New Bounds on The Encoding of Planar Triangulations

Stefan Gumhold
WSI-2000-1
ISSN 0946-3852

March 9, 2000

Wilhelm-Schickard-Institut für Informatik
Graphisch-Interaktive Systeme
Auf der Morgenstelle 10, C9
D-72076 Tübingen
Tel.: +49-7071-2975463
Fax: +49-7071-295466

e-mail: gumhold@uni-tuebingen.de

Abstract

Compact encodings of the connectivity of planar triangulations is a very important subject not only in graph theory but also in computer graphics. In 1962 Tutte determined the number of different planar triangulations. From his results follows that the encoding of the connectivity of planar triangulations with three border edges and v vertices consumes in the asymptotic limit for $v \to \infty$ at least $3.245v + o(\log(v))$ bits. Currently the best compression method with guaranteed upper bounds is based on the encoding of $CRLSE$-Edgebreaker strings and consumes no more than 3.67 bits per vertex.

In this report we improve these results to 3.552 bits per vertex. We also present a new coding scheme for the split indices in a different encoding method - the Cut-Border Machine. We describe an encoding with an upper bound of 4.92 bits per vertex. Finally, we introduce a Cut-Border data structure which allows for linear coding and decoding algorithms.

Contents

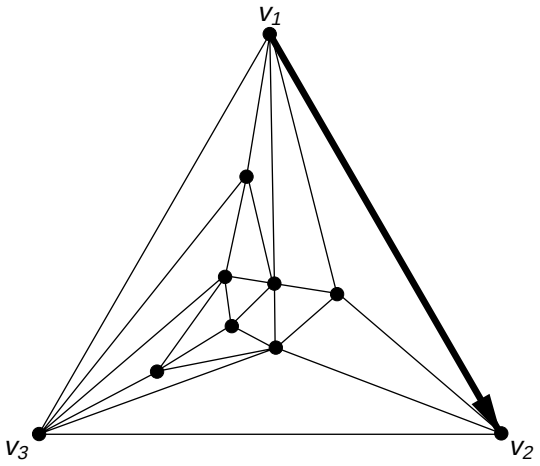


Figure 1: Sample triangulation with the three external vertices v_1, v_2 and v_3.

1 Introduction

In this report we deal with the encoding of planar triangulations with three border edges as defined by Tutte in [10]. Figure 1 shows an example of a planar triangulation with the three border vertices v_1, v_2 and v_3. Two planar triangulations are defined to be equal, if there exists a bijection between their connectivity graphs that maps all border vertices of the first triangulation to the border vertices with the same indices of the second triangulation. Tutte enumerated all different planar triangulations and showed in this way that an optimal encoding uses at least 3.245 bits per vertex for a sufficiently large number of vertices. So far the best encoding schemes [1] and [9] consumed 4 bits. The latter – the Edgebreaker scheme – could be improved to 3.67 bits per vertex [8].

Planar triangulations are a special case of closed manifold triangle meshes where the genus of the triangle mesh is zero. As most encoding schemes for planar triangulations can be extended to manifold triangle meshes with border, the schemes are also important in the representation of surface models and have been studied extensively [2, 6, 3, 5].

The algorithmic scheme of the cut-border machine [5] is very simple. It visits the triangles of an edge-connected component of a triangle mesh in an order defined by the triangle mesh itself. The same traversal is used for encoding and decoding the connectivity – triangle by triangle – into one operation symbol for each triangle. By the use of arithmetic coding and conditional probabilities, the cut-border machine allows to encode the connectivity of typical triangle meshes to an average of 1.9 bits per triangle [4]. As one of the operation symbols – the split symbol – also includes a distance from the current location in the mesh, it is not obvious whether the operation symbols can be encoded in linear time and space in terms of the number of vertices in the triangle mesh. For planar triangulation this is true. We describe a modified version of the cut-border machine in section 2, which encodes the connectivity of planar triangle meshes with less than 4.92 bits per triangle.

The Edgebreaker encoding scheme improves upon the cut-border machine in the way that it allows to avoid encoding the split indices by making the symbol, which closes a compression loop, more expensive. Overall it allows to compress the connectivity with less bits. The Edgebreaker scheme translates the connectivity of a planar triangle mesh into a string over the five symbols $CRLSE$ and the best coding scheme for these strings has been so far the coding presented in [8] to 3.67 bits per vertex. In section 3 we improve this result to 3.552 bits per vertex.

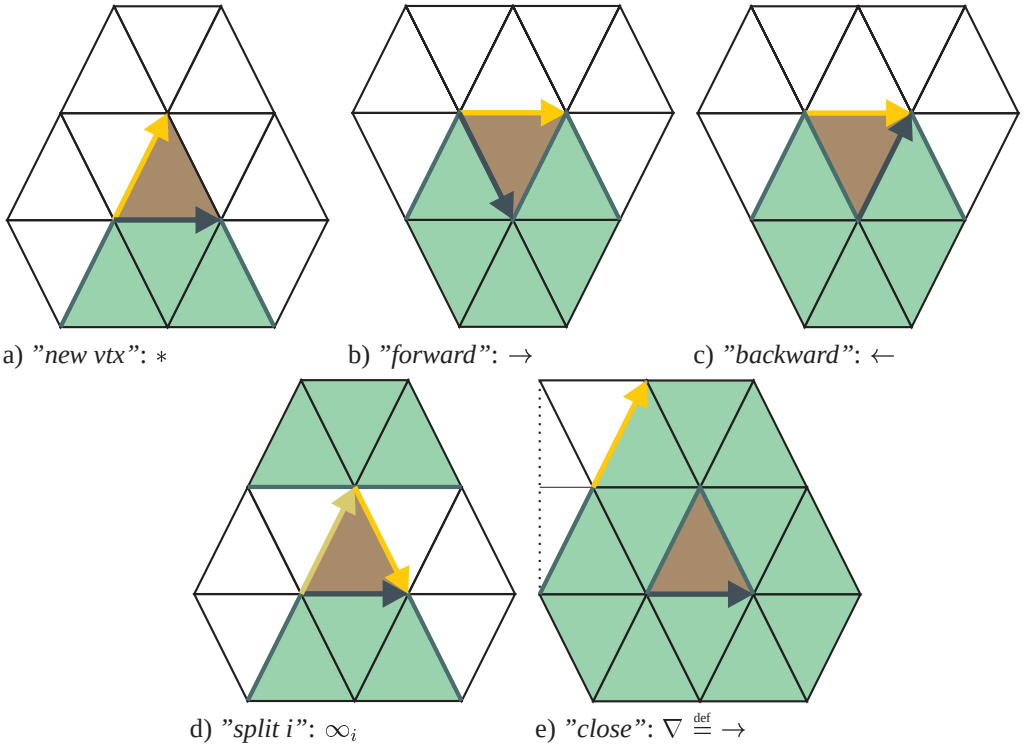


Figure 2: Five possible different operations for encoding one triangle and their corresponding symbols. The so far encoded triangles are shaded in a light green, the cut-border edges are bold and dark green, the gate before the operation is the bold dark arrow, the currently encoded triangle is brown and the new gate(s) after the operation is/are yellow.

2 The Cut-Border Machine

In this section we present a new variant of the Cut-Border Machine which consumes no more than 6 bits per vertex for the encoding of planar triangulations.

First we give a short review of the Cut-Border Machine encoding (section 2.1), followed by a new encoding of the indices of the split operation in section 2.2 and we complete the encoding scheme with the limit of 4.92 bits per vertex in section 2.3. At the end of this section in 2.4 we present a data structure for the cut-border which allows for linear time encoding and decoding with the cut-border machine.

2.1 Cut-Border Machine Encoding

The Cut-Border Machine translates the connectivity of a planar triangulation into a sequence of five different symbols. One of the symbols induces the inclusion of an additional index.

The encoding algorithm is a region growing algorithm, which stores at any time all vertices and edges of the planar triangulation, which divide the so far encoded triangles from the not yet encoded triangles. These vertices and edges form a set of closed loops, which is called the *cut-border*. Before encoding starts, the cut-border is initialized to one loop containing the external edges and vertices.

Triangles are encoded at a specific cut-border edge which is called the *gate*. Each time a triangle has been encoded at the gate, the gate is set to another cut-border edge in a predetermined way. In the beginning the gate is set to the external edge v_1v_2. Figure 2 shows the different operations and their symbols:

- *"new vertex"* ($*$): The newly encoded triangle is formed upon the gate with a new vertex. The gate is set to the right newly introduced cut-border edge.
- *"forward"* ($\rightarrow$): The gate is connected to the next edge on the cut-border and the gate is set to the only newly added cut-border edge.
- *"backward"* ($\leftarrow$): The gate is connected to the previous edge on the cut-border and the gate is set to the only newly added cut-border edge.
- *"split i"* (∞_i): The newly encoded triangle splits the current loop of the cut-border into two loops. The index i specifies the third vertex. Deviating from the index definition in the original work we denote the index of the gate vertices with ± 1, the neighboring vertices with ± 2 and so on, such that the smallest valid vertices are ± 3 and the absolute value of the vertex index defines the length of the shorter cut-border loop. After the split operation the gate is set in both loops to the respectively new cut-border edges.
- *"close"* (∇): The close operation eliminates the current cut-border loop and activates the gate of the cut-border loop, which was split off latestly. The close operation can be encoded with the $\rightarrow$-symbol as both encoding and decoding algorithms know the length of the current cut-border loop.

The encoding is done, when the last available cut-border loop closes. The encoding permutes the vertices. The sample triangulation in Figure 1 is encoded to the string

$$***\rightarrow***\rightarrow\rightarrow\rightarrow*\infty\nabla\rightarrow\nabla,$$

when the edge $v_1 v_2$ is used as initial gate. During decompression the cut-border is updated in exactly the same way as during compression and the planar triangulation with the permuted vertices can be reconstructed. For a more detailed description of the Cut-Border Machine see [5].

2.2 Split Index Encoding in 1.87 Bits per Vertex

In this section we show how to encode the indices for the split operation with proofably no more than 1.87 bits per vertex. For this we describe different variable length coding schemes for signed indices. Theorem 2.1 states that the split indices do not consume more than 1.87 bits per vertex in the cut-border loop independent of the sequence of split operations.

There are two important ideas which contribute to the linear storage space consumption of the split indices. Firstly, the indices are encoded with variable length, such that an index i is encoded with no more than a constant number times the binary logarithm of i. The second idea is that split indices defining vertices on the current cut-border loop before the gate are encoded with negative indices. In this way all split operations, which cut away small parts of the current loop also consume few bits.

2.2.1 Variable Length Index Coding Schemes

Figure 3 illustrates three different simple variable length codings for signed indices. All three codings begin with one bit for the sign of the index. Coding scheme a) encodes each bit with two bits – the bit of the index and an additional control bit specifying whether further bits follow. In scheme a) indices ± 3 and ± 4 are encoded with one sign bit, one index bit and one control bit. The indices $\pm 5 \ldots \pm 10$ are encoded with five bits and so on. Scheme b) packs the index bits in bundles of two index and one control bit. Finally, the third scheme c) mixes both approaches and simple arithmetic coding. The first three bit bundle specifies the two lowest significant bits or equivalently the remainder of the index when divided by four. The second bundle encodes a remainder of a fourth of the index divided by three. Using arithmetic coding this bundle can be encoded with $\mathrm{lb} 3 + 1 \approx 2.585$ bits. The index divided by twelve is encoded with two bit bundles as in scheme a).

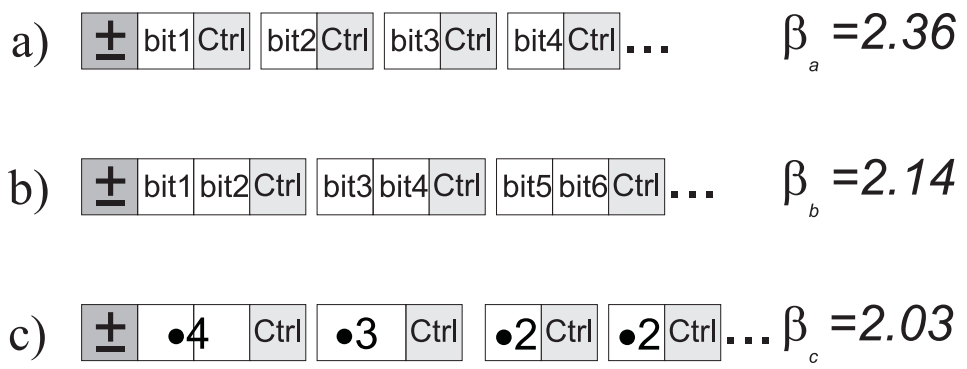


Figure 3: Three different variable length coding schemes for signed indices.

For all of the three schemes in figure 3 the storage space $\mathcal{I}_{\alpha\in\{a,b,c\}}$ for encoding an index i obeys the relation

$$\forall i \geq 3 : \mathcal{I}(i) \leq \beta \mathrm{lb} i + 1, \tag{1}$$

with the different values for β_α as given on the right of figure 3. Let us justify the validity of relation 1 exemplary for scheme c). The problematic indices are the ones which force the usage of a new bundle. In scheme c) these are the indices $\pm 3, \pm 7, \pm 15, \pm 27, \ldots, 12 * 2^k + 3, \ldots$. The first bundle consumes with the sign four bits, the second bundle $\mathrm{lb}3 + 1$ bits and each following bundle further two bits. Thus for the indices ± 3 one must check $4 \leq \beta_c \mathrm{lb}3 + 1$, for the indices ± 7 check $4 + \mathrm{lb}3 + 1 \leq \beta_c \mathrm{lb}7 + 1$ and for the remaining problematic indices relation 1 is valid, iff

$$\forall k \geq 0 : 4 + \mathrm{lb}3 + 1 + 2(k+1) \leq \beta_c \mathrm{lb}\left(12 \cdot 2^k + 3\right) + 1. \tag{2}$$

Solving the equal case of this relation for k yields no real solution and the relation holds true for $k = 0$. Therefore, it must hold true for all values of k. Similar arguments show the validity of relation 1 for the varible length coding schemes a) and b).

The minimal value for β can be achieved by an arithmetic variable length coding scheme. Again the first bit is used for the sign. To the absolute value of each index a subinterval of the unit interval is assigned, the length of which corresponds to the frequency ν_i of the encoded index. In arithmetic coding the consumed bits b_i of a symbol or index i relates to the frequency via the formula

$$\nu_i = 2^{-b_i}. \tag{3}$$

From relation 1 we assume that $b_i = \beta_{\min} \mathrm{lb} i$. As all frequencies of the different indices must sum up to one this yields a condition for $\beta_{\min}$

$$1 = \sum_{i\geq 3} 2^{-\beta_{\min}\mathrm{lb}\, i} = \sum_{i\geq 3} \frac{1}{i^{\beta_{\min}}}. \tag{4}$$

This equation is hard to solve for $\beta_{\min}$, but with the integral $\int \frac{1}{\beta_{\min}}$ we could proof the following relation

$$1.589 < \beta_{\min} < 1.59. \tag{5}$$

An arithmetic coder that achieves $\beta_{\min}$ requires arbitrary precision arithmetic and therefore is not able to encode and decode symbols in constant time. We did not find a simple coding scheme to improve on $\beta_c = 2.03$ but there probably is one. For the remainder we will stick to β_c.

2.2.2 Upper Bound on Index Coding

Theorem 2.1 *For any planar triangle mesh with v vertices the indices of the split operations in the cut-border representation can be encoded with less than $\alpha = 1.87v$ bits.*

Proof: For the coding of the indices of the ∞-symbols we want to use the coding scheme of figure 3 c). The index coding does not depend on the order of the encoded symbols. Therefore we can rearrange the symbols in the following manner. Let

$$**\rightarrow***\infty_4*\rightarrow\leftarrow***\rightarrow\leftarrow**\infty_3\nabla\leftarrow*\rightarrow\nabla***\infty_{-3}*\rightarrow\leftarrow\leftarrow\nabla\nabla$$

be a sample cut-border string. Then we rearrange it by extracting all $*$ symbols to the front:

$$**********\rightarrow\infty_4\rightarrow\leftarrow\rightarrow\leftarrow\infty_3\nabla\leftarrow\rightarrow\nabla\infty_{-3}\rightarrow\leftarrow\leftarrow\nabla\nabla.$$

By going through the cut-border symbols in reverse order, we can keep track of the lengths of the cut-border loops and determine new indices for the split operations. At a close operation the current length is pushed on a stack and a new length of three is generated. At a forward and a backward operation the current loop length is increased by one. At each split operation a new split index is determined for the operation symbol. The absolute value of the new index is smaller length among the current length and the latestly pushed length. It is positive if the current length is smaller, negative otherwise. The split operation is reversed by popping one length, decrementing it by one and adding it to the current length. Finally, each new vertex operation decreases the current length by one, such that if the beginning of the rearranged cut-border symbol sequence is reached, exactly one length remains, which is equal to three. Thus the cut-border sequence with the new split indices is

$$**********\rightarrow\infty_{-8}\rightarrow\leftarrow\rightarrow\leftarrow\infty_3\nabla\leftarrow\rightarrow\nabla\infty_{-3}\rightarrow\leftarrow\leftarrow\nabla\nabla.$$

By construction of the new split indices the resulting sequence is a valid traversal description for the Cut-Border Machine. Furthermore the absolute values of the new indices of each split symbols can only be greater or equal to the original indices as during the reverse tracking of the lengths the missing new vertex operations can only increase the tracked lengths.

There actually exists a planar triangulation producing the rearranged sequence, but this is not important for this proof and requires further mathematical techniques. It is though important that we can now restrict our considerations to the situation when at the beginning of the encoding one cut-border loop with all vertices is built, which is recursively split. Let $\mathcal{L}(v)$ be the maximal storage space consumed by the indices of split operations arising during the encoding of a cut-border loop with v vertices. Then all possible split operations performed on the loop yield the recursive formula for $\mathcal{L}(v)$

$$\mathcal{L}(v) = \max_{i=3}^{i=\left\lceil \frac{v}{2} \right\rceil} \left\{ \mathcal{I}_c(i) + \mathcal{L}(i) + \mathcal{L}(v-i+1) \right\}. \tag{6}$$

$\mathcal{L}(v)$ is the maximum storage space over all possible split operations. The absolute value of the split index i runs from three to $\left\lceil \frac{v}{2} \right\rceil$ and can be encoded with $\mathcal{I}(i)$ bits. Additionally, the maximum storage space of the two remaining loops $\mathcal{L}(i)$ and $\mathcal{L}(v-i+1)$ have to be included. A loop with three or four vertices cannot be split further, for the split of a loop with five vertices there is only one possibility and a loop with six vertices can be split with ∞_3 or ∞_{-3}. Therefore, it holds

$$\mathcal{L}(3) = \mathcal{L}(4) = \mathcal{L}(5) = 0 \qquad\qquad \mathcal{L}(6) = 1. \tag{7}$$

To proof the theorem we have to show the validity of the relation $\mathcal{L}(v) \leq \alpha \cdot v$ for all v. It is obviously true for $v < 7$ if $\alpha \geq 1$. As equation 6 contains the storage space for the index i, we have to prove for the remaining values of v an even stronger upper bound

$$\mathcal{U}(v) \stackrel{\text{def}}{=} \alpha \cdot v - (\beta \mathrm{lb} v + \gamma), \tag{8}$$

i.e. we want to prove the relation

$$\forall v \geq 7 : \mathcal{L}(v) \leq \mathcal{U}(v). \tag{9}$$

To abbreviate the proof we just guess the values of α, β and γ to be $1.87, \beta_c = 2.03$ and 4.92. The values were chosen in a way that $\mathcal{U}(6) = 1$. But we still have to consider all special cases where i or v assumes the values $3, 4$ or 5.

Cut-border loops of length six to nine can be split into two loops of length less than six. For the $6, 7, 8$ and 9-vertex loops we introduce special coding schemes for the indices of split operations performed on these loops. This is possible as the cut-border machine knows the length of the current loop during encoding and decoding. For a split operation on a loop with $l \geq 6$ vertices, there are $l-4$ different split indices. Using arithmetic coding the different indices can be encoded with $\mathrm{lb}(l-4)$ bits. A seven vertex loop can be split in three different ways (∞_3, ∞_{-3} and ∞_4). The three cases can be encoded with $\mathrm{lb}3 < 1.585$ bits. The resulting loops are of length less or equal five and therefore $\mathcal{L}(7) < 1.585$ as no further indices need to be encoded. The eight vertex loop has four possible splits, which are encodable in two bits. As the eight vertex loop can be split into a three vertex and a six vertex loop, an additional bit might be needed. Thus $\mathcal{L}(8) = 2 + \mathcal{L}(6) = 3$. Finally, the nine vertex loop split index consumes $\mathrm{lb}5$ bits and we get $\mathcal{L}(9) = \mathrm{lb}5 + \mathcal{L}(7) < 3.91$. Gathering these cases we state

$$\begin{aligned} \mathcal{L}(7) < 1.585 &< \mathcal{U}(7) > 2.4 \\ \mathcal{L}(8) = 3 &< \mathcal{U}(8) > 3.8 \\ \mathcal{L}(9) < 3.91 &< \mathcal{U}(9) > 5.3 \end{aligned}$$

Next we consider the inductive step for the cases where i is less or equal five in equation 6 and show

$$\forall i \in \{3,4,5\} : \mathcal{I}_c(i) + \mathcal{L}(i) + \mathcal{L}(v-i+1) \leq \mathcal{U}(v). \tag{10}$$

In the following we apply equations 1, 7 and 8 as denoted above the less or equal signs

$$\begin{aligned} &\forall i \in \{3,4,5\} : \\ &\qquad \mathcal{I}_c(i) + \mathcal{L}(i) + \mathcal{L}(v-i+1) \\ &\overset{(1,7)}{\leq} \beta_c \mathrm{lb} i + 1 + \mathcal{L}(v-i+1) \\ &\overset{(9,8)}{\leq} \beta_c \mathrm{lb} i + 1 + \alpha \cdot (v-i+1) - (\beta_c \mathrm{lb}(v-i+1) + \gamma) \\ &\overset{(8)}{=} \mathcal{U}(v) - (i-1)\alpha + \mathrm{lb} i + 1 + \beta_c \mathrm{lb} \frac{v}{v-i+1}. \end{aligned}$$

From the last expression we learn that equation 10 holds true, iff anything in the last expression besides $\mathcal{U}(v)$ is less or equal zero

$$\begin{aligned} (10) &\iff \\ \forall i \in \{3,4,5\} : (i-1)\alpha &\geq \mathrm{lb} i + 1 + \beta_c \mathrm{lb} \frac{v}{v-i+1} \\ &\iff \\ \forall i \in \{3,4,5\} : v &\geq (i-1) / \left(1 - 2^{-\frac{(i-1)\alpha - \mathrm{lb} i - 1}{\beta_c}}\right) \\ &\impliedby \\ v &\geq 7. \end{aligned}$$

The last step was performed by plugging in the values for α, β_c and i and proves equation 10.

With all the preliminaries we can finally attack equation 9 and prove it by induction. That is we validate equation 9 for v under the assumption that equation 9 holds true for all $v' < v$. We start with equation 6:

$$\begin{aligned} \mathcal{L}(v) &= \max_{i=3}^{i=\lceil \frac{v}{2} \rceil} \{\mathcal{I}_c(i) + \mathcal{L}(i) + \mathcal{L}(v-i+1)\} \\ &\overset{(10)}{\leq} \max \left\{ \mathcal{U}(v), \max_{i=6}^{i=\lceil \frac{v}{2} \rceil} \{\mathcal{I}_c(i) + \mathcal{L}(i) + \mathcal{L}(v-i+1)\} \right\} \end{aligned}$$

If $\mathcal{U}(v)$ is the maximum, equation 9 holds true and therefore we do neglect the outer maximum in what follows. We can now plug in the inductive assumption:

$$\begin{aligned}\mathcal{L}(v) &\overset{(9)}{\leq} \max_{i=6}^{i=\lceil \frac{v}{2} \rceil} \{\mathcal{I}_c(i) + \mathcal{U}(i) + \mathcal{U}(v-i+1)\} \\ &\overset{(1,8)}{\leq} \mathcal{U}(v) + 1 + \alpha - \gamma + \beta_c \max_{i=6}^{i=\lceil \frac{v}{2} \rceil} \left\{ \mathrm{lb} \frac{v}{v-i+1} \right\}.\end{aligned}$$

We skipped some simple algebra in the second step. The term inside the logarithmic function evaluates always to a value greater one and less than two, because $i-1$ is always less than $v/2$. Therefore the logarithmic expression is always less as one and we get

$$\mathcal{L}(v) \leq \mathcal{U}(v) + 1 + \alpha - \gamma + \beta_c = \mathcal{U}(v),$$

what proves equation 9 and together with equation 7 the theorem.

□

For a better variable length index coding scheme with $\beta = \beta_{\min}$ theorem 2.1 can be improved to 1.54 bits per vertex with only one change in the proof: the special coding must also be applied to ten vertex loops.

2.3 Upper Bound of 4.92 Bits per Vertex

It is left to discuss how the different symbols $*, \rightarrow, \leftarrow, \infty$ and ∇ are encoded. As there are $t = 2v-2$ triangles in a planar mesh, there are $2v-3$ symbols to be encoded. As there are $v-3$ $*$-symbols, the $*$-symbol is encoded with one bit, contributing $v-3$ bits to the overall storage costs. There are two further constraints which can be exploited.

1. If the current cut-border loop has three vertices only the new vertex and the close operations are possible.

2. After a new vertex operation no backward may follow.

As the close operation may only arise in the situation of the first constraint, it can be encoded with one bit, i.e. if the current cut-border loop contains only three vertices, one bit encodes whether a close or a new vertex operation follows.

To fully exploit both constraints, we define the constant τ as the number of bits, which are consumed for encoding a triangle not introduced by a new vertex operation. With this definition all symbols without the split indices are encoded with less than $(\tau+1)v$ bits. Table 1 shows all possible cases of subsequent symbols which might arise at the current gate position and the number of bits, that may be consumed by each case in the columns "bound". Here we included the observation

bits	case	bound	case	bound
2	$\leftarrow$	τ	$\rightarrow$	τ
3	∞	$2\tau-1$	$*\rightarrow$	$\tau+1$
4	$*\infty$	2τ	$**\rightarrow$	$\tau+2$
5	$**\infty$	$2\tau+1$	$***\rightarrow$	$\tau+3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 1:

that each ∞-operation forces one ∇-operation, which can be encoded with one bit. Thus each ∞-operation may consume $2\tau - 1$ bits.

From the maximum number of bits b in the column "bound", the frequency ν of each case can be computed from 2^{-b}. The sum over the frequencies of all cases in table 1 must yield one. This condition yields an equation for τ and τ computes to 2, which is rather an accident but simplifies the encoding of symbols. In table 1 the cases are arranged in rows, such that each row contains all cases with the same number of bits given in the first column. For each bit number starting with two bits, there are exactly two cases. Thus a case is encoded in two parts. First the row r is encoded with $r-1$ one bits followed by a zero bit and then one bit selects the column. We can conclude the whole discussion with the following theorem.

Theorem 2.2 *A planar triangle mesh with v vertices can be encoded with the cut-border encoding scheme with less than $4.92v$ bits.*

If a better variable length index coding scheme is found an upper bound of 4.5400 bits per vertex can be shown.

As the coding of the symbols without the split indices consumes only 3 bits per vertex and an optimal coding consumes at least 3.245 bits, it is worth while to check whether the split indices can be encoded even better. The best coding scheme we can imaging, which does not allow for a linear time coding algorithm exploits the knowledge of the current loop length better and works as follows: we use the loop storage space $\mathcal{L}(3 \ldots 6)$ from equation 7. For the loop of length $v > 6$ we assume $\mathcal{L}(v)$ is the same for all possible split operations. We distinguish the split operation with index $i = 3 \ldots v-2$, such that we do not need to handle a sign. Finally, we calculate $\mathcal{L}(v)$ recursively from the arithmetic coding equation which sets the sum of the frequencies of all possible split operations for one loop length v to one:

$$\forall v > 6 : 1 = \sum_{i=3}^{v-2} 2^{-(a-\mathcal{L}(i)-\mathcal{L}(v+1-i))}.$$

We calculated the fraction $\mathcal{L}(v)/v$ for $v = 3 \ldots 100$ and the resulting plot converges to 1.15 and crosses 1. Thus it seems to be impossible to encode a planar triangulation with less than four bits per vertex with the cut-border encoding scheme.

2.4 Linear Coding and Decoding Time

In the original work [5] the data structure for the cut-border has been implemented as linked lists and the split operation could neither be encoded nor decoded in linear time yielding a running time worse than $O(n \ln n)$. The advantage of the linked list data structure has been that the gate could be updated after each operation arbitrarily. In this way the compression rates could be improved for regular meshes. In this paper we focus on an optimal upper bound for the encoding of planar triangulations and do not intend to compress regular triangle meshes with better ratios. For this goal we present a new data structure which ensures constant time updates after each operation but restricts the choice of the gate position.

Figure 4 shows the new data structure. It consists of a vertex stack, a loop stack and two markers. The vertex stack is extended by the markers p_s and p_e defining the current loop. On the loop stack marker pairs of loops, which have been pushed during a split operation, are stored together with the split vertex. The current gate is always the edge between the vertex before the p_e marker and the vertex after the p_s marker. In figure 4 the different loops on the vertex stack are visually separated with black blocks, just for the convenience of the reader.

Figure 5 shows how the data structure is updated after each of the five operations.

new vertex: There are two possible situations for a new vertex operation. Either (figure 4 a) the p_s marker points to the beginning of the vertex stack / is identical to the latestly pushed p'_e marker

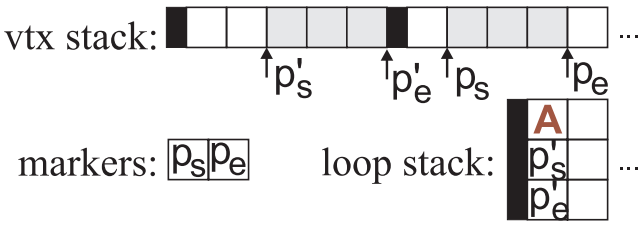
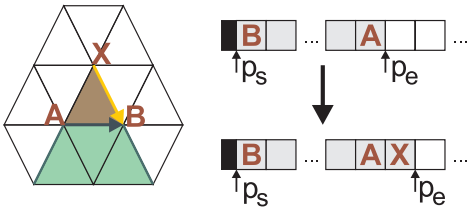


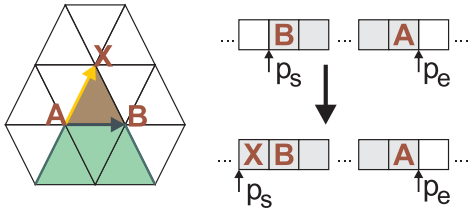
Figure 4: Cut-border data structure, consists of two stacks and two pointers.

(denoted by the black block) or (figure 4 b) some vertices have been removed at the front of the stack by forward operations. In case a) the new vertex X is pushed onto the vertex stack and the p_e marker is moved behind vertex X. Thus the new gate was implicitly chosen to be XB. In case b) the new vertex X is inserted before p_s and p_s is moved before X. In this way the empty vertices at the beginning of the loop can be refilled and the cut-border stack is kept as small as possible. After case b) the gate is chosen differently as before. In order to be able to consider the constraints needed for the efficient encoding, we switch the symbols of the forward and backward operations after case b) has been arosen.

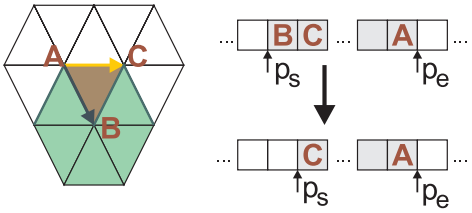
forward: During the forward operation (figure 4 c) the vertex B is removed from the beginning of the current loop by moving the p_s marker one position to the right. The empty vertex location will be filled during the next new vertex operation. The gate is implicitly chosen to be AC.



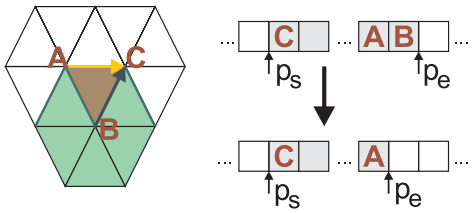
a) $*$: append



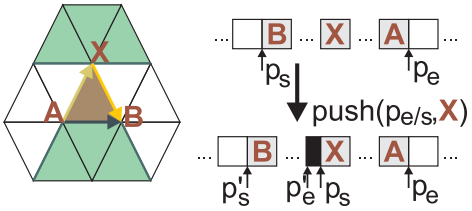
b) $*$: insert



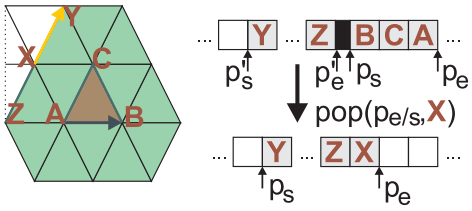
c) $\rightarrow$: delete previous



d) $\leftarrow$: delete next



e) ∞: push loop



f) ∇: pop loop

Figure 5: Update of the optimized data structure after all five different cut-border operations.

backward: The backward operation (figure 4 d) just pops one vertex from the stack and moves p_e one position to the left, implicitly choosing the gate to be AC.

split: During a split operation (figure 4 e) the current loop is split into two loops by setting p_s to p'_s and introducing two new markers p'_e and p_s before the split vertex X. The primed markers together with the split vertex X are pushed onto the loop stack such that the pushed loop can be restored after the right loop has been encoded. In order to find X in constant time we store with each mesh vertex a pointer to the corresponding cut-border vertex. This works as no operation invalidates any of the pointers, i.e. as long as a cut-border vertex exists, it is at the same vertex stack location. The gate location of the right loop after the split operation is AX and the pushed gate location of the other loop is XB.

close: The close operation (figure 4 f) eliminates the current loop of three vertex indices from the vertex stack. If the loop stack is empty, the planar triangulation has been completely encoded / decoded. Otherwise the top loop on the loop stack is popped together with the corresponding split vertex, which is inserted after the popped p_e marker and the marker is moved once to the right. Please notice, that the X vertex will be on the same vertex stack location as during the encoding of the removed loop and therefore the pointer from the mesh vertex to the cut-border vertex needs no update.

Finally, the computation of the split indices can be performed by simple pointer arithmetic in constant time[1]. It is obvious, that the variable length coding schemes in figure 3 can be encoded and decoded in time linear to the encoding size and we can conclude this section with the following theorem:

Theorem 2.3 *With the cut-border machine a panar triangulation with v vertices[2] can be encoded to less than $4.92v$ bits in $O(v)$ time and the triangulation can be decoded in $O(v)$ time.*

[1]Here we assumed that the number of vertices in the mesh can be represented by the pointer/integer format of the used computer. This must be the case as we need to store the mesh itself somehow.

[2], that can be stored in memory,

3 Improved Edgebreaker Coding

cut-border		edgebreaker	
name	symb.	name	symb.
new vertex	$*$		C
forward	$\rightarrow$	right	R
backward	$\leftarrow$	left	L
split	∞	split	S
close	∇	end	E

Table 2: Translations between the cut-border and the edgebreaker symbols.

The edgebreaker encoding scheme is nearly equivalent to the cut-border machine except that it does not encode the split indices. Table 2 gives translations between the cut-border symbols and the edgebreaker symbols. By encoding the C-symbol with one bit and all other symbols with three bit, the edgebreaker scheme allows to encode any planar triangulation with no more than four bits per vertex[3]. In [8] the upper bound for the storage space is improved to 3.67 bits per vertex. Section 3.1 describes a simple linear line decoding algorithm also called Spiral Reversi [7]. Then we show in section 3.2 how to encode the edgebreaker symbol string with no more than 3.557 bits per vertex and in section 3.3 with 3.552 using a reverse coding scheme.

3.1 Spiral Reversi Decoding of Edgebreaker Symbols

The decoding algorithm as described in [7] interprets the string of edgebreaker symbols in reverse order. For this it deletes the tailing E-symbol from the string, reverts the string and adds an S-symbol to the end. Thus the string representation of the sample in figure 1 is mapped to

$$CCCFCCCFFFCSEFE \mapsto FESCFFFCCCFCCCS$$

In this way the tailing S-symbol marks the end of the string representation.

The decoding algorithm knows, that the encoding ended with an E-operation. Therefore, it recreates the triangle encoded by the tailing E-operation and initializes the cut-border to one single loop surrounding this triangle. The gate location is chosen arbitrarily and dummy vertex indices are used for all newly introduced vertices.

Then the decoding algorithm iterates through the symbols of the string representation and performs all encoded operations in an inverse fashion. This is clear for the C-, R- and L-operations: in figure 2 interpret the white triangles as the so far decoded triangles, the brown triangle as the currently decoded triangle, the yellow arrow as the current gate location and the dark green arrow as the new gate location after the next triangle has been decoded. After each C-operation the neighborhood of the new vertex is completely decoded and a new final vertex index is assigned to this vertex. In this way the vertices are assigned indices in the reverse order in which they have been encoded.

It only remains to explain the decoding of the E- and S-operations. Each time an E-symbol is encountered, the current loop is pushed onto a stack together with the current gate location and a new loop is generated with a single triangle and an arbitrary gate location. When an S-symbol is found, one loop together with its gate location is popped from the stack and is merged together with the current loop at the current gate location inserting a triangle as depicted in figure 2 d). In figure 2 d) the greenish yellow arrow (the left one) represents the gate of the popped loop in the decoding algorithm and the pushed loop in the encoding algorithm. During the merging the two dummy vertices of the different loops where the two gates touch are identified in all incident triangles. The

[3]Each symbol introduces one triangle. There are twice as many triangles as vertices. Each vertex corresponds to exactly one C symbol. This sums up to $v + 3v = 4v$ bits.

new gate location is set according to the dark green arrow in figure 2 d). If an S-symbol is found when the stack of loops is empty, this S-symbol is the marker of the end of the string and decoding is complete.

Theorem 3.1 *The connectivity of planar triangulation with v vertices and 3 external edges can be encoded with a unique string of length $2v$ over five different symbols in linear time in v, from which the original connectivity can be decoded also in time linear in v.*

3.2 3.557 Bits per Vertex Encoding of Edgebreaker String

In this section we use two constraints of the edgebreaker string to improve the $4v$ bit encoding to $\mathrm{lb}3 + 2 \approx 3.586$ and then to 3.557 bits per vertex.

The first constraint is that after a C-symbol neither an L- nor an E-symbol may follow, as otherwise the two successive symbols would encode the same triangle twice. We can use this constraint in the following manner. First we notice that the C-symbols constitute half of all the symbols and therefore should be encoded with one bit or in an arithmetic setting with a frequency of $\frac{1}{2}$. Next we assume that all other symbols may consume the same number of bits τ and therefore correspond to the same frequency $\nu_\tau = 2^{-\tau} \in [0, 1]$. Table 3 shows the different possible cases (compare table 1 for the Cut-Border Machine cases). If we use arithmetic coding, we end up with the equation

$$1 = 4\nu_\tau + 2\nu_\tau \sum_{i \geq 1} \frac{1}{2^i} = 6\nu_\tau. \tag{11}$$

From this ν_τ computes to $\frac{1}{6}$ and τ to $\mathrm{lb}6 < 2.585$. As there are v C-symbols and v symbols of other type and the C-symbols consume 1 bit and the others τ bit, we end up with less than 3.585 bits per vertex. Coding and decoding of the symbols is also very simple. The unit interval is subdivided into 6 equal sized sub-intervals assigned to the cases R, L, S, E, C^+R, C^+S. In the C^+-cases the number of C-symbols is encoded with the same number of one bits followed by a zero bit.

The second constraint, which has not been considered yet, makes use of the knowledge of the length of the cut-border. The observation is that during encoding the current cut-border loop is at least of length three. Thus two successive C symbols increase the length to at least five and a following R will reduce the length to not less than four, what prohibits a following E symbol, as this can only appear if the current cut-border loop has exactly length three. To take this constraint into account, we introduce the concept of the *conditional unity*. Let us introduce this concept with the example of the first constraint. For the first symbol there are the five possibilities $CRLSE$. But after a C-symbol has been encoded, only three possibilities are left (CRS). Thus under the condition of a preceding C-symbol the unity is split into the frequencies for the symbols CRS. In

case	bits	freq.	case	bits	freq.
L	τ	$\nu\tau$	E	τ	$\nu\tau$
R	τ	$\nu\tau$	S	τ	$\nu\tau$
CR	$\tau+1$	$\frac{1}{2}\nu\tau$	CS	$\tau+1$	$\frac{1}{2}\nu\tau$
CCR	$\tau+2$	$\frac{1}{4}\nu\tau$	CCS	$\tau+2$	$\frac{1}{4}\nu\tau$
$CCCR$	$\tau+3$	$\frac{1}{8}\nu\tau$	$CCCS$	$\tau+3$	$\frac{1}{8}\nu\tau$
$\vdots$	$\vdots$	$\vdots$	$\vdots$		

Table 3: Different cases of possible the edgebreaker sequences considering the constraint, that no E- nor L-symbol may follow upon C.

we can re-formulate the said as follows

$$1 = 4\nu_\tau + \frac{1}{2}\mathbf{1}_C \tag{12}$$

$$\mathbf{1}_C = 2\nu_\tau + \frac{1}{2}\mathbf{1}_C. \tag{13}$$

In these equations $\mathbf{1}_C$ denotes the conditional unity for the condition that a C symbol is preceding. Solving the system of equations yields the same result $\nu_\tau = \frac{1}{6}$. With the concept of the conditional unity all equations look just like a partitioning of the unit interval.

cond.		*follow*	*equation*
3		$RLSEC$	$1 = 4\nu_\tau + \frac{1}{2}\mathbf{1}_{4,C}$
4	C	RSC	$\mathbf{1}_{4,C} = 2\nu_\tau + \frac{1}{2}\mathbf{1}_{5,C}$
5	C	RSC	$\mathbf{1}_{5,C} = (\mathbf{1}_4 + 1)\,\nu_\tau + \frac{1}{2}\mathbf{1}_{6,C}$
6	C	RSC	$\mathbf{1}_{6,C} = (\mathbf{1}_5 + 1)\,\nu_\tau + \frac{1}{2}\mathbf{1}_{7,C}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	C	RSC	$\mathbf{1}_{i,C} = (\mathbf{1}_{i-1} + 1)\,\nu_\tau + \frac{1}{2}\mathbf{1}_{i+1,C}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
4		$RLSC$	$\mathbf{1}_4 = 3\nu_\tau + \frac{1}{2}\mathbf{1}_{5,C}$
5		$RLSC$	$\mathbf{1}_5 = (2 \cdot \mathbf{1}_4 + 1)\,\nu_\tau + \frac{1}{2}\mathbf{1}_{6,C}$
6		$RLSC$	$\mathbf{1}_6 = (2 \cdot \mathbf{1}_5 + 1)\,\nu_\tau + \frac{1}{2}\mathbf{1}_{7,C}$
$\vdots$		$\vdots$	$\vdots$
i		$RLSC$	$\mathbf{1}_i = (2 \cdot \mathbf{1}_{i-1} + 1)\,\nu_\tau + \frac{1}{2}\mathbf{1}_{i+1,C}$
$\vdots$		$\vdots$	$\vdots$

Table 4: Conditional unities including first and second constraint.

With all this preliminaries we can attack the second constraint. Here we not only want to account for preceding C-symbols but also for the minimal length of the current cut-border loop. If the minimal length is for example known to be at least four, the conditional unity is denoted by $\mathbf{1}_{4,C}$ if a C-symbol is preceding and $\mathbf{1}_4$ otherwise. For each condition we just have to enumerate all possible succeeding symbols and the resulting post-conditions and can easily write down the corresponding equation as shown in table 4. The first column contains the known minimal length of the current cut-border loop. The second column tells whether a C-symbol is preceding. The third column enumerates all symbols which can appear under the precondition in the same order they are accounted for in the equations in the last column. Let us explain the equation for the conditional unity if no C is preceding and the minimal loop length is six. Then the symbols $RLSC$ may follow, not E as the current loop is too long. The symbols R and L each yield the post-condition of no preceding C and a minimal loop length of five as both of them eliminate one vertex from the cut-border. This is represented by the term $2 \cdot \mathbf{1}_5 \cdot \nu_\tau$. After a split symbol nothing about the length of the current cut-border loop is known, which is accounted for be the term ν_τ. Finally, a new vertex operation C will increase the loop length by one and therefore the right side of the equation also contains the term $\frac{1}{2}\mathbf{1}_{7,C}$, where the $\frac{1}{2}$ represents the frequency of the C-symbol.

We solved the set of equations with a computer algebra program for different restrictions $l_{\max}$ of the minimal loop length. If we for example restrict the minimal loop length by six, we replace in the

$l_{\max}$	τ	ν_τ	$\mathbf{1}_{4,C}$	$\mathbf{1}_{5,C}$	$\mathbf{1}_{6,C}$
5	2.56256	.169275	.645798	.614494	
6	2.55779	.169846	.641316	.603290	.591385
7	2.55677	.169955	.640358	.600895	.586446
9	2.55651	.169986	.640111	.600277	.585172

$l_{\max}$	$\mathbf{1}_{7,C}$	$\mathbf{1}_{8,C}$	$\mathbf{1}_{9,C}$	$\mathbf{1}_4$	$\mathbf{1}_5$
5				.815073	
6				.811152	.741053
7	.581919			.810313	.738612
9	.579478	.5773922	.576739	.810098	.737983

$l_{\max}$	$\mathbf{1}_6$	$\mathbf{1}_7$	$\mathbf{1}_8$		
5					
6					
7	.711977				
9	.710618	.700273	.696429		

Table 5: Results for different restrictions $l_{\max}$ for the precondition on the current loop length.

equation for $\mathbf{1}_{6,C}$ the $\mathbf{1}_{7,C}$ on the right side with $\mathbf{1}_{6,C}$. This is valid as if the loop length is at least of length seven then it is also longer than six. Table 5 gives the results for different values of $l_{\max}$ and also the values for the conditional unities, which allow to build an arithmetic coder. τ converges very fast to 2.557 and we conclude this section with the following theorem.

Theorem 3.2 *With the encoding scheme described in this section a planar triangulation with v vertices can be encoded and decoded in linear time to less than $3.557v$ bits.*

3.3 Using More Constraints for 3.552 Bits per Vertex Encoding

If we apply the techniques of the previous section to the reverse decoding, we can consider more constraints. During reverse decoding each E operation starts a cut-border loop with three vertices. During decoding each new vertex operation C decreases the current cut-border loop by one vertex and the R and L operations increase the loop by one. A C operation is never allowed, when the current cut-border loop is of length three or if the previous operation was an L operation. Thus in order to keep track of the current cut-border loop length we define two types of conditional unities. For all $i \geq 3 : \mathbf{1}'_i$ is the unity under the condition that the current cut-border loop is of length i and $\mathbf{1}'_{i,L}$ is the unity under the additional condition that the previous symbol was L. Finally, we define $\mathbf{1}'_L$ to be the unity when nothing about the loop length is known except that L has been the previous symbol. Using these unities to built a system of equations similar to the one in table 4 we can achieve again a value of $\tau = 2.557$.

After a split operation during reverse decoding the last two cut-border loops are merged and we do not know anything about the loop length with the so far described approach. But it is actually feasible to keep track of the lengths of two successive cut-border loops as long as they are short and we define the conditional unities $\mathbf{1}'_{j,i}$ and $\mathbf{1}'_{j,i,L}$ for all all $i, j \geq 3$. The index i represents the loop length of the current cut-border loop and j of the previous loop. With the new unities a subsequence of $EESCCC$ can be correctly excluded as the first two E operations would create two loops of length three each, the split concatenates these two loops to one loop of length five and the three new vertex operations C would reduce the loop length to two what is not possible.

Table 6 gives the different kinds of equations parametrized over the loop lengths of the current loop with length i and the previous loop with length j. In order to calculate the different conditional

cond.		*follow*	*equation*
		$RLSEC$	$\mathbf{1} = (1 + \mathbf{1}'_L + 1 + \mathbf{1}'_3)\nu_\tau + \frac{1}{2}$
3		$RLSE$	$\mathbf{1}'_3 = (\mathbf{1}'_4 + \mathbf{1}'_{4,L} + 1 + \mathbf{1}'_{3,3})\nu_\tau$
i		$RLSEC$	$\mathbf{1}'_i = (\mathbf{1}'_{i+1} + \mathbf{1}'_{i+1,L} + 1 + \mathbf{1}'_{i,3})\nu_\tau + \frac{1}{2}\mathbf{1}'_{i-1}$
$j, 3$		$RLSE$	$\mathbf{1}'_{j,3} = (\mathbf{1}'_{j,4} + \mathbf{1}'_{j,4,L} + \mathbf{1}'_{j+2} + \mathbf{1}'_{3,3})\nu_\tau$
j, i		$RLSEC$	$\mathbf{1}'_{j,i} = (\mathbf{1}'_{j,i+1} + \mathbf{1}'_{j,i+1,L} + \mathbf{1}'_{j+i-1} + \mathbf{1}'_{i,3})\nu_\tau + \frac{1}{2}\mathbf{1}'_{j,i-1}$
	L	$RLSE$	$\mathbf{1}'_L = (1 + \mathbf{1}'_L + 1 + \mathbf{1}'_3)\nu_\tau$
3	L	$RLSE$	$\mathbf{1}'_{3,L} = (\mathbf{1}'_4 + \mathbf{1}'_{4,L} + 1 + \mathbf{1}'_{33})\nu_\tau$
i	L	$RLSE$	$\mathbf{1}'_{i,L} = (\mathbf{1}'_{i+1} + \mathbf{1}'_{i+1,L} + 1 + \mathbf{1}'_{i,3})\nu_\tau$
j, i	L	$RLSEC$	$\mathbf{1}'_{j,i,L} = (\mathbf{1}'_{j,i+1} + \mathbf{1}'_{j,i+1,L} + \mathbf{1}'_{j+i-1} + \mathbf{1}'_{i,3})\nu_\tau$

Table 6: Conditional unities accounting for the constraints induced by two successive loops.

$l_{\max}$	$ll_{\max}$	τ	ν_τ
7	4	2.55197	0.17052
11	5	2.55122	0.17061
17	6	2.55102	0.17063

Table 7: Results for different restrictions $l_{\max}$ for the precondition on the current loop length and $ll_{\max}$ for the precondition on the length of the previous and the current loop.

unities and the value for ν_τ we restricted the maximal loop length for $\mathbf{1}'_i$ to $i \leq l_{\max}$ and the loop lengths for $\mathbf{1}'_{j,i}$ to $i \leq ll_{\max}$. The resulting values for ν_τ and τ are shown in table 7.

Theorem 3.3 *With the encoding scheme described in this section a planar triangulation with v vertices can be encoded and decoded in linear time to less than $3.552v$ bits.*

4 Conclusion and Future Work

In this report we showed that the Cut-Border Machine can be modified in order to encoded and decode planar triangulations in linear time to less than 4.92 bits per vertex, although the split indices are encoded. Without the split indices the Cut-Border Machine allows to encode the operation symbols to 3 bits per vertex what is less than the optimum of 3.245 bits per vertex. Therefore we estimated the minimal storage space consumed by the split indices. The estimation of about 1.15 bits per vertex suggests that it is not possible to achieve an optimal encoding of planar triangulations with the Cut-Border Machine.

Therefore we showed how to improve the encoding of the Edgebreaker operation symbols by the use of arithmetic coding, which allows to take more constraints into account in an optimal fashion. In this way we could present a coding scheme which consumes no more than 3.552 bits per vertex. Here we considered the constraints implied by the length of the current cut-border loop and the length of the previously pushed loop and, finally, the well known constraint that an L-symbol may not be preceded by a C-symbol.

In future work we want to investigate what constraints are not yet considered, how they could improve the encoding and whether they can be considered at all. We also try to find a coding scheme between the Cut-Border Machine and the Edgebreaker which has a better potential for optimal encoding.

References

[1] R. C. Chuang, A. Garg, X. He, and M. Kao. Compact encodings of planar graphs via canonical orderings and multiple parentheses. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, pages 118–129, 1998.

[2] Michael F. Deering. Geometry compression. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 13–20. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[3] M. Denny and C. Sohler. Encoding a triangulation as a permutation of its point set. In *Proceedings of the 9th Canadian Conference on Computational Geometry*, pages 39–43, August 1997. held in Ontario, August 11-14.

[4] Stefan Gumhold. Improved cut-border machine for triangle mesh compression. In *Erlangen Workshop '99 on Vision, Modeling and Visualization*, Erlangen, Germany, November 1999. IEEE Signal Processing Society.

[5] Stefan Gumhold and Wolfgang Straßer. Real time compression of triangle mesh connectivity. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 133–140. ACM SIGGRAPH, Addison Wesley, July 1998.

[6] Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[7] Martin Isenburg and Jack Snoeyink. Spirale reversi: Reverse decoding of the edgebreaker encoding. Technical Report TR-99-08, Department of Computer Science, University of British Columbia, October 4 1999. Mon, 04 Oct 1999 17:52:00 GMT.

[8] Davis King and Jarek Rossignac. Guaranteed 3.67V bit encoding of planar triangle graphs. pages 146–149, 1999.

[9] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1), 1999.

[10] W. Tutte. A census of planar triangulations. *Canadian Journal of Mathemetics*, 14:21–38, 1962.