Single chains to represent groups of objects

Hiram H. López-Valdez[1], Hermilo Sánchez-Cruz[2]
and Magdalena C. Mascorro-Pantoja[1]

[1]Departamento de Matemáticas y Física

[2]Departamento de Ciencias de la Computación

Centro de Ciencias Básicas

Universidad Autónoma de Aguascalientes (UAA)

Av. Universidad 940, Col. Cd Universitaria, CP. 20131

Aguascalientes, Aguascalientes, México. Fax: (+52 449) 9108401

E-mail: hlopez@correo.uaa.mx

E-mail: hsanchez@correo.uaa.mx

E-mail: mcmascorro@correo.uaa.mx

Abstract

A chain code is a common, compact and size-efficient way to represent the contour shape of an object. When a group of objects is studied using chain codes, previous works require to obtain one chain code for each object. In this paper we assign a single chain to a group of objects, in such a way that all the properties of each object of the group can be recovered from the single chain. In order to achieve higher levels of compression, we propose a lossless method, that consists of representing a group of objects by means of a single chain, and then to apply a context-mixing algorithm. Regarding other methods of compression of the state-of-the-art, our experiments demonstrate that the best compression performance is achieved when our lossless method is applied. In this case more than 15% of a better compression level is reached.

Keywords: Single chains, Chain codes, Contour shapes, Compression, Context-mixing, Huffman, Arithmetic.

1 Introduction

A *chain code* is a common and compact way to represent a contour shape.

The "Freeman chain code", proposed by Freeman in 1961 [8], is known as $F8$-*chain code* ($F8$ for short). It is composed of eight directions and travels through the center of the pixels of a contour shape on the basis of eight connectivity. Each movement direction is codified using a symbol $\alpha \in \{0, 1, 2, \ldots, 7\}$ in counter clockwise direction (see Figure 1).

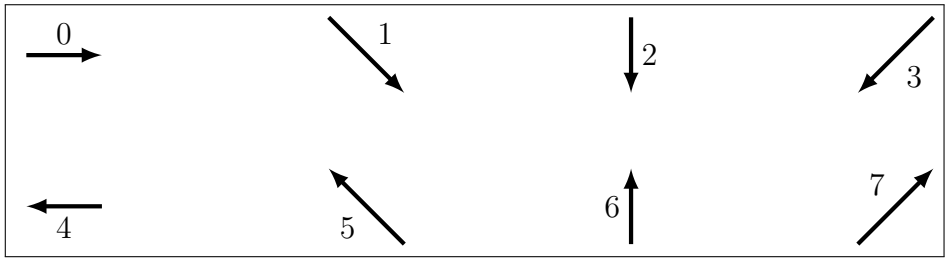

Figure 1: Symbols to encode with $F8$.

Analogous to $F8$, $F4$-*chain code* ($F4$ for short) travels through the edges of the pixels of a contour shape using four connectivity basis. A movement direction is codified using a symbol $\alpha \in \{0, 1, 2, 3\}$ (see Figure 2). It is also known as a *crack* code because it covers the contour shape along edges of border pixels [1, 4, 6, 17, 29].

Figure 2: Symbols to encode with $F4$.

Since Freeman proposed the first chain code [8], a considerable amount of papers using chain codes for a wide variety of issues have appeared. For example, for map representations [1, 18], to look for dominant points [2, 3, 9, 22], and for analysis and document compression [10, 20, 28]. In 1999, Bribiesca [5] proposed the *vertex chain code*, denoted by VCC. Among its most important features, VCC is composed by the symbols $0, 1$ and 2. It is invariant under mirror and rotation transformations and invariant to the

initial point. It has some information between the contour shape and inside information of the object. This code represents the changes made of a contour shape by computing the number of affected pixels (see Figure 3).

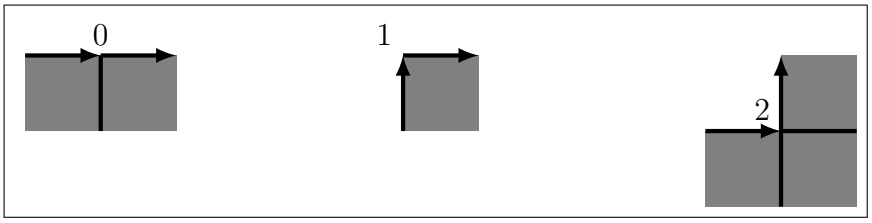

Figure 3: Symbols to encode with VCC.

Sánchez-Cruz and Rodríguez-Dagnino [27] proposed in 2005 the $3OT$-*chain code* ($3OT$ for short). They compared $3OT$ with $F4$ and obtained a better result due to the use of the symbols $0, 1$ and 2 to label the changes generated in relation to orthogonal directions. $3OT$ has the same number of symbols than VCC, however it is composed of 3 vectors to codify the contour: *reference, support* and *change*. The symbol 0 represents no changes between reference and support, 1 represents a change equal to reference and 2 represents a change in contrary sense of the reference (see Figure 4).

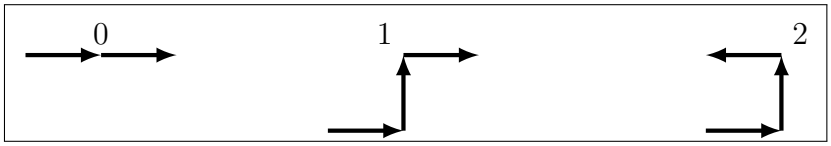

Figure 4: Symbols to encode with $3OT$.

Relied on $F8$, also in 2005, Kui and Žalik [15] proposed a new chain code, that we called here $AF8$-*chain code* ($AF8$ for short). This code is based on changes obtained with every pair of $F8$ code vectors when following the contour, *i.e.* every vector of change in the contour is compared with the previous one, and depending on the angle, a symbol is given (see Figure 5). $AF8$ was compared with $F4$, $F8$, VCC and $3OT$ [23], and the results showed that $AF8$ reported more advantages when using the Huffman algorithm. However, in 2009 the combination $3OT$-Arithmetic coding [25] performed better compression levels than $AF8$-Arithmetic.

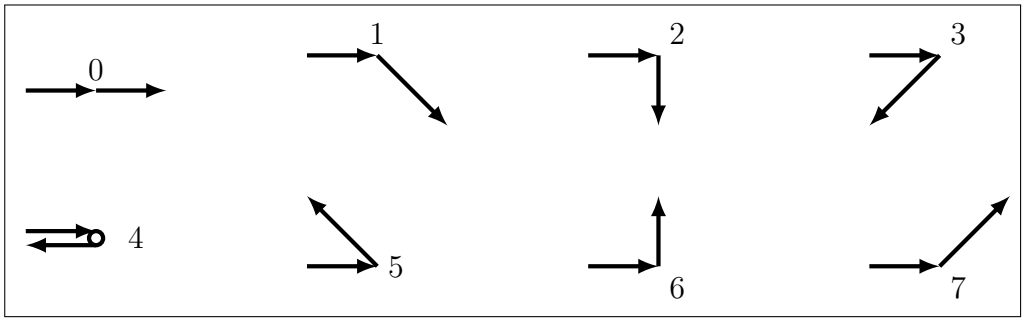

Figure 5: Symbols to encode with $AF8$.

From $F4, VCC, 3OT, F8, AF8$ other chain codes have been derived. The derived codes arise by combining the symbols that appear in the contours, making a probabilistic model to modify the number of bits required to store the coded contour shape. For example, E_VCC, V_VCC and C_VCC-chain codes were proposed in [14]. The E_VCC-chain code was obtained by considering that VCC uses two bits to represent three symbols. The V_VCC-chain code arises by considering a variable-length of VCC. The C_VCC-chain code is based on applying the Huffman algorithm on the VCC code-symbols. On the other hand, the M_3OT-chain code was proposed by considering grouping symbols of $3OT$ [24], whereas $MDF9$-chain code was proposed by considering the $AF8$ patterns in pieces of discrete straight lines of the contour shapes [21]. Recently, NAD-chain code was introduced in [30] and obtained more compression levels in [31]. It is a variation of $AF8$, where instead of using the symbols $0, \ldots, 7$, the authors use four symbols grouped in the following strings of symbols: $0, 2, 310, 311, 312, 301, 300$ and 1 (and labeling the angles according to Figure 6). The $AAF8$-*chain code* is the newest *basic code* (it is not a derivation of another known code), introduced in [26] and inspired in $F8$ and $3OT$. $AAF8$-chain code is composed of three vectors (two angles), in which, regarding the reference vector, a symbol of change direction is given, independently of the support vector direction. It is considered in a basis of eight connectivity. Table 1 shows the basic and the derived codes.

All the chain codes above mentioned have the common particularity that they were designed to represent a contour shape of an isolated object. However, to achieve even higher compression levels, we propose to change the

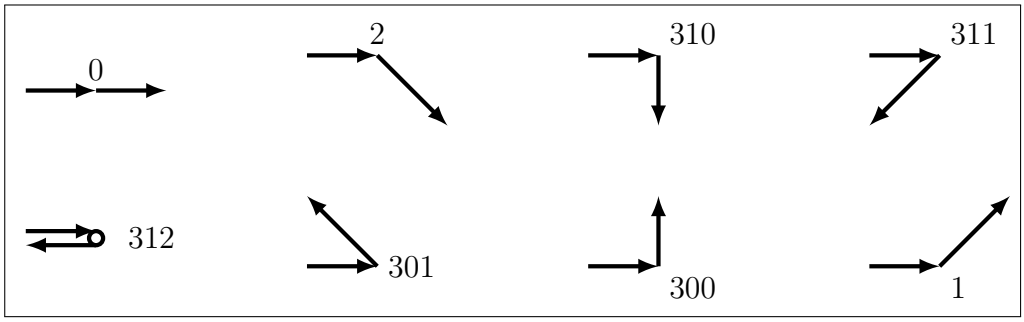

Figure 6: Symbols to encode with NAD.

Year proposed	Basic code	Derived codes
1961	$F8, F4$	
1999	VCC	E_VCC, C_VCC, V_VCC
2005	$3OT$	M_3OT
2005	$AF8$	$MDF9$, NAD
2014	$AAF8$	

Table 1: Basic codes and their derived codes.

paradigm, instead we now represent groups of objects by using a *single chain*. The idea is simple: we concatenate the chain codes of each contour shape of the group. In general the concatenation of chains is always possible, but in order to recover the single chains that form the concatenated chain, it is necessary to have more information, like the length of each single chain and the position of each object. We avoid all this extra storage thanks to the characteristics of the chain codes of objects with pairwise disjoint.

The contents of this work are as follows. In Section 2, some preliminaries are given. In Section 3 the concept of a chain code for a contour shape is extended to a group of objects by using only a single chain. Experiments and results exploiting the idea of using a single chain code for a group of objects are given in Section 4. Finally, in Section 5 some conclusions of this paper are given.

2 Preliminaries

In this section we introduce some concepts, algorithms and notations used throughout this work.

An *alphabet* $\mathcal{A}$ is a set of symbols. A *chain* C of *length* n *over an alphabet* $\mathcal{A}$ is a sequence of n elements of $\mathcal{A}$

$$C := c_1 c_2 \cdots c_n, \quad c_i \in \mathcal{A} \text{ for all } i = 1, \ldots, n.$$

If there is not ambiguity hazard about the alphabet $\mathcal{A}$, we just say that C is a *chain*.

Let $C := c_1 \cdots c_n$ and $T := t_1 \cdots t_m$ be chains of length n and m, respectively. We define the *concatenation* of C and T, denoted by $C \cdot T$ or CT, as the chain:

$$CT := c_1 \cdots c_n t_1 \cdots t_m.$$

Let Γ be a *contour shape* of an object. A *chain code* of Γ is a chain $C(\Gamma)$ that represents (*encode*) the contour shape Γ, this means that it is possible to recover (*decode*) the contour shape Γ using only the information of the chain code $C(\Gamma)$.

The contour shape of an object can be seen as a set of vectors [26], so, a chain code is a representation of a set of vectors. As a consequence, all the basic codes can be handled as vector components, VCC inclusive, as it is explained also in [26].

2.1 Compression algorithms

For data compression, several coding schemes have been used. One of the most popular is the Huffman algorithm [11]. It is probably the most used component of compression methods like GZIP, JPEG and also in compressing of binary objects. Huffman algorithm is simple and easily described in terms of the prefix-code tree generated from the probability that a symbol appears in a message, giving a short code to the most frequent symbol and a larger to that which appear with lower probability. For a code of size n the algorithm runs in $O(n \log n)$ time, however the algorithm has the disadvantage that the code lengths must be rounded to a whole number of bits.

Another of the most known code-schemes, is the Arithmetic algorithm [19]. Arithmetic coding works better with sources of small alphabets. It can be computed by updating the probabilities of each symbol of a message in an

interval between 0 and 1, dividing the interval in proportion to the probability distribution for each input symbol, narrowing the interval by a factor of p_i on each symbol i. This procedure is carried out iteratively until reading the last symbol.

On the other hand, although PPM (Prediction by Partial Matching) algorithm [7] is slower to run, it has better compression performance. The main idea of PPM is to consider previous symbols (context) of a message to generate conditional probability of the current symbol. Because the probability distribution tend to be high it can be used by Arithmetic algorithm to obtain a bit sequence, working much better than Huffman algorithm for this approach.

Content-mixing algorithm is related to PPM in the sense that the compressor is divided into a predictor and an Arithmetic coder, except that it uses a binary alphabet to simplify encoding and model merging [16].

Particularly, PAQ archivers are a family of lossless data compressors based on context-mixing, and they are distributed as free software under the GNU general public license.

Among the PAQ archivers, one of them is called $PAQ8L$, which is used for lossless compression of files with alphabets of few symbols. Throughout the work we use $PAQ8L$ because we want to apply it to chain codes, which come from alphabets of size $2, 3, 4$ or 8.

In last decade, the Huffman and Arithmetic algorithms have been widely used to prove the binary object compression methods [14, 15, 18, 21, 23, 24, 25, 27]. Relied on RLE and $MTFT$ transform to decrease the chain code entropy, Zalik and Lucak [30], and Zalik *et al.* [31] reached the best rate compression to isolated binary objects. On the other hand, methods for document image compression have been carried out in [20] and [28], achieving better compression performance than $JBIG2$ and $DjVu$ by a wide range, where $PAQ8L$ was utilized to compress the resuling file text of such proposed methods.

3 Single chains

In this section we give two algorithms. One of them is to encode a group of N objects. The idea is simple but powerful, to create a single chain formed by the concatenation of the chain codes of each object. The second algorithm is used to decode the single chain that represents the group of N objects.

Algorithm 3.1 *Encode N objects*
Input *N objects $\Gamma_1, \ldots, \Gamma_N$.*
Output *A single chain S that represents the N objects.*

1.- *Encode each of the N objects $\Gamma_1, \ldots, \Gamma_N$ to obtain the chain codes $S_1, \ldots, S_N$.*

2.- *Define the single chain that represents all the objects as $S := S_1 \cdots S_N$.*

Example 3.2 The $F4$-single chain of Figure 7 is

$$\begin{aligned} S_{F4} &:= F4(a) \cdot F4(b), \\ \text{where} \quad F4(\text{a}) &= 000101111212122232330100033222323003 \\ \text{and} \quad F4(\text{b}) &= 001010300121212112332211233233030 3. \end{aligned}$$

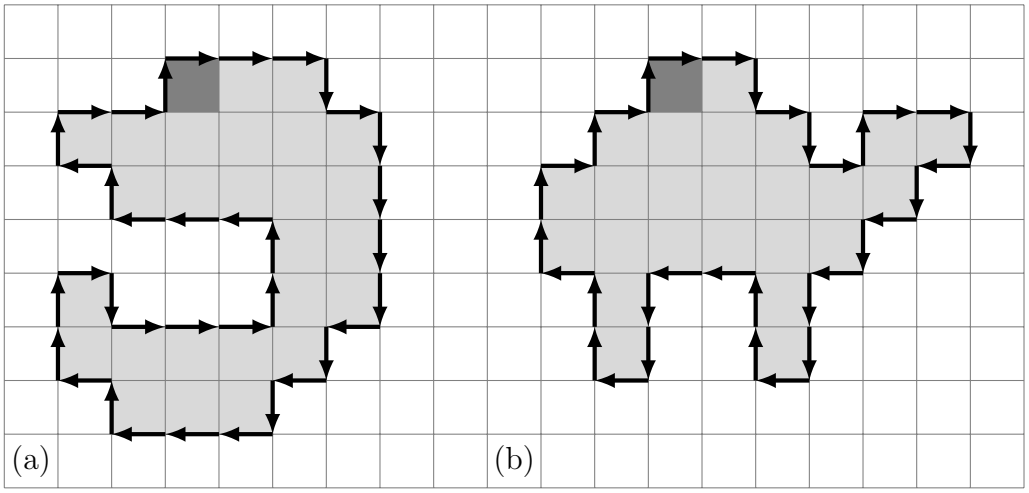

Figure 7: Image composed by two objects.

Observe that given N arbitrary chains, it is always possible to create the concatenation S from these N chains, but in order to recover the original objects, it is necessary to have more information, as the length of each individual chain and the position of each of them in S. We avoid this extra storage exploiting the characteristics of the chain codes.

As we mentioned in Introduction, all the basic codes can be handled as vector components. This means that as a consequence, when a chain code is decoded, we obtain the sequence of vectors that represents the contour shape.

Lemma 3.3 *Let C be a crack code of a contour shape Γ and let $v_1, \ldots, v_n$ be the vectors obtained when C is decoded. There exists a unique vector v_i such that the end of v_i is the beginning of v_1.*

Proof. The existence is easy, it is v_n. Assume v_1 is the vector from the point $(0,0)$ to the point $(1,0)$. Let v_i be a vector such that the end of v_i is the point $(0,0)$, the beginning of v_1. The beginning of v_{i+1} is the point $(0,0)$. v_{i+1} is not the vector to $(-1,0)$ neither $(0,1)$ because v_1 is the leftmost top vector of the contour shape. v_{i+1} is not the vector to $(1,0)$ or $(0,-1)$ because in this case $v_{i+1} = v_1$ or $v_{i+1} = -v_i$, but vectors of a contour shape don not overlap. This means there is not a v_{i+1} vector and thus $v_i = v_n$. □

Remark 3.4 The contour shapes that we are considering in this work are always *closed curves* because they come from a *real object*, so, for this work, the previous Lemma is also valid for chain codes that are not crack codes, i.e., for codes that cover the contour shape traveling through the center of the pixels. The proof of this fact is similar.

The pair of previous results say that if we have a single chain S as the result of applying Algorithm 3.1 to N objects, then, in order to recover the N objects, we need to decode the chain S as a simple chain code, until the end of one vector is the beginning of the first vector, in this moment we have completed an object.

Algorithm 3.5 *Decode N objects*
Input *A single chain S that represents N objects.*
Output *N objects $\Gamma_1, \ldots, \Gamma_N$ represented by S.*

1.- *Let v_1 be the first vector found when S is decoded as a chain code.*

2.- *Define $j := 1, i := 1$ and the set of vectors $V := \{v_1\}$.*

Do

3.- *$i := i + 1$. Let v_i the following found vector when S is decoded as a chain code.*

4.- $V = V \cup \{v_i\}$.

Until *The end of* v_i *is the beginning of* v_1.

5.- *Define* $\Gamma_j := V$ *and* $j = j + 1$.

6.- *Eliminate the symbols of* S *used to find the object* V.

If S *is no empty.* **go** *to Step* 1.-
Else *The sets* Γ_j *represent the objects encoded by* S.

Observe that condition **Until** guarantees that an object was found, because by Lemma 3.3, the first found vector with this property should be the end of an object.

4 Experiments and results

In this section we apply our proposed method of single chains to represent groups of objects in order to obtain high levels of compression. To test our method, we use the 44 images of Figure 8. The first 12 objects of Figure 8 were downloaded from the Repository [12]. The rest of the objects were downloaded from the Repository [13].

4.1 Applications of single chains

Let C be a chain. By $C + PAQ8L$ we mean the result of applying $PAQ8L$ to the chain C. The size, in bytes, of C is denoted by $\text{Size}(C)$ and the size of $C + PAQ8L$ is denoted by $\text{Size}(C + PAQ8L)$. If C is a chain code, to specify what kind of code it is we write $C(P)$, where P is an element of $\mathcal{B} := \{F4, F8, VCC, AF8, 3OT, NAD\}$.

Let $C_1, \ldots, C_N$ be the chain codes of N objects $\Gamma_1, \ldots, \Gamma_N$ with pairwise disjoint and let S be the single chain that represents the N objects. Results of this section show that for all $P \in \mathcal{B}$ we have

$$\text{Size}(S(P) + PAQ8L) < \sum_{i=1}^{N} \text{Size}(C_i(P) + PAQ8L).$$

Moreover, minimum of left side is attached when $P = 3OT$ and in this case, $\text{Size}(S(3OT) + PAQ8L)$ is up to 25% less than the sum $\sum_{i=1}^{N} \text{Size}(C_i(P) +$

$PAQ8L$), for any other $P \in \mathcal{B}$. This means that in order to improve the compression of a figure composed by objects with pairwise disjoint in up to 25%, it is better to create a single chain with $3OT$ that represents all the objects of the image, instead of manage isolated chain codes of each object.

The chain codes of the objects that appear in Figure 8 are denoted by $C_1(P), \ldots, C_{44}(P)$, where $P \in \mathcal{B}$. Table 2 shows the values $\text{Size}(C_i(P) + PAQ8L)$, for $i = 1, \ldots, 44$ and $P \in \mathcal{B}$.

In order to see the behavior of single chains we have formed 12 groups of objects from Figure 8. There are objects with a short length in comparison of other objects. For instance the length of objects $6, 7, 8$ and 9 are $2264, 1236, 5996$ and 1096, respectively. Thus the sum of the lengths of objects $6, 7$ and 9 is less than the length of object 8. For that reason we have decided that the smallest group, the Group 1, is composed by the objects of the first repository, *i.e.* the first 12 objects. Groups $2, 3, 4$ and 5 are composed by objects from 1 to 13, to 14, to 15 and to 16, respectively. Group 6 is formed by objects 1 to 20 and then we added 4 objects to each new group until Group 12, which is formed by the 44 objects.

For $j = 1, \ldots, 12$, let N_j be the number of objects of the group j and let $S_j(P)$ be the single chain that represents the objects of same group j with the codification $P \in \mathcal{B}$. Table 3 shows the values $\text{Size}(S_j(P) + PAQ8L)$ (row "Single chain") and $\sum_{i=1}^{N_j} \text{Size}(C_i(P) + PAQ8L)$ (row "Isolated") for $j = 1, \ldots, 12$ and $P \in \mathcal{B}$. In both cases, computing the sum and using single chains, the lowest value is always given when the codification is made with $3OT$. Column "Percent of improvement" shows the percent of memory storage saved, when a group of objects is encoded with $3OT$ in a single chain rather than have encoded as isolated objects.

Figure 9 shows that the performance by increasing the groups is not linear, but rather, results in a slightly warped curve, *i.e.* decreasing its slope for larger groups. As can be observed, this behavior tells that while larger is the concatenated string (*i.e.* as more objects are compressed), then greater is the performance of the use of single chains.

4.2 Comparing with Huffman and Arithmetic

Among the most used algorithms to compress chains of characters are Huffman, Arithmetic and context-mixing. Regarding the first two of them, versions of variable-length of Huffman and Arithmetic adaptive have also been

Object	$F4$	$F8$	VCC	$AF8$	$3OT$	NAD
			$+$ $PAQ8L$			
Bird	297	296	289	292	292	294
Butterfly	251	248	242	238	242	242
Circle	114	113	91	82	95	83
Fire	224	223	214	215	209	215
Horse	341	340	337	339	326	343
Plane	128	126	127	127	129	132
Shuttle	150	150	146	149	142	152
Spider	487	481	457	440	455	443
Square	45	41	39	44	43	45
Star	105	96	90	90	85	93
Thunder	162	154	157	165	167	170
Tiger	456	458	456	482	439	488
Bull	305	305	297	305	292	309
Mexico	261	268	260	272	258	277
Giraffe	350	347	344	362	338	369
Frog	271	273	263	266	260	271
Omega	170	176	164	164	162	168
VW	199	202	196	208	190	209
Bat	281	283	276	281	268	284
Motorcycle	543	548	531	538	529	543
Camel	277	279	277	285	271	292
Eagle	188	187	184	192	185	192
Dolphin	217	222	210	222	214	227
Gear	276	277	270	275	266	280
Pitbull	203	205	196	203	195	208
Rex	331	336	328	335	322	339
Horse	284	288	278	282	277	289
Boot	179	181	175	175	172	180
Car	157	161	157	161	151	164
Star	192	194	195	204	192	208
Chicken	178	180	173	178	171	182
Jump	236	235	227	236	226	240
Hand	231	236	224	228	224	228
Penguin	161	164	152	157	154	160
Chevy	198	204	207	225	200	228
Dog	231	232	223	223	220	224
Fish	192	197	189	195	185	198
Girl	307	306	292	303	290	305
Spiral	327	321	312	307	309	317
Guitar	87	88	83	90	84	94
Rex	226	229	218	232	217	233
Plane	167	168	165	167	163	167
T-shirt	128	129	121	124	122	122
Jeans	152	151	144	152	141	155
AVERAGE	233.30	234.05	226.73	232.05	224.37	235.5

Table 2: Bytes of individual objects of Figure 8.

Group		$F4$	$F8$	VCC	$AF8$	$3OT$	NAD	Percent of improvement
		$+$ $PAQ8L$						
1 to 12	Isolated	2760	2726	2645	2663	2624	2700	22.26%
	Single chain	2140	2140	2070	2050	2040	2060	
1 to 13	Isolated	3065	3031	2942	2968	2916	3009	21.81%
	Single chain	2380	2370	2310	2290	2280	2300	
1 to 14	Isolated	3326	3299	3202	3240	3174	3286	21.86%
	Single chain	2580	2570	2510	2500	2480	2500	
1 to 15	Isolated	3676	3646	3546	3602	3512	3655	21.70%
	Single chain	2860	2850	2790	2790	2750	2800	
1 to 16	Isolated	3947	3919	3809	3868	3772	3926	21.79%
	Single chain	3070	3050	2990	2990	2950	3000	
1 to 20	Isolated	5140	5128	4976	5059	4921	5130	21.76%
	Single chain	3980	3960	3890	3890	3850	3910	
1 to 24	Isolated	6098	6093	5917	6033	5857	6121	22.66%
	Single chain	4680	4650	4570	4580	4530	4610	
1 to 28	Isolated	7095	7103	6894	7028	6823	7137	23.2%
	Single chain	5400	5360	5290	5300	5240	5330	
1 to 32	Isolated	7858	7873	7646	7807	7563	7931	23.97%
	Single chain	5910	5860	5780	5810	5750	5840	
1 to 36	Isolated	8679	8709	8452	8640	8361	8771	24.53%
	Single chain	6480	6430	6350	6380	6310	6410	
1 to 40	Isolated	9592	9621	9328	9535	9229	9685	24.80%
	Single chain	7130	7090	6980	7020	6940	7050	
1 to 44	Isolated	10265	10298	9976	10210	9872	10362	25.55%
	Single chain	7550	7500	7400	7430	7350	7460	

Table 3: Comparison between single chains and isolated chain codes.
Row "Single chain" of a Group shows the bytes needed to save a single chain that represents all the objects of the group.
Row "Isolated" of a Group shows the bytes needed to save all the chain codes that represent all the objects of the group.
In yellow the lowest number of bytes needed to save the information of a group of image.
Column "Percent of improvement" shows the percent of memory storage saved, when a group of objects is encoded with $3OT$ in a single chain instead of to be encoded as isolated objects.

used. Furthermore, the context-mixing algorithm already has an enhanced version of the Arithmetic algorithm.

Given a group of objects, in terms of compression, we already know by Table 3 that it is better to use a single chain for the whole group and then $PAQ8L$ instead of a chain code for every object and the $PAQ8L$ for every chain code. Now some questions arise with respect to the performance of Huffman and Arithmetic and context-mixing algorithms: What is the behavior of these algorithms when

(a) a single chain for the whole group is obtained and then algorithms for compression are applied to this single chain?, or

(b) isolated chain codes for every object are obtained and then algorithms for compressing are applied to each of these chains?

Which method gives the best compression level? In order to give an answer, we have compared the performance of these three known algorithms. Table 4 shows the number of bytes required to compress individual objects using the Huffman algorithm. On the other hand, Table 5 shows the number of bytes required to compress individual objects using the Arithmetic algorithm.

Finally, Figure 10 shows the behavior of bytes employed when chain codes and a single chain are used to encode the Figure 8. The left graph shows the average of bytes needed to encode an object of Figure 8, when the figure is encoded element by element, i.e., for every object of the figure, a chain code is obtained, and then, an algorithm to compress is applied to every chain code. The right graph shows the average of bytes needed to encode an object of Figure 8, when the figure is encoded in a simple chain and then, an algorithm to compress is applied.

From these results, it can be seen that, considering all the different chain codes, broadly speaking Huffman and Arithmetic algorithms compress slightly better when objects are treated in isolation, moreover, the efficiency remains practically equal to the best performances given by $3OT$, $AF8$ and NAD, in this order. However, it is clear that when context-mixing algorithm is applied, the compression significantly improves the case of single chains by a wide margin.

In the case of compression for chain codes, Huffman has better performance on $AF8$, followed by NAD and then $3OT$. A similar behavior on $3OT$

Object	$F4$	$F8$	VCC	$AF8$	$3OT$	NAD
	+ Huffman					
Bird	594	596	458	377	441	381
Butterfly	503	517	409	330	351	332
Circle	268	284	213	144	190	144
Fire	728	714	558	478	536	479
Horse	1079	1207	629	667	629	789
Plane	566	555	469	306	375	309
Shuttle	309	309	222	182	222	182
Spider	1499	1460	1222	922	1035	923
Square	274	273	137	137	137	139
Star	363	384	288	206	258	208
Thunder	604	451	486	325	360	326
Tiger	1067	1218	638	647	638	748
Bull	531	577	383	339	383	346
Mexico	390	407	305	246	286	253
Giraffe	690	745	495	459	495	472
Frog	426	440	346	273	303	279
Omega	373	424	265	220	265	225
VW	347	361	256	206	256	208
Bat	514	534	409	323	365	327
Motorcycle	1261	1360	969	807	932	815
Camel	449	486	341	297	341	302
Eagle	304	293	251	172	209	175
Dolphin	349	373	267	229	259	233
Gear	373	399	294	258	278	269
Pitbull	332	365	243	204	243	207
Rex	552	598	415	373	415	382
Horse	459	478	369	293	328	298
Boot	365	405	260	227	260	231
Car	301	338	198	178	198	181
Star	295	319	226	187	217	193
Chicken	242	254	192	147	175	149
Jump	345	355	280	216	242	223
Hand	466	501	356	292	348	294
Penguin	208	212	170	125	146	127
Chevy	345	384	222	232	222	250
Dog	391	421	297	240	294	244
Fish	290	313	224	180	213	185
Girl	477	519	369	313	357	322
Spiral	782	830	619	493	559	497
Guitar	58	59	43	39	43	42
Rex	365	374	283	227	270	232
Plane	287	287	223	200	223	201
T-shirt	215	241	152	140	152	141
Jeans	347	370	248	247	248	248
AVERAGE	476.74	499.74	356.69	297.74	333.90	307.06

Table 4: Bytes when Huffman algorithm is applied to the chain code of every object of Figure 8.

Object	$F4$	$F8$	VCC	$AF8$	$3OT$	NAD
	+ Arithmetic					
Bird	584	600	465	346	348	349
Butterfly	511	522	405	289	284	291
Circle	276	292	217	152	143	151
Fire	707	712	565	420	385	420
Horse	1072	1196	448	554	398	670
Plane	575	557	456	279	277	281
Shuttle	308	314	228	164	171	164
Spider	1476	1452	1194	801	784	802
Square	282	282	10	12	12	17
Star	369	386	292	191	193	192
Thunder	595	453	464	289	221	290
Tiger	1075	1209	494	553	421	651
Bull	531	584	388	319	320	325
Mexico	398	410	310	239	233	246
Giraffe	675	731	489	429	388	444
Frog	433	446	344	257	262	263
Omega	381	427	269	201	205	204
VW	343	368	263	190	194	192
Bat	518	537	410	295	282	298
Motorcycle	1267	1353	975	731	698	739
Camel	454	490	348	278	273	282
Eagle	312	299	248	165	174	168
Dolphin	357	377	274	215	200	219
Gear	378	404	299	244	243	258
Pitbull	338	371	250	192	198	195
Rex	556	601	423	354	339	363
Horse	466	482	369	274	274	279
Boot	371	410	265	212	207	215
Car	307	342	191	156	162	157
Star	303	326	233	181	165	188
Chicken	250	261	197	143	150	145
Jump	352	361	280	210	203	215
Hand	470	504	363	265	267	267
Penguin	216	218	172	124	128	126
Chevy	346	390	206	226	176	245
Dog	397	425	305	226	234	229
Fish	298	319	231	175	169	179
Girl	478	521	376	302	298	310
Spiral	791	834	619	443	424	447
Guitar	63	66	49	45	45	47
Rex	368	378	289	219	220	223
Plane	286	286	230	186	172	187
T-shirt	222	247	157	132	125	133
Jeans	342	372	251	225	188	225
AVERAGE	479.48	502.61	347.98	270.52	255.75	279.34

Table 5: Bytes when Arithmetic algorithm is applied to the chain code of every object of Figure 8.

and $AF8$ had already been observed in [23]. On the other hand, Arithmetic performs better for $3OT$, followed by $AF8$, as happened in [25]. Finally, $PAQ8L$ helps all alike, especially when a single chain is considered. Although the performance is almost the same, except for a few bytes, $3OT$ is the best option to compress.

4.3 A single chain is better

Why is better to use a single chain to represent the whole objects than to use a chain of each object? As can be seen from the results obtained in our work, the compression rates are higher if chain codes are concatenated than if we manage them separately. Previous works like [21, 24, 30] have made improvements in the compression of chain codes than come from binary objects, thanks in part to the geometrical analysis of the shape-of-objects, trying to catch the redundancy occurred in the contours, which has resulted in finding pattern substrings and repetitions of bit streams, both to reduce the entropy to improve compression of the chains. However, a deeper statistical analysis as the one conducted by the $PAQ8L$ archiver shows that the search for better patterns can be replaced if better statistical models are sought.

The frequency and context information of the symbols contained in a single string obtained by the Arithmetic and context-mixing to a single chain, is much poorer if all are concatenated. Thus, statistical information given by the frequency of each symbol is better enriched in a longer chain than in a shorter, causing, in the case of mixing context prediction of contexts is markedly improved, and thus the compression. The advantage of using the $PAQ8L$ method is that it uses statistical models to estimate the probability distribution of symbols. It relies mainly on variable order models based on PPM (Prediction by Partial Matching), which perform a statistical analysis of models with different orders at the same time and then uses matching of larger context. It is based on the premise that the larger context which is available for statistical model is the best predictor.

4.4 Comparison with the state-of-the-art

Perhaps a comparison with existing methods in the literature is not entirely fair, since to date these methods have been performed on chains for each of the objects in the image, while we have proposed to use only one chain.

Moreover, in our model we do not set out to reduce the entropy of chains, as is done, for example, in [30, 31], who use RLE and Move-to-front Transform to reduce entropy of the chains. However, highlighting the advantage of encoding with a single chain to all objects, and even without reducing the entropy of the chain, let us consider the average number of bytes required to store the set of objects: for one hand, the best compression levels appeared in [31] for the first 12 objects tested, an average of 198.21 bytes is reported, while using our method our best average is 170 bytes, which is 14.23% of gain. However, as the number of objects increases, it is interesting to see that the gain with our method is increased, as shown in Table 3. Thus, for the case of the 44 objects, we obtained an average of 167 bytes, and our profit would be 15.7%. However, it is important to note that we did not reduce the entropy of the chains, so surely, the gain can be increased if this concept is considered in a future work.

5 Conclusions and further work

As we know, in literature, a number of interesting articles about methods to represent, separately, contours of binary objects have appeared. Authors have carried out an analysis of length and compression performance. In this paper we have used the following strategy: a single chain that represents all the objects of a image has been found and then it has been applied the context-mixing algorithm. The combination of a single chain created with $3OT$ and $PAQ8L$ achieve up to 25% of efficiency about compression in comparison with manage chain codes representing each object of the image, and 15.7% better compression regarding the state-of-the-art.

Our experiments demonstrate that no matter the code, obtaining a single chain and then apply context-mixing algorithm is the best compression method.

Of course, we applied the algorithms of compression on the chain codes without tranforming them to reduce its entropy, however, as a further work, if algorithms to reduce the entropy of a single chain code are implemented, the compression levels can be improved even more.

As a consequence of single chains, giving one more step to what has been done so far in the literature to encode individual objects, we have proposed an new method to improve compression efficiency when considering groups of objects that appear in an image.

6 Acknowledgements

The first author would like to thank to Universidad Autónoma de Aguascalientes for the moral and financial support to obtain a Ph.D. in the Centro de Investigaciones y de Estudios Avanzados del Instituto Politécnico Nacional.

References

[1] Akimov, A. Kolesnikov, P. Franti, Lossless compression of map contours by context tree modeling of chain codes, Pattern Recognition, **40**(3)(2007) 944–952. 2

[2] F. Arrebola, F. Sandoval, Corner detection and curve segmentation by multi resolution chain-code linking, Pattern Recognition, **38**(10)(2005) 1596–1614. 2

[3] H. Beus, S. Tiu, An improved corner detection algorithm based on chain coded plane curves, Pattern Recognition, **20** (1987), 291–296. 2

[4] S. Blackburn, Extraction of color region boundaries, IAPR Workshop on Machine Vision Applications, (1992), 63–66. 2

[5] E. Bribiesca, A New Chain Code, Pattern Recognition, **32** (2)(1999) 235–251. 2

[6] L. Cederberg, Chain-link coding and segmentation for raster scan devices, Comput. Graphics Image Process, **10** (1979) 224–234. 2

[7] J.G. Cleary and I.H. Witten. Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications, **32** (4) (1984) 396–402. 7

[8] H. Freeman, On The Encoding of Arbitrary Geometric Configurations, IRE Trans EC-10 **2** (1961) 260–268. 2

[9] H. Freeman, L. Davis, A Corner-Finding Algorithm for Chain-Coded Curves, IEEE Trans. Comput, **26** (1977) 297–303. 2

[10] S. Hoque, K. Sirlantzis, M. Fairhust, A new chain-code quantization approach enabling high performance handwriting recognition based on multi-classifier schemes, Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003) IEEE, (2003), pp. 834–838. 2

[11] D.A. Huffman. A method for the construction of minimum redundancy codes. Proc. IRE, 40 (1951) 1098–1101. 6

[12] Chain Codes Datasets Repository, http://gemma.uni-mb.si/chaincodes 10

[13] Hsanchez images, in Bilevel_images.zip, https://sites.google.com/site/herssan/images 10

[14] Y. Kui-Liu, W. Wei, P. Wanga, B. Žalik, Compressed vertex chain codes, Pattern Recognition, **40** (11) (2007) 2908–2913. 4, 7

[15] Y. Kui-Liu, B. Žalik, An efficient chain code with Huffman coding, Pattern Recognition, **38**(4)(2005) 553–557. 3, 7

[16] M. Mahoney, Adaptive weighing of context models for lossless data compression, Technical report (CS-2005-16). Florida Institute of Technology, Melbourne, FL, 2005, http://cs.fit.edu/ mmahoney/compression/cs200516.pdf. 7

[17] T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, MD, (1982). 2

[18] A.J. Pinho, Adaptive context-based arithmetic coding of arbitrary contour maps, IEEE Signal Processing Letters, **8** (1)(2001) 4–6. 2, 7

[19] J. Rissanen, Generalized Kraft Inequality and Arithmetic Coding, IBM Journal of Research and Development 20(3) (1976) 198–203. 6

[20] M. A. Rodríguez-Díaz and H. Sánchez-Cruz. Refined fixed double pass binary object classification for document image compression. Digital Signal Processing., **30** (2014), 114–130. 2, 7

[21] H. Sánchez-Cruz, Proposing a new code by considering pieces of discrete straight lines in contour shapes., J. Vis. Commun. Image R, **21**(4) (2010) 311–324. 4, 7, 17

[22] H. Sánchez-Cruz, E. Bribiesca, Polygonal Approximation of Contour Shapes Using Corner Detectors, Journal of Applied Research and Technology, **7** (3)(2009) 275–291. 2

[23] H. Sánchez-Cruz, E. Bribiesca, R. Rodríguez-Dagnino, Efficiency of chain codes to represent binary objects, Pattern Recognition, **40** (6)(2007) 1660–1674. 3, 7, 17

[24] H. Sánchez-Cruz; M. López-Cruces, H. Puga, A proposal Modification of the 3OT Chain Code, Proceedings of the Computer Graphics and Imaging, Innsbruck, Austria, (2008), pp. 6–11. 4, 7, 17

[25] H. Sánchez-Cruz and M. A. Rodríguez Díaz. Coding Long Contour Shapes of Binary Objects. Progress in Pattern Recognition, Image Analysis, Computer Vision and Applications, **5856** (2009) pp 45–52. 3, 7, 17

[26] H. Sánchez-Cruz, H. López-Valdez, Equivalence of chain codes, Journal of Electronic Imaging, **23** (1)(2014) 1–11. 4, 6

[27] H. Sánchez-Cruz, R. Rodríguez-Dagnino, Compressing Bilevel Images by Means of a Three-bit Chain Code, Optical Engineering, SPIE-INT Society Optical Engineering, **44** (2005), no. 9. 3, 7

[28] H. Sánchez-Cruz, M. Rodríguez-Díaz, Binary document image compression using a three-symbol grouped code dictionary, J. Electron. Imaging., **21** (2)(2012) 1–14. 2, 7

[29] G. Wilson, Properties of contour codes, IEE Proc. Vision Image Signal Process, **144** (3) (1997) 145–149. 2

[30] B. Žalik, N. Lukač, Chain code lossless compression using move-to-front transform and adaptive run-length encoding, Signal Processing: Image Communication, **29** (1)(2014) 96–106. 4, 7, 17, 18

[31] B. Žalik D. Mongus, N. Lukač, J. Vis. Commun. Image R. 29 (May, 2015) 8–15 4, 7, 18

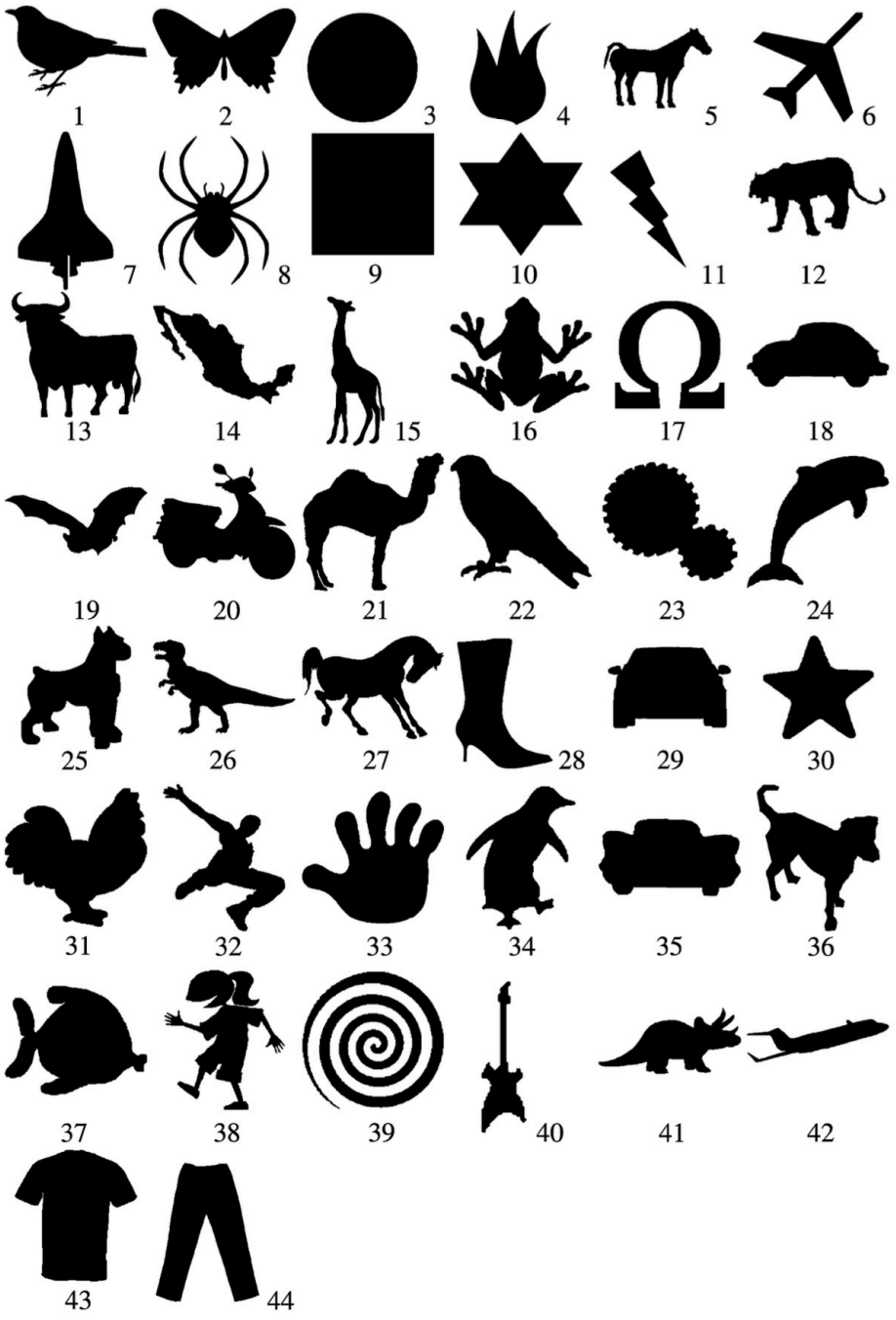

Figure 8: Image composed by 44 objects.

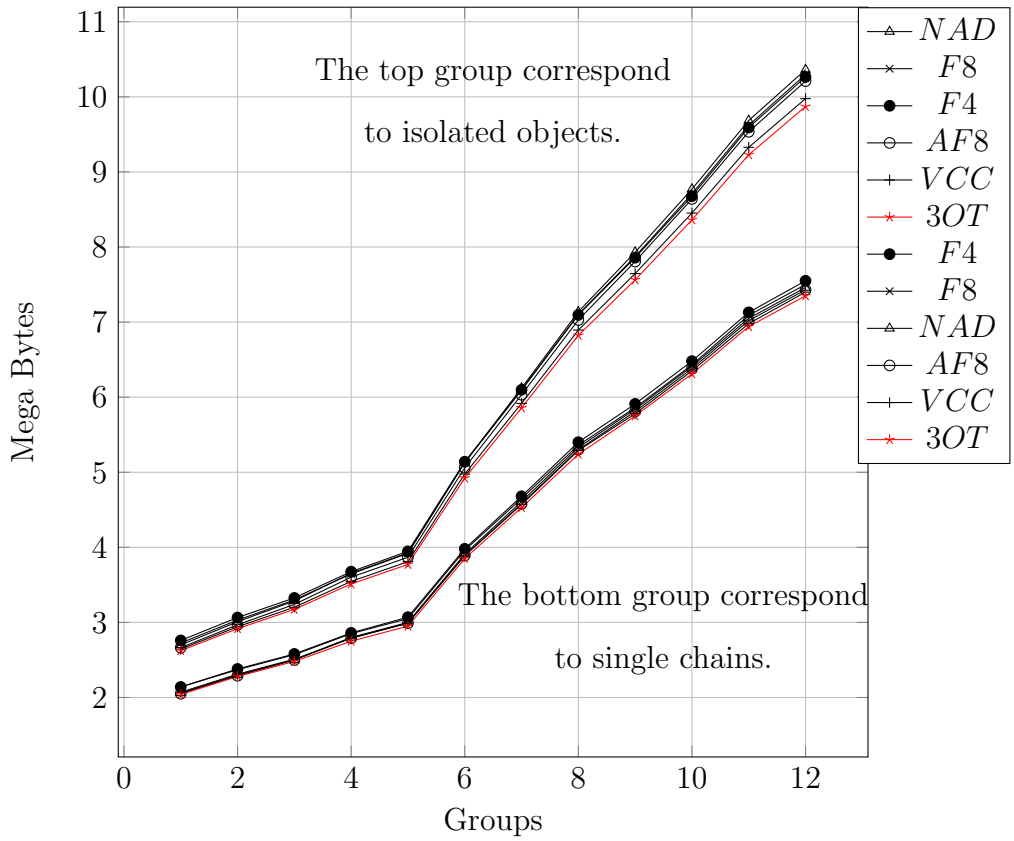

Figure 9: Mega bytes of the twelve tested groups.
The lowest value is attached when all the objects of a image are represented by a $3OT$-single chain.

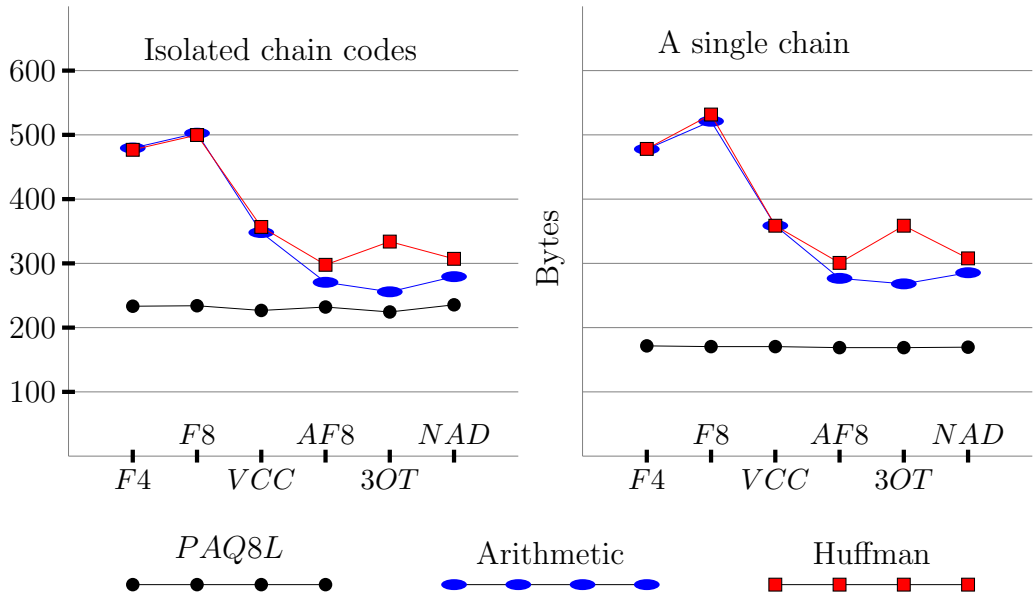

Figure 10: Average of bytes needed to encode an object of Figure 8.
Left: when the figure is encoded element by element, i.e., for every object of the figure, a chain code is obtained, and then, an algorithm to compress is applied to every chain code.
Right: when the figure is encoded in a simple chain and then, an algorithm to compress is applied to this chain.