AUTHOR QUERY FORM

ELSEVIER	**Journal: PARCO** **Article Number: 2211**	**Please e-mail or fax your responses and any corrections to:** **E-mail: corrections.esch@elsevier.sps.co.in** **Fax: +31 2048 52799**

Dear Author,

Please check your proof carefully and mark all corrections at the appropriate place in the proof (e.g., by using on-screen annotation in the PDF file) or compile them in a separate list. Note: if you opt to annotate the file with software other than Adobe Reader then please also highlight the appropriate place in the PDF file. To ensure fast publication of your paper please return your corrections within 48 hours.

For correction or revision of any artwork, please consult http://www.elsevier.com/artworkinstructions.

Any queries or remarks that have arisen during the processing of your manuscript are listed below and highlighted by flags in the proof. Click on the 'Q' link to go to the location in the proof.

Location in article	Query / Remark: click on the Q link to go Please insert your reply or correction at the corresponding line in the proof
Q1	Please confirm that given name(s) and surname(s) have been identified correctly.
Q2	Please check the typeset of author names in Ref. [10], and correct if necessary.

Please check this box if you have no corrections to make to the PDF file ☐

Thank you for your assistance.

Highlights

• We present CMS, an algorithm used to search for geometric motifs in proteins. • Complete cross-proteins analysis calls for parallel processing. • Data parallel problem decomposition favors a shared-memory implementation. • OpenMP implementation meets expectations, but scales only up to 8 threads. • Hybrid OpenMP/MPI approach required for further analysis.

Parallel Computing xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Geometrical motifs search in proteins: A parallel approach

Marco Ferretti[1], Mirto Musci*

Dipartimento di Ingegneria Industriale e dell'Informazione, University of Pavia, Via Ferrata 3, 27100 Pavia, Italy

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Proteins
Secondary structure
Motif extraction
Hough transform
Data parallelism
OpenMP

ABSTRACT

The analysis of the 3D structures of proteins is a very important problem in life sciences, since the geometric set-up of proteins has a deep relevance in many biological processes. The complexity of the analysis and the continuous increase in the number of proteins whose 3D structure is known, call for efficient and quick algorithms. Parallel processing is becoming an enabling tool for such research. A key component in the geometric description of a protein is the structural motif, a 3D element which appears in a variety of molecules and is usually made of just a few simpler structures, the secondary structures elements (SSEs).

This paper is an extended version of Ferretti and Musci (2013), and presents the Cross Motif Search (CMS) and the Complete CMS (CCMS) algorithms, two highly optimized and efficient parallel methods to detect the presence and location of all common motifs of secondary structures in a given protein pair (CMS) or across an arbitrary large dataset of proteins (CCMS). The analysis builds on existing approaches, such as Secondary Structure Co-Occurrences (SSC), based on the General Hough Transform (GHT) technique. The main difference between our proposal and the state of the art is the innovative focus that CMS puts on the geometric description of the structural motifs, which could be simply viewed as vectors in a 3D space, rather than on the topological/biological description employed by competing algorithms, such as ProSMoS, PROMOTIF or MASS. The advantage of a geometrical approach is that it enables to retrieve the exact location of the common substructures in a protein pair.

The paper analyzes all possible forms of serial and parallelism optimization of the proposed algorithms, both shared memory and message passing. It introduces a complete parallel implementation of CMS, based on OpenMP, and discusses its scalability on shared-memory architectures. Both small-scale and medium-scale testing shows that the methods produces very interesting results in real applications, and scales nicely up to the eight-processor limit. More in-depth testing also shows that the scalability limit is due to the inner structure of the problem, and that the similarities among proteins and the chosen tolerance for the analysis highly affect the overall performance.

1. Introduction

The analysis of the 3D structures of proteins is a very important problem in life sciences, since the geometric set-up of proteins has a deep relevance in many biological processes. The complexity of the analysis and the continuous increase in

* Corresponding author.
E-mail addresses: marco.ferretti@unipv.it (M. Ferretti), mirto.musci01@universitadipavia.it (M. Musci).
[1] Principal corresponding author.

http://dx.doi.org/10.1016/j.parco.2014.09.007
0167-8191/

the number of proteins whose 3D structure is known, call for efficient and quick algorithms. Parallel processing is becoming an enabling tool for such research. A key component in the geometric description of a protein is the structural motif, a 3D element which appears in a variety of molecules and is usually made of just a few simpler structures, the secondary structures elements (SSEs).

This paper presents the Cross Motif Search (CMS) and the Complete CMS (CCMS) algorithms, two highly optimized and efficient parallel methods to detect the presence and location of all common motifs of secondary structures in a given protein pair (CMS) or across an arbitrary large dataset of proteins (CCMS). The analysis builds on existing approaches, such as Secondary Structure Co-Occurrences (SSC) [2,3] based on the General Hough Transform (GHT) technique [4]. The main difference between our proposal and the state of the art is the innovative focus that CMS puts on the geometric description of the structural motifs, which could be simply viewed as vectors in a 3D space, rather than on the topological/biological description employed by competing algorithms, such as ProSMoS [5,6], PROMOTIF [7] or MASS [8]. The advantage of a geometrical approach is that it enable to retrieve the exact location of the common substructures in a protein pair, with varying degrees of tolerance. With respect to other geometrical algorithm, such as SSM [9], CMS is way more precise, as SSM only gives a similarity score for the geometrical structures of pair of proteins. As we will show, our main goal is to look for previously unknown common geometrical structures in so-called "unfamiliar" proteins. The main shortcoming of our method with respect to the state of the art is performance, however, as precise geometrical approach are inherently slower. Thus, an efficient implementation becomes even more important.

The paper analyzes all possible forms of serial and parallelism optimization of the proposed algorithms. It introduces a complete parallel implementation of CMS, based on OpenMP, and discusses its scalability on shared-memory architectures. Both small-scale and medium-scale testing shows that the method produces very interesting results in real applications, and scales nicely up to the eight-processor limit. More in-depth testing also shows that the scalability limit is due to the inner structure of the problem, and that the similarities among proteins and the chosen tolerance for the analysis greatly impact the overall performance.

The paper is organized as follows: Section 2 discusses the basics of the application background and the terminology for describing proteins in terms of their secondary structure elements; Section 3 introduces the Hough transform approach to geometrical motif identification and details our proposed algorithms and their computational complexity; Section 4 analyzes the possible optimizations of the serial versions of the algorithms, including a greedy version of CMS that reduces computation time up to two order of magnitudes; Section 5 examines the sources of parallelism in the application, both shared memory and message-passing and thoroughly described our OpenMP parallel implementation. In Section 6 we present a performance study of our implementation, and showcase some application results. Section 7 concludes the paper and introduces our next research project, that is a full-scale implementation on a massively parallel systems with a mixed OpenMP/MPI approach.

This paper is an extended version of Ferretti and Musci [1]. Generally speaking the previous contribution was mostly concerned about an old version of the algorithm, the Entire Motif Search, which is briefly treated here in Section 3.3. In more detail, this paper presents the following differences with respect to Ferretti and Musci [1]: (1) the application background is presented more clearly and in more details, for the benefit of he reader; (2) two new algorithms are introduced and discussed, namely Cross Motif Search (CMS) and Complete Cross Motif Search (CCMS); (3) the complexity analysis has been extended to include both the CMS and the CCMS case; (4) more serial optimizations are discussed, including the extremely efficient greedy variant of CMS; (5) the parallel implementation is discussed in much more details; (6) the testing suite has been vastly extended, to present a more sound assessment of the parallel performance of the algorithm; the focus has obviously been shifted from the old EMS algorithm to CMS.

2. Application background

2.1. Protein structure: a hierarchy of descriptions

Proteins are large and complex polymers, and scientists are accustomed to describing their structure at several levels, in order to better understand their spatial arrangement and behavior.

1. The **primary structure** describes a protein as the sequence of all the residues (amino-acids) composing its polypeptide(s).
2. The **secondary structure** describes a protein in terms of its **secondary structure elements** or **SSEs**.
3. The **tertiary structure** describes the overall 3D shape of a single polypeptide, comprising the atoms it is made of and their exact position.
4. For proteins that are made up of more than one polypeptide, the spatial arrangement of the polypeptides (also called protein subunits) is referred to as **quaternary structure**.

For the purpose of our work, we are mainly interested in the secondary structure. A secondary structure element is a small segment of the protein chain which can be identified as a recurring pattern among a multitude of proteins. Each SSE is made of multiple residues shaped in a peculiar 3D structure.

The first two and the most commonly recurring SSEs are the α-helix and the β-sheet as described by Pauling et al. [10].

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

Publicly available protein data is stored inside the Protein Data Bank (PDB) [11], a huge online database which contains a fine grained description of each known protein, down to the relative position of each atom. The characteristics of secondary structures can be extracted from the PDB entries (the proteins) using the Dictionary of Protein Secondary Structure (DSSP) [12]. DSSP is both a dictionary of the secondary structure elements recurring in proteins and the de facto standard algorithm for identifying them in polypeptide chains.

2.2. Structural motifs

Protein folding is not a random process and is subject to a number of physical and chemical constraints. Some recurring "folding patterns" (i.e. small groups of SSEs) have been identified at the secondary level, which are called **super secondary structures** or **structural motifs**. Hairpin-β, Greek Key, and $\alpha\beta\alpha$ are three of the most interesting structural motifs. Since special biological functions are deemed to be tightly connected with these 3D structures, motif pattern searching and discovery has become an important field in the biotechnological domain.

Structural motifs can be defined through either a *topological* description, or a *geometrical* one. The first description is the most widespread, and it is easier to understand for a human (i.e. a biologist). We can say that a topological description is mostly concerned with what kind of connections do exist between each SSE in a motif. In literature there are various tools that deal with topological motifs (e.g. PROMOTIF [7] or ProSMoS [5]).

Our proposals, instead, are tailored to the retrieval of geometrical-defined structural motifs of secondary structures, or simply geometrical motifs. A geometrical motif, instead, is seen as a collection of geometrical-defined elements, such as points or line segments. A geometrical description, albeit somehow cumbersome, allows the retrieval of an **entire new class of protein substructures**, complete with their precise locations, and gives the flexibility to define varying degrees of tolerance in the analysis. Interested readers can refer to [2] to better understand the rationale of this choice.

2.3. Secondary structure representations

A convenient geometrical description for the SSEs is essential when developing a method for the retrieval of geometrical motifs. In fact, a geometrical motif can be simply modeled as a set of geometrical SSEs.

According to DSSP (Section 2.1), each SSE is described by a tagged group of related amino-acids (let us call it DSSP-SSE). As each amino-acid can also be seen as a tagged 3D point, an entire protein can be seen as one or more linked lists (*chains*) of groups of 3D points (*secondary structures*). Each group has various tags (*biological information*).

For the purpose of this paper, most biological parameters are not needed, with the exception of the *string* of amino-acids that compose the SSE, which can be used as a sorting key, as it is explained in Section 4.1. Moreover, the DSSP description is extremely fine-grained, and could be conveniently abstracted. To do that, we apply linear regression to each group of points (that is, to each DSSP-SSE). In this way, it is possible to obtain a line-segment approximation for each secondary structure, as shown in Fig. 1. During this process, not every DSSP-SSE is considered: the structures which are less significant from a motif extraction point-of-view, like turn and coils [13], are discarded. Actually, only *beta sheets* and *alpha helices*are considered. All relevant information is then stored in a convenient XML format.

According to this format, each SSE is modeled through a 3D vector (see Fig. 1), defined by its start point A (in the reference space of the first residue generated by the DSSP algorithm), its end point B, its orientation and its barycenter.

3. Overview of the algorithms

In what follows, we give a description of the algorithms we designed for the extraction of geometrical motifs. In Section 3.1 we describe the theoretical basis of our proposals, the Generalized Hough Transform, while in Section 3.2 we explain how it can be exploited in the motif retrieval domain with the Secondary Structure Co-Occurrences (SSC) algorithm. A first extension of SSC, the Entire Motif Search (EMS) algorithm, is described in Section 3.3.

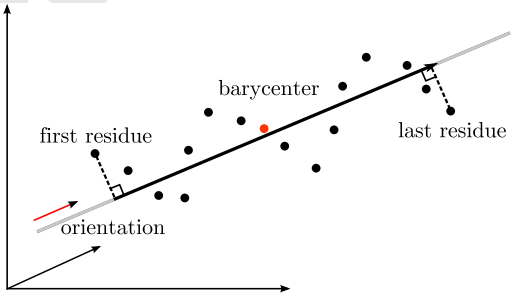

Fig. 1. Linear regression approximation of an alpha helix SSE: from the DSSP point-by-point description to a single line segment description.

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

Sections 3.4 and 3.5 introduce the core of our proposal; that is two new efficient algorithms, based on SSC, which are suited for real, parallel applications of geometrical motif identification and retrieval.

Please note that Sections (3.1)–(3.3) largely draw from Ferretti and Musci [1] and are repeated here for convenience of the reader.

3.1. The Generalized Hough Transform

Our approach for the retrieval of geometrical motifs is based on a well-known pattern recognition algorithm, the Generalized Hough Transform [4].

The GHT uses a model for a geometric pattern consisting of a *reference point* (in a coordinate system local to the pattern) and of a set of *features elements* (points, segments, etc.); these feature elements are stored in the so-called *reference table*, where each entry describes the relative position and orientation of each feature element with respect to the reference point.

The transform (*voting rule*) produces, for each feature element in the *feature space* (the object under scrutiny), a number of candidates for the reference point of the pattern equal to the cardinality of the reference table. These votes are accumulated in a *voting space* and evidence of instances of the pattern is built by the co-occurrences of votes onto the same 3D point. The voting space can be referred to also as *search space*, because it is the space where the search for accumulated peaks is carried out.

The computational complexity of the GHT can be quite relevant, since it depends on the number elements that make up the model, (that is, the cardinality of the reference table), on the number of feature elements present in the feature space to be analyzed, and on the resolution at which the voting space is quantized.

3.2. Secondary Structures Co-Occurrences

The Secondary Structures Co-occurrences (SSC) [2,3] and the Secondary Structures Triplets (SST) [14] are two algorithms, based on the GHT, which search for geometrical motifs (*patterns*) of SSEs (*feature elements*) inside a given protein (*search space*). The SSC algorithm uses pairs (co-occurrences) of SSEs, while SST uses terns. In both cases, the reference table for a motif describes a set of pairs (terns) of SSEs in a 3D space and the voting rule is built using the geometric features of the line segments that approximate the SSEs.

If a motif instance is indeed present in the protein, the transform yields an accumulation of votes that peaks in a 3D location corresponding to the reference point for that instance; the peak of votes amounts to the number of rows of the reference table. Let us consider a simple case: a protein motif composed of four SSEs, as show in Fig. 2 (a 2D case, easily generalized to 3D). In the SSC algorithm, one has to account for all possible $\binom{4}{2}$ pairs produced by the four SSEs. In this case there are six combinations of pairs, namely: AB, AC, AD, BC, BD and CD (reverse pairs, such as BA, are not considered, as will be explained later in Section 4.1). So the reference table contains exactly six rows, one for each pair, and the voting space will show a peak of six accumulated votes (one from each pair), for all instances of motifs actually present in the protein.

The searching process consists of two phases: initialization and voting. In the initialization phase, one chooses a *reference point* (RP), builds a *reference table* (RT) for the current motif, using convenient geometric parameters (e.g. the angle between two line segments), and sets up a voting rule (i.e. a conveniently defined transformation matrix): the properties describing the pairs (or the terns) that make up the model of the motif are *independent* both of *rotation*and of *translation*, a feature which gives great strength to this approach.

In the voting phase, one examines every secondary structure in the feature space and retains only those that have a geometric match with the SSEs of one of the rows inside the RT. For each such match, the mapping rule is applied to the selected structure in the protein and evidence (votes) is accumulated for a candidate reference point. Once the whole set of SSEs in the object (protein) have been considered, peaks inside the voting space are identified (the voting space is an array of 3D points).

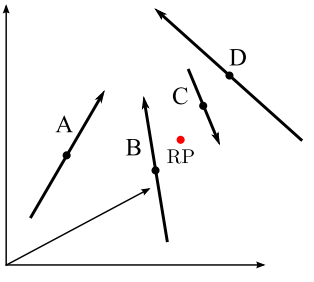

Fig. 2. Geometric representation of a motif of SSE as a set of line segments.

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

3.2.1. Complexity analysis

The computational complexity of the SSC algorithm is estimated as follows. In a motif with n SSEs, the number of pairs, that is the number of rows in the reference table (Section 3.2), is $\binom{n}{2}$; the number of pairs in a protein with N SSEs is $\binom{N}{2}$. So the cost C_{SSC} of $SCC(n,N)$ is:

$$C_{SSC}(n,N) \sim n^2 \times T_1 + n^2 \times N^2 \times T_2 + n^2 \times T_3 = O\left(n^2N^2\right) \tag{1}$$

where T_1 is the time required to compute a single entry in the reference table, and T_2 the average time for a single comparison and for the vote accumulation. The third term accounts for the analysis of the voting space and the search for accumulation points (i.e. clustering): its contribution to the overall complexity is kept to a reasonable quantity, only by using an approximated version of the algorithm, since the exact implementation can be mapped to a maximum clique problem on graphs, and thus would be exponential in N [15]. Asymptotic complexity gives a quantitative assessment for all problem instances; in this application, however, $n \ll N$; so that the cost of the algorithm is largely dominated by the voting phase (the second term in the equation).

The equivalent equation for SST is

$$C_{SST}(i,N) \sim n^3 \times T_1' + n^3 \times N^3 \times T_2' + n^3 \times T_3' = O\left(n^3N^3\right) \tag{2}$$

In the rest of the paper, we disregard the SST algorithm; its computational complexity is too high with respect to its presumed benefits in the precision of the motif retrieval.

3.3. Entire Motifs Search

SSC allows to search for a *single* motif in a *single* protein. Building on it, one can devise a first true, yet limited in scope, application: searching for *all* possible motifs instances in a single protein. We call this extended algorithm Entire Motifs Search (EMS). The complexity of this algorithm is established by defining the maximum number of SSEs contained in the motifs to be searched (n_{max}). In all practical cases, $n_{max} \ll N$, even is the range of values for n_{max} extends up to $N-1$.

Given a protein P, with N SSEs, the EMS requires running $M = \sum_i^{n_{max}} \binom{N}{i}$ SSC instances on the same search space (P itself), where $\binom{N}{i}$ is the number of all possible motifs of dimension i in P and thus M is the total number of geometrical motifs of interest.

So, the cost for EMS can be estimated as follows:

$$\sum_{i=3}^{n_{max}} C_{SSC}(i,N)\binom{N}{i} = O\left(N^2 \sum_{i=3}^{n_{max}} i^2 \binom{N}{i}\right) = O\left(N^2 n_{max}^2 \sum_{i=3}^{N} \binom{N}{i}\right) \tag{3}$$

As $N > n_{max}$, we obtain:

$$C_{EMS}(N) = O\left(N^4 2^N\right) \tag{4}$$

Cases $i = 1$ and $i = 2$ are excluded, as they are either nonsense or trivial. Obviously, a fixed value for n_{max} yields a complexity of $O\left(N^2 \sum_i \binom{N}{i}\right)$.

The strategy to attack the huge computations that arise, even in a single protein analysis is twofold: minimizing the cost of a single motif search (mainly with a preliminary serial optimization) and distributing the work due the combinatorial factor $\binom{N}{i}$ across multiple computational units.

3.4. Cross Motif Search

The EMS is a powerful proof-of-concept algorithm, but it is useless in real applications, since it does not allow to discover new occurrences of the same geometric pattern in new proteins. A further extension of the algorithm is needed, which should be able to work with *different* proteins, and thus to actually search for previously unknown geometrical motifs. We call this extension Cross Motif Search, or CMS.

First of all we need to modify the SSC itself so that it can work with an arbitrary template, that is a motif which is not *a priori* known to be part of the search space. The first phase (the creation of the reference table) remains unmodified. However, in both the voting and accumulation phase, it is now necessary to account for *tolerances*. Indeed, in the modified SSC, two pairs of SSEs (taken from different proteins) produce a vote if their geometric parameters are "*close enough*". A similar reasoning can be extended to the accumulation of vote points, which is now quite sparse and should be searched within a

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

certain tolerance radius. The problem of choosing adequate tolerances its outside the scope of this paper. However it is clear that this extension introduces a new source of computational complexity into the algorithm.

With the above definition, it is easy to describe the CMS algorithm itself. Given two proteins P_1 and P_2 composed respectively by N_1 and N_2 SSEs, the CMS algorithm applies the modified SSC to every $\sum_i^{n_{max}} \binom{N_1}{i}$ candidate motif of P_1 on the search space P_2. It can also be proved that, for the chosen geometric parameters, a CMS run on (P_1, P_2) yields the same result as a CMS run on (P_2, P_1), as it would be expected.

It can be easily shown (similarly to what has been done in Section 3.3) that the CMS asymptotic complexity is:

$$O\left(N_1^2 N_2^2 \sum_{i=3}^{n_{max}} \binom{N_1}{i}\right) = O\left(N_1^2 N_2^2 2^{N_1}\right) \tag{5}$$

Keep in mind, however, that the actual execution times for CMS runs are higher than comparable EMS runs and depend in a non-trivial way on the choice of the tolerance parameters. *Looser* tolerances, of course, produces many more votes, and thus need more matrix manipulations (the computation of the voting rules). Moreover, CMS can be further extended to work on each pair of proteins in an arbitrary large set of proteins, up to the whole Protein Data Bank. We call this extension Complete CMS, or CCMS.

From the above discussion it is clear that both the algorithm and its implementation need to be highly optimized if the CMS is going to be used in any real application. A first version of the CMS employs a *brute-force* approach; that is it checks every possible motif in P_1, even motifs that can not possibly return a positive result due to geometrical considerations, and it does not employ any strategy to reduce the size of the search space. Fortunately it is possible to devise a much faster variant of the CMS algorithm, which is described in Section 4.3, which applies a greedy strategy and greatly reduces overall complexity.

3.5. Complete Cross Motif Search

The generalization of CMS is running the algorithms for each pair of proteins in a given set: formally, given a set S of proteins P, the *Complete Cross Motif Search* problem on S, that is $CCMS(S)$ is the execution of $CMS(P_i, P_j),\ \forall (P_i, P_j) \in (S \times S)$.

The definitive application of CCMS would be analyzing the whole PDB, looking for similarities at the motifs level between proteins pairs, thus setting S to match the PDB.

The amount of computations associated with this tasks is very large, since the cardinality of the PDB is currently in the order of 10^5. Moreover, researchers continuously add new entries to the PDB, so that an efficient method to update the analysis of similarities is also required. Section 5 will discuss the parallel approaches to tackle this tasks. From the application point of view, one can note that the complexity of this application can be reduced since the PDB can be structured in a hierarchy way: for instance, the SCOP [16] database introduces the notion of classes, domains, super-families, families etc. The lower a level in the hierarchy, the higher the similarity of any pair of proteins belonging to the level. According to this categorization, there are 1962 super-families,[2] and proteins belonging to the same super-family share a very large section of their secondary structure. Thus a reduced complexity instance of CCMS would be obtained by selecting S to match a proper subset of the whole PDB, that is by choosing just a single representative for each super-family of proteins. This avoids wasting a lot of time in running instances of $CMS(P_k, P_l)$ where P_k and P_l are very similar to each other: one is likely to find little new information in terms of previously unknown structural motifs if two proteins share the large majority of their 3D structure.

Even with this conservative approach, a $CCMS(S)$ with $|S| = 1962$ does indeed require a carefully designed parallelization strategy.

4. Optimizations and a greedy approach

The combinatorial nature of the algorithms described so far calls for optimizations that can reduce the complexity of the algorithm. Here we present two such optimizations and a greedy version of the CMS algorithm. The first optimization use a sorting technique to avoid duplicate computations for symmetric pairs of SSEs. The second optimization uses a caching policy to speedup the computation of the geometric parameters. While neither optimization changes the asymptotic behavior of the algorithm, they greatly reduce the constants of Eq. (1), and have a significant effect on the performance.

Finally, we introduce an algorithm modification (the *greedy*variant), which is able to greatly reduce the average execution times of CMS. The greedy variant is able to prune the search space of each CMS iteration thanks to several geometric considerations. As we will see later, the greedy algorithm has been used in the final parallel implementation.

4.1. Pre-sorting

Previously (Section 3.2) we assumed that SSC requires to create an entry of the reference table for each of the two permutations of a given motif pair. I.e. if we consider the pair AB composed of the SSEs A and B, then we should also consider the

[2] http://scop.mrc-lmb.cam.ac.uk/scop/count.html

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

pair BA in the whole process. The reason is that voting requires a unique and unambiguous voting rule, but the computation of the geometric parameters depends on the relative order of the SSEs.

Luckily, there are two techniques which allow to ignore the permutations of each SSE pair, and thus allow to reduce the size of the reference table, the computations and the number of votes. The first one is to redefine the voting rule, so that it become independent from the permutations of the pairs. This approach, however, increases both T_1 and T_2. While the time increase in the creation of the reference table (T_1) is largely inconsequential, the increase in the comparison time (T_2) significantly degrades the performance. The second method is to order the set of the SSEs so that they appear always in the same relative position in both source and search space. This can be obtained by enumerating and *pre-sorting* the SSEs in the protein before a motif is selected. Of course, pre-sorting increase the overall complexity, in a way that depends on the choice of the sorting algorithm. For instance, using quick-sort, the complexity of EMS becomes:

$$C'_{EMS}(n,N) = O(N\log N) + \sum_{i=3}^{n} C'_{SSC}(i,N)\binom{N}{i}. \tag{6}$$

.

The added cost due to this *global* sorting (an order of magnitude lower than the dominant term) is paid only once, since sorting is performed off-line, and thus the performance in almost unaffected. The positive side-effect, however, is that T_2 is *halved*; and this improvement pays off for all the $\binom{N}{i}$ combinations.

The strategy used to sort the SSE can have either a geometric, a biological (e.g. we could sort the string of amino-acids of each SSE in a lexicographic order) or a mixed key; it is however required to be *consistent* between source and search proteins. Lack of consistency would surely lead to missed matches in the voting phase. While the definition of the EMS obviously implies consistency for any sorting key (as the source protein and the search proteins coincide), consistency is hard to prove in general for the CMS algorithm.

To avoid the complex issue of studying the consistency of all possible global sorting keys, it is possible to tackle the problem from another direction, and recur to *local* pre-sorting. According to this approach each pair of SSEs is sorted internally during the creation of the reference table, to guarantee a unique mapping rule. This amounts to a negligible increase in T_1 (Eq. (1)) and to an overall performance boost. In the final implementation we choose a geometric key based on the relative lengths of the SSEs. The only caveat is the case of very similar SSEs: if the difference in length is less than or equal to the chosen tolerances (machine tolerance in the EMS case) then we must use secondary or tertiary geometrical sorting keys.

4.2. Caching geometric parameters

In a straightforward implementation of the algorithm, the geometric parameters of each pair of SSEs are computed *online*, directly from the XML description, each time the structures are considered. This is clearly a waste of resources. A significant saving in time (on term T_1 but mostly on term T_2 of Eq. (1)) is possible by storing partial results associated to each pair of SSEs in a special-purpose cache: before searching on each single protein, this cache is precomputed completely. The serial implementation of the algorithm benefits in the speed-up in the range 20–40%, according to the protein size.

Moreover, a closer examination of the algorithm shows that, for most SSE pairs, the local voting rule is needed more than once. To improve performance even more, it is possible to pre-calculate such values and store them in the shared cache, together with the geometric parameters. However, in the CMS case, the voting rule is needed just for some SSE pairs, as not every one of them produces a vote. The chosen solution is to calculate the value of the voting rule only *on demand*: we avoid unnecessary pre-computations, but still optimize reuse.

Analyzing the use of the cache just introduced, one can identify an inefficient pattern of accesses. In fact, due to the pre-sorting optimization, the cache is an upper triangular matrix but it is still accessed by rows. To greatly improve both time and space locality of accesses, the matrix can be transformed into a single-index array. This new index is obtained by using the

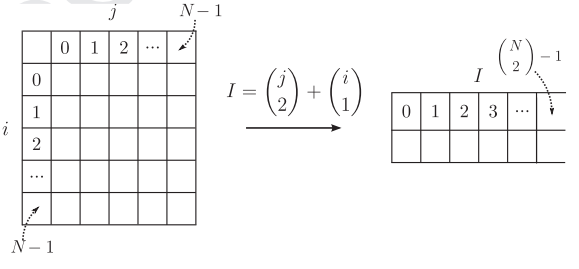

Fig. 3. Linearization in the cache: from matrix to 1D array. The goal is to improve memory usage and access patterns. Each entry in the matrix stores the geometrical parameters of one of the $\binom{N}{2}$ pair of SSEs in a protein with N SSEs.

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

combinatorial number system by Knuth [17] (see Fig. 3). This *linearization* can also lead to better parallel performance, as will be shown in Section 5.3.

4.3. A greedy variant of the algorithm

In the EMS case, there is no need to devise a strategy to reduce the search space. In fact, as the goal of the EMS is an *exhaustive* search for every possible motif of a given dimension n, the algorithm will eventually cover the search space in its entirety. If considering CMS, however, the situation changes, as we are only interested in the common structures between two different proteins (see Section 3.4).

Given two motifs S and M, we say that S is a sub-motif of M ($S \subset M$) if the SSEs in S are a proper subset of the SSEs in M. The key idea is that a motif of dimension n is present in the search space if and only if all its sub-motifs of dimension $n-1$ are also present. Starting from this observation, it is possible to greatly reduce the size of the search space, pruning all the SSC calls which would lead to useless calculations, and *greedily* searching the most promising motif templates.

It can be show that the asymptotic complexity of the greedy-CMS is worse than the CMS one (Eq. (5)):

$$O\left(N_1^3 N_2^2 2^{N_1}\right) \tag{7}$$

However, we showed [15] that the average execution times on a real data set are up to two order of magnitude better than a *brute-force* implementation. Another advantage of the greedy variant, is that the n_{max} value introduced in Section 3.3 has no longer any significance, as the algorithm greedily searches for the structural similarities with the largest size, up to entire tertiary structures. Moreover, note that as the CMS extension added a non-trivial dependence of the execution times on the tolerance parameters, the greedy variant adds an even more complex dependence on the similarity between the source and the search proteins. Tentatively, as a P_1 becomes more similar to P_2 according to some biological metrics (e.g. see [13]) the greedy performance worsen, as predicted by the complexity analysis.

A more detailed discussion is outside of the scope of this paper. For further information, see [15].

5. Parallelism analysis

The available sources for parallelism in the algorithms introduced previously are best explained with some visual help. Fig. 4 shows the decomposition of a CCMS problem into the correspondent CMS and SSC algorithms. The decomposition works on three levels:

- level 1: given a set S of proteins P_i, each sub-task performs a single $CMS(P_i, P_j)$ problem; the number of such sub-tasks is $\binom{|S|}{2}$;
- level 2: each $CMS(P_i, P_j)$ task generates $K = \sum_{c=3}^{n_{max}} \binom{N_i}{c}$ instances of a $SSC(M, P_j)$ problem; that is one instance for each possible candidate motif M built with up to n_{max} SSEs among the N_i that make up protein P_i. Each ~~M~~is retrieved in the same search space, that is protein P_j. The data of P_j used by each instance of SSC is organized in the special data structure described in Section 4.2 and is actually a series of items that describe the geometrical features of each SSE pair of P_j;
- level 3: each $SSC(M, P_j)$ looks for the SSEs that make up M within the whole search space P_j, that is it builds a reference table of M and ~~cast~~ votes for each pair of SSEs of P_j, by using the Hough voting process described in Section 3.2.

We can anticipate at the outset that level 3 offers too small a granularity of computations for a useful parallel implementation; that level 2 is actually the one for which an OpenMP implementation is viable and has been carried out; and that level 1 shows a coarse-grain structure of independent computations, that will be implemented in future works with a message

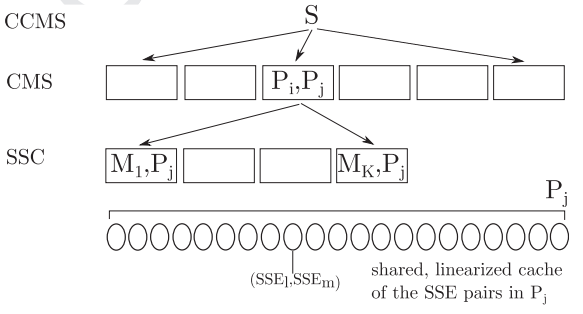

Fig. 4. A coarse-level decomposition of the CCMS algorithm.

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

passing approach. In what follows, we briefly discuss levels 3 and 1, and offer a detailed description of the OpenMP solution for level 2.

The EMS problem described in Section 3.3 is not analyzed here in its parallel implementation. A preliminary study is reported in Ferretti and Musci [1]; we have not pursued this algorithm any further because it has very little relevance for true applications.

5.1. Parallelizing SSC

With reference to Fig. 4, the SSC algorithm can be implemented in parallel using a data-parallel approach, that is partitioning the search space (the SSEs pairs in the search protein P_j). According to this approach, each thread can perform the GHT voting phase on its data-chunk using a shared reference table for M_k. In an hypothetical OpenMP implementation, data distribution to threads could be done with a fixed-size chunk and a static allocation.

Partitioning is quite easy, as the search space is first linearized (using the combinatorial number system, see Section 4.2), and then evenly divided among threads. In this way, one can obtain near-perfect load balancing. However, the granularity of the computation is quite small, and the exponential terms in Eq. (3) and Eq. (5), tend to amplify even a relatively small overhead.

5.2. Parallelizing CCMS

As for the CCMS algorithm at the opposite scale of granularity, each CMS problem can be carried out independently from the others, and each has a consistent computational load. The data structures used by each CMS task are separate; so, the algorithm lends itself to an implementation based on multiple, separate computational units. Furthermore, the amount of inter-process communication is minimal (if not null at all). At the end of the computation, the results, that is the positions of the reference points of all motifs and the indexes of the secondary structures which constitute them, are to be collected by a master node in a convenient XML format. The amount of output data is significantly higher than the amount of input data, so this may represent a bottleneck for the application.

These characteristics call for a parallelization on a systems consisting of fairly powerful nodes, with loose interconnections, and no need for memory sharing; possibly, each computational node could itself be a parallel subsystem, with multiple processors, provided this subsystem is well suited to running the CMS instances it receives. The description of the OpenMP implementation of the CMS algorithm in Section 5.3 will corroborate this statement.

At the coarse level of the CCMS problem, however, partitioning and load balancing can be the real problem. Partitioning amounts to subdividing the set S of proteins in all couples and assigning them in chunks to each computational node. This partitioning depends on n_{max}, the maximum size of motifs to be searched: if this number grows, the optimal number of proteins pair per computational node approaches one, leading to dispatching one protein pair to each computational node. Due to the structure of proteins, the number N of SSEs in a protein can vary significantly, even with a fixed value for n_{max}. To obtain a good chunk of protein pairs to be dispatched, either a priory knowledge has to be gained on the proteins, so that a proper pairing can be prepared off line, or a dynamic dispatcher has to be devised, that is able to feed computational nodes with proper new work. One can obtain load balancing using progressively shorter (or longer) computations across all computational units. Rounds of smaller proteins can also induce a different partitioning, where it is convenient to dispatch multiple protein pairs at the same time to the same computational unit.

The implementation of the CCMS algorithms on a coarse-grain parallel system, and the associated dispatcher, with an MPI approach is currently undergoing refinement, and is not yet reported here.

5.3. Parallelizing CMS: an OpenMP approach

The second level of the decomposition in Fig. 4 is at the execution of all SSC instances of a given CMS problem. In what follows, we describe both the general approach and the actual implementations of the naive (i.e. brute force) version of the algorithm and of the greedy variant introduced in Section 4.3. The description is given with reference to Fig. 5.

A CMS problems requires building the set of all possible motifs M in protein P_i, and searching each of them in the target space P_j. The search involves calling a sequence of SSC instances, assigning each a single motif M and having the SSC serially analyze P_j to look whether M is present or not in P_j. This general brute force scheme is depicted in Fig. 5(a). This figure is deceptively simple: each rectangle represents a motif, that is a set of SSEs, and rectangles are logically grouped according to the number of SSEs they contain. The dotted oval shapes that show this grouping should not imply that the number of motifs for a given value of n is constant in protein P_j. On the contrary, proteins exhibit a large fluctuation in such a number.

A straightforward OpenMP parallelization can use a work-sharing policy based on a fixed chunk-size and a dynamic distribution: each thread receives its chunk of motifs and instantiates the required SSC serial processes that carry out the work on each motif. All SSC instances use the same shared linearized cache of target couples of SSEs in P_j for executing their internal processing. Since the linearized enumeration of couples of SSEs in P_j is also used to generate the sequence of couples which make up the motifs M in P_i, a perfect match is obtained between the iteration that controls the scanning of P_i and the access to the linearized target cache. This apparently simple strategy works well if the load on each thread is well

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

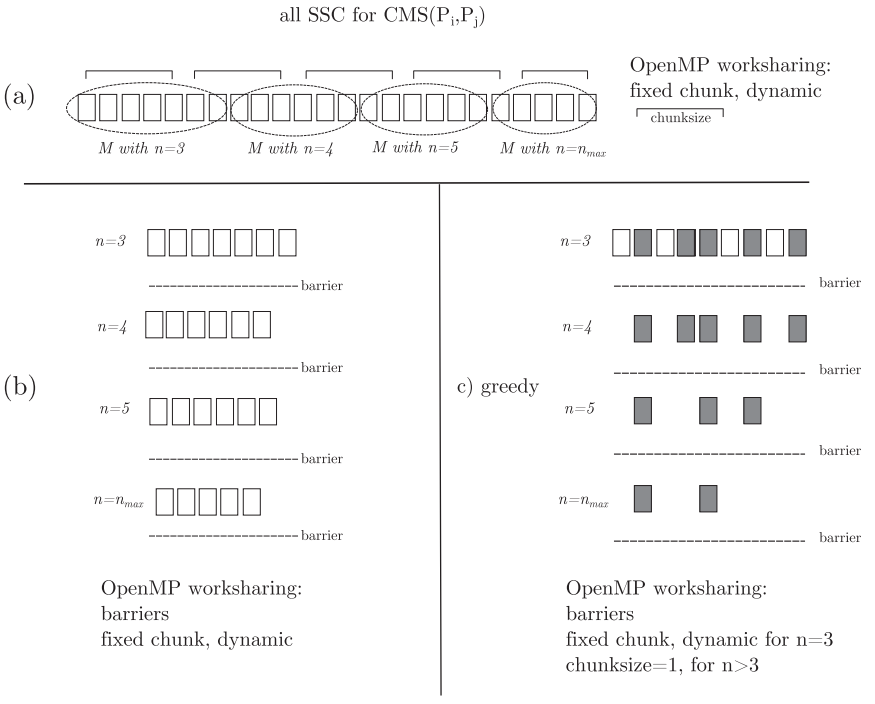

Fig. 5. (a) The general OpenMP decomposition in the brute-force approach; (b) a barrier serialization of tasks handling motifs of the same cardinality; (c) the greedy approach.

balanced; and this is not guaranteed if a thread receives motifs with different number of SSEs. Owing to the just mentioned variance in proteins structures, this is quite common.

Fig. 5(b) shows a second implementation of the brute force approach, actually the one that has been coded and for which experimental results will be described in Section 6. In this case, OpenMP barriers are used to synchronize threads that work on motifs of the same size in SSEs. Data distribution still obeys the fixed chunk, dynamic policy, and this structure optimizes the use of resources for a given value of n.

Fig. 5(c) depicts the parallel implementation of the greedy variant of the CMS algorithm. As described in Section 4.3, the brute force approach wastes a lot of computational resources looking for motifs of size n from scratch, without considering the results of the search for motifs of size $n-1$. In the figure, dark rectangles in level $n=3$ are those for which a match has been found in P_j and motifs for $n=4$ are only examined if they contains all these successfully matched motifs of size $n=3$. The process is iterated for following values of n.

The OpenMP implementation still uses barriers, that in this case are **mandatory**, and perform a function completely different from that in the brute form approach: the barrier is required so that threads working on n do wait for completions of threads working on $n-1$. The work-sharing policy adopted for the greedy CMS is different; while for $n=3$ the fixed chunk, dynamic policy is still valid, for subsequent processing a chunk-size of 1 proves to be better from the view point of load-balancing.

This is a high level description of the OpenMP version of the greedy algorithm; the barrier construct is not the only one required for proper synchronization of threads, since other mutual exclusion constructs are required for accessing other shared resources. For instance, a shared map is needed to preserve the information of the motif that the "greedy threads" are constructing, so to avoid repetitions and potential conflicts. Another example, is the semaphore which regulates writing access on the shared cache of P_j. All in all, these synchronization operations are likely (but not overly critical) sources of overhead.

6. Testing and results

We developed a complete OpenMP implementation of the algorithms discussed above. The source code is available on demand, for research purposes. In [1] we presented a suite of test which showed the performance of the EMS algorithm (Section 3.3) on a small database on a desktop machine. We will not repeat those results here, as they are not very relevant anymore. In this extended paper we will describe a new suite of tests, which will ~~shows~~ the capabilities and the performance of both the CMS and the CCMS algorithms (Sections 3.4 and 3.5).

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

Table 1
CCMS, number of common substructures found. Strict tolerances: midpoints distance 0.5 Å; line distance 0.5 Å; angle 5°; cluster 1.0 Å. Loose tolerances: midpoints distance 1.0 Å; line distance 1.0 Å; angle 7°; cluster 2.0 Å.

Tolerances	Cardinality	Number of common substructures found
Strict	3	1026
	4	793
	5	383
	6	105
	7	12
	8	0
	9	0
	10	0
Loose	3	2481
	4	3489
	5	3546
	6	2642
	7	1506
	8	656
	9	208
	10	42

All testings were performed on a small server with four Intel Xeon E5-4620 CPUs (4×8 cores, hyper-treading disabled, 252 GB of RAM) and the latest stable Debian distribution. If not specified, all results are assumed to be the average results of ten different runs.

6.1. Capabilities assessment

As we said in the introduction, it is meaningless to compare our results with the ones of other motif retrieval algorithms, such as ProSMoS, PROMOTIF, SSM or MASS, as they have a completely different nature, and different goals. However, we can craft an application example for our motif extraction proposal. As we have shown in the Section 3.5, a CCMS run can be performed on any set of proteins: as our OpenMP parallelization operates on the CMS level, the size of the data-set does not have any effect on performance values, other than the global execution time.

To show the capabilities of the CMS algorithm, several CCMS runs have been performed on a small protein data set [3] by using two different configurations of CMS. The first configuration employs *strict* tolerances, while the second configuration employs *loose* ones. These two concepts are intentionally left vague, as a detailed analysis of tolerance is outside of the scope of this paper. The same data set has been used in [15] to show the advantages of the greedy approach.

The goal of this suite of tests is threefold: (1) give an idea of the number of common substructures that can be identified thanks to the CCMS algorithm; (2) evaluate the effect of a small variation of tolerance on the final results; (3) show what happens if "familiar" proteins are kept in the data-set.

Table 1 reports a histogram of the common geometrical sub-structures that have been found, grouped by their cardinality. The greedy variant has been employed, and the maximum cardinality of the common motifs being searched has been set to 10.

First of all we can note how much a small difference in the tolerance values has an impact on the number of extracted motifs. Unfortunately, there is no simple formula to describe a behavior like this, as it solely depends on the structural nature of the protein in the data-set.

The second things that we can note is the outstandingly high number of common structures being found. However, these values are biased by the choice of the protein set. In fact, the set contains pairs of proteins that are very similar to one another: PDB entries `2qmy`, `2qx8` and `2qx9` are basically the same protein.

As already said, CMS is not a structural alignment algorithm; it performs an exhaustive enumeration of the common geometrical motifs between two proteins. When two proteins are very similar to one another, there is no point in using CMS to have a list of all the common geometrical structures, because the outcome would be an enormous number of not very useful results. For example, if we apply CMS to a pair of *identical* proteins having 100 SSEs each, the number of common motif being found would be almost 2^{100}! In fact, processing the pair (`2qmy`, `2qx9`) takes 2.050 s, whereas processing the pair (`7fab`, `2qx9`) takes only 0.079 s (greedy variant, loose tolerances, cardinality from 3 to 5), even if `7fab` contains 10 more SSEs than `2qmy`. Table 2 shows the result of the same set, obtained just by removing the proteins `2qx8` and `2qx9`.

The conclusion we can draw is that CMS should be used with pairs of proteins that are not similar to one another (this can be assessed by using a structural alignment tool like PDBeFold or looking at their SCOP [16] classification). Its main draw is

[3] PDB entries `1fnb`, `2prl`, `2pzn`, `2qmy`, `2qx8`, `2qx9`, `2z7g`, `2z7k`, `2z98`, `2z9b`, `2z9c`, `3c3u`, `3c94`, `3c95`, `3dc7`, `3dhp`, `3e9l`, `3e9o`, `4gcr`, `7fab`.

Table 2
CCMS, number of common geometrical substructures found (removing `2qx8` and `2qx9` from the protein set). Strict tolerances: midpoints distance 0.5 Å; line distance 0.5 Å; angle 5°; cluster 1.0 Å. Loose tolerances: midpoints distance 1.0 Å; line distance 1.0 Å; angle 7°; cluster 2.0 Å.

Tolerances	Cardinality	Number of common substructures found
Strict	3	170
	4	79
	5	17
	6	1
	7	0
	8	0
	9	0
	10	0
Loose	3	491
	4	431
	5	237
	6	68
	7	9
	8	0
	9	0
	10	0

indeed the capability to detect common structures that are not easy to identify with unaided eye, and are not considered significant by alignment tools and topological algorithms.

As a final remark, please note that the result of a CCMS is completely deterministic: given the same configuration, the number of common substructure is constant every time. Moreover, CMS (and thus CCMS) are completely symmetric. That is $\mathrm{CMS}(P_i, P_j) = \mathrm{CMS}(P_j, P_i)$.

6.2. Performance assessment: greedy-CMS

The goal of the second batch of tests is to assess the performance of the OpenMP implementation of *greedy-CMS* (see Section 4.3) on a real machine, with special regards to the multi-core scalability of the implementation. It is important to understand that even if we are able to achieve **almost perfect speedups and scalability** for a brute-force implementation of CMS, it would be still completely unpractical due to the high execution times (see [15]). So again, we are only interested in the performance of greedy-CMS.

The testing database is larger than the previous one (200 proteins) and it has been chosen as follows. As we are mainly interested in looking for previously unknown geometrical motifs and thus structural similarities, we chose to perform the tests on proteins from different protein "*types*". More precisely, the database includes the first (lexicographic order) proteins listed from the first (yet again lexicographic order) 200 *super-families*, as defined in the SCOP database [16]. A complete list of all the proteins in the test database can be found in Table 3.

Table 3
The complete list of all 200 proteins in the testing dataset.

101m	102l	10gs	10mh	117e	11as	11ba	121p	12ca	12e8
1a0a	1a0b	1a0c	1a0g	1a0i	1a0j	1a0k	1a0n	1a0o	1a0p
1a1u	1a1w	1a1x	1a21	1a22	1a25	1a26	1a27	1a28	1a2a
1a31	1a32	1a33	1a34	1a36	1a37	1a39	1a3a	1a3c	1a3h
1a4h	1a4i	1a4l	1a4s	1a4y	1a50	1a59	1a5c	1a5e	1a5j
1a6c	1a6d	1a6f	1a6i	1a6l	1a6q	1a6r	1a6s	1a6x	1a70
1a7w	1a7x	1a80	1a85	1a8a	1a8d	1a8h	1a8i	1a8m	1a8p
1ab4	1ab8	1abe	1abr	1abv	1aca	1acb	1aci	1acm	1aco
1af0	1af2	1af5	1af7	1afa	1afc	1afh	1afi	1afv	1afw
1ahs	1ahu	1ai9	1aig	1aip	1aiq	1air	1ais	1aiw	1aj3
13pk	155c	16vp	19hc	1a02	1a03	1a04	1a05	1a06	1a07
1a0r	1a12	1a14	1a15	1a16	1a17	1a18	1a19	1a1m	1a1s
1a2f	1a2k	1a2n	1a2o	1a2p	1a2q	1a2t	1a2v	1a2z	1a30
1a3s	1a3w	1a3z	1a40	1a43	1a44	1a45	1a47	1a48	1a4e
1a5k	1a5r	1a5t	1a5v	1a5y	1a5z	1a62	1a67	1a68	1a69
1a73	1a76	1a79	1a7c	1a7d	1a7g	1a7k	1a7t	1a7u	1a7v
1a8r	1a9n	1a9x	1aa1	1aa2	1aa3	1aa6	1aa7	1aab	1aam
1acp	1acx	1ad1	1ad2	1ad6	1adj	1adr	1adu	1aep	1aew
1ag1	1ag2	1ag9	1agm	1agr	1agx	1ah5	1ah7	1ahj	1ahq
1aja	1ak1	1akd	1ako	1akz	1al0	1al7	1alq	1am2	1am4

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

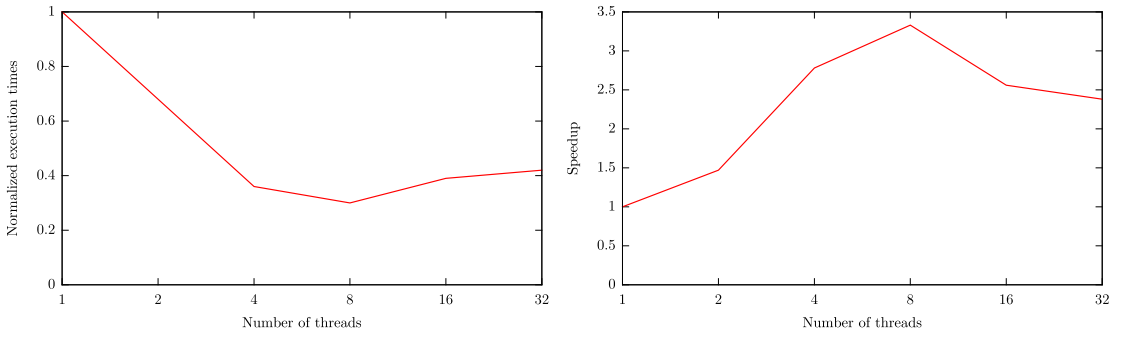

Fig. 6. Scalability testing. The figures show the global execution times and global scalability for a greedy-CMS run on the whole 200 proteins database. The serial execution time is about 1 h 53 min: the *y*-scale is normalized with respect to said value (1 in the figure equals to 1 h 53 min).

First, we present a global execution time and scalability analysis. Fig. 6 shows the normalized execution times (1 is the serial execution time) and the speedup for this test case with different number of cores.

In these experiments we chose to search only for very similar structures belonging to each pair of protein, thus we used strict tolerance parameters. The n_{max} parameter was left unbounded thanks to the exhaustivity property of the greedy variant.

The plots in Fig. 6 shows that the parallel implementation of greedy-CMS does indeed yields a boost in performance, even if it yields worse speedups than the parallel EMS implementation presented in Ferretti and Musci [1]. For instance, the speedup obtained using 4 cores, is 2.7, instead of 3.2 as in the EMS case. The reason for this difference is clearly the introduction of the synchronization constructs presented in Section 5.3, which are not needed in the EMS case.

Moreover, another interesting fact arises: the algorithm scalability is intrinsically limited. The best results are obtained with 8 cores: further increases in the number of cores yield worse performance. It is worth noting that, even on a database of this scale (a medium one at best, when compared to the full PDB), the serial execution time is almost 2 h, and would grow quadratically as the size of the database increase, according to Eq. (5).

We can speculate on the reasons of this behavior. The most probable one is the nature of the greedy variant, which can lead to extreme variations in the amount of computation to be performed by each iteration of the linearized loops (see Section 5.3), so that a lot of time is spent with idle processors, especially as the number of cores grows.

Let us verify this theory. As explained previously, execution times are dependent in a non-trivial way on: (1) the size in SSEs of each protein pair (N_1 and N_2), (2) the maximum motif cardinality (n_{max}); (3) the prescribed tolerances, both on the geometric parameters and on the accumulation points; (4) the relative similarity of the two proteins. The dependance on factors (1) and (2) can be clearly understood by a close analysis of Eq. (5), which ~~describe~~ the computational complexity of CMS. However, factors (1) and (2) do not affect granularity. ~~Factor~~ (3) and (4), instead, have a huge impact on the performance of the CMS algorithm, and may be the limiting factor in the scalability of the parallel implementation. A theoretical analysis of the sensitivity of the speedup with respect to each factor would be extremely difficult to carry out. However we can try to understand the problem with a few more experiments.

With regards to factor (3), we can analyze the effects of various degrees of tolerance thanks to Fig. 7. As one can see, looser tolerances obviously lead to longer execution times, but also to better scalability result. The reason for this behavior is that looser tolerances tend to increase the granularity of the parallelization scheme. As the two-hundred proteins testing suite uses strict tolerances, one reason for the lack of scalability has been identified. But this is still not enough to explain the result shown in 6.

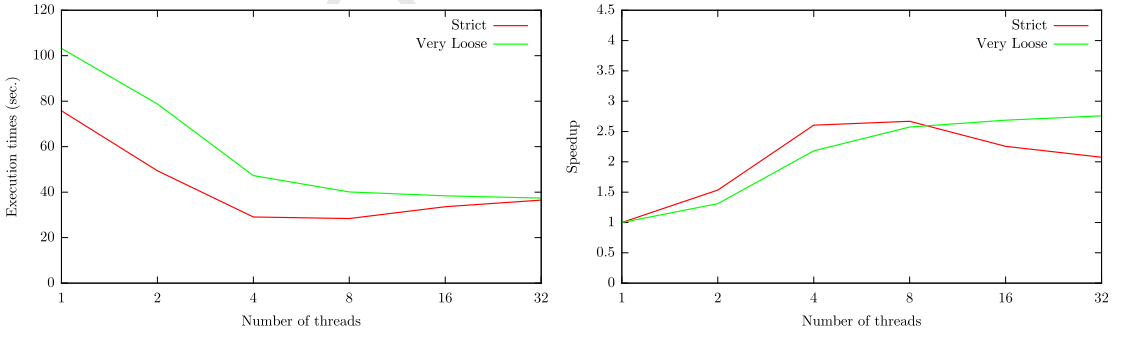

Fig. 7. The effect of tolerance on the execution times, with number of threads ranging from 1 to 32. Small dataset (20 proteins), greedy-CMS. Strict tolerance as defined in Section 6.1. Very loose tolerances: midpoints distance 5.0 Å; line distance 5.0 Å; angle 10°; cluster 3.0 Å.

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

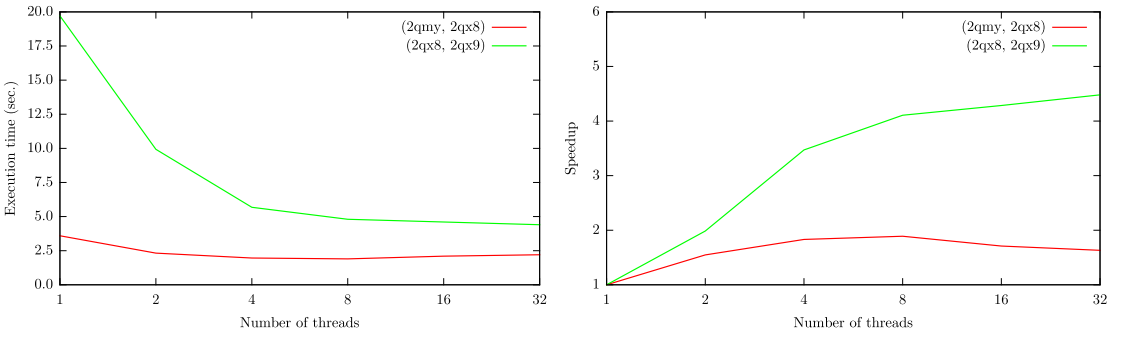

Fig. 8. The effect of protein similarity on execution time and speedup of parallel greedy-CMS. `2qx8` and `2qx9` are essentially the same protein as they belong to the same SCOP domain. Even if `2qmy` and `2qx9` are in the same family, they are still significantly dissimilar.

As to factor (4), the similarities of proteins in the data set, we can try to isolate the effect on the scalability by analyzing single pairs of proteins. Fig. 8 shows the execution times and speed-up of a *single* greedy-CMS run on three pair of proteins. Recall from the previous section, that `2qx8` and `2qx9` are essentially the same protein; while `2qmy`, even if still similar to `2qx8`, is dissimilar enough to make a huge difference in the behavior of the algorithm.

From these results, it is clear that a CMS run on similar proteins, while obviously being slower to complete, gives extremely better speedup and scalability with respect to a CMS run on less similar proteins. This behavior is even more pronounced if we choose completely unfamiliar proteins.

Before drawing any conclusion we would like to ~~note~~ the conditions for the two-hundred proteins testing suite closely approximate a real-case application of the greedy-CMS. We are interested in strict tolerances, as we would like to identify very similar geometrical sub-structures. Moreover, choosing proteins belonging to different protein super-families is mandatory if one ~~expect~~ to find previously unknown geometrical motifs. The only parameter that we can see to be different is the size of the database of proteins, as it was discussed in Section 3.5.

So, in conclusion, we can say that the lack of scalability is unavoidable with the current parallelization scheme and the expected amount of similarity and tolerance in the set. A change of the parallelization scheme is unfortunately impossible at the OpenMP level, as we firmly believe that the usage of barriers and synchronization constructs has been kept to the bare minimum, and that the greedy scheme is inherently limited in the amount of available parallelism (see in particular Fig. 5).

The lack of scalability, the relative high execution times and the impossibility to improve the shared-memory parallelization are a clear indication that, in order to extend the CMS to a larger database, that is in order to perform a fully-blown CCMS run, it is necessary to compare multiple protein pairs concurrently. That is, a hybrid shared-memory/message-passing approach is needed, and we are currently working on it (see Section 7). However, we can still conclude that, with a pure shared-memory approach, we have obtained satisfactory results up to a reasonable number of cores.

7. Conclusions and future work

We have introduced and discussed a novel set of algorithms, based on the Generalized Hough Transform, which are able to perform the task of motif identification and retrieval at the secondary structure level of proteins. The most important of these algorithm is the Cross Motif Search, or CMS. The main feature of CMS is that is based on a geometrical description of the structural motifs, and thus is able to pin-point previously unknown similarities among a lot of different pair of proteins, especially "unfamiliar" ones. As we have shown, even in a small database of 20 proteins, a Complete CMS can identify thousands of common substructures.

Much effort has been directed to consistently reduce both the computational complexity and the execution time of our algorithms, as handling tolerances in the geometrical description of motif is especially demanding in terms of computations. The goal to carry out extensive cross searches, coupled with a wide range of geometrical tolerances, brought us to sophisticated parallel approaches.

At the fine-grain level of a single protein analysis, the computations can be easily carried out with an optimized serial version of the algorithm, and this holds true even with a small database. The biggest contribution to the performance at the serial level ~~come~~ from the greedy optimization of CMS.

However, cross-protein and entire database processing (from hundreds up to thousands of proteins) do require a careful conceived OpenMP approach. We described a complete OpenMP implementation of CMS, and showed how its many sub-computations are distributed across shared-memory threads. Unfortunately, a detailed scalability analysis showed that there is a limit in the exploitations of cores (8 maximum) due to the nature of the problem. On the other hand, for most real-case applications, this level of parallelism is more than enough.

It is possible to conceive applications with even bigger dataset, up in theory to the whole PDB, so that a pure OpenMP approach is unable to properly handle the huge computations that arise. In this scenario an hybrid approach would therefore

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007

be the only viable solution; a massively parallel system that supports MPI, with a good support for a "local" OpenMP implementation, would appear to be the best choice. Indeed, a large scale application of CCMS can be partitioned into sub-problems that can be mapped to shared-memory sub-units, so that the OpenMP implementation we have described in this paper would be simply replicated.

Preliminary experiments shows that the hybrid OpenMP-MPI implementation has extremely good performance. Currently, the port of a Complete Cross Motif Search problem on a very large subset of the SCOP database is being prepared for execution on the HPC facility available in the Italian Supercomputing Center at CINECA.[4]

References

[1] M. Ferretti, M. Musci, Entire motifs search of secondary structures in proteins: a parallelization study, in: Proc. Pbio 2013 Intern. Workshop on Parallelism in Bioinformatics, Sept. 17, ACM Digital Library, Madrid, 2013, pp. 199–204.
[2] V. Cantoni, A. Ferone, O. Özbudak, A. Petrosino, Structural analysis of protein secondary structure by GHT, in: 21st International Conference on Pattern Recognition, ICPR'12, Nov. 11–15, IEEE Computer Society Press, Tsukuba, Japan, 2012, pp. 1767–1770.
[3] V. Cantoni, A. Ferone, O. Özbudak, A. Petrosino, Motif retrieval by exhaustive matching and couple co-occurrences, in: Ninth International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics, CIBB'12, July 12–14, Texas, 2012, pp. 1767–1770.
[4] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognit. 13 (2) (1981) 111–122.
[5] S. Shi, Y. Zhong, I. Majumdar, S.S. Krishna, N.V. Grishin, Searching for three-dimensional secondary structural patterns in proteins with ProSMoS, Bioinformatics 23 (11) (2007) 1331–1338.
[6] S. Shi, B. Chitturi, N.V. Grishin, ProSMoS server: a pattern-based search using interaction matrix representation of protein structures, Nucleic Acids Res., 1362-4962 37 (Web server issue) (2009) W526–W531, http://dx.doi.org/10.1093/nar/gkp316.
[7] G. Hutchinson, J.M. Thornton, PROMOTIF – a program to identify and analyze structural motifs in proteins, Protein Sci. 5 (1996) 212–220.
[8] O. Dror, H. Benyamini, R. Nussinov, H. Wolfson, MASS: multiple structural alignment by secondary structures, Bioinformatics, 1367-4803 19 (1) (2003) i95–i104, http://dx.doi.org/10.1093/bioinformatics/btg1012. ISSN: 1460-2059 <http://bioinformatics.oxfordjournals.org/content/19/suppl_1/i95>.
[9] E. Krissinel, K. Henrick, Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions, Acta Crystallogr. Sect. D: Biol. Crystallogr. 60 (2004) 2256–2268.
[10] L. Pauling, R.B. Corey, H.R. Branson, ~~The anatomy and taxonomy of protein structure~~, Proc. Natl. Acad. Sci. USA 37 (4) (1951) 205–211.
[11] Research Collaboratory for Structural Bioinformatics, Protein Data Bank, 2013, <http://www.rcsb.org/pdb> (online; accessed 5-December-2013).
[12] W. Kabsch, C. Sander, Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features, Biopolymers 22 (12) (1983) 2577–2637.
[13] C. Branden, J. Tooze, Introduction to Protein Structure, 2nd ed., Garland Science, 1999.
[14] V. Cantoni, A. Ferone, O. Özbudak, A. Petrosino, Protein motifs retrieval by SS terns occurrences, Pattern Recognit. Lett. 34 (5) (2013) 559–563.
[15] G. Drago, M. Ferretti, M. Musci, CCMS: a greedy approach to motif extraction, in: New Trends in Image Analysis and Processing — ICIAP 2013, Lecture Notes in Computer Science, vol. 8158, Springer, 2013, ISBN 978-3-642-41189-2, pp. 363–371, http://dx.doi.org/10.1007/978-3-642-41190-839.
[16] T.J. Hubbard, B. Ailey, S.E. Brenner, A.G. Murzin, C. Chothia, SCOP: a structural classification of proteins database, Nucl. Acids Res. 27 (1) (1999) 254–256.
[17] D.E. Knuth, The Art of Computer Programming, Addison-Wesley, 2005.

[4] http://www.hpc.cineca.it

Please cite this article in press as: M. Ferretti, M. Musci, Geometrical motifs search in proteins: A parallel approach, Parallel Comput. (2014), http://dx.doi.org/10.1016/j.parco.2014.09.007