MADMX: A Novel Strategy for Maximal Dense Motif Extraction

Roberto Grossi\textsuperscript{1}, Andrea Pietracaprina\textsuperscript{2,*}, Nadia Pisanti\textsuperscript{1}, Geppino Pucci\textsuperscript{2,*}, Eli Upfal\textsuperscript{3,*,**}, and Fabio Vandin\textsuperscript{2,*,**,**}

\textsuperscript{1} Dipartimento di Informatica, Università di Pisa, Pisa, Italy
\{grossi,pisanti\}@di.unipi.it
\textsuperscript{2} Dipartimento di Ingegneria dell’Informazione, Università di Padova, Padova, Italy
\{capri,geppo,vandinfa\}@dei.unipd.it
\textsuperscript{3} Department of Computer Science, Brown University, Providence RI, USA
eli@cs.brown.edu

Abstract. We develop, analyze and experiment with a new tool, called MADMX, which extracts frequent motifs, possibly including don’t care characters, from biological sequences. We introduce density, a simple and flexible measure for bounding the number of don’t cares in a motif, defined as the ratio of solid (i.e., different from don’t care) characters to the total length of the motif. By extracting only maximal dense motifs, MADMX reduces the output size and improves performance, while enhancing the quality of the discoveries. The efficiency of our approach relies on a newly defined combining operation, dubbed fusion, which allows for the construction of maximal dense motifs in a bottom-up fashion, while avoiding the generation of nonmaximal ones. We provide experimental evidence of the efficiency and the quality of the motifs returned by MADMX.

1 Introduction

The discovery of frequent patterns (motifs) in biological sequences has attracted wide interest in recent years, due to the understanding that sequence similarity is often a necessary condition for functional correlation. Among other applications, motif discovery proves an important tool for identifying regulatory regions and binding sites in the study of functional genomics. From a computational point of view, a major complication for the discovery of motifs is that they may feature some sequence variation without loss of function. The discovery process must therefore target approximate motifs, whose occurrences are similar but not necessarily identical. Approximate motifs are often modeled through the use of

* Support for these authors was provided, in part, by the European Union under the FP6-IST/IP Project AEOLUS.
** Supported in part by NSF awards IIS-0325838 and DMI-0600384, and ONR Award N000140610607.
*** Contact Author. This work was done, in part, while the author was visiting the Department of Computer Science of Brown University.
the *don’t care* character in certain positions, which is a wild card matching all characters of the alphabet, called *solid characters* [10].

Finding interesting approximate motifs is computationally challenging. As the number of don’t cares increases and/or the minimum frequency threshold decreases, the output may explode combinatorially, even if the discovery targets only maximal motifs—a subset of the motifs which implicitly represents the complete set. Moreover, even when the final output is not too large, partial data during the inference of target motifs might lead to memory saturation or to extensive computation during the intermediate steps.

A large body of literature in the last decade has dealt with efficient motif discovery [9, 12, 4, 16, 8, 6, 5, 2], and an excellent survey of known results can be found in the book [10]. In order to alleviate the computational burden of motif extraction and to limit the output to the most promising or interesting discoveries, some works combine the traditional use of a frequency threshold with restrictions on the flexibility of the extracted motifs, often captured by limitations on the number of occurring don’t cares.

In a recent work, Apostolico et al. [2] study the extraction of *extensible motifs*, comprising standard don’t cares and extensible wild cards. The latter are spacers of variable length that can take different size (within pre-specified limits) in each occurrence of the motif. An efficient tool, called *varun*, is devised in [2] for extracting all maximal extensible motifs (according to a suitable notion of maximality defined in the paper) which occur with frequency above a given threshold \( \sigma \) and with upper limits \( D \) on the length of the spacers. *Varun* returns the extracted motifs sorted by decreasing z-score, a widely adopted statistical measure of interestingness. The authors demonstrate the effectiveness of their approach both theoretically, by proving that each maximal motif features the highest z-score within the class of motifs it represents, and experimentally, by showing that the returned top-scored motifs comprise biologically relevant ones when run on protein families and DNA sequences.

A slightly more general way of limiting the number of don’t cares in a motif has been explored in [13]. The authors define \( \langle L, W \rangle \) motifs, for \( L \leq W \), where at least \( L \) solid characters must occur in each substring of length \( W \) of the motif. They propose a strategy for extracting \( \langle L, W \rangle \) motifs which are also maximal, although their notion of maximality is not internal to the class of \( \langle L, W \rangle \) motifs. As a consequence, the algorithm is not complete, since it disregards all those \( \langle L, W \rangle \) motifs that are subsumed by a maximal non-\( \langle L, W \rangle \) one.

**Our results.** Our work focuses on the discovery of *rigid motifs*, which contain blocks of solid characters (solid blocks) separated by one or more don’t cares. We propose a more general approach for controlling the number of don’t cares in rigid motifs. Specifically, we introduce the notion of *dense motif*, a frequent pattern where the fraction of solid characters is above a given threshold. Our density notion is more flexible and general than the one considered in [10, 2], since it allows for arbitrarily long runs of don’t cares as long as the fraction of solid characters in the pattern is above the threshold. We define a natural notion of *maximality* for dense patterns and devise an efficient algorithm, called