Programming to generate geminal configurations

Son Nguyen

Ohio University, Mathematics Department

March 11, 2010
Others make geminal configurations by hand, and they are not totally confident about getting all the cases.

I made a computer program (using Python) to generate configurations, and check if the results achieved by hand is sufficient.

In this presentation, I use the following geminal configuration to illustrate the algorithm. Then, I give the results for the other geminal configurations, when the number of nodes are higher.

$$\begin{array}{c}
\bullet & \bullet & \bullet & \bullet \\
\end{array}$$
1. Represent the curves of configuration by a sequence:
   - Example: The graph of
     \[
     \begin{align*}
     &0 &1 &2 &3 \\
     &\circ &\circ &\circ &\bullet \\
     &\end{align*}
     \]
     is represented as the list of:
     \[
     [(0,0),(1,1),(2,3),(2,3)]
     \]

2. Represent the operation of switching nodes by the permutation \((0,1,2,3)\).
   - Example: Switch 0 to 1 is presented as the list \((1,0,2,3)\),
     switch 0 to 2 and 1 to 3 is presented as the list \((2,3,0,1)\)

3. Apply an operation to a configuration is illustrated by this example:
   - Operation: \(g=(2,3,0,1)\)
   - Configuration: \(a= [(0,1),(0,2),(1,3),(2,3)]\)
   - \(g(a) = [(2,3),(2,0),(3,1),(0,1)]\)
   - Note that \(g(a)\), and \(a\) are duplicated configuration
The determinant of the matrix is represented symbolically:

- Determinant of a $2 \times 2$ matrix:
  \[
  \begin{vmatrix}
  a & b \\
  c & d \\
  \end{vmatrix}
  \rightarrow [(a, d), (b, c)]
  \]

- Determinant of a $3 \times 3$ matrix:
  \[
  \begin{vmatrix}
  a & b & c \\
  x & y & z \\
  m & n & p \\
  \end{vmatrix}
  \rightarrow [(a, y, p), (a, z, n), (x, b, p), (x, c, n), (m, b, z), (m, c, y)]
  \]

- **Note:**
  - If the matrix is $n \times n$ then the determinant is the list of $n!$ elements
  - The element’s signs are all neglected
Algorithm Outline

The algorithm basically has three main steps

- Step 1-Generate the configurations of curved edges.
- Step 2-Generate the symmetry groups.
- Step 3-Remove duplicates configuration of curved edge using the symmetry groups.
1. Construct all the configurations of curved edges by compute symbolically the determinant of the matrix:

\[ M = \begin{bmatrix}
(0,0) & (0,1) & (0,2) & (0,3) \\
(1,0) & (1,1) & (1,2) & (1,3) \\
(2,0) & (2,1) & (2,2) & (2,3) \\
(3,0) & (3,1) & (3,2) & (3,3)
\end{bmatrix} \]

2. Order all the determinant’s elements. Rules of ordering are as follows:
   - \((a_i, a_j)\) has \(a_i < a_j\)
   - \([(a_{00}, a_{01}), (a_{10}, a_{11}), (a_{20}, a_{21}), \ldots]\) has \(a_{00} < a_{10} < a_{20} < \ldots\)

3. Delete the same elements received after ordering.
There are 17 configurations of curved edges in the case 4 dots:

\[ A = \left[ \left( (0, 0), (1, 1), (2, 2), (3, 3) \right), \left( (0, 1), (0, 1), (2, 3), (2, 3) \right), \left( (0, 1), (0, 1), (2, 2), (3, 3) \right), \left( (0, 1), (0, 2), (1, 3), (2, 3) \right), \left( (0, 2), (0, 3), (1, 2), (1, 3) \right), \left( (0, 0), (1, 2), (1, 3), (2, 3) \right), \left( (0, 3), (0, 3), (1, 1), (2, 2) \right), \left( (0, 0), (1, 1), (2, 3), (2, 3) \right), \left( (0, 0), (1, 3), (1, 3), (2, 2) \right), \left( (0, 0), (1, 2), (1, 2), (3, 3) \right), \left( (0, 2), (0, 2), (1, 1), (3, 3) \right), \left( (0, 1), (0, 2), (1, 2), (3, 3) \right), \left( (0, 1), (0, 3), (1, 2), (2, 3) \right), \left( (0, 3), (0, 3), (1, 2), (1, 2) \right), \left( (0, 1), (0, 3), (1, 3), (2, 2) \right), \left( (0, 2), (0, 2), (1, 3), (1, 3) \right), \left( (0, 2), (0, 3), (1, 1), (2, 3) \right) \right] \]
We will consider the case:

\[ \begin{array}{ccc}
\text{g}_0 & \text{g}_1 & \text{g}_2 & \text{g}_3 \\
\text{g}_1 & \text{g}_2 & \text{g}_3 & \\
\text{g}_2 & \text{g}_3 & \\
\text{g}_3 & & & \\
\end{array} \]

1. Establish the list \( G_0 \) of the generators:

\[ G_0 = [g_0, g_1, g_2, g_3] = [(0, 1, 2, 3), (1, 0, 2, 3), (0, 1, 3, 2), (2, 3, 0, 1)] \]

2. Create matrix \( G_1 \) from the list \( G_0 \):

\[
\begin{array}{cccc}
g_0 & g_1 & g_2 & g_3 \\
g_1 & g_1 & g_2 & g_3 \\
g_2 & g_2 & g_3 & \\
g_3 & g_3 & & \\
\end{array}
\]

3. Remove the duplicates from elements of matrix \( G_1 \) to establish the list \( G_1 \):

\[ G_1 = [g_{10}, g_{11}, \ldots, g_{1k}] \]

4. In the similar procedure, create the list \( G_2 \) from the list \( G_1 \)

5. Keep generating \( G_3, G_4 \ldots \) until the length of \( G_n \) is equal to the length of \( G_{n+1} \), at this point we have generated the whole groups.
There are 8 group elements in this case:

\[
G = [(1, 0, 3, 2), (2, 3, 0, 1), \\
(3, 2, 0, 1), (3, 2, 1, 0), \\
(0, 1, 2, 3), (1, 0, 2, 3), \\
(0, 1, 3, 2), (2, 3, 1, 0)]
\]
1. Apply $G[0]$ to $A[0]$ to get $G[0](A[0])$

2. Reorder (using the rules already mentioned) $G[0](A[0])$ to get $K$

3. Check $K$ with all $A[i]$: 
   - If $K = A[i]$, delete $A[i]$
   - If $K \neq A[i]$, keep $A[i]$

4. Redo the above steps with $G[1]$, $G[2]$, ...

5. Put $A[0]$ into $F$ (empty list)

6. Put the remaining $A[i]$ into $A_1$

7. Redo the previous procedure with $A_1$

8. Keep doing this procedure until the list $A_k$ is empty

9. $F$ is the list that contains all the distinct configurations.
There are 8 distinct configurations in the case of 4 dots

This result is the same as the result calculated by hand

\[ F = \left[ ((0, 0), (1, 1), (2, 2), (3, 3)), \\
((0, 1), (0, 1), (2, 3), (2, 3)), \\
((0, 1), (0, 1), (2, 2), (3, 3)), \\
((0, 1), (0, 2), (1, 3), (2, 3)), \\
((0, 2), (0, 3), (1, 2), (1, 3)), \\
((0, 0), (1, 2), (1, 3), (2, 3)), \\
((0, 3), (0, 3), (1, 1), (2, 2)), \\
((0, 3), (0, 3), (1, 2), (1, 2)) \right] \]
In case of:

- By hand, they found 20 configurations.
- By computer, I have 21 configurations.
- After coordinating with Dhinali, I found that they missed this one:

\[
\begin{array}{c}
\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
\end{array}
\]
In the case:

- By hand, they found 32 configurations; however, 2 of them are duplicates. By computer, I found 33 ones.
- The 3 new ones I found are:
Computers beat humans!
Generating by hand missed some configurations, especially when the number of nodes increase.

Symmetry groups are useful in finding the duplicates.

So far, I have done 3 cases. There are still other cases to do.

Finally, I wish to express my gratitude to Dr Martin J. Mohlenkamp for providing the idea for the algorithm, and helping me while I write the program.