# MusicBLAST — GAPPED SEQUENCE ALIGNMENT FOR MIR

*Jürgen Kilian*
Darmstadt University of Technology
FB Informatik / AFS
e-mail:kilian@noteserver.org

*Holger H. Hoos*
University of British Columbia
Department of Computer Science
e-mail:hoos@cs.ubc.ca

**ABSTRACT**

We propose an algorithm, MusicBLAST, for approximate pattern search/matching on symbolic musical data. MusicBLAST is based on the BLAST algorithm, one of the most commonly used algorithms for similarity search on biological sequence data [1, 2]. MusicBLAST can be used in combination with an arbitrary similarity measure (*e.g.,* melodic, rhythmic or combined) and retrieves multiple occurrences of a given search pattern and its variations. Different from many other pattern matching techniques, it can find incomplete and imperfect occurrences of a given pattern, and produces a significance measure for the accuracy and quality of its results. Like BLAST — and different from many musical pattern matching approaches — MusicBLAST retrieves heuristically optimised bi-directional alignments searching iteratively in forward and backward direction by starting at a dedicated *seed note* position of a performance.
**Keywords**: Similarity, pattern matching, retrieval.

## 1. MOTIVATION

Searching for a given musical pattern or fragment (based on melodic, rhythmic, or arbitrary types of similarity) in a piece of music or a musical database is a common task in the context of music information retrieval (MIR). Related to pattern searching are the problems of pattern induction in the context of musical analysis (*i.e.,* inferring a global structure, such as AABA) and the task of score-following.

Typical queries may be inexact, and consequently, a search algorithm should be able to support approximate pattern matching and gapped alignments between a search pattern and given performance data or pieces in a database. Especially in the MIR domain, matching algorithms typically need be able to search large databases, and hence must be optimised for performance. A similar situation arises in the domain of bioinformatics, where the identification of approximate similarities between biological sequences, such as genomic DNA, is an extremely important task; in this field, an algorithm called *Basic Local Alignment Search Tool* (BLAST) has become one of the most widely used methods for accomplishing this task.



The similarity between the problem of finding approximate matches for a given biological sequence (*e.g.,* a gene) in large biological sequence databases (such as GenBank) and typical retrieval tasks in the musical domain suggests the adaptation of the basic features of BLAST to musical data. In the following we give a short outline about the original BLAST algorithm and explain its adaption to retrieval on symbolic musical data. This is followed by a summary of preliminary results on the performance of the new MusicBLAST algorithm, and an outlook to future work.

## 2. BLAST

The gapped BLAST algorithm [2], based on an earlier *ungapped* version [1], is a commonly known and widely used search tool in biological sequence analysis. Applied to amino acid sequences, it works as follows: [1]

First a *window based similarity matrix* of size $m \times n$ between two arbitrary amino acid strings of size $m$ and $n$, respectively, is created. Each entry of the matrix represents the similarity between subsequences of the two amino acid strings with a certain length $l$; $l$ is also called *window size*. The similarity between two *single characters* of the strings is calculated by using a *position specific score matrix* which specifies a similarity measure between the different amino acids. A commonly used score matrix for amino acids is the $20 \times 20$ PAM matrix. [2]

Next, a *limited number of high-scoring hits* within the window-based similarity matrix (best matching windows) are selected and used for determining the start positions (*seeds*) for possible alignments. A high-scoring window will only be used for determining a seed position if in the same diagonal of the matrix another high-scoring entry appears within a certain distance. As a start position (seed), any entry of an high-scoring window (*e.g.,* the one with the highest score, or the centre of the window as used in [2]) might be selected. Starting from the selected seeds, a *bi-directional gapped alignments* will be retrieved using a performance optimised version of string matching by dynamic programming (DP), which produces a cost optimised local alignment.

As shown in Figure 1, the two DP tables for the right and left directed alignments are filled alternatingly via the inverse diagonals, starting at the seed position.

[1] While in [2], BLAST is applied to protein sequences, *i.e.,* sequences of amino acid symbols, in general, it is also frequently applied to DNA and RNA sequences.

[2] see `http://www.cmbi.kun.nl/bioinf/tools/pam.shtml`

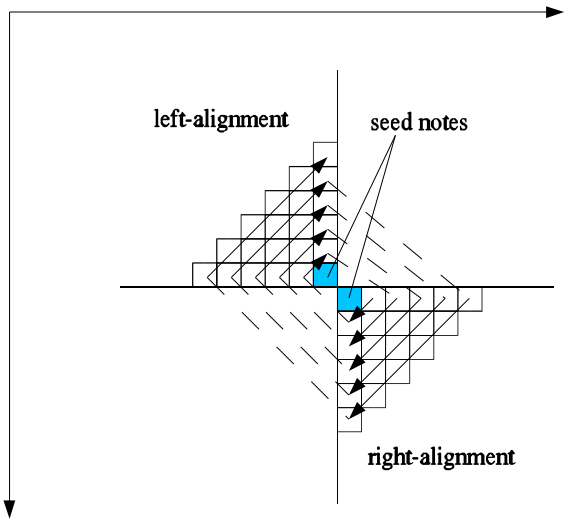


**Figure 1**. BLAST: iterative filling of the DP table along the inverse diagonals of the right and left alignment.

Although in principle, the DP procedure guarantees an optimal local alignment, because of the use of heuristically selected seed positions and abort criteria, BLAST may not always find an optimal alignment; however, in practice, it has been found to give an excellent combination of accuracy and efficiency. To achieve this efficiency, the number of evaluated cells of each DP table is limited by a threshold on the score of cells relative to the current best alignment score. Cells are marked as invalid if their score falls below this threshold. For invalid cells located at the border of the DP table, the remaining cells of the corresponding row or column need not be evaluated. Different from other efficiency improvements to DP (*e.g.*, [1, 4]), this heuristic does not directly limit the length or the course of the best path of the DP table.

The most time consuming task within BLAST is the generation of gapped alignments; but by using the threshold optimisations, the complexity for computing a single alignment can typically be minimised far below $O(n \cdot m)$, because then large parts of the DP table must not be filled. Furthermore, the window-based similarity matrix is used to limit the number of calls to the DP procedure to promising start positions.

## 3. BLAST ON MUSICAL DATA

By using the outline of the original gapped BLAST approach and adapting the similarity measures to musical data, it is possible to exploit the performance advantages of the BLAST algorithm for pattern retrieval and similarity analysis in the musical domain. For creating the similarity matrix on musical data (a series of notes and chords), the scoring matrix for amino acids (or nucleotides) needs to be replaced by a similarity function that assigns a score to any combination of two notes or chords. Depending on the precise application context, this similarity function can be based on any feature of a single note or chord, in particular: pitch, pitch ratio (interval), duration, inter-onset interval (IOI), IOI ratio, or intensity. Analogously to the biological scoring matrix, the similarity function should give positive results for highly similar notes and negative results for notes that are not similar. In general, any similarity function can be used that satisfies the general requirements of a DP cost function, especially similarity functions used in context of other DP based musical string matching (*e.g.*, [4]). Here eventually the function's output range must be normalised to the intended range. By evaluating the IOI ratio and/or the duration ratio instead of absolute IOI and duration, it is possible to allow rhythmical pattern matching between any combination of unquantised performance data and quantised score data. To start the bi-directional alignment, a *seed note* needs to be selected within each high-scoring window of the similarity matrix; this can be the centre note or any other note inside the window (*e.g.*, the longest note). For our evaluation we used the note with the highest similarity score as start position.

It should be noted that standard DP methods for string matching have already been successfully applied to score following (see [3]) and MIR (*e.g.*, [10]); however, compared to the approach proposed here, these suffer from several limitations. For example, a bi-directional heuristic alignment (starting from the seed note in forward and backward direction) promises advantages for all scenarios where two patterns have a high (ungapped) similarity on a small range of notes only. By starting the alignment search in both directions from that high similarity region, the number of calculations during the dynamic programming can be decreased significantly. In these situations standard approaches such as [10] would require the calculation of all $n \cdot m$ positions of the DP table, where the query length $m$ is usually significantly smaller than the performance length or database size $n$. Assuming that the calculation of an alignment will be aborted by the optimisation features of BLAST if the number of insertions (gaps) exceeds $k \cdot m$. the bi-directional approach would require the calculation of only $2 \cdot k \cdot m^2$ entries. The rather simple calculation of the similarity matrix still requires time $O(n \cdot m)$, but the complexity of the alignment retrieval depends only on the query length.

After retrieving a set of cost optimised gapped alignments — built by concatenating of pairs of left- and right-alignments — these can be ranked by their significance or matching quality. This significance can be determined as the total cost already calculated by the DP procedure or calculated by applying a general cost function, which need not satisfy the constraints for DP. By allowing gaps in the alignment of a search pattern and the performance data, the query will be more robust against extraneous or missing notes in the pattern and/or the performance data. Using BLAST on musical data, as described above, it is also possible to retrieve substring alignments and multiple occurrences of a pattern in a piece or performance. Mongeau and Sankoff proposed an approach based on dynamic programming that allows fragmentation (splitting of notes) and consolidation (merging of notes) during retrieving a cost optimal alignment [8]. In principle, it should be possible to integrate their improvements into BLAST, where this would require a re-design of the advanced aborting criteria of BLAST, which might decrease its efficiency.

## 4. PRELIMINARY RESULTS AND DISCUSSION

The MusicBLAST algorithm described in the previous section has been prototypically implemented within a more general system for inferring score level information from low level musical data called *midi2gmn*, which supports as input mechanically or live performed MIDI files or symbolic representation in GUIDO Music Notation. The MusicBLAST module can be used in two ways: for retrieving approximate (complete or partial) occurrences of a search pattern given as a single voice GUIDO file, or for analysing the overall structure of an arbitrary input file by performing a self-similarity analysis. For the example discussed in the following, we used a similarity measure that evaluates the absolute (transposition invariant) pitch distance between two notes; intervals smaller than three semitones resulted in positive costs and intervals larger than three semitones in negative costs. Gaps were assigned a penalty equal to that of a five-semitone interval.

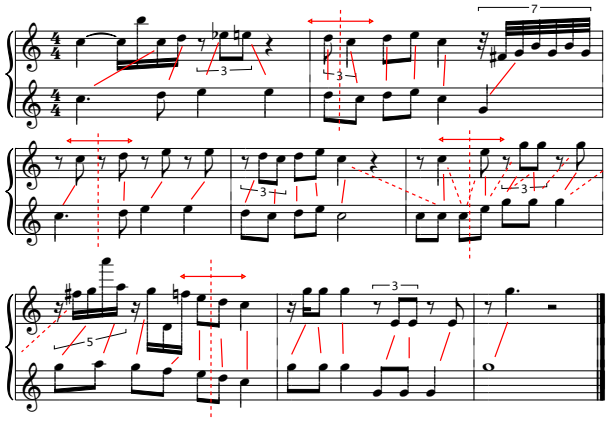

**Figure 2**. Example showing a performance and the original score of *Alouette* (the same example is discussed in [9]). The solid lines connect pairs of notes that have been aligned by our implementation of the MuscBLAST algorithm. Dashed vertical lines (with arrows on top) indicate the start positions of the alignments included in the path of the four overlapping alignments around the main diagonal of the similarity matrix (the two other alignments are not shown here, see Figure 4). Dashed connections between notes indicate that here the different alignments paths produced different alignments. Note that the alignment at the beginning of the first measure is a result of the fact that the similarity measure used in this experiment ignores any rhythmic information.

For the input data shown in Figure 2, our MusicBLAST implementation selected six high scoring entries of the similarity matrix, each of which triggered the bi-directional DP procedure for retrieving an optimum alignment starting at the best match of the high scoring window (see Figure 4b). The evaluation of our tests showed that the use of the BLAST optimisations for DP decreased the number of cells (of the DP tables) that needed to be evaluated for the retrieval of all six alignments from $12\,054$ $(= 41{\cdot}49{\cdot}6)$ to $2\,143$ (17.8%). As shown in the trace of the alignment paths in Figure 4c, the start positions of the alignments 3 to 5 are subsets of the path of alignment 2. Assuming that it is possible (without increasing the overall time complexity) to mark start positions as invalid, that are part of an already retrieved alignment, then the number of evaluated cells would have been decreased to only 11.8% compared to a non-optimised DP implementation. The roughtly constant ratio between the number of evaluated cells and the length of the retrieved pattern seen in Figure 3 gives an indication that the assumption that with using the optimised method for filling the DP table in BLAST (marking high penalty cells as invalid), on average, a single bi-directional alignment can be computed in time $O(m)$, where $m$ is the length of the query.

| nr | length (l/r/l+r) | #cells/length (l/r/l+r) |
|---|---|---|
| 1 | 7 / 12 / 19 | 8.8 / 7.9 / 8.3 |
| 2 | 5 / 42 / 47 | 7.5 / 10.17 / 9.85 |
| 3 | 25 / 21 / 46 | 9.32 / 12.23 / 10.67 |
| 4 | 13 / 31 / 44 | 7.31 / 11.9 / 10.52 |
| 5 | 36 / 9 / 45 | 10.86 / 9.78 / 10.64 |
| 6 | 2 / 9 / 11 | 4.5 / 9.0 / 8.18 |
| | average 1-6: | 11.08 |
| | average 1,2,6: | 8.24 |

**Figure 3**. Evaluation of the retrieved alignments for the *Alouette* example. Each column shows the values for the right-, left-alignments, and the combination of both. The column length shows the lengths of the retrieved alignments including inserted gaps, the right column gives the relation between the number of evaluated cells of the DP tables and the retrieved alignment length.

One of the algorithms for pattern matching proposed in [5] (named Algorithm 3) appears to be similar to MusicBLAST (and gapped BAST, respectively), but there are some differences between both algorithms: Algorithm 3 searches in two directions for start and end points of a discovered pattern, but not in the iterative bi-directional way used within the BLAST approach. In our model, the abort criterion depends on the distance between local alignment costs and the best local alignment costs achieved so far; this seems to be less restrictive than the global threshold proposed by Dannenberg and Hu. The dynamic abort criterion of BLAST avoids the insertions of gaps at the end of the alignment (where they have to be trimmed later). The window based MusicBLAST selection strategy for start positions of alignments seems to be more selective (higher discrimination rate) than the single-note-similarity-based strategy of Algorithm 3. As shown in Figure 4, the window based similarity matrix (b) achieves a significantly higher discrimination rate than a matrix based on note-to-note similarity (a). With the MusicBLAST approach it is possible — if desired — to retrieve overlapping matches, which seems to be not the case for the approach described in [5]. Finally, different from Dannenberg's and Hu's implementation, where a complete second $n \times m$ matrix is used for marking invalid cells, MusicBLAST requires only two arrays (with size $n$ and $m$) for storing the index of the last evaluated cell (first invalid cell, respectively) in each row and each column.

There are faster algorithms for exact pattern matching

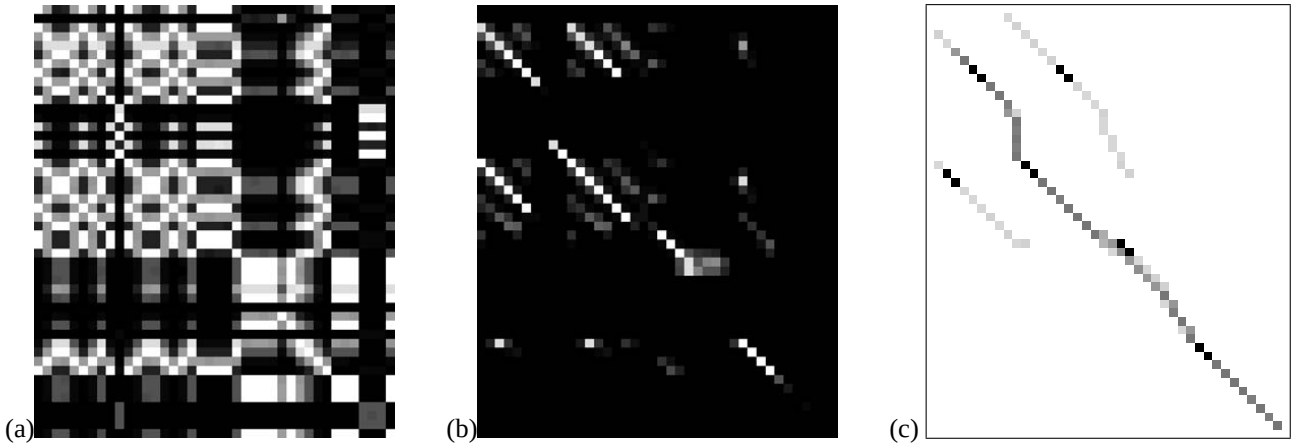

**Figure 4**. *Alouette* example, score *vs* performance, absolute pitch similarity measure: (a) one-by-one similarity matrix; (b) similarity matrix (window size 4) the window similarity has been calculated as the product of the single note-to-note similarity values of all notes within in a window); (c) trace of retrieved alignments. In (a) and (b), the bright regions have a high similarity. In (c), the start positions of the traces are indicated by a pair of black squares (one for the left and one for the right direction). Positions included only in a single alignment are coloured in light grey. A dark grey coloured position indicates that it is included in more than one alignment. The slight deviations between the four overlaid traces around the main diagonal (nr 2–5 in Figure 3) are caused by ambiguities of the cost function (*e.g.*, for a series of three perfect matches and a single gap, the cost function is independent of the gap position) and the different directions (left/right) in which they have been passed by the alignment.

(such as suffix-tree-based methods with time complexity $O(m)$ [7]), but it is somewhat unclear whether the underlying indexing techniques can be applied to approximate matching approaches for handling queries with missing or additional notes and arbitrary similarity measures. The MusicBLAST approach is robust against these errors, can be adapted to different similarity measures, and can be used for quantised and live performed input data.

Because of the voice separation (stream segregation) functionality available in midi2gmn [6], MusicBLAST can be applied to non-separated polyphonic data as well as to the single voices (containing notes and chords) obtained from voice separation. First results from analysing the melodic and rhythmic similarity in live performed and mechanical MIDI files showed that MusicBLAST can retrieve significant occurrences of a search pattern and analyse self-similarities within a performance with an adequate response time. A detailed evaluation of MusicBLAST will be conducated in the near future. With the current direction in developing data formats for representing hybrid combinations audio and symbolic information (*e.g.*, MPEG7), the number of databases with symbolic musical data and the need for performance optimised retrieval algorithms should increase even more in future. Given the popularity and success of BLAST in biological sequence analysis and retrieval, we believe that MusicBLAST has substantial potential MIR research and applications.

## 5. REFERENCES


[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[2] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI–BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[3] T. Crawford, C. S. Iliopoulos, and R. Raman. String-Matching Techniques for Musical Similarity and Melodic Recognition. *Computing in Musicology*, 11:73–100, MIT Press, 1998.

[4] R. B. Dannenberg, An On-Line Algorithm for Real-Time Accompaniment. *Proceedings of ICMC 1984*, pages 193–198, 1984.

[5] R. B. Dannenberg and N. Hu. Pattern Discovery Techniques for Music Audio. *Proceedings of ISMIR 2002*, pages 63–70, 2002.

[6] J. Kilian and Holger H. Hoos. Voice separation — a local optimisation approach. *Proceedings of ISMIR 2002*, pages 39–46. IRCAM – Centre Pomdidou, Paris, France, 2002.

[7] K. Lemström, A. Haapaniemi, and E. Ukkonen. Retrieving music — to index or not to index. *ACM Multimedia '98*, 1998.

[8] M. Mongeau and D. Sankoff. Comparison of musical sequences, Computers and the Humanities, 24:161–175, 1990.

[9] B. Pardo, W. P. Birmingham. Improved Score Following for Acoustic Performances, *Proceedings of ICMC 2002*

[10] L. A. Smith, R. J. McNab, and I. H. Witten. Sequence-based melodic comparison: a dynamic programming approach. *Computing in Musicology*, 11:101–117. MIT Press, 1998.