EFFICIENT EVALUATION ALGORITHMS FOR SOUND EVENT DETECTION

Vincent Lostanlen[1*] *and Brian McFee*[2†]

[1] Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France
[2] New York University, Music and Audio Research Lab & Center for Data Science, New York, NY, USA

ABSTRACT

The prediction of a sound event detection (SED) system may be represented on a timeline by intervals whose bounds correspond to onset and offset respectively. In this context, SED evaluation requires to find all non-empty intersections between predicted and reference intervals. Denoting by M and N the number of predicted events and reference events, the time complexity of exhaustive search is $O(MN)$. This is particularly inefficient when the acoustic scene of interest contains many events (typically above 10^3) or when the detection threshold is low. Our article presents an algorithm for pairwise intersection of intervals by performing binary search within sorted onset and offset times. Computational benchmarks on the BirdVox-full-night dataset confirms that our algorithm is significantly faster than exhaustive search. Moreover, we explain how to use this list of intersecting prediction–reference pairs for the purpose of SED evaluation: the Hopcroft-Karp algorithm guarantees an optimal bipartite matching in time $O((M+N)^{3/2})$ in the best case (all events are pairwise disjoint) and $O((M+N)^{5/2})$ in the worst case (all events overlap with each other). The solution found by Hopcroft-Karp unambiguously defines a number of true positives, false positives, and false negatives; and ultimately, information-retrieval metrics such as precision, recall, and F-score.

Index Terms— Evaluation procedures, sound event detection.

1. INTRODUCTION

Given a sound category of interest, the task of sound event detection (SED) aims to identify occurrences of this sound category within an audio recording. SED systems are optimized to pinpoint each instance of the target sound over the time axis. This is known as "strong" labeling, as opposed to "weak" labeling which only reports presence versus absence. In recent years, the renewed interest for deep learning in SED has found many fruitful applications to conservation biology, urban science, industry, and healthcare.

Evaluating the performance of an SED system is not so simple as evaluating a classifier of acoustic scenes. Let us denote the prediction of the system by $\boldsymbol{x}$ and the reference by $\boldsymbol{y}$. We use symbols $\wedge$, $\vee$, and $\neg$ for conjunction (AND), disjunction (OR), and negation (NOT) respectively. With weak labels, $\boldsymbol{x}$ and $\boldsymbol{y}$ boil down to a single bit, and may be compared with elementary logical operations: $(\boldsymbol{x} \wedge \boldsymbol{y})$ for a true positive (TP), $(\boldsymbol{x} \wedge \neg \boldsymbol{y})$ for a false positive (FP), and $(\neg \boldsymbol{x} \wedge \boldsymbol{y})$ for a false negative (FN). The time complexity of this evaluation is independent of the content of $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e., $O(1)$.

The situation is different with strong labels since they are localized in time and potentially repeated over multiple instances. For this matter, we may express the prediction $\boldsymbol{x}$ in terms of a list of M time intervals over $\mathbb{R}$: $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_M) = ([a_1, b_1], \dots, [a_M, b_M])$. Each of these intervals represents a different predicted instance of the target sound, with the lower and upper bound denoting sound onset (start time) and offset (end time) respectively. Likewise, we define the reference $\boldsymbol{y}$ as a list of N intervals: $\boldsymbol{y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_N) = ([u_1, v_1], \dots, [u_N, v_N])$. The evaluation procedure specifies a binary operator, later denoted by $\approx$, which determines whether a predicted interval $\boldsymbol{x}_m$ may be matched to a reference interval $\boldsymbol{y}_n$. An important example of such operator consists in checking whether $\boldsymbol{x}_m$ and $\boldsymbol{y}_n$ have a non-empty intersection:

$$\begin{aligned}(\boldsymbol{x}_m \approx \boldsymbol{y}_n) &\iff \big(\boldsymbol{x}_m \cap \boldsymbol{y}_n\big) \neq \emptyset \\ &\iff (a_m \leq v_n) \wedge (b_m \geq u_n). \end{aligned} \tag{1}$$

The predicted number of events is equal to M and the true number of events is equal to N. Meanwhile, computing the number of true positives (TP) is more challenging because it posits that each interval cannot be matched more than once during evaluation. Formally speaking, we consider a graph $\mathcal{G}$ whose vertices are partitioned into two subsets, $\boldsymbol{x}$ and $\boldsymbol{y}$, and whose edges $\mathcal{E}$ correspond to all interval pairs $(\boldsymbol{x}_m, \boldsymbol{y}_n)$ satisfying Equation (1). We seek three subsets $\mathcal{X} \subseteq \boldsymbol{x}$, $\mathcal{Y} \subseteq \boldsymbol{y}$, and $\mathcal{Z} \subseteq \mathcal{E}$ of highest cardinal, under the constraint that each vertex $\boldsymbol{x}_m \in \mathcal{X}$ and each $\boldsymbol{y}_n \in \mathcal{Y}$ must be incident to at most one of the edges in $\mathcal{Z}$. Hence, SED evaluation comprises two stages:

1. Construct the full edge set

$$\mathcal{E} = \{(\boldsymbol{x}_m, \boldsymbol{y}_n) \subseteq \mathcal{X} \times \mathcal{Y} \mid \boldsymbol{x}_m \approx \boldsymbol{y}_n\}, \tag{2}$$

2. Identify a maximal subset $\mathcal{Z} \subseteq E$ such that each interval of $(\boldsymbol{x} \cup \boldsymbol{y})$ appears at most once. In mathematical terms:

$$\begin{aligned}&\forall \boldsymbol{x}_m \in \boldsymbol{x}, \forall \boldsymbol{x}_{m'} \in \boldsymbol{x} \setminus \{\boldsymbol{x}_m\}, \forall \boldsymbol{y}_n \in \boldsymbol{y}, \forall \boldsymbol{y}_{n'} \in \boldsymbol{y} \setminus \{\boldsymbol{y}_n\}, \\ &(\boldsymbol{x}_m, \boldsymbol{y}_n) \in \mathcal{Z} \implies \big((\boldsymbol{x}_{m'}, \boldsymbol{y}_n) \notin \mathcal{Z}\big) \wedge \big((\boldsymbol{x}_m, \boldsymbol{y}_{n'}) \notin \mathcal{Z}\big)\end{aligned} \tag{3}$$

The second stage is efficiently solved by the Hopcroft-Karp algorithm [1]. Meanwhile, our work focuses on the first stage: i.e., to efficiently identify all candidate matchings. This problem is solvable in time $O(MN)$ by comparing all pairs of intervals. However, in SED settings which cover long time periods, the number of detected intervals may range in the thousands, making the all-pairs matching approach inefficient in practice.

The key observation is that most comparisons of the form $\boldsymbol{x}_m \approx \boldsymbol{y}_n$ will evaluate to false and can be discarded in advance. Indeed, if we know a lower bound b on the onset u_n of $\boldsymbol{y}_n$, then we can conclude that any interval $\boldsymbol{x}_m = [a_m, b_m]$ such that $b_m < b$ is necessarily disjoint from $\boldsymbol{y}_n$; and likewise if we know an upper bound a on the offset v_n such that $a_m > a$.

*VL is supported by CNRS grant CAPTEO, WeAMEC grant PETREL, and Horizon Europe BioacAI.

†BM is supported by NSF award 1955357.

In this article, we propose an algorithm for evaluating SED efficiently; i.e., without examining all pairs. More precisely, our algorithm implements maximum cardinal matching on interval bigraphs [2], and has an asymptotic time complexity of

$$O\left((M+N)(\log M + \log N) + |\mathcal{E}|\sqrt{M+N}\right). \quad (4)$$

We begin by explaining why a greedy approach, in which all intervals are visited once, is not guaranteed to return the optimal number of true positives, and thus should not be used. Then, we present the two stages of our algorithm: construction of the interval bigraph and maximum cardinality matching. We discuss the use of our algorithm since 2021 as part of Task 5 of the DCASE challenge on few-shot bioacoustic event detection; and its connection with an existing algorithm for efficient evaluation of sound event detection in mir_eval. We conclude with a performance benchmark on a realistic use case, namely, automatic detection of avian flight calls in the BirdVox-full-night dataset [3].

2. SUBOPTIMAL GREEDY ALGORITHM

Algorithm 1 Exhaustive search of matching pairs $(\boldsymbol{x}_m, \boldsymbol{y}_n)$. The length of $\mathcal{E}$ is an upper bound on TP. Complexity: $O(MN)$.

$\mathcal{E} = \text{list}()$
for $m = 1$ to M **do**
  for $n = 1$ for N **do**
    if $\boldsymbol{x}_m \approx \boldsymbol{y}_n$ **then**
      append $(\boldsymbol{x}_m, \boldsymbol{y}_n)$ to $\mathcal{E}$
    end if
  end for
end for
return $\mathcal{E}$

Algorithm 2 Greedy search of matching pairs $(\boldsymbol{x}_m, \boldsymbol{y}_n)$. Lists $\mathcal{X}$ and $\mathcal{Y}$ have the same length as $\mathcal{Z}$ and contain non-repeating elements only. This algorithm gives a lower bound on TP and should not be used for SED evaluation. Worst-case complexity: $O(MN)$.

$\mathcal{Z} = \text{list}()$
$\mathcal{X} = \text{list}()$
$\mathcal{Y} = \text{list}()$
for $m = 1$ to M **do**
  for $n = 1$ for N **do**
    if $(\boldsymbol{x}_m \approx \boldsymbol{y}_n) \wedge (\boldsymbol{y}_n \notin \mathcal{Y})$ **then**
      append $(\boldsymbol{x}_m, \boldsymbol{y}_n)$ to $\mathcal{Z}$
      append $\boldsymbol{x}_m$ to $\mathcal{X}$
      append $\boldsymbol{y}_n$ to $\mathcal{Y}$
      break
    end if
  end for
end for
return $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$

At first glance, the evaluation of an SED system may seem easy. Since the number of true positives (TP) involves non-disjoint pairs between a prediction interval $\boldsymbol{x}_m$ and a reference interval $\boldsymbol{y}_n$, one could list those pairs exhaustively with a double loop, as in Algorithm 1. Yet, this algorithm involves $O(MN)$ comparisons and only produces an upper bound on TP, since the same interval may appear in multiple pairs of the list $\mathcal{E}$. A potential workaround consists in defining a list $\mathcal{X}$ containing all the prediction intervals that have been matched so far, and grow it as we traverse the list $\boldsymbol{x}$; and likewise for $\mathcal{Y}$ and $\boldsymbol{y}$. By only admitting a new pair $(\boldsymbol{x}_m, \boldsymbol{y}_n)$ if $\boldsymbol{x}_m$ does not already belong to $\mathcal{X}$ nor $\boldsymbol{y}_n$ to $\mathcal{Y}$, one guarantees that each prediction interval is matched to at most one reference interval and vice versa: see Algorithm 2. This is a form of "greedy" search: it makes a locally optimal choice at each stage, yet is globally suboptimal. In this instance, the inner loop in Algorithm 2 consists in looking for an unmatched reference interval ($\boldsymbol{y}_n \notin \mathcal{Y}$) such that $\boldsymbol{x}_m \approx \boldsymbol{y}_n$, given an unmatched prediction interval ($\boldsymbol{x}_m \notin \mathcal{X}$).

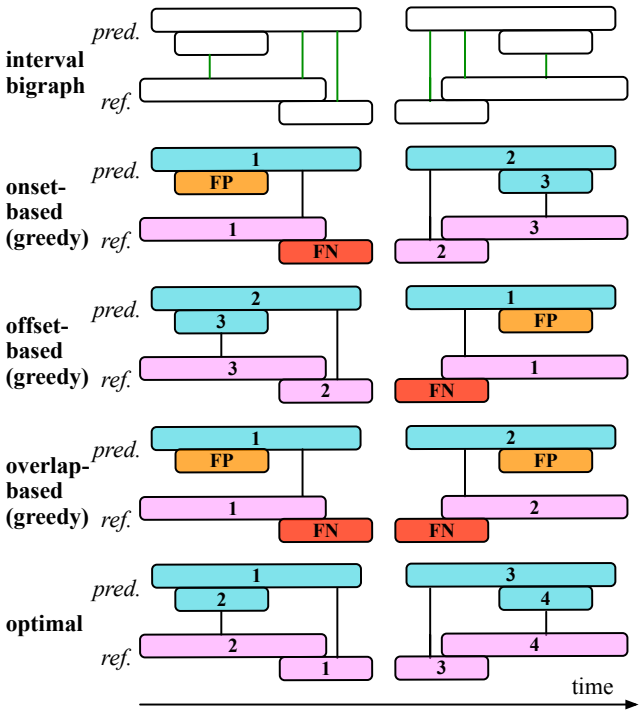

Figure 1: Top: exhaustive search (Algorithm 1) overestimates the number of true positives (TP) of sound event detection, defined as the maximum cardinality matching between prediction (blue) and reference (orange). Center: greedy search (Algorithm 2) underestimates TP. Bottom: our algorithm (Algorithm 3) returns the correct value of TP. See Section 2 for details.

Figure 2 illustrates the problem of SED evaluation between a prediction of $M = 4$ intervals and a reference of $N = 4$ intervals. The top row, in white, implements Algorithm 1 on this example, yielding a list $\mathcal{E}$ of six pairs, each of the form $(\boldsymbol{x}_m, \boldsymbol{y}_n)$. Greedy algorithms, such as Algorithm 2, achieve this by traversing the list $\mathcal{E}$ once, in a predefined order. In this way, they define a sublist $\mathcal{Z} \subset \mathcal{E}$, which is initialized as the empty list and grown progressively until all pairs in $\mathcal{E}$ have been examined. The second, third, and fourth rows in Figure 2 show instances of such greedy algorithms, with variations in priority: i.e., based on earlier onset time a_m, on later offset time b_m, or on greatest overlap (as done in MIREX). We observe that, even though these greedy algorithms differ in terms of pairing sublist $\mathcal{Z}$, both of them leave one prediction event and one reference event unmatched. Hence, they evaluate the prediction as yielding three TP, one FP, and one FN.

Yet, all the greedy algorithms mentioned above are suboptimal. Indeed, in Figure 2, there exists a matching which yields four TP and no FP nor FN: see bottom row. In the next section, we present an algorithm which finds this optimal solution in polynomial time.

3. MAIN CONTRIBUTION

Our proposed method comprises two stages: construction of the interval bigraph $\mathcal{G} = (\boldsymbol{x} \cup \boldsymbol{y}, \mathcal{E})$ and maximum cardinality matching. For the first stage, we write a custom iterative algorithm which is based on dichotomic search: see Algorithm 3. For the second stage, we reuse the algorithm of Hopcroft and Karp for maximum cardinality matching of bipartite graphs [1]. The novelty of our method resides in the faster construction of the set of edges $\mathcal{E}$ in the first stage, by exploiting the temporal logic of intervals in Equation 1.

Algorithm 3 presents our main contributions, i.e., the fast construction of the edge set $\mathcal{E}$. We begin by sorting predicted offsets $(b_1, \ldots, b_M)$, yielding a permutation σ satisfying $b_{\sigma(1)} \leq \ldots \leq b_{\sigma(M)}$. Thanks to divide-and-conquer algorithms such as quicksort, this operation incurs an average-case time complexity of $O(M \log M)$. Likewise, we sort reference onsets $(u_1, \ldots, u_N)$, yielding a permutation ϕ satisfying $u_{\phi(1)} \leq \ldots \leq u_{\phi(N)}$ with average-case $O(N \log N)$ time complexity. Then, we initialize two integers n and μ, pointing to the first element of ϕ and σ respectively. We define the partial list of sorted predicted offsets $L = (b_{\sigma(\mu)}, \ldots b_{\sigma(M)})$ and increment μ by the index of the first element in L exceeding $u_{\phi(n)}$. Since L is already sorted, this element may be found via binary search, whose number of comparisons is logarithmic in M in the worst case.

After having updated μ, we store the set $\{\sigma(\mu), \ldots, \sigma(M)\}$ at the entry $\phi(n)$ of a set-valued array $\mathcal{S}$. We increment n by one and repeat the operation of dichotomic search over the list L with the new value $u_{\phi(n)}$. We halt this procedure as soon as $u_{\phi(n)}$ exceeds $b_{\sigma(M)}$: indeed, for greater values of n, $\mathcal{S}(\phi(n))$ is known to be empty. By construction, each set $\mathcal{S}(\phi(n))$ contains all the indices of the predicted events whose offsets happen after the onset of the reference event $\phi(n)$. Formally:

$$\begin{aligned} \mathcal{S}(\phi(n)) &= \left\{1 \leq \sigma(m) \leq M \,|\, b_{\sigma(m)} \geq u_{\phi(n)}\right\} \\ &= \left\{1 \leq m \leq M \,|\, b_m \geq u_n\right\} \end{aligned} \tag{5}$$

The latter formula in the equation above is obtained after applying inverse permutations σ^{-1} and ϕ^{-1} to indices m and n respectively. Going back to Equation 1, we observe that $(\boldsymbol{x}_m \approx \boldsymbol{y}_n)$ implies $(m \in \mathcal{S}(n))$ but the converse is not necessarily true. However, if $(m \in \mathcal{S}(n))$ for some pair (m, n), a necessary and sufficient condition for $(\boldsymbol{x}_m \approx \boldsymbol{y}_n)$ is $(a_m \leq v_n)$. Thus, we propose to refine each set $\mathcal{S}(n)$ by intersecting it with the set of all indices m such that $a_m \leq v_n$.

We sort predicted onsets $(a_1, \ldots, a_M)$, yielding a permutation π which satisfies $a_{\pi(1)} \leq \ldots \leq a_{\pi(M)}$. Likewise, we sort reference offsets $(v_1, \ldots, v_N)$, yielding a permutation ψ which satisfies $v_{\psi(1)} \leq \ldots \leq v_{\psi(N)}$. Similarly to σ and ϕ in the paragraph above, these sorting operations incur a cumulated asymptotic cost of $O(M \log M + N \log N)$ in the average case. We reset the integer to n to N. We set μ to the maximum value of m such that $a_{\pi(m)}$ is below $v_{\psi(n)}$. Thanks to sorting, this may be achieved by binary search, whose worst-case complexity is $O(\log M)$. We update the list $\mathcal{S}(\psi(n))$ by intersecting it with $(\pi(1), \ldots, \pi(\mu))$. This intersection may be implemented efficiently with a hash table, as it does not involve any numerical comparison. We decrement n by one and repeat the process until $v_{\psi(n)}$ falls below $a_{\pi(1)}$. Finally, we build $\mathcal{E}$ incrementally by looping through every value $m \in \mathcal{S}(n)$ for n from 1 to N and constructing the pair $(\boldsymbol{x}_m, \boldsymbol{y}_n)$. In practice, since $\mathcal{S}(n)$ has much fewer than M elements, the number of iterations in this nested loop is typically much smaller than $O(MN)$. Thus, most of the computational cost of Algorithm 3 is spent in binary search.

	naïve	proposed
graph construction	MN	$(M+N)(\log M + \log N) + \lvert\mathcal{E}\rvert$ (Algorithm 3)
event matching	$2^{\lvert\mathcal{E}\rvert}$	$\lvert\mathcal{E}\rvert\sqrt{M+N}$ (Hopcroft-Karp)

Table 1: Upper bounds on the asymptotic time complexities of algorithms for constructing non-disjoint interval pairs $\mathcal{E}$ (left column) and maximum cardinality matching $\mathcal{Z}$ (right column) in the worst case, up to a constant multiplicative factor. See Section 3 for details.

Algorithm 3 Our algorithm lists all edges of an interval bigraph.

$\mathcal{E} \leftarrow \emptyset$
$\mathcal{S}(1), \ldots, \mathcal{S}(N) \leftarrow \emptyset$ {initialize list of matching indices}
$\sigma \leftarrow \arg\operatorname{sort}(b_1, \ldots, b_M)$ {sort predicted offsets}
$\phi \leftarrow \arg\operatorname{sort}(u_1, \ldots, u_N)$ {sort reference onsets}
$n \leftarrow 1$
$\mu \leftarrow 1$
while $u_{\phi(n)} \leq b_{\sigma(M)}$ {up to last predicted offset} **do**
  $L \leftarrow (b_{\sigma(\mu)}, \ldots, b_{\sigma(M)})$ {sublist of predicted offsets}
  $\mu \leftarrow \mu + \min\left\{0 \leq i \leq (M-\mu) \,|\, u_{\phi(n)} \leq L_{i+1}\right\}$
  $\mathcal{S}(\phi(n)) \leftarrow \{\sigma(\mu), \ldots, \sigma(M)\}$
  $n \leftarrow n + 1$
end while
$\pi \leftarrow \arg\operatorname{sort}(a_1, \ldots, a_M)$ {sort predicted onsets}
$\psi \leftarrow \arg\operatorname{sort}(v_1, \ldots, v_N)$ {sort reference offsets}
$n \leftarrow N$
while $v_{\psi(n)} \geq a_{\pi(1)}$ {down to first predicted onset} **do**
  $\mu \leftarrow \max\left\{1 \leq m \leq \mu \,|\, v_{\psi(n)} \geq a_{\pi(m)}\right\}$
  $\mathcal{S}(\psi(n)) \leftarrow \mathcal{S}(\psi(n)) \cap (\pi(1), \ldots, \pi(\mu))$
  $n \leftarrow n - 1$
end while
for n from 1 to N {for every reference event} **do**
  for $m \in \mathcal{S}(n)$ {for every matching prediction} **do**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(I_m, J_n)\}$ {include edge}
  end for
end for
return $\mathcal{E}$

As shown in Table 1, Algorithm 3 accelerates graph construction from $O(MN)$ to $O(M \log M + N \log N + |\mathcal{E}|)$. If the detector is perfect ($\boldsymbol{x} = \boldsymbol{y}$) and if the reference consists of disjoint intervals, one has $M = N = |\mathcal{E}|$: under this important special case, the time complexity of graph construction is $O(N^2)$ for exhaustive search (Algorithm 1) versus $O(N \log N)$ for binary search (Algorithm 3). Furthermore, the complexity of event matching is 2^N for exhaustive search versus $N\sqrt{N}$ for Hopcroft-Karp. The interest behind our contribution is that $N \log N$, unlike N^2, is dominated by $N\sqrt{N}$; thus, after replacing Algorithm 1 by Algorithm 3, the cost of graph construction is asymptotically negligible, and most of SED evaluation is spent in event matching.

4. PRACTICAL CONSIDERATIONS

4.1. Selecting pairs based on intersection-over-union ratio

We have implemented Algorithm 3 as part of the official evaluation toolkit[1] of the DCASE challenge task on "Few-shot bioacoustic event detection" [4]. This task was inaugurated in 2021 and maintained through 2022 and 2023. As part of the challenge rules, we have stated that, in order to be considered a valid matching, an interval pair $(\boldsymbol{x}_m, \boldsymbol{y}_n)$ should not only overlap but also have at least 50% of intersection-over-union ratio (IoU). This is a refinement of Equation 1 in the criterion $\boldsymbol{x}_m \approx \boldsymbol{y}_n$. To accelerate the construction of the bipartite graph $\mathcal{G}$, we run Algorithm 3 as a prefiltering stage and then evaluate IoU explicitly on all non-disjoint pairs.

4.2. Interoperability with sed_eval

Running Algorithm 3 in conjunction with Hopcroft-Karp yields a maximal matching $\mathcal{Z}$ for $\mathcal{G}$. The set cardinal of $\mathcal{Z}$ corresponds to the number of true positives (TP) of the detector. From this number, we deduce the following information-retrieval metrics:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{|\mathcal{Z}|}{M}, \tag{6}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{|\mathcal{Z}|}{N}, \tag{7}$$

$$F_1\text{-score} = \frac{2}{(\text{Precision})^{-1} + (\text{Recall})^{-1}} = \frac{2|\mathcal{Z}|}{M + N}, \tag{8}$$

which are already in widespread use in the DCASE community thanks to the sed_eval toolbox [5]. While sed_eval relies on exhaustive search (Algorithm 1) for graph construction, we rely on Algorithm 3. For event matching, both sed_eval and our implementation rely on the Hopcroft-Karp algorithm, as made available by the SciPy toolbox. Therefore, our implementation returns the same output as sed_eval while being more computationally efficient.

4.3. An important special case: evaluating onset detection

Algorithm 3 generalizes another algorithm, implemented under the name of "_fast_hit_windows" in mir_eval v0.5 and later [6]. In combination with Hopcroft-Karp, this other algorithm serves to evaluate sound onset detection efficiently. For future reference, we present its pseudocode in Algorithm 4. The premise of Algorithm 4 is that a predicted onset a_m may be matched to a reference onset u_n if and only if they are within a certain time lag δ of each other. Formally:

$$\begin{aligned}(a_m \approx u_n) &\iff |a_m - u_n| \leq \delta \\ &\iff (u_n \geq a_m - \delta) \wedge (u_n \leq a_m + \delta)\end{aligned} \tag{9}$$

Like Algorithm 3, Algorithm 4 begins by sorting reference onsets, which incurs an $O(N \log N)$ time complexity. Then, for every predicted onset u_m, it performs binary search over the sorted list of reference onsets $(u_{\phi(1)}, \ldots, u_{\phi(N)})$, while accounting for the maximum admissible lag δ. This later stage incurs a time complexity of $O(M \log N)$; hence, the time complexity of Algorithm 4 is $O((N + M) \log N)$. Note that, if there are many more predicted onsets than reference onsets ($N \gg M$), the algorithm may be accelerated by a factor $(\log N / \log M)$ by swapping the roles of prediction and reference.

[1] Source code: https://github.com/c4dm/dcase-few-shot-bioacoustic. The metrics module implement functions slow_intersect (Algorithm 1) and fast_intersect (Algorithm 3).

Algorithm 4 An efficient evaluation algorithm for sound onset detection, as implemented in the mir_eval v0.5 and later.

$\mathcal{E} \leftarrow \emptyset$
$\phi \leftarrow \arg\text{sort}(u_1, \ldots, u_N)$
for m from 1 to M **do**
  $n_{\min} \leftarrow \min \{1 \leq n \leq N \mid u_{\phi(n)} \geq a_m - \delta\}$
  $n_{\max} \leftarrow \max \{1 \leq n \leq N \mid u_{\phi(n)} \leq a_m + \delta\}$
  for n from $n_{\min}$ to $n_{\max}$ **do**
    $\mathcal{E} \leftarrow \mathcal{E} \cup (a_m, u_{\phi(n)})$
  end for
end for
return $\mathcal{E}$

5. EXAMPLE APPLICATION

To evaluate the speed of exhaustive search (Algorithm 1) versus our algorithm (Algorithm 3), we evaluate a deep convolutional network for automatic detection of avian flight calls on an audio recording from the BirdVox-full-night dataset [3]. This audio recording lasts for roughly 11 hours and has been annotated by an expert ornithologist. The reference $\boldsymbol{y}$ contains $N = 9113$ events. We set the threshold of the convnet detector to a value such that the prediction $\boldsymbol{x}$ contains $M = 2N = 18226$ events.

In the Python programming language, Algorithm 1 takes 65 ± 1 seconds to find all matching pairs between $\boldsymbol{x}$ and $\boldsymbol{y}$ on a personal computer (2.3 GHz Quad-Core Intel Core i7). On the same computer, Algorithm 3 returns the same output within 11.7 ± 0.1 seconds. Beyond the raw comparison, we note that the speed could be improved further by resorting to a high-performance compiler such as Numba. We should also keep in mind that, in practice, computing the area under the precision–recal curve (AUPRC) requires to recompute the bipartite graph $\mathcal{G}$ for many values of the detection threshold, including low values when $M \gg N$. Furthermore, SED evaluation is typically performed over several initializations of the system and across several hyperparameter choices, as in [7]. Hence, the gain in speed by switching from Algorithm 1 to Algorithm 3 becomes significant when conducting a full-scale benchmark.

6. CONCLUSION

With this article, we have stressed the difficulty of making SED evaluation both correct and computationally efficient, by pointing out the shortcomings of greedy methods and of exhaustive search. We have presented an algorithm evaluating sound event detection, which generalizes an evaluation algorithm for onset detection in mir_eval. Our theoretical analysis and speed benchmark on a long-duration audio recording demonstrate the interest of this algorithm.

7. ACKNOWLEDGMENT

We thank Colin Raffel for maintaining the mir_eval package, which serves as an inspiration to this work. We also thank Veronica Morfi, Inês Nolasco, and Shubhr Singh, and Dan Stowell for maintaining the evaluation software of DCASE Task 5.

8. REFERENCES

[1] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on comput-*

ing, vol. 2, no. 4, pp. 225–231, 1973.

[2] A. K. Das and R. Chakraborty, "New characterizations of proper interval bigraphs," *AKCE International Journal of Graphs and Combinatorics*, vol. 12, no. 1, pp. 47–53, 2015.

[3] V. Lostanlen, J. Salamon, A. Farnsworth, S. Kelling, and J. P. Bello, "Birdvox-full-night: A dataset and benchmark for avian flight call detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 266–270.

[4] V. Morfi, I. Nolasco, V. Lostanlen, S. Singh, A. Strandburg-Peshkin, L. F. Gill, H. Pamula, D. Benvent, and D. Stowell, "Few-shot bioacoustic event detection: A new task at the DCASE 2021 challenge," in *Proceedings of the International Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2021, pp. 145–149.

[5] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[6] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *Proceedings of the International Society of Music Information Retrieval (ISMIR) Conference*, 2014, pp. 367–372.

[7] V. Lostanlen, J. Salamon, A. Farnsworth, S. Kelling, and J. P. Bello, "Robust sound event detection in bioacoustic sensor networks," *PLOS ONE*, vol. 14, no. 10, p. e0214168, 2019.