhood, all the differences of global penalized entropy scores $\Delta_{a,j}$, associated with each admissible one-move $< a, A_i, A_j >$ are computed and the move corresponding to the lowest value of $\Delta_{a,j}$ is applied (because it is a minimization problem).

For a move $< a, A_i, A_j >$ applied to the current partition $\Pi$ and resulting in a new partition $\Pi' = \Pi \oplus < a, A_i, A_j >$, only the penalized entropy of the changing groups and the switch process need to be reevaluated. Indeed, according to Equation 10,

$$\Delta_{a,j} = C_{\Pi', \mathbf{k}'^*} - C_{\Pi, \mathbf{k}^*} \tag{14}$$

$$= C_{A'_i, \hat{k'}^*_i} - C_{A_i, \hat{k}^*_i} + C_{A'_j, \hat{k'}^*_j} - C_{A_j, \hat{k}^*_j} + C_{\sigma_{\Pi'}, \hat{k'}^*_w} - C_{\sigma_{\Pi}, \hat{k}^*_w}, \tag{15}$$

where $C_{A'_i,\hat{k'}^*_i}$ and $C_{A'_j,\hat{k'}^*_j}$ are respectively the optimal penalized entropy of the new sub-alphabet $A'_i = A_i \backslash a$ and $A'_j = A_j \cup a$ (after moving the letter $a$ from $A_i$ to $A_j$) with optimal order $\hat{k'}^*_i$ and $\hat{k'}^*_j$; $C_{\sigma_\Pi,\hat{k}^*_w}$ and $C_{\sigma_{\Pi'},\hat{k'}^*_w}$ are respectively the optimal entropy of the switch process of the partitions $\Pi$ and $\Pi'$.

Since $|\mathcal{A}|$ letters can be displaced to at most $|\mathcal{A}| - 1$ sub-alphabets, the size of this neighborhood is bounded by $O(|\mathcal{A}|^2)$. Evaluating a transition toward a neighbor with the one-move operator required to evaluate new penalized entropy, whose time complexity is in $O(n \times k_{max} \times |\mathcal{A}|^{k_{max}+1})$ ($n$ the length of the sequence). Therefore, the overall complexity of this tabu search procedure is $O(nb_{iter} \times n \times k_{max} \times |\mathcal{A}|^{k_{max}+3})$.

### 4.4 Greedy Likelihood-based Crossover Operator

The popular greedy partition crossover (GPX) [6] has proven to be very effective for graph coloring [9,12]. The two main principles of GPX are: 1) a solution is a partition of vertices (letters) into color classes (sub-alphabet) and not an assignment of colors to vertices, and 2) large color classes are transmitted to the offspring.

For the DDP, we introduce a new greedy likelihood-based partition crossover called GLPX. GLPX relies on the main principles of the GPX crossover with specific adaptations to our problem. Instead of only prioritizing large groups of letters, which does not make much sense for our problem, we prioritize groups as large as possible, but with as low entropy as possible, because our problem is to minimize the global entropy of the partition over the whole alphabet. A GLPX score for a group $A_i$ is introduced as

$$\begin{cases} \widehat{C}_{A_i}(z^n) = \frac{C_{A_i,k^*_i}(z^n)}{|A_i|-1} \text{ if } |A_i| > 1, \\ \widehat{C}_{A_i}(z^n) = +\infty \text{ if } |A_i| = 1. \end{cases} \tag{16}$$

Given two parent partitions $\Pi_1$ and $\Pi_2$, the GLPX procedure alternates two steps. First, it transmits to the child the sub-alphabet $\bar{A}$ with the lowest score $\widehat{C}_{\bar{A}}$. After having withdrawn the letters of this sub-alphabet in both parents and having recomputed all scores, it transmits to the child the sub-alphabet $\bar{B}$ with the lowest score $\widehat{C}_{\bar{B}}$ of the second parent. This procedure is repeated until all the letters of the alphabet $\mathcal{A}$ are assigned to the child. For a given parent, if two or more processes have the same lowest score, one of them is selected at random. Note that singletons have infinite scores, and then are randomly selected at the end of the process, when no more groups of at least two letters remain. The GLPX procedure is described in Algorithm 2.

This crossover is asymmetrical like the GPX crossover. As noticed in [12], starting the crossover with parent 1 or parent 2 can produce different offspring solutions. Therefore when used in the MAAP algorithm to generate two new offspring solutions $\Pi_1 = GLPX(\Pi'_1, \Pi'_2, z^n)$ and $\Pi_2 = GLPX(\Pi'_2, \Pi'_1, z^n)$, the two children $\Pi_1$ and $\Pi_2$ can be very different (in the sense of the set-theoretic partition distance defined in Section 4.2).

**Algorithm 2** GLPX crossover procedure

**Input: parents partitions** $\Pi_1 = \cup_{i=0}^{m} A_i$, $\Pi_2 = \cup_{i=0}^{q} B_i$ **and observed sequence** $z^n$.
**Output: Child partition** $\Pi_c$
$\Pi_c \leftarrow \emptyset$
**while** $\Pi_1$ **or** $\Pi_2$ **are not empty do**
  **for** $i = 1, 2$ **do**
    $\bar{A} \leftarrow \underset{A \in \Pi_i}{argmin}\ \widehat{C}_A(z^n)$
    $\Pi_c \leftarrow \Pi_c \cup \bar{A}$
    **for** $a \in \bar{A}$ **do**
      $\Pi_1 \leftarrow \Pi_1 \backslash a$
      $\Pi_2 \leftarrow \Pi_2 \backslash a$
    **end for**
  **end for**
**end while**
**return** $\Pi_c$

## 5 Experiments and Computational Results

This section is dedicated to the computational assessment of the proposed algorithm on both synthetic datasets and realistic datasets. Before showing the computational results, we first present the experimental condition.

### 5.1 Experimental Condition and Reference Algorithm

**Parameter Settings** For the TabuAP procedure, the tabu tenure parameter $\alpha$ is set to the value of 0.6 according to [6,12]. The maximal number of iterations for each TabuAP run is set to 50. The penalization parameter $\beta$ in Equation (5) is set to $\frac{1}{2}$, which is a common value used in the literature [2], allowing to retrieve the Bayesian Information Criterion (BIC). The maximum order $k_{max}$ for entropy estimation is set to the value of $k_{max} = 1$. Table 1 summarizes the parameter setting for the MAAP algorithm which can be considered the default and was used for all our experiments.

Table 1: Parameter setting in MAAP

| Parameter | Description | Value |
|---|---|---|
| $nb_{iter}$ | Number of iterations of the TabuAP local search | 50 |
| $\alpha$ | Tabu tenure parameter | 0.6 |
| $\beta$ | Penalization parameter entering in Equation 5 | $\frac{1}{2}$ |
| $k_{max}$ | Maximum order for entropy estimation | 1 |

**Reference Algorithm** Our MAAP algorithm is compared to the iterated greedy algorithm (iteratedGreedy) for alphabet deinterleaving pulse trains (see Algorithm 1 in [5]). For this iteratedGreedy algorithm, the radius of jump $r$ is set to the value of 2 and the neighborhood radius is set to 1 like in [5]. The maximum number of jumps $N$ is not limited. For this iteratedGreedy algorithm, the entropy evaluation is done with the same function used in the MAAP algorithm, and with the same parameters ($\beta = \frac{1}{2}$ and $k_{max} = 1$). The only difference between MAAP and iteratedGreedy is thus the search heuristic in the space of partition $\Omega_{\Pi}$. Both MAAP and iteratedGreedy are coded in Python with the Numpy library and are launched on a computer equipped with Intel Xeon ES 2630, 2.66 GHz CPU.

**Evaluation Metric and Stopping condition** To assess the quality of the best partition $\Pi_{best}$ found by an algorithm, we compute the set-theoretic partition distance between $\Pi_{best}$ and the ground truth partition $\Pi_{truth}$. The stopping condition for each experiment (on synthetic data and electronic warfare data) is indicated in the corresponding section.

### 5.2 Experiments on Synthetic Datasets

This section is dedicated to a first computational assessment of the proposed memetic algorithm for the DPP. The data are simulated with an interleaved Markov process $P = \mathcal{I}_{\Pi}(P_1, ..., P_m; P_w)$ over disjoint sub-alphabets, in the *ideal* framework presented in Section 2: independent time-homogeneous, ergodic and finite memory component processes $P_w$ and $P_i$ for $i = 1, \ldots, m$.

**Synthetic Dataset generation** The datasets are based on synthetic sequences of size $n$ with different numbers of letters $|\mathcal{A}|$ and maximal order equal to 1 to limit the computation time required for entropy estimation.

The following parameters are randomly set up to generate a sequence $z^n$ with an IMP $P = \mathcal{I}_{\Pi}(P_1, ..., P_m; P_w)$ according to Equation (1):

- from an alphabet $\mathcal{A}$ of size $|\mathcal{A}|$, a *ground truth* partition $\Pi_{truth} = \cup_{i=1}^{m} A_i$ is generated with the random initialization procedure as described in Section 4.2. $m$ is the number of emitters (groups) associated with this partition;
- for each emitter $i$ ($i = 1, \ldots, m$), a probabilistic transition matrix $Q_i$ associated with the stochastic process $P_i$ of size $|A_i| \times |A_i|$ is randomly drawn;
- for the switch process, a probabilistic transition matrix $Q_w$ of size $m \times m$ is drawn;
- initial state (letter) of each process $P_i$ is randomly drawn in its correspondent sub alphabet $A_i$;
- the first emitter is randomly drawn in the set of $m$ emitters;

We consider 4 different configurations $(|\mathcal{A}|, n)$ with $|\mathcal{A}| = \{20, 50\}$ and $n = \{10000, 50000\}$. For each configuration, 10 different datasets $(z^n, \Pi_{truth})$ are generated. So a total of 40 datasets are obtained. These datasets will be made publicly available.

| # | Config | | iteratedGreedy | | | MAAP | | | Config | | iteratedGreedy | | | MAAP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|\mathcal{A}\|$ | $n$ | $\bar{D}$ | $\bar{C}$ | time | $\bar{D}$ | $\bar{C}$ | time | $\|\mathcal{A}\|$ | $n$ | $\bar{D}$ | $\bar{C}$ | time | $\bar{D}$ | $\bar{C}$ | time (s) |
| 1 | 20 | 10000 | 0 | 24612 | 110 | 0 | 24612 | 101 | 50 | 10000 | 2.4 | 34606 | 2253 | **1.4** | 34593 | 1771 |
| 2 | 20 | 10000 | **0** | **26155** | 208 | 0.06 | 26161 | 358 | 50 | 10000 | 7.8 | 34063 | 1631 | **6.57** | **34051** | 2240 |
| 3 | 20 | 10000 | 0 | 24381 | 92 | 0 | 24381 | 121 | 50 | 10000 | 2.1 | 34968 | 2909 | **1.26** | **34958** | 2173 |
| 4 | 20 | 10000 | 0 | 26240 | 96 | 0 | 26240 | 99 | 50 | 10000 | 4.23 | 33398 | 1132 | **3.57** | **33391** | 2055 |
| 5 | 20 | 10000 | 0 | 24386 | 77 | 0 | 24386 | 80 | 50 | 10000 | 0.43 | 34626 | 1834 | 0 | **34621** | 1901 |
| 6 | 20 | 10000 | 0 | 26843 | 66 | 0 | 26843 | 58 | 50 | 10000 | 11.46 | 33881 | 5339 | **10** | **33859** | 4426 |
| 7 | 20 | 10000 | 0 | 24100 | 449 | 0 | 24100 | 103 | 50 | 10000 | 1.56 | 37051 | 3041 | **0.13** | **37027** | 3119 |
| 8 | 20 | 10000 | 0 | 26763 | 70 | 0 | 26763 | 69 | 50 | 10000 | **5.56** | 34698 | 1691 | 6.1 | **34682** | 3544 |
| 9 | 20 | 10000 | 0 | 24821 | 82 | 0 | 24821 | 98 | 50 | 10000 | 2.33 | 33758 | 3218 | **1.63** | **33741** | 3605 |
| 10 | 20 | 10000 | 0 | 26588 | 82 | 0 | 26588 | 75 | 50 | 10000 | 5.87 | 36155 | 909 0 | **4** | **36134** | 3250 |
| 1 | 20 | 50000 | 0 | 128191 | 324 | 0 | 128191 | 324 | 50 | 50000 | 0 | 175904 | 5216 | 0 | 175904 | 4690 |
| 2 | 20 | 50000 | 0.73 | 119944 | 298 | **0** | **119533** | 383 | 50 | 50000 | 0 | 163835 | 8388 | 0 | 163835 | 8350 |
| 3 | 20 | 50000 | 1.3 | 124721 | 630 | **0** | **124461** | 682 | 50 | 50000 | 0 | 166655 | 7287 | 0 | 166655 | 7184 |
| 4 | 20 | 50000 | 0.87 | 127508 | 507 | **0** | **127165** | 413 | 50 | 50000 | 0 | 177233 | 5979 | 0 | 177233 | 5669 |
| 5 | 20 | 50000 | 1.86 | 132787 | 388 | **0** | **132272** | 449 | 50 | 50000 | 0 | 167299 | 7288 | 0 | 167299 | 6534 |
| 6 | 20 | 50000 | 0.5 | 124718 | 425 | **0** | **124559** | 382 | 50 | 50000 | 0 | 180056 | 4943 | 0 | 180056 | 5006 |
| 7 | 20 | 50000 | 0.37 | 128701 | 360 | 0 | 128701 | 366 | 50 | 50000 | 0 | 166328 | 7619 | 0 | 166328 | 7238 |
| 8 | 20 | 50000 | 0.4 | 120087 | 583 | 0.4 | 128516 | 493 | 50 | 50000 | 0 | 166755 | 7514 | 0 | 166755 | 7450 |
| 9 | 20 | 50000 | 0.87 | 127186 | 379 | **0** | **126900** | 417 | 50 | 50000 | 0 | 173295 | 5828 | 0 | 173295 | 6045 |
| 10 | 20 | 50000 | 0 | 127150 | 552 | 0 | 127150 | 396 | 50 | 50000 | 0 | 173497 | 5980 | 0 | 173497 | 5952 |

Table 2: Comparison of MAAP and iteratedGreedy on synthetic datasets generated with interleaved Markov processes. Dominating results (lower scores) are indicated in boldface. Significantly better values are underlined (t-test with p-value of 0.05).

**Results on Synthetic Data** For each dataset, given the stochastic nature of both algorithms, 30 independent runs are launched. The time limit in seconds for each run is $T_{limit} = 200 * |\mathcal{A}|$ when $n = 10000$ and $T_{limit} = 500 * |\mathcal{A}|$ when $n = 50000$. Once the algorithm reaches this time limit, it returns the best partition $\Pi_{best}$ found so far with its associated minimum penalized entropy score $C_{\Pi_{best}}(z^n)$.

Table 2 displays the results obtained by the algorithms MAAP and iteratedGreedy on the 40 different datasets generated with 4 different configurations. Columns $\bar{D}$ indicate the average distance relative to the ground truth partition. Columns $\bar{C}$ show the average lowest penalized global entropy obtained by an algorithm and columns *time* correspond to the average time in seconds required by the algorithm to reach its best result. Values in bold mean the algorithm has a better score than the other one. Underlined values mean that the average score obtained for a given algorithm is significantly better than the average score of the other algorithm according to a t-test with p-value of 0.05.

Table 2 shows that both algorithms work efficiently since the distance to the truth partition is often close to zero which validates the relevance of the likelihood-based method used in this context. The comparison between the two algorithms reveals that MAAP obtains significantly better results for several configurations, due to more effective exploration of the search space of all possible partitions.

### 5.3 Experiments on Electronic Warfare Datasets

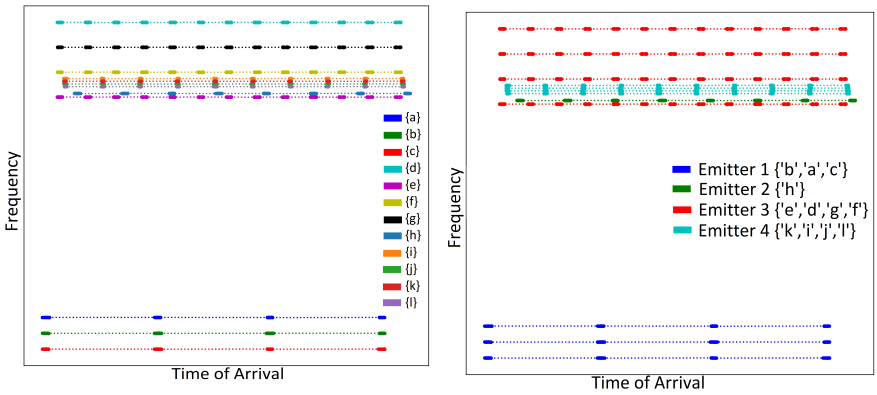
In this section we present results on datasets coming from an Electronic Warfare data generator which simulates realistic situations with mobile radar warning receivers. One configuration corresponds to a random draw in a list of known radars and a draw in their relative phasing. We cannot share the content of the generator. For each simulation, a dataset $\mathcal{D}$ consisting in a sequence of pulses with their corresponding frequency (CF) and time of arrival (ToA) is generated. The *ground truth* $\Pi_{truth}$ (i.e the association of each pulse to each emitter) is known. The objective is then to retrieve $\Pi_{truth}$ from the data.

**Preprocessing of the data** A preprocessing step is first performed to obtain the alphabet $\mathcal{A}$ from the dataset $\mathcal{D}$. It consists of clustering pulses with the DBSCAN algorithm [4] based on their frequency. Then, each obtained cluster is associated with a letter in $\mathcal{A}$. The sequence $z^n$ is then obtained by ordering these letters by increasing order of time of arrival (ToA). Since we only use the frequency, the $\epsilon$-neighborhood parameter of DBSCAN corresponds to our precision parameter and is a fixed number of the order of the MHz not specified here.

**Illustrated Example** Figure 1a shows an example of a pulse train measured with the frequency and the time of arrival of the different signals. The scales are hidden on purpose. Pulses regrouped in the same cluster after the first preprocessing phase have the same color and are associated with the same letter (from $a$ to $l$). Figure 1b corresponds to the known ground truth for this scenario (4 emitters):
{{a,b,c},{h},{d,e,f,g},{i,j,k,l}}. Pulses generated by the same emitter have the same color.

We ran the MAAP algorithm on this dataset with default parameters (see Table 1) and Figure 2 shows the evolution over time of the distance to the ground truth (blue) and the best penalized entropy (red) reached during the search (average, minimum and maximum over 10 runs). The two curves, distance and entropy, have similar variations, meaning that in this case, minimizing the penalized global entropy allows to get closer to the target partition. We observe that in some experiments, MAAP reaches the best target partition (distance of 0) within a few seconds, while in others, it never reaches it in the allotted time with a distance of 4. This highlights that finding a good partition in a limited amount of time is not always easy for these realistic datasets and may depend a lot on the random initiating solution from which the search starts.

**Results on Electronic Warfare Data** 10 different scenarios with 5 emitters are generated by the Eletronic Warfare data simulator. The number of observed pulses varies from 10000 to 100000 for these scenarios (the scenarios were cut if the number of points exceeded 100000 and couldn't contain less than 10000 points). We launched the MAAP and iteratedGreedy algorithm [5] with the




(a) Alphabet, 12 letters (b) Ground truth, 4 different emitters

Fig. 1: Illustrated example of radar pulses deinterleaving

same configuration and parameters as presented in Section 5.2. Each algorithm is launched 30 times (independent runs) on each dataset with a time limit of three hours.

Table 3 reports the result of these experiments, with the measures of the best distance ($D^*$) and the average distance ($\bar{D}$) to the known ground truth, the average best penalized global score ($\bar{C}$) obtained over the different runs and the time in seconds required to obtain the best scores.

We first observe in this table that for some scenarios, both algorithms are able to recover or come close to the target partition, but for others, such as scenario 9, they remain far from it. This is because the IMP representation used in this work is not always completely valid for some of these scenarios, as some assumptions are violated. In particular for some scenarios, a certain number of emitters are only active for a short period of time over the whole time frame, which violates the time-homogeneous assumption (see Section 1). Therefore, for these datasets, minimizing the global penalized entropy score does not always allow to identify the target partition.

We observe that for three scenarios (2, 7 and 10), MAAP is significantly better than iteratedGreedy [5], but less good on scenario 8; this still highlights the value of improving the search heuristic for solving the DPP in this realistic setting.

## 6 Conclusions

A memetic algorithm for alphabet partitioning was presented in this work. It is used for the problem of deinterleaving pulse trains generated by multiple emitters and described by interleaved Markov processes.

The results show that the proposed heuristic almost always finds the best partitions for synthetic datasets generated with Markov processes and obtains

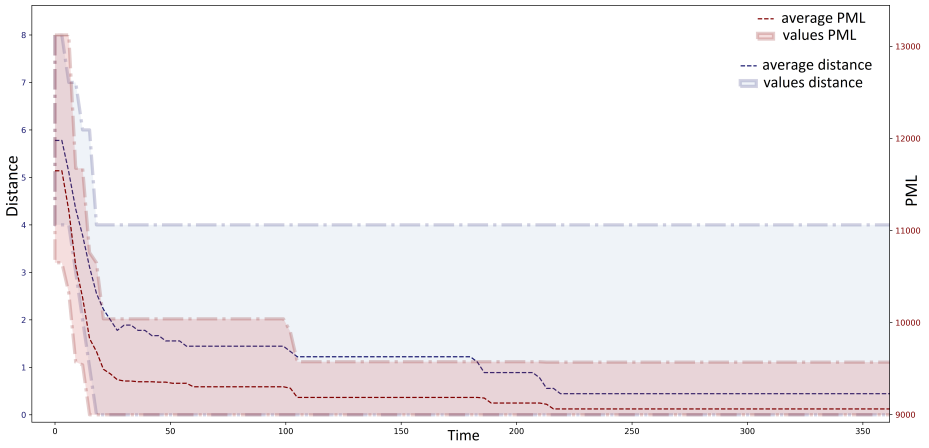



Fig. 2: Evolution over time (seconds) of the distance between the ground truth partition and the current best solution (in blue and left scale), and the corresponding penalized global entropy (in red and right scale), during the search process of the MAAP algorithm.

good results for electronic warfare datasets generated under realistic conditions. For some datasets, it can obtain significantly better results than the recent iterated greedy algorithm [5].

A future work could be to take into account the time delay between different signals to improve the estimation of the different component and switch processes.

## Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions which helped us to improve the paper.

## References

1. Batu, T., Guha, S., Kannan, S.: Inferring mixtures of Markov chains. In: Hutchison, D., et al. (eds.) Learning Theory, vol. 3120, pp. 186–199. Springer, Berlin, Heidelberg (2004)
2. Csiszar, I., Shields, P.: The consistency of the BIC Markov order estimator. In: 2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060). p. 26. IEEE, Sorrento, Italy (2000)
3. Davies, C.L., Hollands, P.: Automatic processing for ESM (1982)
4. Ester, M., Kriegel, H., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA. pp. 226–231. AAAI Press (1996)

| sc | Config | | iteratedGreedy | | | | MAAP | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\|\mathcal{A}\|$ | $n$ | $D^*$ | $\bar{D}$ | $\bar{C}$ | time | $D^*$ | $\bar{D}$ | $\bar{C}$ | time |
| 1 | 13 | 13302 | 2 | 2.0 | 3354 | 34 | 2 | 2.0 | 3354 | 45 |
| 2 | 28 | 10000 | 0 | 7.5 | 31445 | 8122 | 0 | **5.5** | **31237** | 6047 |
| 3 | 11 | 69342 | 3 | 3.0 | 18895 | 78 | 3 | 3.0 | 18895 | 114 |
| 4 | 11 | 68571 | 3 | 3.0 | 18872 | 106 | 3 | 3.0 | 18872 | 123 |
| 5 | 19 | 78783 | **4** | 7.9 | **26443** | 265 | 6 | 7.6 | 26548 | 733 |
| 6 | 24 | 90891 | 2 | 4.9 | 19550 | 4711 | 2 | **4.2** | **19271** | 3086 |
| 7 | 23 | 100000 | 1 | 7.7 | 25575 | 3253 | 1 | **5.6** | **25195** | 4470 |
| 8 | 13 | 100000 | 5 | **5.0** | **88414** | 683 | **4** | 6.0 | 88725 | 871 |
| 9 | 21 | 100000 | 8 | 8.0 | 107021 | 2573 | 8 | 8.0 | 107021 | 3151 |
| 10 | 27 | 100000 | 1 | 6.2 | 56363 | 2431 | 1 | **3** | **54717** | 4263 |

Table 3: Comparison of MAAP and iteratedGreedy on 10 electronic warfare datasets generated with 5 emitters. Dominating results (lower scores) are indicated in boldface. Significant better values are underlined (t-test with p-value 0.05).


5. Ford, G., Foster, B.J., Braun, S.A.: Deinterleaving pulse trains via interleaved markov process estimation. In: 2020 IEEE Radar Conference (RadarConf20). pp. 1–6. IEEE (2020)
6. Galinier, P., Hao, J.K.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization **3**(4), 379–397 (1999)
7. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing **39**(4), 345–351 (1987)
8. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**(1-2), 83–97 (1955)
9. Lü, Z., Hao, J.K.: A memetic algorithm for graph coloring. European Journal of Operational Research **203**(1), 241–250 (2010)
10. Mardia, H.: New techniques for the deinterleaving of repetitive sequences. IEE Proceedings F (Radar and Signal Processing) **136**(4), 149–154 (1989)
11. Milojević, D., Popović, B.: Improved algorithm for the deinterleaving of radar pulses. IEE Proceedings F (Radar and Signal Processing) **139**(1), 98–104 (1992)
12. Moalic, L., Gondran, A.: Variations on memetic algorithms for graph coloring problems. Journal of Heuristics **24**(1), 1–24 (2018)
13. Moore, J., Krishnamurthy, V.: Deinterleaving pulse trains using discrete-time stochastic dynamic-linear models. IEEE Transactions on Signal Processing **42**(11), 3092–3103 (1994)
14. Seroussi, G., Szpankowski, W., Weinberger, M.J.: Deinterleaving finite memory processes via penalized maximum likelihood. IEEE Transactions on Information Theory **58**(12), 7094–7109 (2012)
15. Zhou, Y., Hao, J., Duval, B.: Reinforcement learning based local search for grouping problems: A case study on graph coloring. Expert Systems with Application **64**, 412–422 (2016)