

AN APPLICATION OF VOGEL'S ALGORITHM

ANDREW BARTHOLOMEW

February 2011

ABSTRACT

Building on work by Yamada [3], Vogel describes in [1] how a Reidemeister type II move may be used to manipulate a link diagram in such a way that an equivalent braid word may be read from it. In [2] Choi described a computer algorithm that performs the moves prescribed by Vogel to yield a braid word from a Gauss code description of a classical link diagram. Here we describe an algorithm to produce braid words for arbitrary link diagrams from a labeled peer code. This includes classical, virtual, welded and flat knots and links.

The contents of the paper are as follows:

1. Preliminary Definitions and Concepts
2. Seifert Diagrams and Seifert Graphs
3. Vogel's Move
4. Description of the Algorithm

References

1 Preliminary Definitions and Concepts

1.1 Peer Codes

Let D be an oriented diagram of a knot or link with k components and let $\mathcal{I}(D)$ be the immersion of D in the plane. Thus $\mathcal{I}(D)$ is a 4-regular plane graph with n vertices and $2n$ edges, which inherits an orientation from D . Each vertex in $\mathcal{I}(D)$ has two incoming edges and two outgoing edges, with respect to the orientation. The two incoming edges are called *peers*, or *peer edges*.

Choose an initial component of D and a basepoint b for that component that does not map to a vertex in $\mathcal{I}(D)$. Number the edges in $\mathcal{I}(D)$ consecutively from zero starting at the edge containing image of b and following the path determined by the orientation of D . This results in crossings of $\mathcal{I}(D)$ whose incoming edges have both been numbered having one peer assigned an even number and the other peer an odd number.

Now proceed successively through the remaining components of D in a similar manner, each time choosing the next component so that its image in $\mathcal{I}(D)$ involves a crossing that has already had a number assigned to one of its incoming edges. Extend the numbering to the corresponding edges in $\mathcal{I}(D)$ choosing a basepoint on the next component so that crossings whose incoming edges have both been

numbered have one peer assigned an even number and the other peer an odd number. It is always possible to choose such a basepoint.

The numbering of edges in $\mathcal{I}(D)$ induces a unique numbering of the vertices $0, \dots, n - 1$ by assigning the number i to the vertex at which edge $2i$ terminates. We refer to the edge $2i$ as the *naming edge* for vertex i .

From the numbering of the edges of $\mathcal{I}(D)$ we may write a list of the odd numbered peers of the naming edges in the order determined by the vertex numbering. We separate this list into the peers of those naming edges that are associated with the same component of D

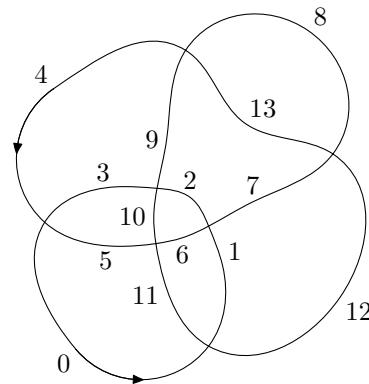


Figure 1.

For example, for the immersion and edge numbering shown in Figure 1, the list of odd peers is 11 9, 3 1 13 5 7

1.2 Type I and type II crossings

There are two possibilites for the relative numbering of incoming edges at a vertex of $\mathcal{I}(D)$, as shown in Figure 2.

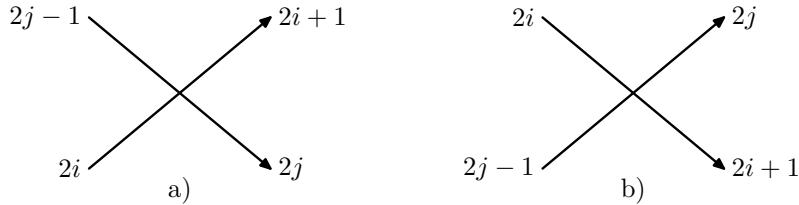


Figure 2.

Definition 1.1 A crossing of the type shown in Figure 2 a) is called a *type I* crossing and a crossing of the type shown in Figure 2 b) is called a *type II* crossing.

The list of odd peers may be supplemented to record the type of each crossing by writing each odd peer associated with a type I crossing as a negative number.

For the immersion in Figure 1 we get $-11\ 9,\ -3\ 1\ -13\ 5\ -7$. We refer to this code as a *peer code*.

1.3 Labelled Peer Codes

We may describe a knot or link diagram D fully by giving its peer code together with a set of labels that describe each crossing. For classical crossings we assign the label + if the naming edge in $\mathcal{I}(D)$ forms part of the over-arc of the crossing and the label – if it forms part of the under-arc. For virtual or welded crossings we assign the label * and for flat crossings we assign the label #.

Definition 1.2 A *labelled peer code* for a diagram D is a peer code for D together with a set of labels, one for each crossing. It is written as the peer code followed by a '/' character, followed in turn by the labels. The labels appear in the order corresponding to the numbering of the vertices.

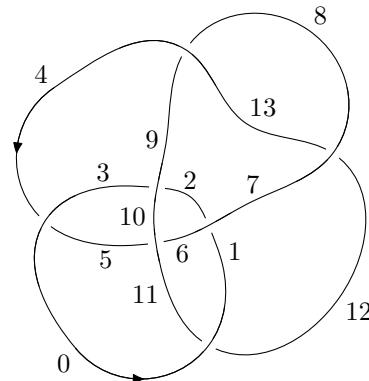


Figure 3.

Thus, the labelled peer code for the link and numbering shown in Figure 3 is

$$-11\ 9,\ -3\ 1\ -13\ 5\ -7 / +\ -\ -\ +\ -\ +\ -$$

1.4 Seifert Circles and Link Diagrams

Sieffert circles are features of the immersion of a link diagram in the plane and are independent of the nature of the crossings, as be seen from Figure 4.

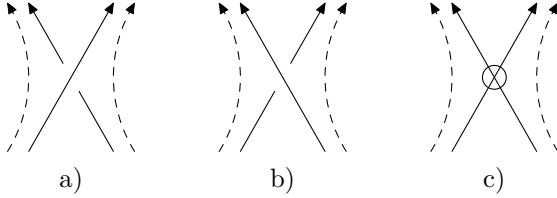


Figure 4. Seifert circles from different crossing types

Therefore for an arbitrary link diagram we may determine Seifert circles from the shadow immersion using the standard approach of leaving a crossing on the other strand than the one on which we arrived, without consideration of the nature of the crossing in the diagram.

Definition 1.3 In a link diagram, each pair of Seifert circles s_i, s_j cobounds a unique annulus A_{ij} on S^2 and we say that s_i and s_j are *coherently oriented* if they represent the same element of $H_1(A_{ij}; \mathbb{Z})$, otherwise they are *incoherently oriented*.

Notice that two connected Seifert circles in a link diagram are necessarily coherently oriented.

It was Yamada [3] who first noticed that if all the Seifert circles in a diagram D are coherently oriented then the diagram is essentially that of a closed braid, and our observations above regarding Seifert circles and virtual crossings show that this is the case for arbitrary links not just classical ones. Vogel's algorithm involves manipulating a link diagram until all the Seifert circles are coherently oriented.

2 Seifert Diagrams and Seifert Graphs

Definition 2.1 The *Seifert graph* Γ of a Seifert diagram D is a graph where $V(\Gamma)$ is in one-to-one correspondance with the set of Seifert circles in D and $E(\Gamma)$ is in one-to-one correspondance with the set of crossings in D . An edge in Γ connects two vertices iff the corresponding crossing joins the two corresponding Seifert circles in D .

Of course, if two Seifert circles are joined by multiple crossings in a Seifert diagram, then the corresponding vertices in a Seifert graph will be connected by multiple edges.

Definition 2.2 The *reduced Seifert graph* Γ' of a Seifert Diagram D is a graph derived from a Seifert graph Γ of D with the following properties:

- (i) $V(\Gamma') = V(\Gamma)$
- (ii) Two vertices are joined by an edge in Γ' iff they are joined by an edge in Γ
- (iii) There is at most one edge joining two vertices in Γ'

2.1 Properties Inherited from Labelled Peer Codes

We make the following observations about the Seifert circles associated with a diagram described by a labelled peer code:

- (i) The crossings between two connected Seifert circles are either all type I or all type II crossings
- (ii) If C_1 and C_2 are the components of the complement of a Seifert circle s then all crossings connecting s to another Seifert circle in C_i have the same type.
- (iii) Seifert circles are comprised wholly of even labelled edges or wholly of odd labelled edges;

The first observation allows us to assign a type to each edge in Γ' depending on whether the Seifert circles corresponding to the vertices in its boundary are connected by type I or type II crossings.

Definition 2.3 For each even numbered edge e in a connected immersion $\mathcal{I}(D)$ there is a sequence of edges e_0, \dots, e_k with $e = e_0 = e_k$ called the *left turning cycle* obtained by turning left at each crossing we encounter as we trace around $\mathcal{I}(D)$ starting by moving along e following the orientation of $\mathcal{I}(D)$. Similarly we define the *right turning cycle* for e as the corresponding sequence obtained by always turning right. We define left and right turning cycles for odd numbered edges in the same way but require that we start by moving along the edge against the orientation of $\mathcal{I}(D)$.

Clearly every edge in a left (right) turning cycle will determine the same left (right) turning cycle.

3 Vogel's Move

Vogel's move, introduced in [1], is to perform a Reidemeister type II move between two incoherently oriented Seifert circles resulting in an equivalent diagram that contains two additional crossings but has fewer pairs of incoherently oriented Seifert circles.

The Vogel move amalgamates two incoherently oriented Seifert circles s_1 and s_2 into one, which we shall refer to as the *amalgamated* Seifert circle, s_a , to which all of the crossings originally connected to s_1 and s_2 are now connected. It also creates a Seifert circle connected only to the two crossings introduced by the Vogel move, which we shall refer to as the *new* Seifert circle, s_n . All other Seifert circles are left unchanged, so the total number of Seifert circles is unchanged but whereas s_1 and s_2 were incoherently oriented s_a and s_n are coherently oriented. Clearly any Seifert circle that was originally connected to both s_1 and s_2 is now connected to one fewer Seifert circle in the revised diagram, so the order of the corresponding vertex in the revised Γ' is reduced by one.

The operation of the algorithm below hinges on the following lemma.

Lemma 3.1 Let v be a vertex in the reduced Seifert graph Γ' of a diagram. If two of the edges in $\text{star}(v)$ have the same type then two of the Seifert circles corresponding to the vertices in $\text{star}(v) - v$ are incoherently oriented and admit a Vogel move between them.

Proof Let s be the Seifert circle in the Seifert diagram D corresponding to v . If $e_1, e_2 \in \text{star}(v)$ have the same type then there are at least two Seifert circles connected to s that lie in the same component of its complement. Any two such Seifert circles are incoherently oriented with respect to each other since they are both coherently oriented with s . Furthermore, two of these Seifert circles must intersect the same component of the complement of the Seifert diagram and therefore admit a Vogel move. \square

Corollary 3.2 If $v \in V(\Gamma')$ has order greater than 2 then two of the Seifert circles corresponding to the vertices of $\text{star}(v) - v$ admit a Vogel move. \square

After a finite number of Vogel moves the vertices of $V(\Gamma')$ will all have order ≤ 2 and each vertex of order 2 will lie in the boundary of one type I edge and one type II edge. At this point the diagram cannot contain a pair of incoherently oriented Seifert circles, and so is a closed braid.

4 Description of the Algorithm

The algorithm takes as input a labelled peer code. The strategy is to perform the following steps:

- A1. Create a code table describing the peer edges for each crossing, the crossing types (type I or II), the terminating and originating odd and even edges, and the labels assigned to each crossing (+, -, * or #).
- A2. Determine the set of distinct left and right turning cycles from the code table.

- A3. Determine the Seifert circles as cyclic lists of crossings *and* as cyclic lists of edges.
- A4. Construct the reduced Seifert graph Γ' from the Seifert circles.
- A5. Determine from Γ' whether another Vogel move is required, if not go to A8.
- A6. Evaluate the labelled peer code corresponding to the diagram after the Vogel move has been performed.
- A7. Go to A1.
- A8. Number the Seifert circles as braid strands and read the braid word from the Seifert circles, strand numbering and crossing labels.

The following subsections will describe each of these steps in more detail, using the diagram shown in Figure 4 as an example.

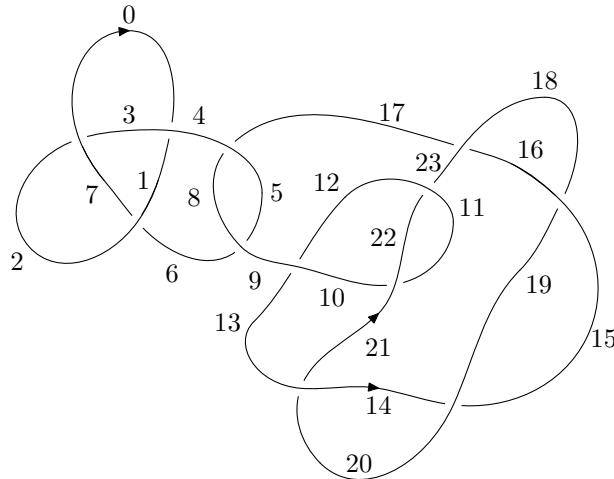


Figure 5.

4.1 A1, Creating the Code Table

The labelled peer code for the diagram and numbering shown in Figure 5 is

3 7 - 17 1, -5 21 9 - 19 - 23, -15 - 13 11 / - - + - + - - - - - - -

We create a table known as the *code table* that records in its rows the various pieces of information carried by the labelled peer code. For each crossing i we

record the peer of the even edge $2i$, the peer of the odd edge $2i + 1$, the type of crossing, type I or type II, and the label associated with the crossing. We also store for each crossing the component number to which the naming edge $2i$ belongs, the components being implicitly numbered from zero by the edge numbering. From this information we are able to derive the terminating and originating odd and even edges at each crossing.

Thus, for the peer immersion code above the code table is:

| Crossing | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|-----------------------|-----------|-----------|----------|-----------|----------|-----------|-----------|----------|----------|----------|----------|-----------|
| Peer of even edge | 3 | 7 | 17 | 1 | 5 | 21 | 9 | 19 | 23 | 15 | 13 | 11 |
| Peer of odd edge | 6 | 0 | 8 | 2 | 12 | 22 | 20 | 18 | 4 | 14 | 10 | 16 |
| Crossing type | <i>II</i> | <i>II</i> | <i>I</i> | <i>II</i> | <i>I</i> | <i>II</i> | <i>II</i> | <i>I</i> | <i>I</i> | <i>I</i> | <i>I</i> | <i>II</i> |
| Crossing label | – | – | + | – | + | – | – | – | – | – | – | – |
| Component | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| Even terminating edge | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 |
| Odd terminating edge | 3 | 7 | 17 | 1 | 5 | 21 | 9 | 19 | 23 | 15 | 13 | 11 |
| Even originating edge | 4 | 0 | 8 | 2 | 6 | 22 | 10 | 20 | 18 | 16 | 14 | 12 |
| Odd originating edge | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 |

4.2 A2, Determine the Left and Right Turning Cycles

From the code table we may easily determine the edge we encounter when turning left or right at a crossing. Thus to determine the set of distinct left turning cycles we start by following edge 0 to arrive at crossing 0. In our example, crossing 0 is a type II crossing and so arriving on the even terminating edge means that turning left we encounter the even originating edge, 4. We continue following the orientation of the diagram and arrive at crossing 2, which is a type I crossing, so turning left means we encounter the odd terminating edge, 17. As we follow edge 17 we are moving against the orientation and arrive at crossing $(17 - 1)/2 = 8$. This is another type I crossing, so arriving on the originating odd edge means that turning left we next encounter the even originating edge, 18. Continuing in this manner until we return to edge 0, we are able to determine the first left turning cycle $(0, 4, 17, 18, 15, 20, 13, 9, 6, 2)$.

The other left turning cycles are determined similarly, starting from an edge not yet included in a left turning cycle and moving with the orientation along even numbered edges or against it along odd numbered edges. Once we have a complete set of distinct left turning cycles we may use the same procedure to determine the right turning cycles.

In our example, the set of left and right turning cycles are:

| Left turning cycles | Right turning cycles |
|-------------------------------------|----------------------|
| (0, 4, 17, 18, 15, 20, 13, 9, 6, 2) | (0, 3) |
| (1, 3, 7) | (1, 4, 8, 6) |
| (5, 8) | (2, 7) |
| (10, 22, 12) | (5, 17, 23, 12, 9) |
| (11, 21, 14, 19, 16, 23) | (10, 21, 13) |
| | (11, 22) |
| | (14, 20) |
| | (15, 19) |
| | (16, 18) |

4.3 A3, Determining Seifert Circles From an Immersion Code

In a similar manner to turning cycles, the code table allows us to determine the edge on which we leave a crossing when tracing a Seifert circle. From Figure 2 it may be seen that if we arrive at a crossing at the terminating even (odd) edge at a crossing and follow the Seifert circle, we leave on the originating even (odd) edge.

To enumerate the set of Seifert circles we follow both the odd and even edges according to their orientation, ensuring that each edge is encountered exactly once. We list the sequence of crossings that comprise each Seifert circle, and separately the sequence of edges; the crossing lists are used to create the reduced Seifert graph and when we create the final braid word, the edge lists are used when performing a Vogel move.

The Seifert circles thus determined for our example are as follows:

| Crossings: | Edges: |
|--------------------------------|-----------------------------------|
| $s_1:$ (0, 2, 4, 3, 1) | $s_1:$ (0, 4, 8, 6, 2) |
| $s_2:$ (3, 1, 0) | $s_2:$ (1, 7, 3) |
| $s_3:$ (4, 6, 10, 5, 11, 8, 2) | $s_3:$ (5, 9, 13, 21, 11, 23, 17) |
| $s_4:$ (5, 11, 6) | $s_4:$ (10, 22, 12) |
| $s_5:$ (7, 10) | $s_5:$ (14, 20) |
| $s_6:$ (9, 7) | $s_6:$ (15, 19) |
| $s_7:$ (8, 9) | $s_7:$ (16, 18) |

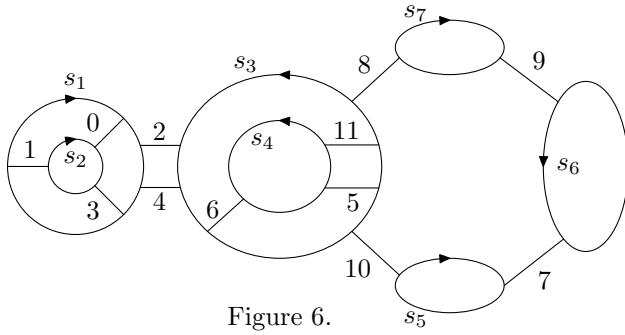


Figure 6.

4.4 A4, Constructing the Reduced Seifert Graph

From the set of Seifert circles we may determine the reduced Seifert graph, Γ' . Each of the crossings on a Seifert circle appears in exactly one other and corresponds to an edge in Γ . If there are multiple crossings between the same two Seifert circles, to construct Γ' we consider only the first that we encounter as we work along a particular Seifert circle, as recorded in section 4.3.

We record Γ' as a list of vertices, and associated with each one the set of edges incident with that vertex. These edges are recorded by noting the other vertex in their boundary. The numbering of vertices corresponds to the numbering of the Seifert circles. We record the type of each edge in the Seifert graph by prefixing the vertex it takes us to with a minus sign if and only if the corresponding edge is a type I edge.

Applying this process to our example results in the following lists of edges, where we have retained the Seifert circle notation, s_i , for clarity.

$s_1: s_2, -s_3$
 $s_2: s_1$
 $s_3: -s_1, s_4, -s_5, -s_7$
 $s_4: s_3$
 $s_5: -s_6, -s_3$
 $s_6: -s_7, -s_5$
 $s_7: -s_3, -s_6,$

4.5 A5, Determining Whether a Vogel Move is Required

Lemma 3.1 and Corollary 3.2 give us a simple mechanism for testing whether another Vogel move is required. We look for a vertex of Γ' with order greater than 2, or of order 2 where both of the attached edges are of the same type.

Having identified such a vertex, v , we then look for a pair of vertices v_1 and v_2

in $\text{star}(v) - v$ joined to v by edges of the same type whose corresponding Seifert circles, s_1 and s_2 , considered as lists of edges in D , have an edge in the same turning cycle C . Since v_1 and v_2 are joined to v by edges of the same type, s_1 and s_2 lie in the same component of the complement of s_v , the Seifert circle corresponding to v , and so are incoherently oriented. We may therefore perform a Vogel move between them across the component of the diagram's complement corresponding to C .

We shall refer to the type of edge in Γ' joining v_1 and v_2 to v as the *edge type* associated with this Vogel move.

In our example, since the edges attached to vertex s_1 have different types, we consider vertex s_3 and note that s_1 and s_5 are both joined to s_3 by type 1 edges. From the edge lists for s_1 and s_5 it can be seen that edges 0 and 20 both appear in the left turning cycle $(0, 4, 17, 18, 15, 20, 13, 9, 6, 2)$ so we may perform a Vogel move between these two edges. From Figure 5 it can be seen that we have correctly identified edges that admit a Vogel move.

Note that the edges identified by the above process will always be either both odd or both even, since Seifert circles are comprised entirely of odd edges or entirely of even edges, and connected Seifert circles are comprised of edges of opposite parity.

4.6 A6, Evaluating the New Code Table

We apply the Vogel move by deriving a labelled peer code for the diagram after the move has been performed. Suppose that the move is to be performed between edges e_1 and e_2 , where $e_1 < e_2$ and that the original immersion contained n crossings.

We start the new numbering of the edges in the changed diagram from the same point as we did for the original numbering, so that in the case where $e_1 > 0$ the first $e_1 - 1$ edges have the same number as before. The two edges numbered e_1 and e_2 in the original diagram each produce three edges in the changed diagram, numbered $e'_{11}, e'_{12}, e'_{13}$ and $e'_{21}, e'_{22}, e'_{23}$ respectively, as shown in Figure 7.

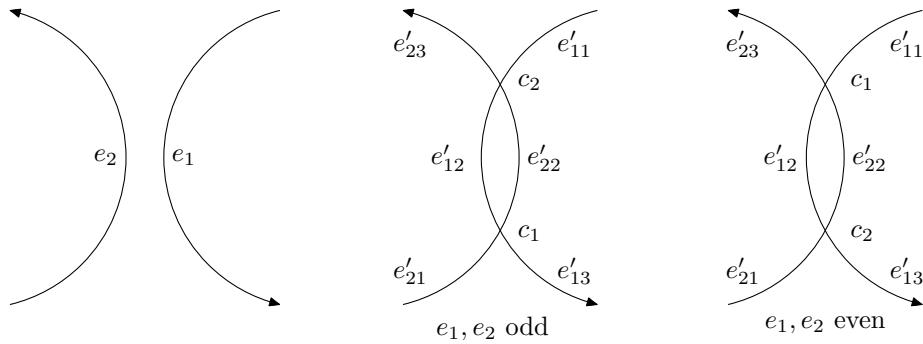


Figure 7

We have $e'_{11} = e_1$, $e'_{12} = e_1 + 1$, $e'_{13} = e_1 + 2$, and $e'_{21} = e_2 + 2$, $e'_{22} = e_2 + 3$, $e'_{23} = e_2 + 4$ as a result of the additional crossings in the new immersion. If $e_2 < 2n$ then for those edges after e_2 in the original immersion, we have the new number $e'_i = e_i + 4$ for $i = e_2 + 1, \dots, 2n$. Note that odd edges remain odd in the new numbering and even edges remain even.

Using the new edge numbering we may determine the crossing numbers for the new crossings as follows (see figure 7). If e_1 and e_2 are both odd then the two crossings are numbered $c_1 = e'_{12}/2$ and $c_2 = e'_{22}/2$; that is $c_1 = (e_1 + 1)/2$ and $c_2 = (e_2 + 3)/2$. Otherwise e_1 and e_2 are both even and the two crossings are $c_1 = e'_{11}/2$ and $c_2 = e'_{21}/2$; that is $c_1 = e_1/2$ and $c_2 = (e_2 + 2)/2$, and in either case, $c_1 < c_2$. If $c_1 > 0$ then the first $c_1 - 1$ crossings retain the same number as before, between crossings c_1 and c_2 the crossing number is increased by one and for crossings beyond c_2 the crossing number is two greater than before.

The type of the two new crossings is the opposite type to the edge type associated with the Vogel move, since s_n and s_v lie in different components of the complement of s_a .

If e_1 and e_2 are both odd the naming edge e'_{12} has the odd edge $e'_{21} = e_2$ as its peer and the naming edge e'_{22} has the odd edge $e'_{11} = e_1$ as its peer. If e_1 and e_2 are both even the naming edge e'_{11} has the odd edge $e'_{22} = e_2 + 3$ as its peer and the naming edge e'_{21} has the odd edge $e'_{12} = e_1 + 1$ as its peer.

Clearly the edges $e'_{11}, e'_{12}, e'_{13}$ lie on the same component as e_1 and the edges $e'_{21}, e'_{22}, e'_{23}$ lie on the same component as e_2 .

Finally, notice that the labels assigned to the new crossings will be $+$ and $-$ but we are free to chose which label to assign to which crossing.

Given the crossing numbers, the odd numbered peers of the naming edges, the crossing types, the components to which the naming edges belong and the labels, we are able to write the labelled peer code for the diagram after the Vogel move has been performed.

4.7 A7, The Iterative Step

Once a labelled peer code for the changed knot has been calculated, we may repeat the process of identifying turning cycles, Seifert circles and the Seifert graph. Since the effect of a Vogel move is to reduce the number of incoherently oriented Seifert circles, this process terminates after a finite number of iterations with a diagram whose Seifert circles are all coherently oriented.

When the process terminates, Γ' has been reduced to a graph where exactly two vertices have order 1 and any other vertex has order 2. If v is a vertex in Γ' of order 2 then the edges in $\text{star}(v)$ have different types and therefore the Seifert circles corresponding to the vertices in $\text{star}(v) - v$ lie in different components of the complement of the Seifert circle corresponding to v . It follows that the Seifert diagram is comprised of at most two sets of concentric Seifert circles in the plane and therefore a suitable choice of infinity realizes the diagram as a set of concentric circles from which the braid word may be read.

4.8 A8 Reading the Braid Word

We read the braid word from Seifert circles and crossing types to produce a braid of n strands where n is the number of Seifert circles. We number the braid strands in the order determined by Γ' , starting at a vertex of order 1. By a suitable choice of the point at infinity we may consider the Seifert circle corresponding to the first braid strand to be the innermost Seifert circle of our concentric set.

The braid word is built up from the record of the Seifert circles by replacing crossing numbers with braid crossings. If a crossing joins Seifert circles numbered i and $i + 1$ it represents the braid crossing $\sigma_i^{\pm 1}$ or t_i depending upon the label associated with the crossing. The labels assigned to real crossings in a labelled peer code indicate whether the naming edge forms part of the over-arc or under-arc of the crossing. In order to determine whether a real crossing is positive or negative in the braid sense, we must also consider the type of the crossing. From Figure 2 it may be seen that if a type I crossing is labelled ‘–’ or a type II crossing is labelled ‘+’, then it is a positive crossing. If a type I crossing is labelled ‘+’, or a type II crossing is labelled ‘–’, it is negative. Virtual or flat crossings of either type are replaced by ‘ t_i ’.

Imagine that a ray is drawn from the centre of the concentric Seifert circles so that it intersects each of them transversely. As this ray is rotated by 2π following the orientation of the Seifert circles we can read off the braid crossings represented

as described above for each crossing. We may also assume that our diagram is such that this ray never encounters more than one crossing at a time.

We build up the braid word by successively considering the Seifert circles from the middle of our concentric set outwards. We shall speak of a *child* Seifert circle and its *parent* when we consider two Seifert circles corresponding to adjacent vertices in Γ' , the parent being the Seifert circle with the greater strand number.

Consider then parent and child Seifert circles p and s respectively, written as lists of crossing numbers, and suppose that s corresponds to a univalent vertex in Γ' . Suppose that c_1, c_2, c_3 are consecutive crossings on p and that c_2 also appears in s . By replacing c_2 in p with its braid crossing equivalent we have a record in p of the braid equivalent of the strands s and p as sweep our imaginary ray from c_1 to c_3 . We shall write this modified form of p as

$$p = (\dots, c_1, w(c_2, p), c_3, \dots)$$

we shall refer to replacing all crossings to s in p in this manner as *putting s into its parent*. If we have consecutive crossings in p that are also in s the result of putting s into p will include one or more sequences of the following form:

$$p = (\dots, c_k, w(c_{k+1}, p), \dots, w(c_{k+n}, p), c_{k+n+1}, \dots)$$

where c_k and c_{k+n+1} do not appear in s . By concatenating

$$w(c_{k+1}, p), \dots, w(c_{k+n}, p)$$

we obtain a braid word, $w^*(c_k, p)$, representing the braid word equivalent of the strands s and p as we sweep our imaginary ray from c_k to the next crossing of p not on s .

Suppose now that p has a parent p' in Γ' and assume c_1 is a crossing on p' , say $p' = (\dots, c_a, c_1, c_b, \dots)$. Since p has its own child, we replace c_1 in p' with the braid crossing equivalent of c_1 followed by $w^*(c_1, p)$, denoting the concatenation of the two $w(c_1, p')$. We then have a record in p' of the braid word equivalent of the strands s , p and p' as we sweep our imaginary ray from c_a to c_b . By repeating this process for all crossings in p' that are also in p , and concatenating successive crossings to create the required $w^*(c_i, p')$ we obtain the result of putting p into p' .

To compute the whole braid word then we proceed inductively along Γ' putting the Seifert circles corresponding to its vertices into their parent. When we reach other univalent vertex of Γ' the Seifert circle corresponding to the root vertex will have been completely replaced by exactly the braid word representation of the diagram that we have been striving for.

References

- [1] P. Vogel. Representation of links by braids, a new algorithm. Comment. Math. Helvetici 65 (1990), 104-113.
- [2] D. H. Choi. Algorithm Reading Off a Braid Word From Yamada-Vogel Construction. Master's Thesis, <http://knot.kaist.ac.kr/> dhchoi/
- [3] S. Yamada. The minimal number of Seifert circles equals the braid index of a link. Invent. Math. 89 (1987), 347-356.
- [4] L. Kauffman. Virtual Knot Theory.
- [5] R. Fenn, M. Jordan-Santana, L. Kauffman and G. Wraith. Biracks and Virtual Links.