Rule Based Knot Breaking and Resolution of Certain Unknotting Numbers

Seenu S. Reddi
ReddiSS at aol dot com
November 22, 2012
Available at www.rspq.org/pubs

Summary: We present a simple rule based algorithm for breaking knots and finding their unknotting numbers (see References[1] and [2]). We give examples of using these for some simple knots and discuss discrepancies with existing unknotting numbers.

We represent the knot as a sequence of numbers by tracing the path of the strand composing the knot. We assign numbers like 1, 2, 3, .. if the strand is passing through the junctions/vertices on the upper plane ; otherwise we assign 1’, 2’, 3’, .. As an example for the 5_2_1 knot listed, the knot is specified by 1 2’ 3 4’ 5 3’ 2 1’ 4 5’ for the numbering assigned shown in the figure. We start with 1 as the strand passes thru’ the top, cross 2 on the bottom, pass thru’ 3 on the top, .. We use the following rules:

1. Reduce the sequence .. X X’ … by eliminating nodes X and X’.
1C. Reduce the sequence .. X’ X … by eliminating nodes X and X’.

As an example, 5’ 6 6’ 5 gets reduced to 5’ 5 and eventually an unknot. For the next rule we introduce some notation. We say a vertex X is neighbored-up if the path contains ..XY .. or ..YX.. for some Y and neighbored-down if the path contains ..X’Z’.. or ..Z’X’.. for some Z. X has an up-neighbor in Y and a down-neighbor in Z.

2. Eliminate X and X’ if X is neighbored-up and neighbored-down. Note up-neighbor and down-neighbor need not be the same.

As an example, if we have 1’ 2’ 3’ 4 5’ 1 2 6 4’ 5 6’ 3, vertex 1 has 2 as up- and down-neighbor. By Rule 2 we eliminate 1 and 1’ thus reducing the knot to 2’ 3’ 4 5’ 2 6 4’ 5 6’ 3. Note 2 and 2’ can be eliminated because 2 has up- and down-neighbors and the path is reduced to 3’ 4 5’ 6 4’ 5 6’ 3. Applying Rule 1, the path is further reduced to 4 5’ 6 4’ 5 6’ which can be recognized as a trefoil with break number of 1. Rule 2 is analogous to how the humans unknot by identifying a consecutive path without entanglements and shortening the path.

We illustrate with a simple example how these rules are used to find the unknotting numbers. Consider the knot 5_2_1 (corresponds to 5-2 knot with an unknot number of 1) represented by 1 2’ 3 4’ 5 3’ 2 1’ 4 5’. Select 5 to break: change 5 to 5’ and 5’ to 5.
Essentially this process *breaks* the knot by reversing the junction. Now we have 1 2' 3 4' 5' 3' 2' 1' 4 5 (4 removed by Rule 2) -> 1 2' 3 5' 3' 2 1' 5 (5 removed by Rule 2) -> 1 2' 3 3' 2 1' (1 removed by Rule 1) -> 2' 3 3' 2 -> Unknot. Thus we get an unknotting factor of 1 for this particular knot. This agrees with what we have in Rolfsen’s knotting table [3]. (Curiously Rolfsen [4] himself does not list the unknotting numbers in his book).

Though in most of the cases this procedure gives the right unknotting number, there are certain situations where the procedure gives unknotting numbers higher than listed in the table. Denoting by RU(K) and PU(K) the unknotting factors listed in Rolfsen’s table and obtained by our procedure respectively for a given knot K, we conjecture:

CONJECTURE: \( PU(K) \geq RU(K) \).

In the following examples we indicate the unknotting solution for a given knot by showing the selection of vertices; for instance, the above solution is indicated by 5_2_1.txt S: 5 -> Breaks: 1 (1) showing vertex 5 selected for breaking. 5_2_1.txt contains the path of the knot namely 1 2' 3 4' 5 3' 2 1' 4 5'. We provide a program written for the Windows to implement the unknotting procedure.

For the 6-2 knot (6_2_1 in our case), our procedure gives an unknotting factor of 2 regardless of how many different variations attempted in selecting vertices to break. However we can introduce a twisting operation that can result in the minimum number. The concept behind it is explained by noting that an unknot corresponds to the figure of a circle and a twisted unknot to an eight, and the latter can be untwisted. The specific twisting operation we introduce at a vertex V consists of reversing all the vertices between V and V'. For instance, in 6_2_1.txt we have after selecting 1 to break:

1 2' 3 4' 5 1' 2 3' 6 5' 4 6' (Break 1) -> 3 4' 5 3' 6 5' 4 6'

Twisting at 3 (which is different from twisting at 3') will reverse vertices 4 and 5 resulting in:

3' 4 5' 3 6 5 4' 6' -> 4 5' 6 5 4' 6' -> 4 5' 5 4' -> unknot

At this time we do not know whether a twist of this kind is legal or illegal (the reader can verify himself that performing twists of this kind randomly will lead to absurd results) but it seems to be a necessity in obtaining the minimal unknotting numbers. It would have been highly beneficial how the unknotting numbers are obtained in [3] and this would have helped us in some kind of understanding when twists are needed and how to differentiate legal and illegal twists.

We list the knots up to seven crossings and present solutions without twist and with possible legal twist (though we cannot assure the legality). A program written for Windows is available as well as the source code at [www.rspq.org/pubs](http://www.rspq.org/pubs). Also we show the Haken knot and a couple of knots with ten crossings for which we cannot get the minimum unknotting number. These are left as a challenging exercise to the interested reader.
References:


In the following examples, we freely use the diagrams found in Rolfsen’s page and the respective copyrights remain.
Examples of Unknotting

Path: 1 2' 3 4' 5 3' 2 1' 4 5'  
Solution: F: 5_2_1.txt  S: 5 -> Breaks: 1 (1)

Path: 1 2' 3 4' 5 6' 4 3' 2 1' 6 5'  
Solution: F: 6_1_1.txt  S: 6 -> Breaks: 1 (1)
Path: 1' 2' 3' 4' 5 1 2 3 6 5' 4 6'  
Solution: F: 6_2_1.txt  S: 1 3 -> Breaks: 2 (1) 
Solution: F: 6_2_1.txt  T: 3 S: 5 -> Breaks: 1? (1) 

Path: 1' 2 3' 4 5' 6 2' 3 6' 1 4' 5  
Solution: F: 6_3_1.txt  S: 1 3 -> Breaks: 2 (1) 
Solution: F: 6_3_1.txt  T: 6 S: 5 -> Breaks: 1? (1)
Path: 1 2 3 4 5 6 4 3 2 1 7 5 6 7 7 2 1
Solution: F: 7_2_1.txt  S: 5 -> Breaks: 1 (1)

Path: 1 2 3 4 5 6 1 2 3 4 7 6 5 7 3 2
Solution: F: 7_3_2.txt  S: 1 3 -> Breaks: 2 (2)
Path: 1' 2' 3 4' 5 6' 2 1' 7 3' 6 5' 4 7'  
Soln.: F: 7_4_2.txt  S: 1 3 → Breaks: 2 (2)

Path: 1' 2' 3 4' 5 6' 7 1' 6 5' 2 3' 4 7'  
Soln: F: 7_5_2.txt  S: 1 3 → Breaks: 2 (2)
Path: 1 2' 3 4' 5 3' 7 6' 4 5' 2 1' 6 7'  
Soln.: F: 7_6_1.txt  S: 7 3 -> Breaks: 2 (1)
Soln.: F: 7_6_1.txt  S: 5 T: -2 -> Breaks: 1 (1)

Path: 1 2' 3 4' 5 3' 7 6' 4 5' 2 1' 6 7'  
Soln.: F: 7_7_1.txt  S: 4 S: 6 -> Breaks: 2 (1)
Soln.: F: 7_7_1.txt  S: 5 T: 6 -> Breaks: 1 (1)