Byte Structure Variable Length Coding (BS-VLC): A New Specific Algorithm Applied in the Compression of Trajectories Generated by Molecular Dynamics

André Melo,*,† André T. Puga,‡ Fernanda Gentil,† Nelson Brito,† Artur P. Alves,‡ and Maria João Ramos†

CEQUP/Departamento de Química, Faculdade de Ciências do Porto, Rua do Campo Alegre 687, 4169-007 Porto, Portugal, and FEUP/INESC-Porto, Largo Mompilher 22, 4007 Porto Codex, Portugal

Received July 2, 1999

Molecular dynamics is a well-known technique very much used in the study of biomolecular systems. The trajectory files produced by molecular dynamics simulations are extensive, and the classical lossless algorithms give poor efficiencies in their compression. In this work, a new specific algorithm, named byte structure variable length coding (BS-VLC), is introduced. Trajectory files, obtained by molecular dynamics applied to trypsin and a trypsin:pancreatic trypsin inhibitor complex, were compressed using four classical lossless algorithms (Huffman, adaptive Huffman, LZW, and LZ77) as well as the BS-VLC algorithm. The results obtained show that BS-VLC nearly triplicates the compression efficiency of the best classical lossless algorithm, preserving a near lossless behavior. Compression efficiencies close to 50% can be obtained with a high degree of precision, and the maximum efficiency possible (75%), within this algorithm, can be performed with good precision.

1. INTRODUCTION

Molecular dynamics (MD) has been applied to biomolecular systems and has been shown to be able to elucidate on their energetic, dynamical, and statistical mechanical properties. MD is a deterministic procedure in which the atoms in a molecule move according to classical mechanics. Thus, in a MD calculation we have to integrate Newton's equations of motion over time for the N atoms of the molecular system which is being studied

$$\partial^2\mathbf{r}(i,t)/\partial t^2 = m(i)^{-1}\,\mathbf{F}(i,t) \qquad i = 1, ..., N \qquad (1)$$

with $\mathbf{F}(i,t) = -\partial V(\mathbf{r}(1,t), ..., \mathbf{r}(N,t))/\partial\mathbf{r}(i,t)$ as the force on atom i at time t, $V(\mathbf{r}(1,t), ..., \mathbf{r}(N,t))$ as the potential energy function, $\mathbf{r}(i,t)$ as the position of atom i at time t, and $m(i)$ as the mass of atom i. In the present work, the leap-frog algorithm[1] has been used to compute the position vectors $\mathbf{r}(i,t)$ using the forces and previous positions of the atoms at a series of time intervals which differ by Δt.

The set of atomic positions occupied in a given time t_j is called *conformation* (vector $\mathbf{r}(i,t_j)$, $i = 1, ..., N$), and a succession of conformations, in n time intervals, is named a *trajectory* (matrix $[\mathbf{r}(i,t_j)]$, $i = 1, ..., N$, $j = 1, ..., n$). For simplicity, we shall use $\mathbf{r}(i,j)$, $i = 1, ..., N$, to represent a conformation and $[\mathbf{r}(i,j)]$, $i = 1, ..., N$, $j = 1, ..., n$, to represent a trajectory. A trajectory is thus given by the following matrix

$$\begin{array}{cccccc}
\mathbf{r}(1,1) & \mathbf{r}(2,1) & \cdots & \mathbf{r}(N,1) & \Rightarrow & \text{first conformation} \\
\mathbf{r}(1,2) & \mathbf{r}(2,2) & \cdots & \mathbf{r}(N,2) & \Rightarrow & \text{second conformation} \\
\vdots & \vdots & \cdots & \vdots & & \\
\mathbf{r}(1,n) & \mathbf{r}(2,n) & \cdots & \mathbf{r}(N,n) & \Rightarrow & n\text{th conformation}
\end{array} \qquad (2)$$

The trajectories generated by a MD calculation are the basis for the calculation of the system properties. However, the correspondent binary trajectory files are extensive and create problems of storage space. The classical lossless compression algorithms, such as the Huffman coding[2] used in the compression pack utility, adaptive Huffman[3] used in compact, LZW[4] used in compress, and LZ77[5] used in gzip, give poor efficiencies in the compression of these types of files. Therefore, specific lossy algorithms, which increase significantly the compression efficiency preserving a high degree of precision, are of great importance in attaining a better approach to this problem.

This work introduces the reader to a new specific algorithm, named byte structure variable length coding (BS-VLC), which increases significantly the compression efficiencies of the best classical lossless algorithms preserving a high degree of precision.

Frequently, the MD simulations are performed on very powerful parallel supercomputers and the analysis of the corresponding trajectory files are carried out on smaller workstations. This stresses the importance of the portability of the compressed data. The floating point binary formats differ usually very much from one machine to another, and this creates difficulties on their portability. On the other hand, the integer binary formats differ only in the byte ordering (big endian or little endian), and it is quite easy to convert this type of data between computers with different architec-

* To whom correspondence should be addressed. Phone: 351-22-6082803. Fax: 351-22-6082959. E-mail: asmelo@fc.up.pt.
† CEQUP/Faculdade de Ciências do Porto.
‡ FEUP/INESC-Porto.

10.1021/ci990069u CCC: $19.00
Published on Web 01/22/2000

tures. The BS-VLC algorithm works with integer differential coordinates, and the writing of a special subroutine is under way to be included in the compression code to enable the portability of the compressed files.

In this work, the BS-VLC algorithm was used in the compression of trajectory files generated by MD applied to the biological systems trypsin and the trypsin:pancreatic trypsin inhibitor (PTI) complex.

2. METHODS

2.1. Compression with the BS-VLC Algorithm. The initial data in this process is the trajectory matrix $[\mathbf{r}(i,j)]$, $i = 1, ..., N$, $j = 1, ..., n$, which in Cartesian space can be substituted by three matrixes $[\mathbf{r}_\mathbf{u}(i,j)]$ with $\mathbf{u} = \mathbf{x}$, $\mathbf{y}$, or $\mathbf{z}$, respectively, and $i = 1, ..., N$, $j = 1, ..., n$. The trajectories are usually stored in binary files with every coordinate figuring as a 4 bytes real number. All of these data are submitted to three steps in a BS-VLC compression: preprocessing, quantization, and variable length coding. These will be analyzed separately.

Preprocessing. The proposed methodology here is based on the conversion of the initial data into differential trajectory matrixes $[\Delta\mathbf{r}_\mathbf{u}(i,j)]$, $i = 1, ..., N$, $j = 1, ..., n$, in which the different components are given by coordinate differences; i.e.

$$\Delta r_\mathbf{u}(i,j) = \mathbf{r}_\mathbf{u}(i,j) - \mathbf{r}_\mathbf{u}^{\mathrm{ref}}(i,j) \tag{3}$$

where $\mathbf{r}_\mathbf{u}^{\mathrm{ref}}(i,j)$ is the reference coordinate associated with $\mathbf{r}_\mathbf{u}(i,j)$. It is necessary to store n_0 conformations (standard integral conformations $([\mathbf{r}_\mathbf{u}^{\ominus}(i,k)]$, $i = 1, ..., N$, $j = 1, ..., n_0$) in its original form to allow the rebuilding of trajectory matrices $[\mathbf{r}_\mathbf{u}(i,j)]$ from the correspondent differential matrices $[\Delta\mathbf{r}_\mathbf{u}(i,j)]$, in the decompression. The number of standard integral conformations depends on the criteria used in the selection of the reference coordinates $\mathbf{r}_\mathbf{u}^{\mathrm{ref}}(i,j)$ and will be discussed later in section 2.4.

Quantization. This information is further transformed into integer differential trajectory matrices $[\mathbf{u}(i,j)]$, $i = 1, ..., N$, $j = 1, ..., n$, with components given by coordinate differences converted to integers multiplied by a scaling factor (scale) as follows

$$\mathbf{u}(i,j) = \Delta\mathbf{r}_\mathbf{u}(i,j) \times \mathrm{scale} \tag{4}$$

The higher the scaling factor, the higher the precision of the process is and the lower the compression efficiency becomes; the contrary also applies; i.e., to increase the compression efficiency, one can lower the scaling factor with the added disadvantage of lowering the precision. It is important to find a middle point which will provide a good compression coupled with a reasonable error.

Variable Length Coding. Subsequently, the elements of the integer differential matrices $[\mathbf{u}(i,j)]$ are compressed using a variable length code, where the trajectories are subdivided into sets (structures) which are separately coded.

2.2. Coding within the BS-VLC Algorithm. Byte Integer Signal Coding. The integers, components of the integer differential trajectory matrices $[\mathbf{u}(i,j)]$, can be represented by implicit signal coding or explicit signal coding. The former is the conventional form of partitioning integers into bytes: i.e., 1 bit is always kept to represent the signal, and the other $N - 1$ bits represent the respective absolute value; in the latter, signal bits are explicitly codified and grouped into signal bytes. This representation can be less expensive if all the data are centered in certain regions. In a 32 bit machine the minimum number of bytes necessary for the representation of absolute values, according to the interval where they are, is

$$\begin{aligned} 1\ \text{byte} &\leftrightarrow [0{:}(2^8 - 1)] \\ 2\ \text{byte} &\leftrightarrow [2^8{:}(2^{16} - 1)] \\ 3\ \text{byte} &\leftrightarrow [2^{16}{:}(2^{24} - 1)] \\ 4\ \text{byte} &\leftrightarrow [2^{24}{:}(2^{32} - 1)] \end{aligned} \tag{5}$$

However, when we are working with signed integers, half of this space has to be used to represent negative integers. Therefore, we end up with the following association between the number of bytes and intervals

$$\begin{aligned} 1\ \text{byte} &\leftrightarrow [0{:}(2^7 - 1)] \cup [-1{:}-2^7] \\ 2\ \text{byte} &\leftrightarrow [2^7{:}(2^{15} - 1)] \cup [(-2^7 - 1){:}-2^{15}] \\ 3\ \text{byte} &\leftrightarrow [2^{15}{:}(2^{23} - 1)] \cup [(-2^{15} - 1){:}-2^{23}] \\ 4\ \text{byte} &\leftrightarrow [2^{23}{:}(2^{31} - 1)] \cup [(-2^{23} - 1){:}-2^{31}] \end{aligned} \tag{6}$$

On the other hand, the signal can be explicitly coded, allowing the absolute values to be represented according to eq 5. One possible way to perform this is to subdivide the data into groups of eight elements and to associate the value 1 to their signals when they are positive and the value 0 when they are negative.

for example: $+ - + + - - + + \leftrightarrow 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1$

Each group of eight signal bits is then stored in a signal byte, and the entire sequence of signal bytes is stored in a signal vector. The dimension (nsbyte) of this vector is equal to the number of coordinates over 8. Associating eqs 5 and 6, the 32 bit integer space can be subdivided into seven intervals which need different minimum numbers of bytes for coding according to whether implicit or explicit signal coding is used. All this information has been collected in Table 1. It can be observed from the table that explicit signal coding is favored when data are concentrated in intervals 2, 4, or 6.

Concept of Structure. Structures are sets within the integer differential trajectory matrices where elements exhibit some sort of correlation. BS-VLC is based on the fact that a structure could be represented by fewer bytes than entire trajectory matrices. All the elements within a structure (S_i) are represented by the number of bytes $(\mathrm{nbmax}(S_i))$ necessary to codify the element with the largest absolute value. However, it is necessary to keep one byte extra for each structure to indicate the number of bytes used in its representation. If a trajectory matrix is subdivided into n_0 standard integral conformations, n_1 structures that use implicit signal coding, and n_2 structures that use explicit signal

Table 1. Minimum Number of Byte Coding for Different Intervals of Integer Numbers

		min no. of byte coding	
interval identification	interval	implicit signal coding	explicit signal coding
1	$[0:(2^7-1)] \cup [-1:-2^7]$	1	1
2	$[2^7:(2^8-1)] \cup [(-2^7-1):(-2^8+1)]$	2	1
3	$[2^8:(2^{15}-1)] \cup [-2^8:-2^{15}]$	2	2
4	$[2^{15}:(2^{16}-1)] \cup [(-2^{15}-1):(-2^{16}+1)]$	3	2
5	$[2^{16}:(2^{23}-1)] \cup [-2^{16}:-2^{23}]$	3	3
6	$[2^{23}:(2^{24}-1)] \cup [(-2^{23}-1):(-2^{24}+1)]$	4	3
7	$[2^{24}:(2^{32}-1)] \cup [-2^{24}:-2^{31}]$	4	4

coding, the byte length of a compressed file (CFBL) is given by

$$\mathrm{CFBL} = \mathrm{ovh} + (n_1 + n_2) + \mathrm{nsbyte} + \sum_{i=1}^{n_1} n(S_i)\ \mathrm{nbmax1}(S_i) + \sum_{j=1}^{n_2} n(S_j)\ \mathrm{nbmax2}(S_j) + \mathrm{ns}_0 \quad (7)$$

In eq 7, ovh is the structure organization overhead, which indicates how the trajectory file was compressed. This information is necessary to the decompression process but represents a very insignificant part (less than 0.01%) of CFBL. In the same equation, $n(S_i)$ and $n(S_j)$ are the number of elements of structures S_i and S_j, respectively; nbmax1(S_i) is the number of bytes necessary to represent the element of structure S_i with the largest absolute value determined using the third column of Table 1; nbmax2(S_j) is the number of bytes necessary to represent the element of structure S_j with the largest absolute value determined using the fourth column of Table 1, ns_0 is the number of bytes necessary to store the standard integral conformation, and nsbyte is the total number of elements represented with explicit signal coding over 8:

$$\mathrm{ns}_0 = 4 \times 3N \times n_0 \quad (8)$$

$$\mathrm{nsbyte} = \sum_{j=1}^{n_2} n(S_j)/8 \quad (9)$$

Types of Structures. The BS-VLC algorithm considers two types of structures: structures associated with temporal correlations (blocks and conformations within a block) and structures associated with spatial correlations (atoms within a block and atomic Cartesian coordinates within a block).

(a) Block. The block structure is the largest one considered in the BS-VLC method. A block is a set of sequential conformations, and this is a natural structure with the preprocessing and quantization methodologies adopted in this work. If implicit signal coding is assumed, the byte length of block B_i (BBL(B_i)) is given by

$$\mathrm{BBL}(B_i) = 3N\ \mathrm{nconf}(B_i)\ \mathrm{nbmax1}(B_i) \quad (10)$$

where nconf(B_i) is the number of conformations of block B_i. If explicit signal coding is assumed, BBL(B_i) is given by

$$\mathrm{BBL}(B_i) = \mathrm{nsbyte}(B_i) + 3N\ \mathrm{nconf}(B_i)\ \mathrm{nbmax2}(B_i) \quad (11)$$

where nsbyte(B_i) is calculated as

$$\mathrm{nsbyte}(B_i) = \mathrm{nconf}(B_i)\ 3N/8 \quad (12)$$

(b) Conformation within a Block. The conformation within a block is the smallest temporal structure considered in the BS-VLC method. In this case, BBL(B_i) with implicit signal coding is given by

$$\mathrm{BBL}(B_i) = \mathrm{nconf}(B_i) + 3N \sum_{k=1}^{\mathrm{nconf}(B_i)} \mathrm{nbmax1}(\mathrm{conf}(k)) \quad (13)$$

If explicit signal coding is assumed, BBL(B_i) is calculated as

$$\mathrm{BBL}(B_i) = \mathrm{nconf}(B_i) + \mathrm{nsbyte}(B_i) + 3N \sum_{k=1}^{\mathrm{nconf}(B_i)} \mathrm{nbmax2}(\mathrm{conf}(k)) \quad (14)$$

(c) Atom within a Block. In an MD trajectory it is possible to establish spatial correlation. An atom within a block is a structure that reflects this type of correlation. For example, in a protein the differential coordinates of the side chain atoms are usually larger than the differential coordinates of the main chain atoms. In this case, BBL(B_i) with implicit signal coding is given by

$$\mathrm{BBL}(B_i) = N + 3\mathrm{nconf}(B_i) \sum_{k=1}^{N} \mathrm{nbmax1}(\mathrm{at}(k)) \quad (15)$$

If explicit signal coding is assumed, BBL(B_i) is calculated as

$$\mathrm{BBL}(B_i) = N + \mathrm{nsbyte}(B_i) + 3\mathrm{nconf}(B_i) \sum_{k=1}^{N} \mathrm{nbmax2}(\mathrm{at}(k)) \quad (16)$$

(d) Atomic Cartesian Coordinate within a Block. The atomic Cartesian coordinate within a block is a subdivision of the atom structure, and it is appropriate when the potential energy function exhibits some type of anisotropy. Here, BBL(B_i) with implicit signal coding, is given by

$$\mathrm{BBL}(B_i) = 3N + \mathrm{nconf}(B_i)\{\sum_{k=1}^{N} \mathrm{nbmax1}(x(k)) + \sum_{k=1}^{N} \mathrm{nbmax1}(y(k)) + \sum_{k=1}^{N} \mathrm{nbmax1}(z(k))\} \quad (17)$$

If explicit signal coding is assumed, BBL(B_i) is calculated as

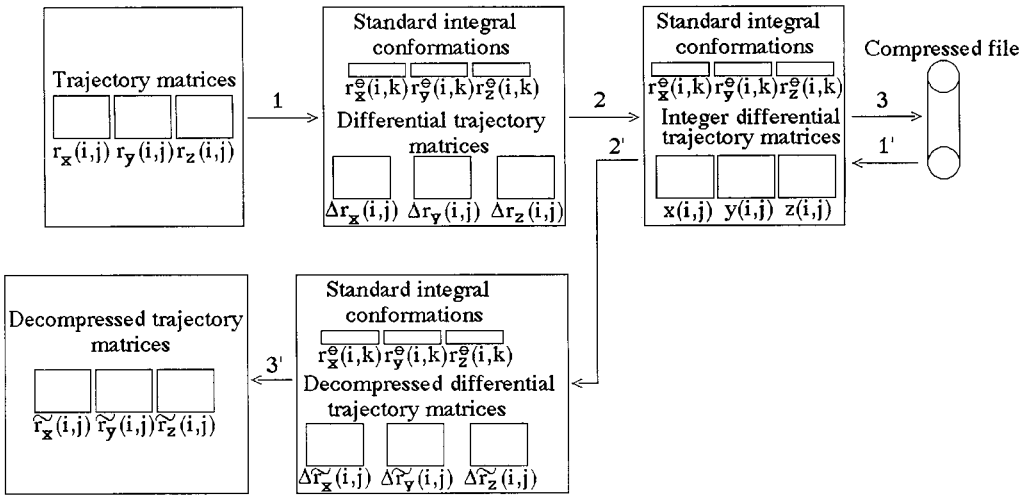

Figure 1. Flow chart of the BS-VLC algorithm. The following steps have to be considered: for compression, (1) preprocessing, (2) quantization, and (3) VLC; for decompression, (1′) VLD, (2′) inverse quantization, and (3′) postprocessing.

$$\mathrm{BBL}(B_i) = 3N + \mathrm{nsbyte}(B_i) + \mathrm{nconf}(B_i) \times \{\sum_{k=1}^{N}\mathrm{nbmax2}(x(k)) + \sum_{k=1}^{N}\mathrm{nbmax2}(y(k)) + \sum_{k=1}^{N}\mathrm{nbmax2}(z(k))\} \quad (18)$$

In eqs 17 and 18 $x(k)$, $y(k)$, and $z(k)$ are the Cartesian coordinates x, y, and z of atom k in block B_i, respectively.

2.3. Decompressing the Files Obtained with the BS-VLC Algorithm. A BS-VLC decompression of files follows three steps which will be analyzed separately, namely, variable length decoding, inverse quantization, and postprocessing.

Variable Length Decoding. In this step, the compressed file is decoded, restoring the original integer differential matrices $[\mathbf{u}(i,j)]$.

Inverse Quantization. The differential matrices $[\mathbf{u}(i,j)]$ are used to produce the decompressed differential matrixes $[\Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j)]$ with

$$\Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{u}(i,j)/\mathrm{scale} \quad (19)$$

The elements of the decompressed differential matrices differ from the original differential matrices from a given quantity $e(\Delta\mathbf{r}_\mathbf{u}(i,j))$:

$$\Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \Delta\mathbf{r}_\mathbf{u}(i,j) + e(\Delta\mathbf{r}_\mathbf{u}(i,j)) \quad (20)$$

The magnitude of the error $e(\Delta\mathbf{r}_\mathbf{u}(i,j)$ depends on the scaling factor selected.

Postprocessing. The decompressed trajectory matrices $[\tilde{\mathbf{r}}_\mathbf{u}(i,j)]$ are then calculated from the decompressed differential trajectory matrices $[\Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j)]$ and from the decompressed reference coordinates $\tilde{\mathbf{r}}^{\mathrm{ref}}_\mathbf{u}(i,j)$.

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \tilde{\mathbf{r}}^{\mathrm{ref}}_\mathbf{u}(i,j) + \Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j) \quad (21)$$

The elements of the decompressed trajectory matrices differ from the elements of the original trajectory matrices by a quantity $e(\mathbf{r}_\mathbf{u}(i,j))$.

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{r}_\mathbf{u}(i,j) + e(\mathbf{r}_\mathbf{u}(i,j)) \quad (22)$$

The characteristics of the errors $e(\mathbf{r}_\mathbf{u}(i,j))$ depend on the type of the coordinates $\mathbf{r}^{\mathrm{ref}}_\mathbf{u}(i,j)$ selected. The nature of these dependencies will be discussed later in section 2.4. A flow chart of the BS-VLC algorithm is given in Figure 1.

2.4. Selection of the Reference Coordinates. Several alternative selection criteria are possible to choose the reference coordinates $\mathbf{r}^{\mathrm{ref}}_\mathbf{u}(i,j)$ used in eq 3. The characteristics of the errors $e(\mathbf{r}_\mathbf{u}(i,j))$, in eq 22, are conditioned by the reference coordinates selected. We have considered the following types of reference coordinates.

Atomic Coordinates. The reference coordinates are defined as the atomic coordinates immediately before the present $\mathbf{r}_\mathbf{u}(i,j)$ coordinates

$$\mathbf{r}^{\mathrm{ref}}_\mathbf{u}(i,j) = \mathbf{r}_\mathbf{u}(i,j-1) \quad (23)$$

The standard integral conformation is the first conformation of the trajectories $\mathbf{r}_\mathbf{u}(i,1)$, with the necessary number of bytes to be stored, given as

$$\mathrm{ns}_0 = 4 \times 3\mathrm{N1} \quad (24)$$

Equation 21 becomes

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \tilde{\mathbf{r}}_\mathbf{u}(i,j-1) + \Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j) \quad (25)$$

Substituting eq 20 into eq 25, one obtains

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \tilde{\mathbf{r}}_\mathbf{u}(i,j-1) + \Delta\mathbf{r}_\mathbf{u}(i,j) + e(\Delta\mathbf{r}_\mathbf{u}(i,j)) \quad (26)$$

Repeating this substitution for all the $\tilde{\mathbf{r}}_\mathbf{u}(i,j-1)$, we obtain

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{r}_\mathbf{u}(i,1) + \sum_{k=2}^{j}\Delta\mathbf{r}_\mathbf{u}(i,k) + \sum_{k=2}^{j}e(\Delta\mathbf{r}_\mathbf{u}(i,k)) \quad (27)$$

or

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{r}_\mathbf{u}(i,j) + \sum_{k=2}^{j}e(\Delta\mathbf{r}_\mathbf{u}(i,k)) \quad (28)$$

Comparing eqs 28 and 22, the errors $e(\mathbf{r}_\mathbf{u}(i,j)$ can be calculated as

$$e(\mathbf{r_u}(i,j)) = \sum_{k=2}^{j} e(\Delta\mathbf{r_u}(i,k)) \quad (29)$$

If the successive values $e(\Delta\mathbf{r_u}(i,j))$ are not well-compensated for, this will originate cumulative errors $e(\mathbf{r_u}(i,j))$.

Atomic Coordinates within a Block. One possible methodology which partially prevents the cumulative nature of the error inherent to the previous selection consists of truncating its accumulation at the end of each block. In this situation, it is necessary to store not only the first conformation of the trajectory but also the first conformation of all the blocks. Consequently, the error, eqs 26−28, is still cumulative within a block but becomes null for the transition between the last conformation of a block and the first conformation of the next block. The compression efficiencies are also lightly reduced by this procedure:

$$\mathrm{ns}_0 = 4 \times 3N \times \mathrm{nblo} \quad (30)$$

where nblo is the total number of blocks used.

Decompressed Atomic Coordinates. Here, the reference coordinates are defined as

$$\mathbf{r}_\mathbf{u}^{\mathrm{ref}}(i,j) = \tilde{\mathbf{r}}_\mathbf{u}(i,j-1) \quad (31)$$

Equation 3 becomes

$$\Delta\mathbf{r_u}(i,j) = \mathbf{r_u}(i,j) - \tilde{\mathbf{r}}_\mathbf{u}(i,j-1) \quad (32)$$

and eq 20 can be rewritten as

$$\Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{r_u}(i,j) - \tilde{\mathbf{r}}_\mathbf{u}(i,j-1) + e(\Delta\mathbf{r_u}(i,j)) \quad (33)$$

Equation 33 can be rearranged as

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j-1) + \Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{r_u}(i,j) + e(\Delta\mathbf{r_u}(i,j)) \quad (34)$$

or

$$\tilde{\mathbf{r}}_\mathbf{u}(i,j) = \mathbf{r_u}(i,j) + e(\Delta\mathbf{r_u}(i,j)) \quad (35)$$

Comparing eqs 35 and 22, we conclude that the errors $e(\mathbf{r_u}(i,j))$ can be calculated as

$$e(\mathbf{r_u}(i,j)) = e(\Delta\mathbf{r_u}(i,j)) \quad (36)$$

Here, the error becomes noncumulative, preserving the compression efficiency obtained with reference coordinates a. This formulation corresponds to the classical scheme DPCM (differential pulse code modulation),[6] which is the basis of several audio, image, and video compression algorithms such as JPEG[7] and MPEG.[8]

The best way to evaluate the cumulative characteristics of the error associated with the compression/decompression process is to represent the conformational root mean square deviation (rms) between the decompressed $\tilde{\mathbf{r}}_\mathbf{u}(i,j)$ and the original coordinates $\mathbf{r_u}(i,j)$,

$$\mathrm{rms} = \{[\sum_{i=1}^{N}(\tilde{\mathbf{r}}_\mathbf{x}(i,j) - \mathbf{r_x}(i,j))^2 + (\tilde{\mathbf{r}}_\mathbf{y}(i,j) - \mathbf{r_y}(i,j))^2 + (\tilde{\mathbf{r}}_\mathbf{z}(i,j) - \mathbf{r_z}(i,j))^2]/(3N-1)\}^{1/2} \quad (37)$$

as a function of simulation time.

The total mean square deviation ($\overline{\mathrm{rms}}$) is given by

$$\overline{\mathrm{rms}} = \{[\sum_{j=1}^{n}\sum_{i=1}^{N}(\tilde{\mathbf{r}}_\mathbf{x}(i,j) - \mathbf{r_x}(i,j))^2 + (\tilde{\mathbf{r}}_\mathbf{y}(i,j) - \mathbf{r_y}(i,j))^2 + (\tilde{\mathbf{r}}_\mathbf{z}(i,j) - \mathbf{r_z}(i,j))^2]/(3N\,n-1)\}^{1/2} \quad (38)$$

allowing an evaluation of the global precision of the compression/decompression process.

2.5. Implementation of the BS-VLC Algorithm. In the *BS-VLC compression scheme* the following procedure has to be considered.

Preprocessing.

(1) Read initial data $[\mathbf{r_u}(i,j)]$.

(2) Select the number of blocks (nblo).

(3) Select the reference coordinates (atomic coordinates, atomic coordinate within a block, or decompressed atomic coordinates).

(4) Store the standard integral conformations $[\mathbf{r}_\mathbf{u}^{\ominus}(i,k)]$.

(5) Calculate the differential trajectory matrices $[\Delta\mathbf{r_u}(i,j)]$.

Quantization.

(1) Select the scaling factor (scale).

(2) Calculate the integer differential matrices $[\mathbf{u}(i,j)]$.

VLC. For each block:

(1) Select the structure (entire block, conformation within the block, atom within the block, or atomic Cartesian coordinate within the block) and the signal coding (explicit or implicit) which will allow a more efficient compression of the block (eqs 10−18).

(2) Compress the block using the structure and the signal coding selected in 1.

The byte length of the compressed file (CFBL) is given by

$$\mathrm{CFBL} = \mathrm{ovh} + \sum_{i=1}^{\mathrm{nblo}} \mathrm{BBL}(B_i) + \mathrm{ns}_0 \quad (39)$$

In the BS-VLC decompression scheme the following procedure must to be considered:

VLD. For each block:

(1) Read the structure (entire block, conformation within the block, atom within the block, or atomic Cartesian coordinate within the block) and the signal coding (explicit or implicit) used in the compression.

(2) Decompress the block using the structure and signal coding read in 1 (the integer differential matrices $[\mathbf{u}(i,j)]$ are rebuilt).

Inverse Quantization.

(1) Read the scaling factor (scale).

(2) Calculate the decompressed differential matrices $[\Delta\tilde{\mathbf{r}}_\mathbf{u}(i,j)]$.

Postprocessing.

(1) Read the standard integral conformations $[\mathbf{r}_\mathbf{u}^{\ominus}(i,k)]$.

(2) Calculate the decompressed trajectory matrices $[\tilde{\mathbf{r}}_\mathbf{u}(i,j)]$.

3. RESULTS AND CONCLUSIONS

In this work two solvated proteic systems, trypsin and the trypsin:PTI complex, have been studied. Trypsin is a digestive enzyme, and PTI is a natural inhibitor of trypsin.

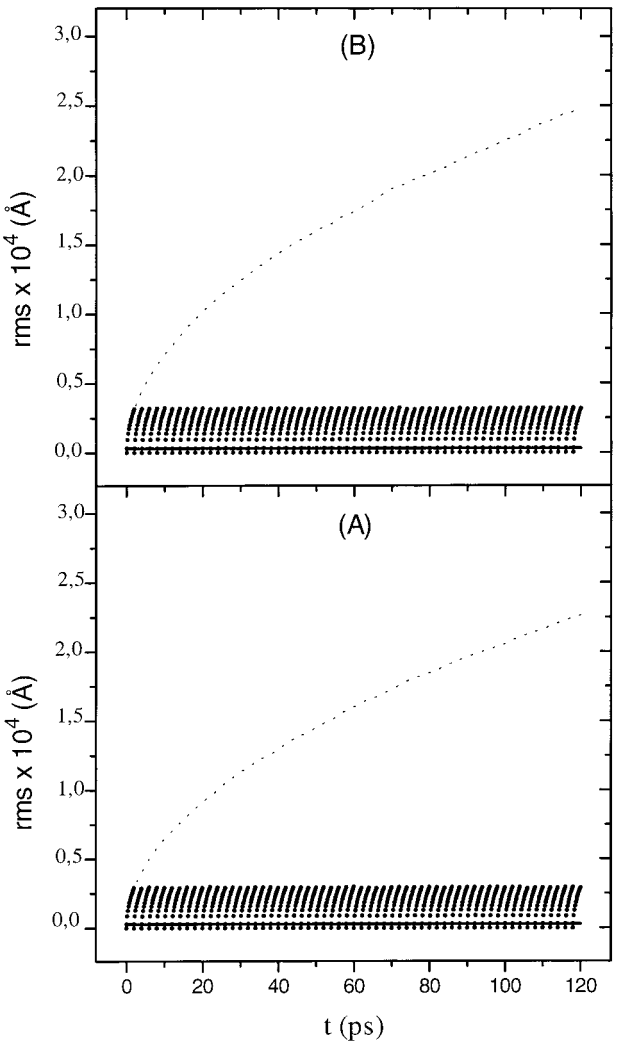

Figure 2. Conformational root-mean-square deviations (rms) between the original trajectories, obtained by molecular dynamics applied to (A) trypsin and (B) the trypsin:PTI complex, and the trajectories obtained by sequential BS-VLC compression and decompression with a scaling factor of 10^5 as function of the simulation time (*t*). Three alternative reference coordinates, atomic coordinates (- -), atomic coordinates within a block (···), and decompressed atomic coordinates (—), were used.

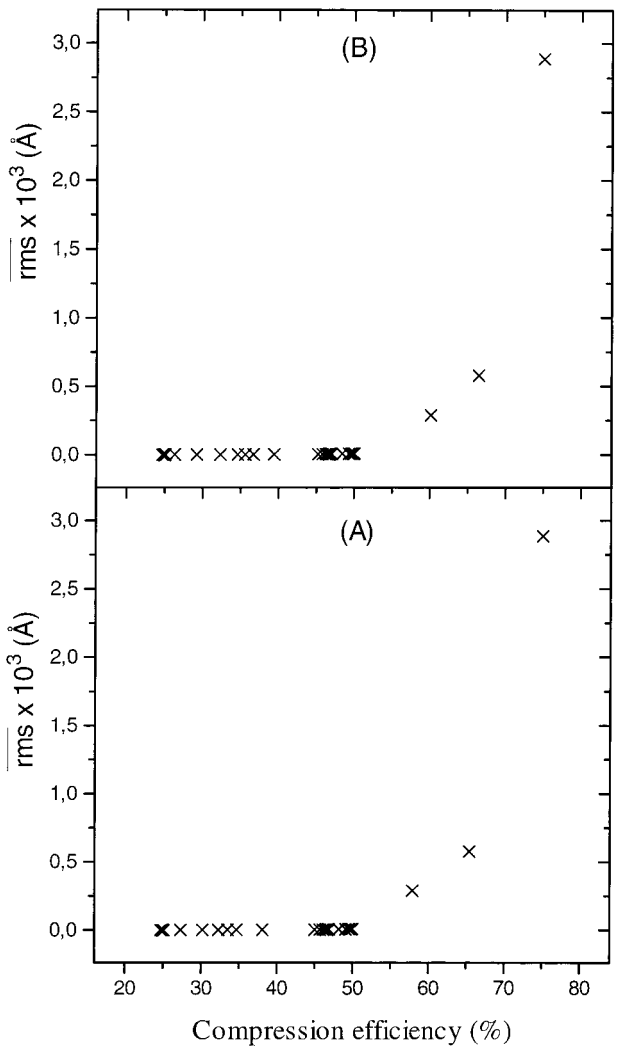

Figure 3. Total root mean square deviations ($\overline{\text{rms}}$) between the original trajectories, obtained by molecular dynamics applied to (A) trypsin and (B) the trypsin:PTI complex, and the trajectories obtained by sequential BS-VLC compression and decompression with 28 different scaling factors as a function of compression efficiency.

In the MD simulations,[9] all the water molecules and amino acid residues within a 15 Å sphere, centered in the active center of trypsin, were allowed to move. Harmonic forces were used to restrain any water molecule from leaving the 15−18 Å boundary; this was achieved by constraining the oxygen atoms of the water molecules to their initial positions using a force constant of 0.6 kcal mol^{-1} $Å^{-1}$. The other residues were included in the determination of the energy and forces but were kept fixed in their starting positions. The choice for performing the MD calculation in this way had its reasons in the size of the simulation which would have been prohibitive otherwise.[9] A total number of 1815 atoms for solvated trypsin and 1666 for solvated trypsin:PTI were allowed to move during the MD simulations. Newton's equations of motion were integrated every 0.001 ps using the leap-frog algorithm,[1] and 120 ps trajectories were generated for both systems studied. The total simulation time was 6.240 ns for trypsin, 6.400 ns for trypsin:PTI, and 12.640 ns for both trypsin and trypsin:PTI. The trajectory matrices have the following dimensions:

$$\mathbf{r}_u(1815,6000) \Rightarrow \text{for solvated trypsin}$$

$$\mathbf{r}_u(1666,6000) \Rightarrow \text{for solvated trypsin:PTI}$$

All simulations were carried out with the program CHARMM,[10] and the two trajectories were compressed using the lossless compressors (compress, gzip, pack, and compact) as well as the BS-VLC algorithm.

In the BS-VLC compressions, a total number of 60 blocks was used. Preliminary compressions, using the three alternative reference coordinates (atomic coordinates, atomic coordinates within a block, and decompressed atomic coordinates) and a scaling factor of 10^5, were performed. The conformational root mean square deviation (rms) was computed as a function of simulation time. The results obtained are presented in Figure 2; they confirm that atomic coordinates, atomic coordinates within a block, and decompressed atomic coordinates lead to a cumulative, a truncated cumulative, and a noncumulative error, respectively. Con-

Table 2. Compression Efficiencies and Total Root Mean Square Deviations ($\overline{rms}$) between the Original Trajectories, Obtained by Molecular Dynamics Applied to Trypsin and the Trypsin:BPTI Complex, and the Trajectories Obtained by Sequential BS-VLC Compression and Decompression with a Scaling Factor (Scale)

trypsin			trypsin:BPTI		
scale	compressn efficiency (%)	$\overline{rms}$ (Å)	scale	compressn efficiency (%)	$\overline{rms}$ (Å)
1.000×10^7	24.7	0.000	1.000×10^7	24.7	0.000
9.400×10^6	24.8	6.000×10^{-9}	9.400×10^6	24.9	6.000×10^{-9}
8.850×10^6	24.9	7.000×10^{-9}	8.300×10^6	25.0	9.000×10^{-9}
8.300×10^6	25.0	7.100×10^{-9}	7.200×10^5	25.1	1.400×10^{-8}
7.200×10^5	25.1	1.000×10^{-8}	4.108×10^5	26.3	3.664×10^{-7}
3.664×10^5	27.3	8.140×10^{-7}	3.664×10^5	29.2	6.090×10^{-7}
3.220×10^5	30.2	9.290×10^{-7}	3.220×10^5	32.3	7.790×10^{-7}
2.776×10^5	32.3	1.055×10^{-6}	2.776×10^5	34.6	9.050×10^{-7}
2.332×10^5	33.5	1.321×10^{-6}	2.332×10^5	35.6	1.055×10^{-6}
1.888×10^5	34.7	1.653×10^{-6}	1.888×10^5	36.7	1.285×10^{-6}
1.444×10^5	38.1	2.182×10^{-6}	1.444×10^5	39.4	1.685×10^{-6}
1.000×10^5	45.0	2.923×10^{-6}	1.000×10^5	45.2	3.197×10^{-6}
9.400×10^4	45.7	3.097×10^{-6}	9.400×10^4	45.8	3.421×10^{-6}
8.850×10^4	46.1	3.268×10^{-6}	8.850×10^4	46.2	3.651×10^{-6}
8.300×10^4	46.4	3.507×10^{-6}	8.300×10^4	46.4	3.884×10^{-6}
7.750×10^4	46.6	3.767×10^{-6}	7.750×10^4	46.6	4.130×10^{-6}
7.200×10^4	46.7	4.077×10^{-6}	7.200×10^4	46.7	4.399×10^{-6}
6.650×10^4	46.8	4.430×10^{-6}	6.650×10^4	46.8	4.699×10^{-6}
6.100×10^4	46.9	4.787×10^{-6}	6.100×10^4	46.9	5.021×10^{-6}
5.550×10^4	47.0	5.221×10^{-6}	5.550×10^4	47.0	5.399×10^{-6}
5.000×10^4	48.2	5.794×10^{-6}	5.000×10^4	48.4	5.849×10^{-6}
4.552×10^4	49.1	6.392×10^{-6}	4.552×10^4	49.3	6.404×10^{-6}
4.108×10^4	49.5	7.057×10^{-6}	4.108×10^4	49.6	7.052×10^{-6}
3.664×10^4	49.8	7.889×10^{-6}	3.664×10^4	49.8	8.042×10^{-6}
3.220×10^4	50.0	9.005×10^{-6}	3.220×10^4	50.0	9.134×10^{-6}
1.000×10^3	57.9	2.886×10^{-4}	1.000×10^3	60.1	2.887×10^{-4}
5.000×10^2	65.4	5.773×10^{-4}	5.000×10^2	66.4	5.772×10^{-4}
1.000×10^2	75.0	2.886×10^{-3}	1.000×10^2	75.0	2.887×10^{-3}

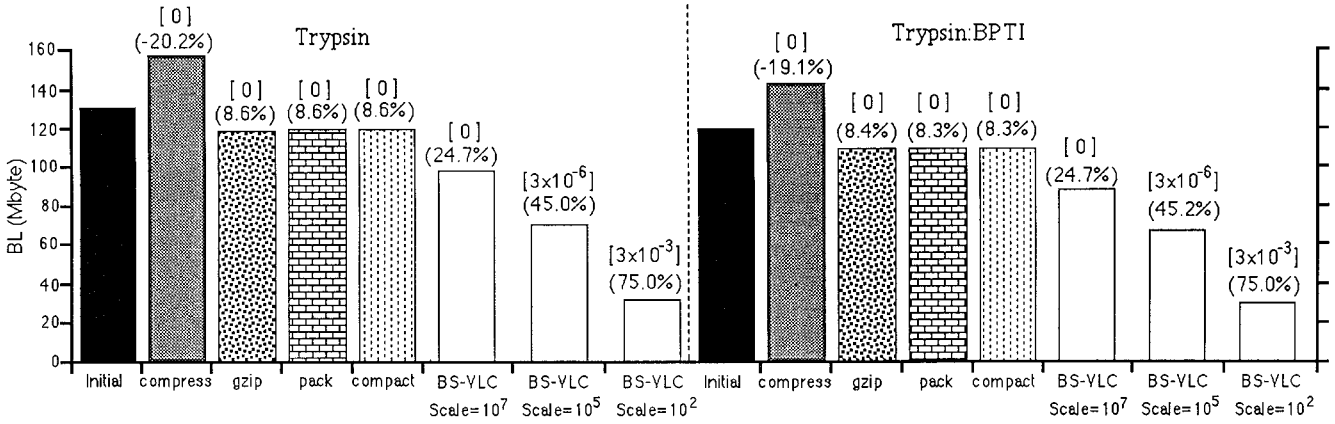

Figure 4. Byte length (BL) of initial trajectory files, obtained by molecular dynamics applied to trypsin and the trypsin:PTI complex, and of compressed files obtained using different algorithms. The compression efficiencies and total root mean square deviations, between the original trajectory and the trajectories obtained by sequential compression and decompression, are also indicated within round and square brackets, respectively.

sequently, as has been pointed in section 2.4, the decompressed atomic coordinates are the most appropriate selection for reference coordinates. This selection was used in the remainder BS-VLC compression presented here.

To evaluate the efficiency of the BS-VLC algorithm, several compressions of trypsin and trypsin:PTI trajectory files were performed, using 28 different values as scaling factors. The total root mean square deviations ($\overline{rms}$) were computed for all the cases. The results obtained are presented in Table 2 and Figure 3. Additionally, the compression efficiencies, obtained with different algorithms, can be visualized in Figure 4.

The analysis of Table 2, Figure 3, and Figure 4 is extremely favorable to the BS-VLC algorithm. In fact, when a scaling factor of 10^7 is used, the BS-VLC algorithm has a lossless behavior and presents a significantly larger compression efficiency than the classical lossless algorithms.

Green et al.[11] have made available a compression algorithm which reaches 70% as compared to the 75% achieved in this work. Additionally, this algorithm leads to a cumulative error in the compression/decompression process, while BS-VLC has noncumulative error behavior.

Here, all the obtained results enable us to conclude that BS-VLC has near lossless behavior ($\overline{rms} = 0$) when a scaling factor close to 10^7 is used. In this situation, BS-VLC nearly triplicates the compression efficiency of the best classical lossless algorithm (LZ77 used in gzip). In addition, larger compression efficiencies ($\approx$50%) can be managed with

BS-VLC preserving a high degree of precision ($\overline{\text{rms}}$ between 10^{-5} and 10^{-6}). For compression efficiencies larger than 50%, the precision decreases significantly. However, a compression with the maximum efficiency possible (75%) within this algorithm can be performed with good precision ($\overline{\text{rms}} \approx 10^{-3}$).

REFERENCES AND NOTES

(1) van Gunsteren, W. F.; Berendsen, H. J. C. *Angew. Chem., Int. Ed. Engl.* **1990**, *29*, 98.
(2) Huffman, D. V. *Proc. IRE* **1952**, *40*, 1098.
(3) Gallager, R. G. *IEEE Trans. 1nf. Theory* **1978**, *IT-24*, 668.
(4) Welch, T. A. *IEEE Comput.* **1984**, *17*, 8.
(5) Ziv, J.; Lempel, A. *IEEE Trans. 1nform. Theory* **1977**, *IT-23*, 337.
(6) Jayan, N. S.; Noll, P. *Digital Coding of Waveforms: Principles and Applications to Speech and Video*; Signal Processing Series; Prentice-Hall: Englewood Cliffs, 1984.
(7) Wallace, G. K. *Commun. ACM* **1991**, *34*, 30.
(8) ISO/IEC JTC 1/SC 29/WG 11, ISO/IEC CD 13818-3: Information Technology—Generic Coding of Moving Pictures and Associated Audio Information, 1994.
(9) Melo, A.; Ramos, M. J. *J. Pept. Res.* **1997**, *50*, 382.
(10) Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D. J.; Swaminathan, S.; Karplus, M. *J. Comput. Chem.* **1983**, *4*, 187.
(11) Green, D. G.; Meacham, K. E.; Surridge, M.; van Hoesel, F.; Berendsen, H. J. C. *Metecc-95 Proceedings*; 1995; p 434.

CI990069U