Vol. 00 no. 00 2008
Pages 1–5

An algorithm based on 3-braids to solve tangle equations arising in the action of Gin DNA invertase

Hugo Cabrera Ibarra[*] and David A. Lizárraga Navarro
Division of Applied Mathematics
Instituto Potosino de Investigación Científica y Tecnológica, IPICYT
Camino a la Presa San José 2055
78216, San Luis Potosí, S.L.P., México
email: cabrera@ipicyt.edu.mx, D.Lizarraga@ipicyt.edu.mx

Received on ; revised on ; accepted on

Associate Editor:

ABSTRACT

The tangle model of Ernst and Sumners is an effective tool in the topological analysis of enzymes, a particular application of which aims at deducing the action mechanism of site-specific recombination mediated by the Gin DNA invertase, an enzyme that involves 3-string tangles. In order to determine the enzyme's action mechanism, the tangle model yields equations that involve tangle indeterminates that must be solved for. While some of the available methods for solving such equations judiciously exploit the theory of 2-tangles, an algorithm is introduced in this note, based on 3-braid–theoretical results in (Cabrera-Ibarra and Lizárraga-Navarro, 2008), which allowed the authors to discover previously unreported solutions for the action of Gin DNA invertase. More generally, the algorithm allows one to exhaustively solve tangle equations under the assumption that two or more rounds of recombinations are given by known 2-bridge knots. Rather than a specific language implementation, the listing below provides a pseudo-code description of the algorithm that permits its implementation in a variety of computer languages and, possibly, its inclusion into more powerful analysis software.

Contact: cabrera@ipicyt.edu.mx

1 INTRODUCTION

In DNA site-specific recombination, a recombination enzyme attaches to a pair of DNA specific sites, breaks both and then recombines them to different ends, thus modifying the original topology of the molecule. Electron micrographs of recombinases bound to DNA show the enzyme as a blob from which two or three DNA loops emerge, depending on the enzyme. In the specific case of Gin, three DNA loops stick out of the blob, thus making of the theory of 3-tangles a particularly useful analysis tool.

The tangle model, introduced in (Sumners *et al.*, 1995), was applied under reasonable biological assumptions to model the site-specific recombinase Tn3 resolvase as well as other enzymes such as λ-Int (Crisona *et al.*, 1999) and Xer (Vazquez *et al.*, 2005). The cases under study in those references involved actions of enzymes on 2-tangles, a favorable situation since rational 2-tangles have been completely classified. Nevertheless, some enzymes, such as Gin and Hin integrase recombinases, act on 3-tangles instead of 2-tangles. In order to cope with the latter case, Vazquez and Sumners (2004) introduced the assumption that one of the strings may be neglected—so that the molecule may be regarded as a 2-tangle—which allowed them to solve the action of the Gin enzyme with inversely and directly repeated sites.

Based on the theory and results developed in (Cabrera-Ibarra, 2003), in the more recent reference (Cabrera-Ibarra and Lizárraga-Navarro, 2008) we applied the properties of standard braid diagrams, along with the main ideas of the tangle model, to analyze knotted products of site-specific recombination mediated by the Gin enzyme. As a result, for both the directly and inversely repeated cases we obtained two solutions to the Gin action under the assumption that the tangles involved were 3-braids. To the extent of our knowledge, two of the four solutions reported in (Cabrera-Ibarra and Lizárraga-Navarro, 2008) were not previously available in the literature. Instrumental in obtaining those solutions was a computer algorithm that allows one to solve 3-braid equations when the product of two rounds of recombinations are equal to two known 2-bridge knots. The purpose of this paper is to give a detailed pseudo-code listing of that algorithm, from which implementations in a variety of computer languages may be easily derived. Our algorithm adds to the list of powerful software implementations developed by other researchers, among whom (Saka and Vázquez, 2002; Darcy and Scharein, 2006), and it might conceivably be incorporated into those or other existing software realizations.

2 2-BRIDGE KNOTS AND THE TANGLE MODEL

The family of 2-bridge knots and links has been profusely studied, to the point that it is completely classified. The classification, which assigns a couple of integers (α, β) (satisfying some requirements) to any 2-bridge knot $b(\alpha, \beta)$, states that two such knots $b(\alpha, \beta)$ and $b(\alpha', \beta')$ are equivalent iff $\alpha = \alpha'$ and either $\beta = \beta' \mod \alpha$ or $\beta\beta' = 1 \mod \alpha$. As shown in Fig. 1, the standard diagram of a 2-bridge knot may be seen as the closure, denoted by $A(\mathcal{T}(a_1, \ldots, a_n))$, of a braid $\mathcal{T}(a_1, \ldots, a_n)$.

The tangle model of (Ernst and Sumners, 1990) permits the analysis of knotted and linked products of site-specific recombination mediated by a given enzyme. Biologically, the

[*]to whom correspondence should be addressed

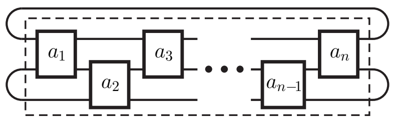

Fig. 1. Standard diagram of a 2-bridge knot as the A-closure of a braid $\mathcal{T}(a_1, \ldots, a_n)$

model assumes that (1) the enzyme acts by a mechanism that is independent of phenomena such as supercoiling and the linking of the substrate population, (2) the recombination takes place in the interior of the enzyme ball, and (3) the substrate configuration outside the enzyme ball remains unchanged while the strand reconnection takes place inside the ball. Mathematically, it is assumed that the recombination process may be modeled by tangle addition (concatenation). The action of an enzyme on an unknotted circular DNA molecule as substrate is schematically shown in Fig. 2.

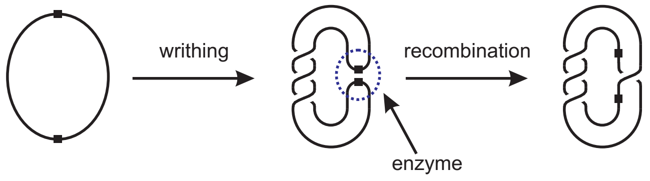

Fig. 2. Site specific enzyme-mediated recombination: An enzyme acts on a writhed molecule, possibly modifying its topology.

In view of the biological and mathematical assumptions, and considering the products obtained after several recombination events of the Gin enzyme from experimental results, its action mechanism may be schematically depicted as in Fig. 3.

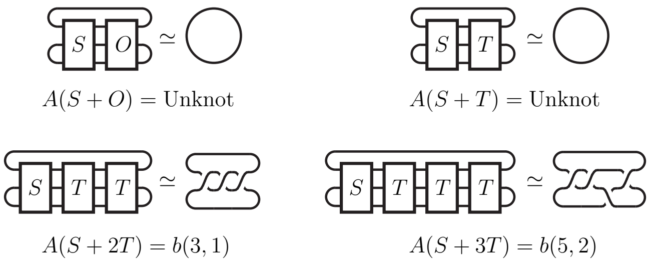

Fig. 3. The tangle model for the Gin enzyme acting on a DNA molecule

3 APPLICATION: TANGLE SOLUTIONS TO THE ACTION OF GIN

In the case of Gin with inversely repeated sites, our algorithm provides 32 families of solutions, which reduce to 12 if they are required to satisfy an additional equation. Moreover, by disregarding inessential differences, the 12 merge into 2 essentially distinct families, one of which was previously unreported. Using these results one may even predict the outcome of the nth recombination. As a matter of fact, for $n = 4$, the predicted recombination coincides with the product obtained in the fourth round in an experiment involving Gin invertase enzyme for substrate with inversely oriented *gix* sites (Kanaar *et al.*, 1990). In (Cabrera-Ibarra and Lizárraga-Navarro, 2008), the algorithm was also applied to the Gin with directly repeated sites, leading to the discovery of 2 essentially different families of solutions, one of them previously unreported as well.

It is worth mentioning that, although we have mainly focused on the Gin DNA invertase enzyme, the essential ideas and the algorithm are equally applicable for the analysis of other enzymes acting in similar ways.

4 THE ALGORITHM

Our algorithm provides an implementable way to solve tangle equations using the theory developed in (Cabrera-Ibarra and Lizárraga-Navarro, 2008). The Reader may wish to consult the latter reference for a detailed description of concepts and results alluded to below. Specific implementations in a variety of computer languages may easily be obtained from the listed pseudo-code, which adheres to the conventions set forth in the now classical reference (Cormen *et al.*, 2001). As an additional convention, if A is an array, $A[j \,..\, k]$ denotes the (finite) sequence $A[j], A[j + 1], \ldots, A[k]$. Due to space limitations, the procedure listing does not include any data validation or exception handling.

4.1 Families of Procedures

The algorithm takes, via GET-P-AND-Q(A, B), two 2-bridge knots $A = b(\alpha, \beta)$ and $B = b(\alpha', \beta')$, specified by two pairs of integers, and returns two families $P_{x,y}$ and $Q_{x,y}$, $x, y \in \mathbb{N}$, of 3-braids satisfying $A(P_{x,y} + Q_{x,y}) = b(\alpha, \beta)$ and $A(P_{x,y} + 2Q_{x,y}) = b(\alpha', \beta')$. The algorithm proceeds by first finding all families S_x and T_y of 3-braids having A closures equal to the given knots: $A(S_x) = b(\alpha, \beta)$ and $A(S_y) = b(\alpha', \beta')$. It then computes $P_{x,y} = S_x - T_y + S_x$ and $Q_{x,y} = -S_x + T_y$. The remaining procedures may be classified into five families according to the data type they operate on or the solutions they provide:

- *Basic structures*: Besides basic data types, (abstract, multidimensional) arrays are a fundamental data structure for the algorithm; APPEND, CONCATENATE and STRIP-ZEROES handle operations on arrays.
- *Continued fractions*: Continued fractions are represented as arrays of integers or multinomials; procedures which handle them include GET-CF-EXPANSION, APPLY-LAGRANGE-AT and REMOVE-ZEROES.
- *Braids*: A braid $\mathcal{T}(a_1, \ldots, a_n) + kE$ is represented by a structure B with fields $\mathit{fraction}[B] = [a_1, \ldots, a_n]$ and $\mathit{index}[B] = k$. Since families of braids (typically indexed by indeterminates) occur in the algorithm, the entries $a_1, \ldots, a_n$ usually represent multinomials. The group operation on braids—concatenation—is performed by CONCATENATE-BRAIDS; the inverse is computed via

INVERT-BRAID; and braids are transformed into their standard form using GET-STANDARD-FORM.

- *Multinomials*: DETECT-SIGN and DETECT-SIGN-CHANGE.
- *Solutions to the 2-bridge knot congruences*: Procedures that allow one to find 3-braids whose closures equal a specified 2-bridge knot are SOLVE-2BK-CONGRUENCES and GET-SOLUTION-EXPANSION.

4.2 Brief description of the procedures

Here we present a brief description of the procedures involved.

- APPEND(A, x) Appends the element x at the end of array A.
- CONCATENATE(A, B) Concatenates arrays A and B, in that order.
- STRIP-ZEROES(F) Returns an array that possibly differs from the array of integers F in that it contains no zero entry.
- GET-CF-EXPANSION$(n, \textit{long-format})$ Gets the continued fraction expansion of a rational n. The argument *long-format* indicates whether the long or short format is required.
- APPLY-LAGRANGE-AT$(F, k, \textit{plus-to-minus})$ Applies a flype move (or Lagrange's rule) to the continued fraction F, represented as an array of multinomials, at position $k \in \{1, \ldots, \textit{length}[F]\}$. If $\textit{plus-to-minus} = $ TRUE, the rule $a + \frac{1}{-b} = a - 1 + \frac{1}{1+\frac{1}{b-1}}$ is applied, so that if $F = [f_1, \ldots, f_{k-1}, a, -b, f_{k+2}, \ldots, f_n]$, it returns $[f_1, \ldots, f_{k-1}, a - 1, 1, b - 1, -f_{k+2}, \ldots, -f_n]$. If $\textit{plus-to-minus} = $ FALSE, the rule $-a + \frac{1}{b} = -a + 1 + \frac{1}{-1+\frac{1}{-b+1}}$ is applied, so that if $F = [f_1, \ldots, f_{k-1}, -a, b, f_{k+2}, \ldots, f_n]$, it returns $[f_1, \ldots, f_{k-1}, -a + 1, -1, -b + 1, -f_{k+2}, \ldots, -f_n]$.
- REMOVE-ZEROES(F) Takes a continued fraction F, given by an array, then removes zeros and simplifies according to the rules of continued fractions.
- CONCATENATE-BRAIDS(A, B, X) Takes braids A and B (cf. GET-STANDARD-FORM (B, X) for the description of braids) and concatenates (i.e., adds) them, expressing the result in standard form.
- INVERT-BRAID(A) Inverts braid A under concatenation in the braid group (cf. GET-STANDARD-FORM(B, X) for the description of braids).
- GET-STANDARD-FORM(B, X) Takes a braid B, given by $\mathcal{T}(\textit{fraction}[B]) + \textit{index}[B]E$ and puts it in the standard form $AD + kE$. $\textit{fraction}[B]$ is an array of multinomials in the indeterminates X.
- DETECT-SIGN(P, X) Takes a degree-one multinomial $P = a_0 + a_1X[1] + \cdots + a_{\textit{length}[X]}X[\textit{length}[X]]$ in the indeterminates $X[i]$ and returns a structure indicating whether P is sign-definite and, in such case, $\text{sign}(P)$. Sign-definiteness is tested under the assumptions that multinomials are evaluated and that the indeterminates take strictly positive values.
- DETECT-SIGN-CHANGE(A, I, X) Takes an array A of multinomials in the indeterminates X and, starting from the Ith position, determines whether there is a sign change. In such case, the position p at which it occurred, and the sign of $A[p]$ are also returned. Sign-detection is based on the rules applied by DETECT-SIGN.
- SOLVE-2BK-CONGRUENCES(α, β) Returns an array of couples of integers $[a, b]$ that solve the two-bridge knot congruences: $a = \alpha$ and ($b \equiv \beta \mod \alpha$ or $b\beta \equiv 1 \mod \alpha$).
- GET-SOLUTION-EXPANSION(α, β, X) Returns an array of braids. Each braid B is in standard form and satisfies $(F(B))_1 \geq 1$. These braids represent the totality of solutions to equation $A(B) = b(\alpha, -\beta)$.
- GET-P-AND-Q(A, B) Given two 2-bridge knots $b(A[1], A[2])$ and $b(B[1], B[2])$, returns braid families P and Q such that $A(P + Q) = b(A[1], A[2])$ and $A(P + 2Q) = b(B[1], B[2])$.

4.3 Pseudo-code listing

APPEND(A, x)

```
1  return $[A[1 \,..\, length[A]], x]$
```

CONCATENATE(A, B)

```
1  return $[A[1 \,..\, length[A]], B[1 \,..\, length[B]]]$
```

STRIP-ZEROES(F)

```
1  $R \leftarrow [\,]$
2  for $i \leftarrow 1$ to $length[F]$
3      do if $F[i] \neq 0$
4          then $R \leftarrow$ APPEND$(R, F[i])$
5  return $R$
```

GET-CF-EXPANSION$(n, \textit{long-format})$

```
1  $R \leftarrow \text{sign}(n)$GETCONTINUEDFRACTION$(|n|)$
2  if $long\text{-}format$
3    then $R[length[R]] \leftarrow R[length[R]] - \text{sign}(n)$:
4         $R \leftarrow$ APPEND$(R, \text{sign}(n))$
5  return $R$
```

APPLY-LAGRANGE-AT$(F, k, \textit{plus-to-minus})$

```
1  $n \leftarrow length[F]$
2  if $n < 2$ or $k \notin \{1, \ldots, n-1\}$
3    then return $F$
4  if $plus\text{-}to\text{-}minus$
5    then return
           $[F[1 \,..\, k-1], F[k] - 1,\ \ 1, -F[k+1] - 1, -F[k+2 \,..\, n]]$
6    else return
           $[F[1 \,..\, k-1], F[k] + 1, -1, -F[k+1] + 1, -F[k+2 \,..\, n]]$
```

REMOVE-ZEROES(F)

```
 1  $R \leftarrow F$;  $last\text{-}zero \leftarrow 3$
 2  while $length[R] > 1$ and any entry of
            $R[last\text{-}zero \,..\, length[R] - 1]$ equals 0
 3      do $n \leftarrow length[R]$
 4         for $i \leftarrow last\text{-}zero - 1$ to $n - 1$
 5             do if $R[i] = 0$
 6                  then $R \leftarrow [R[1 \,..\, i-2],$
                              $R[i-1] + R[i+1], R[i+2..n]]$
 7                       $last\text{-}zero \leftarrow \max\{i, 3\}$
 8                       break  $\triangleright$ Break for loop; jump to 2
 9  if $length[R] > 1$ and $R[length[R]] = 0$
10    then $R \leftarrow [R[1..\, length[R] - 1]]$
11  return $R$
```

```
CONCATENATE-BRAIDS($A, B, X$)
$f_a \leftarrow fraction[A]$;  $f_b \leftarrow fraction[B]$;
$n_a \leftarrow length[f_a]$;  $n_b \leftarrow length[f_b]$
$index[R] \leftarrow index[A] + index[B]$
if $index[A] \in 2\mathbb{Z}$
  then if $n_a \in 2\mathbb{Z}$
         then $fraction[R] \leftarrow$ CONCATENATE($f_a, f_b$)
         else $fraction[R] \leftarrow [f_a[1 \,..\, n_a - 1],$
                             $f_a[n_a] + f_b[1], f_b[2 \,..\, n_b]]$
  else if $n_a \in 2\mathbb{Z}$
         then $fraction[R] \leftarrow [f_a[1 \,..\, n_a - 1],$
                             $f_a[n_a] - f_b[1], -f_b[2 \,..\, n_b]]$
         else $fraction[R] \leftarrow$ CONCATENATE($f_a, -f_b$)
return GET-STANDARD-FORM($R$)
```

```
INVERT-BRAID($A$)
$n \leftarrow length[fraction[A]]$;  $k \leftarrow index[A]$
$F \leftarrow [\,]$
if $k \in 2\mathbb{Z}$
  then if $n \in 2\mathbb{Z}$
         then $F \leftarrow [0]$
       $F \leftarrow$ APPEND($F, [-fraction[A][n \,..\, 1]]$)
  else if $n \in 2\mathbb{Z} + 1$
         then $F \leftarrow [0]$
       $F \leftarrow$ APPEND($F, [fraction[A][n \,..\, 1]]$)
$fraction[R] \leftarrow$ REMOVE-ZEROES($F$)
$index[R] \leftarrow -k$
return $R$
```

```
GET-STANDARD-FORM($B, X$)
$n \leftarrow length[fraction[B]]$;  $F \leftarrow$ REMOVE-ZEROES($fraction[B]$)
$a \leftarrow 0$;  $\ell \leftarrow 4$
$c \leftarrow$ DETECT-SIGN-CHANGE($F, \ell - 3, X$)
while $sign\text{-}changed[c]$
    do $F \leftarrow$ REMOVE-ZEROES(APPLY-LAGRANGE-AT(
                 $F, position[c] - 1, plus\text{-}to\text{-}minus[c]$))
       if $plus\text{-}to\text{-}minus[c]$
         then $a \leftarrow a + (-1)^{position[c]}$
         else $a \leftarrow a - (-1)^{position[c]}$
       $\ell \leftarrow \max\{position[c], 4\}$
       $c \leftarrow$ DETECT-SIGN-CHANGE($F, \ell - 3, X$)
$fraction[R] \leftarrow F$
$index[R] \leftarrow index[B] + a$
return $R$
```

```
DETECT-SIGN($P, X$)
  ▷ $P$ is assumed to be given by
  ▷ $P = a_0 + a_1 X[1] + \cdots + a_{length[X]} X[length[X]]$
  ▷ It is assumed that sign($n$) = 0 if and only if $n = 0$
$s = [\text{sign}(a_0), \ldots, \text{sign}(a_{length[X]})]$
$s \leftarrow$ STRIP-ZEROES($s$)
if $length[s] = 0$
  then $is\text{-}definite[R] \leftarrow$ TRUE
       $sign[R] \leftarrow 0$
  else
       $m \leftarrow \min\{s[1 \,..\, length[s]]\}$
       $M \leftarrow \max\{s[1 \,..\, length[s]]\}$
       if $m = M$
         then $is\text{-}definite[R] \leftarrow$ TRUE
              $sign[R] \leftarrow m$
         else $is\text{-}definite[R] \leftarrow$ FALSE
              $sign[R] \leftarrow 0$
return $R$
```

```
DETECT-SIGN-CHANGE($A, I, X$)
$sign\text{-}changed[R] \leftarrow$ FALSE
$position[R] \leftarrow 0$
$plus\text{-}to\text{-}minus[R] \leftarrow$ FALSE
if $length[A] \leq 1$
  then return $R$
create array $s$ of size $[1 \,..\, length[A] - I + 1]$
for $i \leftarrow I$ to $length[A]$
    do $s[i - I + 1] \leftarrow$ DETECT-SIGN($A[i], X$)
for $i \leftarrow 2$ to $length[s]$
    do if $is\text{-}definite[s[i]]$
         then
              ▷ Detect first A[j] to the left of A[i] with
              ▷ definite sign and compare
              for $j \leftarrow i - 1$ downto 1
                  do if $is\text{-}definite[s[j]]$ and $sign[s[j]] \neq 0$
                       then
                            if $sign[s[j]] \neq sign[s[i]]$
                              then $sign\text{-}changed[R] \leftarrow$ TRUE
                                   $position[R] \leftarrow i + I - 1$
                                   if $sign[s[j]] > sign[s[i]]$
                                     then $plus\text{-}to\text{-}minus[R] \leftarrow$ TRUE
                                   return $R$
return $R$
```

```
SOLVE-2BK-CONGRUENCES($\alpha, \beta$)
$T \leftarrow \{\beta\}$
if $|\alpha + \beta| < \alpha$
  then $T \leftarrow T \cup \{\alpha + \beta\}$
if $|-\alpha + \beta| < \alpha$
  then $T \leftarrow T \cup \{-\alpha + \beta\}$
for $\ell \leftarrow -|\beta|$ to $|\beta|$
    do $b \leftarrow \frac{\alpha}{\beta}\ell + \frac{1}{\beta}$
       if $b \in \mathbb{Z}$ and $|b| < |\alpha|$
         then $T \leftarrow T \cup \{b\}$
create array $R$ of size $[1 \,..\, length[T]]$
for $i \leftarrow 1$ to $length[R]$
    do $a[R[i]] \leftarrow \alpha$
       $b[R[i]] \leftarrow T[i]$
return $R$
```

```
GET-SOLUTION-EXPANSION($\alpha, \beta, x$)
$C \leftarrow$ SOLVE-2BK-CONGRUENCES($\alpha, \beta$)
$R \leftarrow [\,]$
for $i \leftarrow 1$ to $length[C]$
    do
       $F \leftarrow a[C[i]] / b[C[i]]$
       for $j \leftarrow 1$ to 2
           do
              if $j = 1$
                then $fraction[S] \leftarrow$ GET-CF-EXPANSION($F$, FALSE)
                else $fraction[S] \leftarrow$ GET-CF-EXPANSION($F$, TRUE)
              $fraction[S] \leftarrow$ APPEND($fraction[S], \text{sign}(F)x$)
              $index[S] = length[fraction[S]] \mod 2$
              $R \leftarrow$ APPEND($R, S$)
return $R$
```

```
GET-P-AND-Q($A, B$)
1   $Xs \leftarrow$ GET-SOLUTION-EXPANSION($A[1], A[2], a$)
2   $Ys \leftarrow$ GET-SOLUTION-EXPANSION($B[1], B[2], b$)
3   create array $R$ of size $[1 \,..\, length[Xs], 1 \,..\, length[Ys]]$
4   for $i \leftarrow 1$ to $length[Xs]$
5       do for $j \leftarrow 1$ to $length[Ys]$
6              do $Q \leftarrow$ CONCATENATE-BRAIDS(INVERT-BRAID($Xs[i]$),
                                 $Ys[j], [a, b]$)
7                 $P \leftarrow$ CONCATENATE-BRAIDS($Xs[i]$,
                                 INVERT-BRAID($Q$), $[a, b]$)
8                 $X[R[i, j]] \leftarrow Xs[i]$
9                 $Y[R[i, j]] \leftarrow Ys[j]$
10                $P[R[i, j]] \leftarrow P$
11                $Q[R[i, j]] \leftarrow Q$
12  return $R$
```

5 CONCLUSIONS

An algorithm was developed to exhaustively solve pairs of 3-braid equations in the sense that, given two 2-bridge knots K and L, find all 3-braids P and Q such that $A(P + Q) = K$ and $A(P + 2Q) = K$, where "$+$" denotes braid concatenation and $A(\cdot)$ denotes "A closure." Equations of this class typically arise in applications of the tangle model to deduce or analyze the action mechanism of several enzymes. In the particular case of the action of Gin DNA invertase, the algorithm has allowed the authors to find solutions previously unreported in the literature. What is more, the algorithm might also prove useful as a computational tool in applications that extend beyond the determination of the action of enzymes, such as in research or education in braid and tangle theory. Finally, the description of the algorithm using pseudo-code does not enforce coding in any specific language, which enhances its potential for different implementations.

ACKNOWLEDGEMENTS

This research was partially funded by grants from CONACYT, Fdo. Inst. 66912 S-3124 and 66912 S-3122, as well as CONCYTEG No. 06-02-K117-84.

REFERENCES

Cabrera-Ibarra, H. (2003). On the classification of rational 3-tangles. *J. Knot Theory Ramifications*, **12**(7), 921–946.

Cabrera-Ibarra, H. and Lizárraga-Navarro, D. (2008). Braid solutions to the action of the gin enzyme. *(Submitted, manuscript available from the authors)*.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press, 2nd edition.

Crisona, N., Weinberg, R., Peter, B., Sumners, D., and Cozzarelli, N. (1999). The topological mechanism of phage λ integrase. *Journal of Molecular Biology*, **289**, 747–775.

Darcy, I. and Scharein, R. (2006). TopoICE-R: 3D visualization modeling the topology of dna recombination. *Bioinformatics*, **22**(14), 1790–1791.

Ernst, C. and Sumners, D. (1990). A calculus for rational tangles: Applications to DNA recombination. *Math. Proc. Cambridge Philos. Soc.*, **108**, 489–515.

Kanaar, R., Klippel, A., Shekhtman, E., Dungan, J., Kahmann, R., and Cozzarelli, N. (1990). Processive recombination by the phage Mu Gin system: Implications for the mechanisms of DNA strand exchange, DNA site alignment, and enhancer action. *Cell*, **62**, 353–366.

Saka, Y. and Vázquez, M. (2002). TangleSolve: topological analysis of site-specific recombination. *Bioinformatics*, **18**(7), 1011–1012.

Sumners, D., Ernst, C., Cozzarelli, N., and Spengler, S. (1995). Mathematical analysis of the mechanisms of DNA recombination using tangles. *Quart. Review Biophysics*, **28**, 253–313.

Vazquez, M. and Sumners, D. (2004). Tangle analysis of Gin site-specific recombination. *Math. Proc. Cambridge Philos. Soc.*, **136**(3), 565–582.

Vazquez, M., Colloms, S., and Sumners, D. (2005). Tangle analysis of Xer recombination reveals only three solutions, all consistent with a single three-dimensional topological pathway. *Journal of Molecular Biology*, **346**, 493–504.