Finding Motifs with Gaps

Abstract

This paper focuses on a set of string pattern-matching problems that arise in musical analysis, and especially in musical information retrieval. A musical score can be viewed as a string: at a very rudimentary level, the alphabet could simply be the set of notes in the chromatic or diatonic notation, or the set of intervals that appear between notes (e.g. pitch may be represented as MIDI numbers and pitch intervals as number of semitones).

An important example of flexibility required in score searching arises from the nature of polyphonic music. Within a certain time span each of the simultaneously-performed voices in a musical composition does not, typically, contain the same number of notes. So 'melodic events' occurring in one voice may be separated from their neighbours in a score by intervening events in other voices. Since we cannot generally rely on voice information being present in the score we need to allow for temporal '*gaps*' between events in the matched pattern. Typically, the magnitude of such a gap (which might be expressed as a maximum time value, or, more probably, as a maximum number of skipped event-time-slots) will be a parameter set by the user. In our mathematical treatment the allowance for gaps in the query and the score being searched is represented by the constant α.

Fig. 1 shows a short example from a musical score in monophonic format in which we attempt to match a pattern y (also known as the 'query') within a music score t (that we will call the 'text'). This pattern can only be matched by allowing gaps of up to two spaces between pitches in the pattern; Note that the matching of the pattern to the score can be actually 'approximate' (see Cambouropolos et al 1999), in that the difference between the pitches and the sequence of musical events could be bounded by a constant δ (for simplicity we set δ to be zero in this example). We can see that there is an occurrence of the pattern in the text, starting at position three because y_1, y_2, y_3, y_4 and y_5 matches exactly x_3, x_5, x_8, x_9 and x_{11}, respectively, with a sequence of gaps $G=(g_1=1, g_2=2, g_3=0, g_4=1)$. g_1 is the number of spaces between the first two matched pitches in the text (i.e. between (x_3=8) and (x_5=3)), g_2 is the number of spaces between the second and the third matched pitches in the text, and son on. Clearly, this is a valid match because all the gaps are less than or equal to two, which was the given gap restriction. If we want to find matches with a gap of up to one, then this match won't be a valid one. Fig. 2 shows a similar example but for δ=1 so that the matched pitches don't need to be exact, an '*error*' of up to 1 is now allowed. Therefore, the first pitch in the pattern (8) matches the third pitch in the text (7) because |8-7|=1 and that is less or equal to our allowed error of one. The second pith in the pattern (3) matches the fifth pith in the text (4) and so on. The sequence of gaps remains exactly the same.

The problem of matching with gaps, can be formally defined as follows: given a musical sequence x (call the 'text') and a motif y (call the 'pattern') find all occurrences of y in x such that $y_i = x_{j_i} \quad i \quad \{1..m\}$, where m is the length of y. Note that y occurs at position j_1 of x with a gap sequence $G=(g_1, g_2, \ldots, g_{m-1})$, where $g_i = |j_i - j_{i+1} - 1| \quad i \quad \{1..m-1\}$. We will consider this problem under a variety of conditions: the motif matching can be either exact or approximate. The gaps can be bounded, unbounded or all the same length. We have design efficient algorithms and implementations of all the above variants.

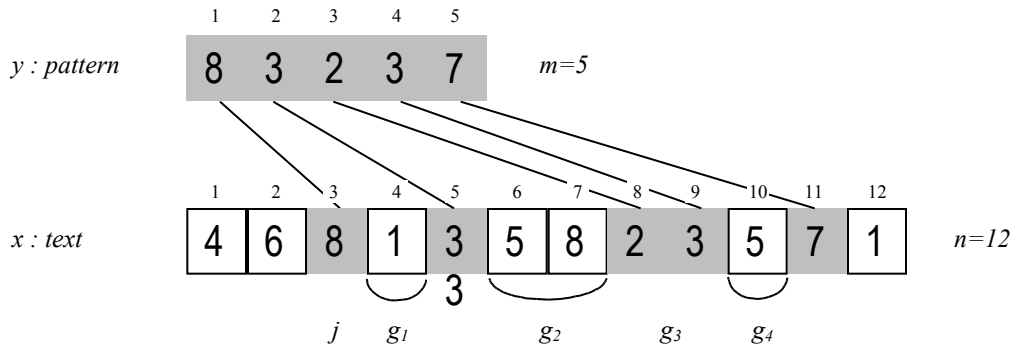

Figure 1 (displayed a δ-occurrence of *y* with α-bounded gaps, for δ=0 and α=2.)

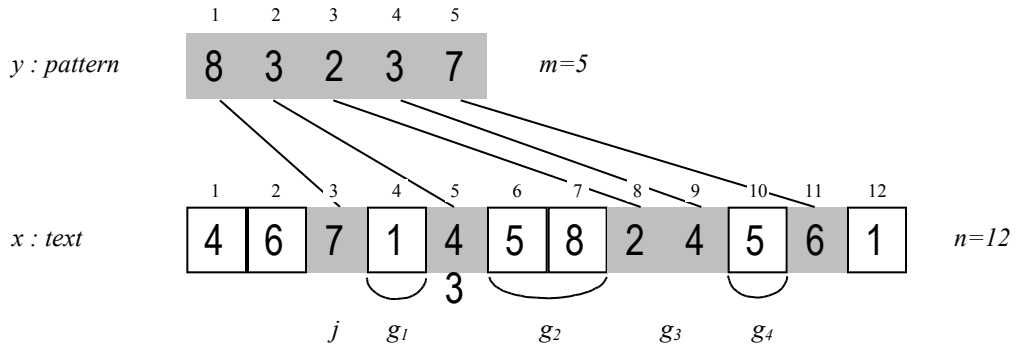

Figure 2 (displayed a δ-occurrence of *y* with α-bounded gaps, for δ=1 and α=2.)

Author Information

Maxime Crochemore
Institut Gaspard-Monge, Laboratoire d'informatique, Université de Marne-la-Vallée
mac@univ-mlv.fr
www-igm.univ-mlv.fr/~mac

Wojciech Rytter
Uniwersytet Warszawski, Banacha 2, 02--097, Warszawa, Poland, and Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, UK.
rytter@csc.liv.ac.uk
www.csc.liv.ac.uk/~rytter

Costas S. Iliopoulos and Yoan J. Pinzon
Dept. Computer Science, King's College London, London WC2R 2LS, UK,
and School of Computing, Curtin University of Technology,
GPO Box 1987 U, WA. Australia
{csi,pinzon}@dcs.kcl.ac.uk
www.dcs.kcl.ac.uk/staff/csi,
www.dcs.kcl.ac.uk/pg/pinzon

Suggested Readings

T. Crawford, C. S. Iliopoulos, R. Raman. 1998, String Matching Techniques for Musical Similarity and Melodic Recognition, Computing in Musicology, Vol 11: 73--100.

E. Cambouropolos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, Y. J. Pinzon. 1999, Algorithms for Computing Approximate Repetitions in Musical Sequences, Proceedings of the 10--th Australasian Workshop on Combinatorial Algorithms, (Eds J. Simpson and R. Raman), Curtin University Press, Vol 3: 114--128.