On the Automation of the Token Graphs

M. K. Christophe Ndjatchi
Department of Physics and Mathematics, Instituto Politécnico Nacional, UPIIZ, P.C. 098160, Zacatecas, México

David Betancourt Montellano
Department of Physics and Mathematics, Instituto Politécnico Nacional, UPIIZ, P.C. 098160, Zacatecas, México

Onder Francisco Campos García
Department of Physics and Mathematics, Instituto Politécnico Nacional, UPIIZ, P.C. 098160, Zacatecas, México

Hugo Pineda Martínez
Unidad de Ingeniería eléctrica, Universidad Autónoma de Zacatecas, Zacatecas, México

ABSTRACT
Let G be a simple graph of order $n \geq 2$ and let $k\epsilon\{1,2,\dots,n-1\}$. The k-token graph $F_k(G)$ of G is the graph whose vertices are the k-subsets of $V(G)$, where two vertices are adjacent in $F_k(G)$ whenever their symmetric difference is an edge of G. The generation and drawing of the token graphs of a given graph are not easy due to their high number of vertices and edges, so the study of such graphs turn out to be a very complex task without using an efficient software. In this paper, an efficient software with a user-friendly interface is presented. This software was developed in the high-level programming languages *C++* and *Wolfram Mathematica*, and it is able to automatically generate token graphs of a given graph so that the study and the teaching of that family of graphs become easier.

General Terms
Combinatory, graph, high-level programming language.

Keywords
Token graphs, symmetric difference, user-friendly interface.

1. INTRODUCTION
Graph theory is often considered as one of the most modern areas of mathematics, although it was founded in 1736 with the work of Leonhard Euler by the famous problem of the seven Königsberg bridges. It is important to mention that there are currently several applications of graph theory in various areas of engineering [1]. In addition, these applications are related to both civil and military problems. Recently, several investigations have been carried out on a special class of graphs called token graphs [2-7]. Recall that, a k-token graph $F_k(G)$ of a graph G is the graph whose vertices are all k-subsets of $V(G)$, and two k-subsets are adjacent whenever their symmetric difference is a pair of adjacent vertices in G. Let G be a simple finite graph of order $|V(G))| = n,\ n \geq 2$ and let $k\epsilon\{1,2,\dots,n-1\}$, then the number of vertices in $F_k(G)$ is $|V(F_k(G))| = \binom{n}{k}$ and in [2], it is proved that the number of edges of $F_k(G)$ is $|E(F_k(G))| = \binom{n-2}{k-1}|E(G)|$.

Therefore, it is not difficult to see that, the number of vertices and edges of a k-token graph is very high for some values of n and k. Thus, the generation, the drawing and, of course the teaching of such graphs turn out to be a very complex task without the use of modern computers, which should be programmed by the teacher and / or researcher. Indeed, the study and the teaching of token graphs become a task with a great complexity for those teachers and / or researchers who do not know how to program, since there is no software with a user-friendly interface that calculates the vertices, the edges and adjacency matrices of token graphs. That is, there does not exist a software that generates them and draws them automatically. In this paper, an efficient software with a user-friendly interface is presented. This software was developed in the high-level programming languages C++ (using *QT creator* open source version) and *Wolfram Mathematica* [8], which is able to automatically generate token graphs and their digital images with Igraph/M [9], so that the study and the teaching of that family of graphs become easier.

2. MATERIALS AND METHODS
2.1 Software
The software of automation of token graphs is based on the next requirements:

- Wolfram Mathematica v11.0.0 or higher.
- WolframScript v12.0.0 or higher.
- IGraph/M a Mathematica interface for igraph v0.4.
- MinGW – GNU Compiler Collection (with C++), 64 bits, v7.3.0.
- Microsoft Windows 10, 64 bits.
- QT Open source, v5.14.2.

2.2 Pipeline
The pipeline represents, at high level, the sequence of steps in the process of automation of token graphs of a given graph. In Figure 1, the input is an undirected graph G of order n; the immediate action is to save the adjacency matrix of the graph G in a plain text file. This adjacency matrix, say A, is an $n * n$ matrix, where the element $A[i][j] = 1$ if (i,j) is an edge of G; otherwise, $A[i][j] = 0$, [10]. Then, the Wolfram script is executed to generate the visual representation (digital image) of G and thus save it in a folder within results; the matrix A is also saved in the same folder. In order to calculate $F_k(G)$ where $k\epsilon\{1,2,\dots,n-1\}$; for each k, three steps are made. In the first step **4.1.1,** the vertex combination is done to generate the subsets of size k (k-subsets). This k-subsets are the vertices of the k-token. In the second step **4.1.2**, the symmetric difference is applied to all pairwise k-subsets, if the symmetric difference of any two k-subsets is a pair of vertices of G and both vertices are adjacent in G, then both k-subsets are adjacent in $F_k(G)$; and generate a value *1* in the adjacency matrix A of $F_k(G)$. With this process, the upper or lower triangular matrix A is created. Since the graph is

undirected. Then to complete with the construction of A, the symmetric reflection of the elements of the matrix is done. Note that A is a $m * m = \binom{n}{k} * \binom{n}{k}$ matrix. In the last step, the digital image is generated, using Mathematica as well as saving A and k-subsets in a plain text file

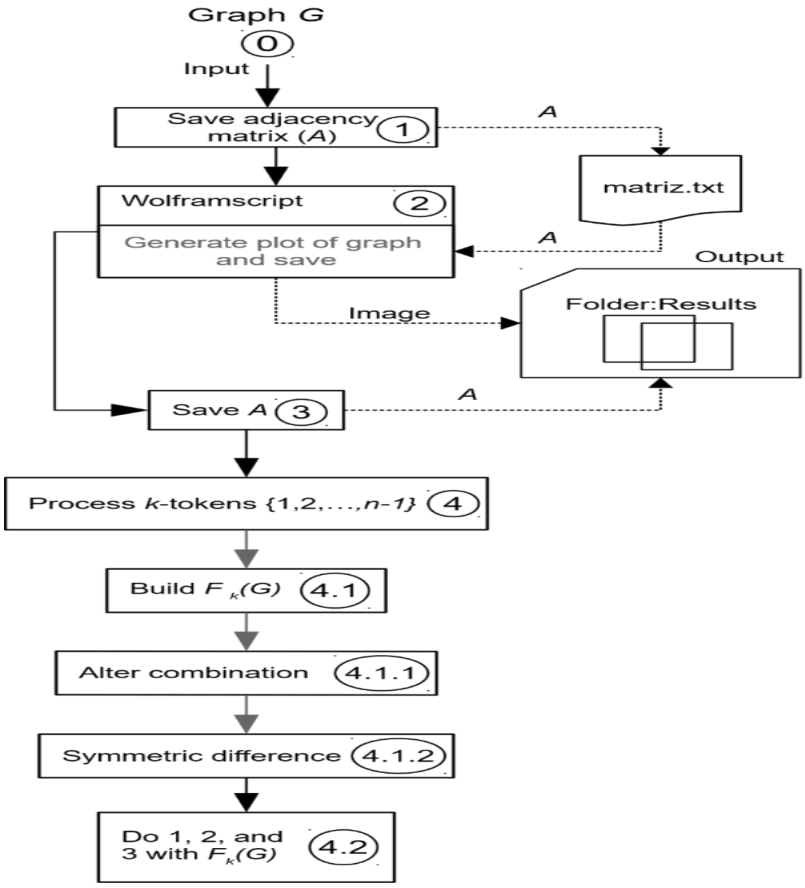

Fig 1. Pipeline to generate the k-token graphs.

2.3 Algorithm and its analysis

Input: Positive integer n that is the number of vertices, and k a positive integer indicating the size of the subsets of $V(G)$ in $F_k(G)$.

Output: $\binom{n}{k}$ k-subsets representing the vertices of $F_k(G)$.

1. Declare $V(F_k(G)$ as a vector of vectors, $vectors$, to store the k-subsets.
2. Declare collection $c[n]$, array of integers of size n.
3. **for** $\forall\ i\epsilon\ \{0,1, \dots, k-1\}$ **do**
4. $c[i] = k - i$, generate the i-th element of the base k-subset, descending order.
5. **end**
6. **while** $True$ **do,** generate all k-subsets.
7. Declare array $k\_subset\_i[k]$
8. **for** $\forall\ i\epsilon\ \{k-1, k-2, \dots, 0\}$ **do**
9. $k\_subset\_i[k-1-i] = c[i]$, assign elements to k-subset.
10. **end**
11. Add k_subset_i to $vectors$
12. **if** $c[0]+1 < n$ **then**, check if all vectors have already been generated for $c[0]$.
13. $c[0] = c[0] + 1$, update the last (descending order) element.
14. **goto** 6, when not all vectors have been generated yet.

15. **else**, increase the values of $c[]$
16. $i = 0$
17. $c[i] = c[i] + 1$, to be able to check if it is already completed.
18. **while** $c[i] >= (n - i)$ **do**
19. $i = i + 1$, increase index i.
20. **if** $i = k$ **then**
21. **goto** 30, when all the vector combinations are done.
22. **end**
23. **end**
24. $c[i] = c[i] + 1$, update $c[i]$ to from it update the other values of $c[]$.
25. **for** $\forall\, j\epsilon$ {i, i − 1, ... ,1} **do**
26. $c[i - 1] = c[i] + 1$, increase (in descending order) $c[i - 1]$ element by 1.
27. **end**
28. **end**
29. **end**
30. Output: $vectors$

Fig *2.* Algorithm of Alter combination for generating subsets of size k (k-subsets), with time complexity : $O(n^k)$.

The algorithmic complexity of Alter combination algorithm can be upper bounded (considering a worst case) with Big-O about time [10]. Since G is an undirected graph of order n, then it is easy to see that the number of k-subsets on n elements is $\binom{n}{k}$. Hence, there are $\binom{n}{k}$ output vectors, which represent the vertices of $F_k(G)$. In [11], it is known that $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. In the pseudocode of Figure 2 the **for** loop of line 8 must be run $\binom{n}{k}$ times. On the other hand, in [12] Shagnik Das proposes an upper bound 2^n. However, it is imprecise. The most precise upper bound is $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$ then$\binom{n}{k} \leq \left(\frac{n^k e^k}{k^k}\right)$. Hence the increasing factor for a large enough n is n^k therefore it is denoted as $O(n^k)$.

Input: $V(F_k(G))$ set of all vector[i], i=0,…, m-1, that are vertices of $F_k(G)$; and adjacency matrix A of G

Output: Adjacency matrix of $F_k(G)$, am_fk_g.

1. Declare an integer m and initialize with $|V(F_k(G))|$.
2. Initialize am_fk_g, with zeros, of size $m * m$.
3. **for** $\forall\, i\epsilon$ {0,1, ... , $m - 1$} **do**
4. **for** $\forall\, j\epsilon$ {i, i + 1, ... , $m - 1$} **do**
5. $subset = symmetric\_difference(vector[i], vector[j])$
6. **if** $|subset| = 2$ **then**
7. **if** $A[subset[0]][subset[1]] == 1$ **then,** $subset[0]$ and $subset[1]$ are adjacent in G.
8. $am\_fk\_g[i][j] = 1$, set an edge in $F_k(G)$.
9. $am\_fk\_g[j][i] = 1$, do symmetric reflex.
10. **end**
11. **end**
12. **end**
13. **end**
14. Output: am_fk_g

Fig 3. Algorithm of Symmetric difference, with time complexity: $O(m^2)$.

Time complexity $O(m^2)$ in symmetric difference is due to nested iterations, lines 3 and 4, wherein the upper iteration is performed m times $\forall\ i\epsilon\{0,1,\ldots,m-1\}$. While nested iteration runs $m-i$ times, inside the previous various operations are executed in constant time p, in this way the number of operations to be performed are $m*((m-i)*p) = m(mp-ip) = m^2p-mip$, where m^2 is the increasing rate factor for sufficiently large values of m, therefore, the upper bound denoted by Big-O is $O(m^2)$.

3. RESULTS AND DISCUSSION

The system was evaluated on a personal computer with Windows 10, 64 bits, RAM 8 GB, processor Intel(R) Core (TM) i3-5005U CPU @ 2.00 GHz.

Next, there are some screenshots of the program's execution:

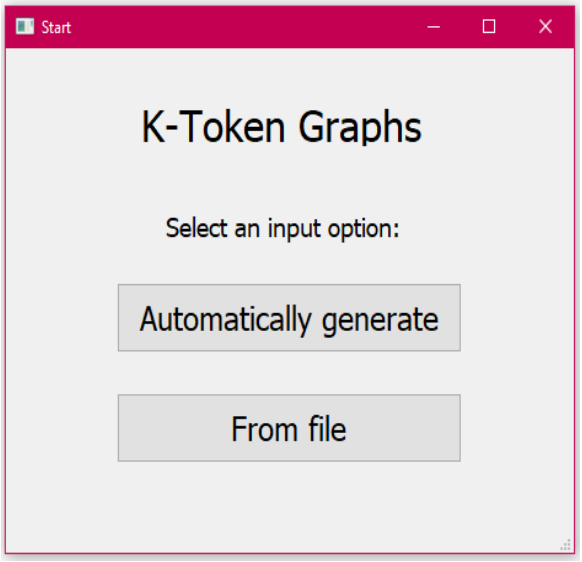

Fig 4. Main menu.

In the main menu, the user constructs the input adjacency matrix of an undirected graph G of order n (see Figure 4). This matrix can be generated in two ways. It can be generated automatically (see, Figure 5) or from an input file, which must contain the adjacency matrix of the graph G.

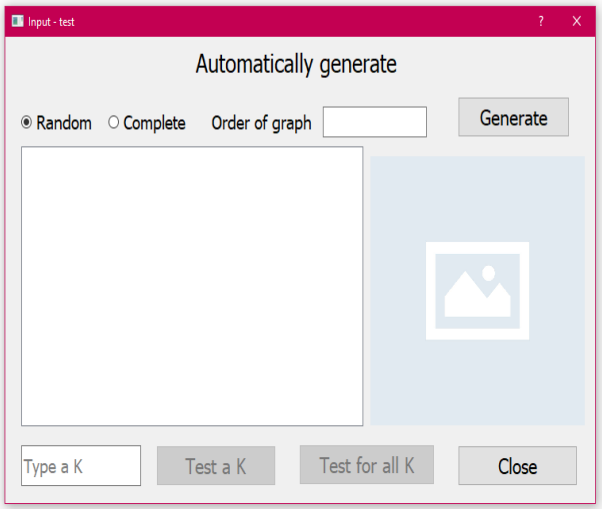

Fig 5. Automatic generation of input adjacency matrix of an undirected graph G.

As a test, an undirected graph G of order 8 is generated (see Figure 6).

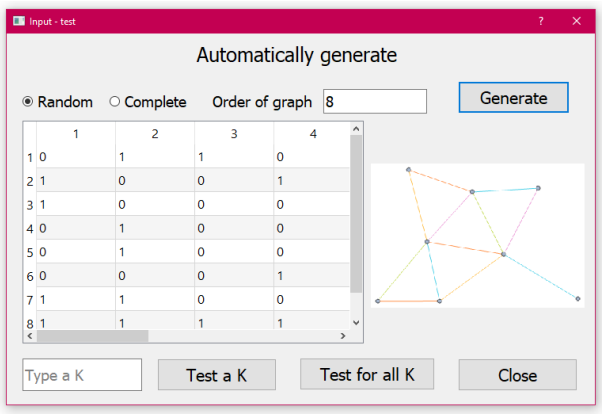

Fig 6. The adjacency matrix of G is shown on the left and a drawing of G on the right.

It is possible to calculate the k-token graphs $F_k(G)$ of a graph G for all $k\epsilon\{1,2,\ldots,n-1\}$ (see Figure 6). The result is displayed when clicking on "Test for all K" (see Figure 7). Note that the adjacency matrix of each k-token graph and its drawing are generated and saved in the directory. In Figure 12, for a graph G of order $n=8$, the files that contain the adjacency matrix of each k-token graph and its drawing are generated. However, it is also possible to only generate the k-token graph of G for a certain value of k, clicking on "Test a K" (see Figure 6).

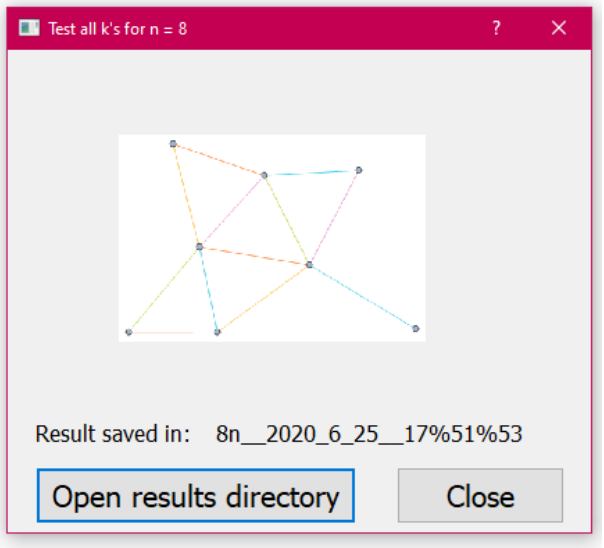

Fig 7. A dialog window of the result, which is displayed when clicking on "Test for all K".

The following figures show some of the generated k-token graphs of a graph G. Inside the results folder in Figure 12, there are images in *.png* format that contain the drawings of the k-token graphs. For example, the file "k_1" is a drawing of the 1-token graph of G (see Figure 12). In addition, the files in *.txt* format contain the adjacency matrix of the k-token graphs of G. Furthermore, they show the list of the vertices of k-token graphs of G. Finally, they exhibit the adjacency matrix of graph G. On the other hand, the user can observe a drawing of G (see Figure 8), when clicking on the file called as *original* (see Figure 12). In Figure 13, the file "n8 k3.txt" represents the adjacency matrix of the 3-token graph of G of order n=8. In "n8 k3.txt", there is also the adjacency matrix

of G. Note that, the adjacency matrix of the 3-token graph of this graph is too large. In fact, the 3-token graph of G has 56 vertices. Then, in Figure 13, it is exhibited part of the 56 vertices of 3-token graph that are in the file "n8 k3.txt".

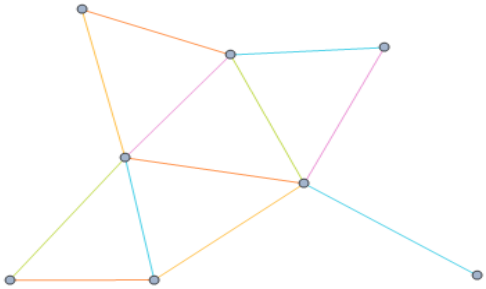

Fig 8. The graph G of order n=8.

Note that, the k-token graph for $k = 1$ is isomorphic to G and $F_{n-1}(G)$ [2]. In Figure 11, it is easy to see that $F_7(G)$ is isomorphic to the graph G. The 2-token graph of an undirected graph G of order n=8 has 28 vertices (see Figure 9). However, the 5-token graph of this graph G has 56 vertices (see Figure 10).

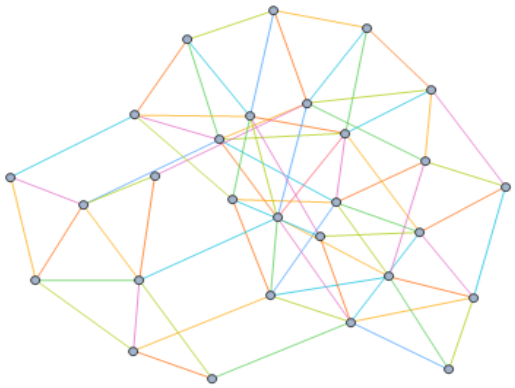

Fig 9. The 2-token graph of G of order n=8.

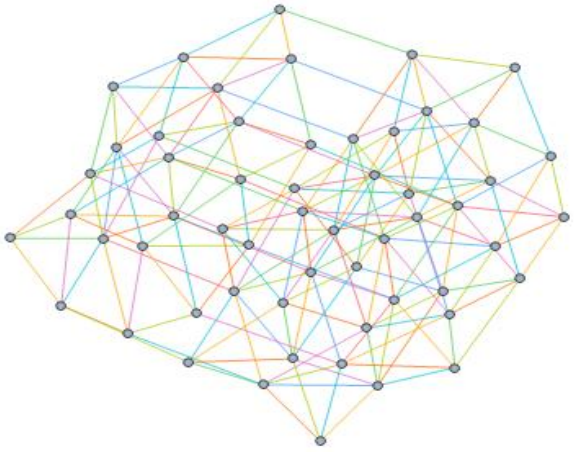

Fig 10. The 5-token graph of G of order n=8.

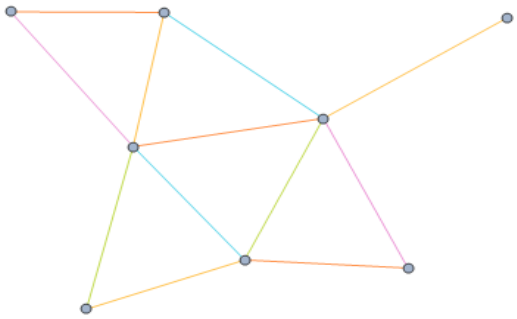

Fig 11. The 7-token graph of G of order n=8.

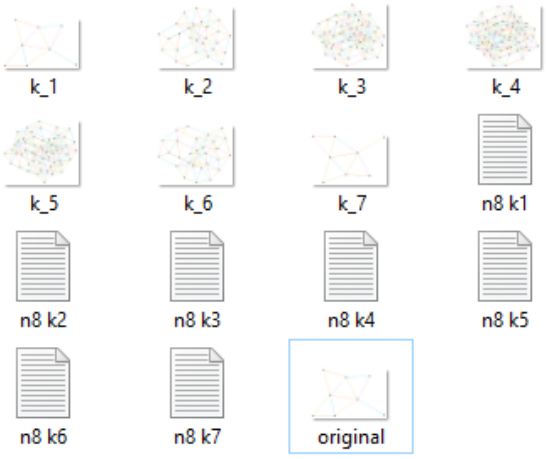

Fig 12. Generated files.

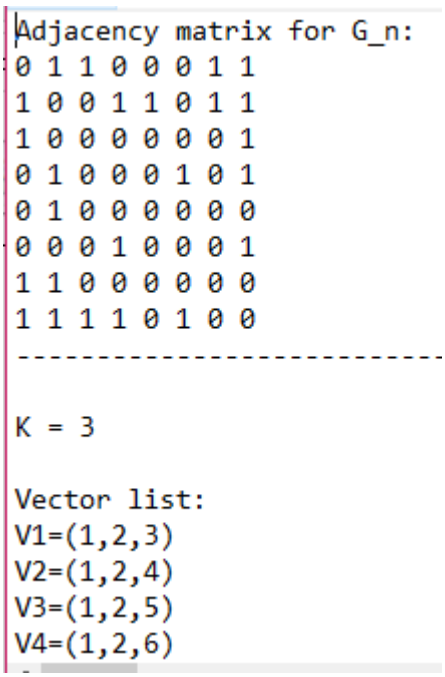

Fig 13. Inside file "n8 k3.txt".

On the other hand, in order to measure the time efficiency, some tests are done. Here, it is generated all the k-tokens of a complete graph K_n, which are known as Johnson graphs $J(n,k)$ [2]. Moreover, the time it takes the program to generate $J(n,k)$ is measured. The results are shown in Table 1.

Table 1. Time efficiency for the generation of the k-tokens of a complete graph of order n .

Order (n) of G	Time (seconds)
2	6.01342
3	12.36
4	18.3497
5	25.0156
6	31.1685
7	37.995
8	51.7211
9	54.2779
10	68.0613
11	90.0575
12	145.017
13	274.383

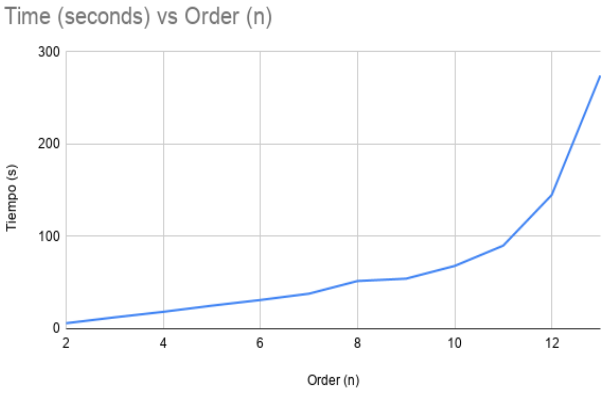

Fig 14. Growth rate for some values of n.

As shown in Figure 14, for the complete graphs of order $n \geq 11$, the computation time, for the generation of the k-tokens of K_n, has an accelerated growth rate. However, this time efficiency for the generation of k-tokens of a complete graph is still good compared to the very high number of vertices and edges of Johnson graphs. That is, $|V(F_k(G))| = \binom{n}{k}$; $|E(F_k(G))| = \binom{n-2}{k-1}\binom{n}{2} = \frac{k(n-k)}{2}\binom{n}{k}$.

4. CONCLUSION

The generation and drawing of token graphs of a given graph are a very complex task due to their high number of vertices and edges. For the graphs of order $n \geq 11$, a combinatorial explosion is obtained [13]. However, time efficiency for the generation of the k-tokens for a given graph is still good. Moreover, the software has been developed in C++. Indeed, a programming language that is efficient and flexible [14], such as C++, minimizes the execution time is minimized as much as possible. Concerning the tool for the generation of the graph image in Mathematica, note that it takes a constant time, but for values of n ($n \geq 11$, see Table 1) the process takes the longest. However, this tool saves a lot of time for the researchers or teachers who are interested in exploring, teaching or learning about k-token graphs of a given graph.

5. ACKNOWLEDGEMENTS

This work was supported by Instituto Politécnico Nacional, UPIIZ. The authors gratefully acknowledge the reviewers.

6. REFERENCES

[1] D. Kornhauser, G. Miller, and P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutations groups, and applications, Proc. 25th IEEE Symposium on Foundations of Computer Science 1 (1984), 241-250.

[2] Ruy Fabila-Monroy, David Flores-Peñaloza, Clemens Huemer, Ferran Hurtado, Jorge Urrutia, and David R. Wood, Token graphs, Graphs Combin. 28 (2012), no. 3, 365-380.

[3] J. M. Gomez Soto, J. Leaños, L. M. Ríos-Castro, and L. M. Rivera, The packing number of the double vertex graph of the path graph, Discrete Appl. Math. 247 (2018), 327-340, DOI 10.1016/j.dam.2018.03.085. MR3843346

[4] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno, On the complexity of reconfiguration problems, Theoret. Comput. Sci. 412 (2011), no. 12-14, 1054-1065.

[5] Koenraad Audenaert, Chris Godsil, Gordon Royle, and Terry Rudolph, Symmetric squares of graphs, J. Combin. Theory Ser. B 97 (2007), no. 1, 74-90.

[6] J. Leaños and A. L. Trujillo-Negrete, The connectivity of token graphs, Graphs and combinatorics 34 (2018), 132-138.

[7] J. Leaños and M. K. Ndjatchi, The Edge-connectivity of Token Graphs, arXiv preprint arXiv: 1909.06698, (2019).

[8] Wolfram Research, Inc., Mathematica, Version 12.0, Champaign, IL (2019)

[9] Szabolcs Horvát. (2020, April 3). IGraph/M (Version v0.4).

[10] Skiena, S. The algorithm design manual. Springer, 2nd ed. (2008), 34-35, 151.

[11] Kay, S. Intuitive Probability and Random Processes Using MATLAB. Boston, MA: Springer US, (2012), 60.

[12] Shagnik Das, A brief note on estimates of binomial coefficients, (2016), 2-3.

[13] Domingos, P. The master algorithm. Basics Books, (2015). 7.

[14] Fco. Javier Ceballos Sierra, Curso de programación C/C++,(1995), 6.