# Two algorithms for finding core-periphery structures in complex networks

**Javier Lobillo Olmedo**[a]

This manuscript was compiled on August 30, 2024

**The field of network science has grown significantly in recent years as we have developed new tools and methods for analyzing the structure and properties of networks. In this paper, we will look at one particular subfield of network science called core-periphery structures. As in its name, we specify two groups within a particular network: the core, a dense central structure, and the periphery, the sparse structure on the periphery of the network. We introduce two algorithms in order to study this structure on particular unweighted and undirected networks. In the first algorithm, we naively classified nodes based on degree, and found that this method was more than solid. Secondly, we introduce the *MRJZ* algorithm by Ma et al. (1), which defines a new centrality measure for classifying the core and periphery. Then we compare both algorithms' performance using an "objective" function as criterion. The tested dataset is based on J.R.R. Tolkien's The Lord of the Rings. Besides comparing the two algorithms, we also want to dive into the topic of applying network science to literature, which we think is a promising and fairly unexplored topic.**

Community structures | Core-periphery structures | Core-periphery algorithms | Lord of the Rings | Literature | J.R.R. Tolkien

## Significance Statement

When studying network science, understanding the underlying structure of networks is key to extract information and further analyze them. Core-periphery structure is a very natural and important case of structure that arises many networks. We study this by exploring the theoretical elements of core-periphery structure as well as tools for computing it. We apply these tools to a better understand the underlying structures present in J.R.R Tolkien's *Lord of The Rings*. This paper improves our understanding of algorithms for computing core-periphery structures, as well as methods for analysing said algorithms and provides new insights enabling future research of this field and the application of network science in literature.

Community structures play an integral role in the analysis and study of networks. They tell us information about groups of nodes within networks and their importance, allowing us to view networks holistically and reduce the complexity of analysis. In this paper, we focus on the core-periphery structures of networks. Although these structures are not precisely defined, the essence is that the core nodes are very well connected with each other, they also are fairly well connected to periphery nodes, and periphery nodes are very poorly connected among themselves. The theory surrounding core and periphery structures also admits differentiating between several cores, one inside the next one, as explained in *Networks* by Newmann (2), although we will focus only on finding one single core.

Currently, we find many algorithms to solve the core-periphery problem. Thus, an issue arises when one needs an implementation of an algorithm to compute a core-periphery structure and, thus, faces the choice of which algorithm to use.

Furthermore, there is not even universal objective criteria to compare two algorithms. Some people will prefer complexity, other people decide based on some characteristic of the results they provide, while some people even decide whether they want the best result or a good enough result. And even what a good enough result is varies from person to person.

In this paper, we will begin by introducing the basic theoretical aspects and definitions on the core-periphery topic. In the second part we will focus on two different algorithms that compute a core-periphery partition for an input network. As explained in Networks (2), simply putting the nodes with the highest degree in the core is a pretty good division, and actually this will be our first algorithm. Our second algorithm, the MRZJ algorithm, will be based on the work of Ma et al. in their article (1), where they propose another method for our task. Afterwards we will also compare the two techniques using the work of Cucuringu et al. in (3), where they present an *objective* function that measures a certain core-periphery partition. This function can be used to try to (locally) improve any given partition. This function was also included in Ma et al. (1), and comes in very handy as an objective criterion to compare algorithms in some fixed networks.

Naturally, core-periphery structures arise in many networks, from transportation networks, to social networks or social sciences networks like networks arising from literature. In our case, we applied our two algorithms to a literary network, which consists of the list of the main entities (characters, places, things, groups) from the series of books The Lord of the Rings by J.R.R. Tolkien, where two entities are related if they appear in the same paragraph at any point of the books.

Author affiliations: [a,2]University of Zurich department of Mathematics, Winterthurerstrasse 190, 8057, Switzerland.

[2]To whom correspondence should be addressed. E-mail: javier.lobilloolmedo@uzh.ch

The reason why we made this choice is based on the importance LOTR has as a literary piece in the history of literature, and understanding it can be a good point of study for many experts in the field.

Finally, the two algorithms are compared based on the output they provide for each of the LOTR network, as well as some synthetic examples we prepared for the study of our chosen algorithms. Again, these comparisons where carried out by the means provided in Cucuringu et al. (3).

At the end of this work, we justify why we think the naive algorithm based on the work by Zhang et al. (4) is a better algorithm for our particular datasets, while we give some hints as to why we think it is a better algorithm in general, although, as we will see, that does not mean the MRZJ algorithm is bad in every aspect.

## 1. Networks and structure

In order to study the core-periphery structure of networks (also called graphs), we need to first discuss general networks, and the network we will be analyzing. A graph, or network $G$ is the set $(V, E)$ where $V$ represents the set of nodes, while $E \subseteq V \times V$ represents the set of edges between pairs of these nodes. In an applied setting, nodes can represent a multitude of things, such as people, places, or things. Edges, on the other hand, represent the connections between these nodes. If the edges have a particular direction from one of the two nodes it connects towards the other one, we will say the graph is directed. If, on the contrary, the edges do not express any direction, we call this network *undirected*, so the edges only represent a symmetric connection between the two nodes. We can denote networks by the adjacency matrix $A$, where we represent $n$ nodes via an $n \times n$ matrix. Here, the matrix elements $a_{ij}$ are equal to 1 if there exists an edge from $i$ and $j$ and zero otherwise. In an undirected network, the adjacency matrix is trivially symmetrical.

One could also assign a numerical value to each edge, called weight of the edge. In such case we would be speaking of a *weighted* network. On the other hand, if we do not assign any weight to the edges, we speak of an *unweighted* network.

In an undirected network, additionally, we denote the degree of a node to be the sum of the weights of the edges connected to each itself. In an unweighted network, we consider the edges to have weight 1. For example, if a node $j$ has 5 unweighted edges to other nodes, then the degree of $j$ is 5. In the directed case, one could naturally define the in-degree and the out-degree.

In our case, we have an undirected and unweighted network where nodes are representing characters, things, places, and groups in the popular *The Lord of the Rings* trilogy, while the edges between these nodes occur if and only if two entities occur together in the same paragraphs within the books. Our network has 73 nodes (entities) and 1444 edges between them. With this structure, we now can analyze the underlying properties of this network, in particular the core-periphery structure of the network.

## 2. Core-periphery structure

The first thing we need to discuss is what the networks we are going to work upon look like. Our networks will be represented by an undirected, unweighted graphs with no self loops. We made this choice as to be able to work upon the work of Ma et al. in (1), which focuses on unweighted networks.

However, these restrictions are not necessary in order to define a core-periphery structure in a network, for which we use the definition in (2), where more details can be found. For a given network, we first define two groups, the core and the periphery, which we are going to call group 1 and 2, respectively. Now, denote by $p_{ij}, i = 1, 2$ the average probability of edges between groups $i$ and $j$. In this notation, it is clear that $p_{12} = p_{21}$, which is the probability of edges between nodes in different groups. If we have $p_{11} > p_{12} < p_{22}$ we obtain the classical assortative community structure, where the probability of edges within groups are higher that the probability of edges between groups. Conversely, if we had $p_{11} < p_{12} > p_{22}$ then we would obtain a dissortative community structure. The remanding case of $p_{11} < p_{12} < p_{22}$ is just a renaming of the first one.

The case we are interested in is that in which $p_{11} > p_{12} > p_{22}$, which is what we mean exactly in this paper by core-periphery structure. In this case, there are two clear different groups, groups 1 and 2 where the first one is a dense core group and the second one is a sparse periphery group with an intermediate probability of connections between these two groups. As we will later see, a natural way of visualizing this structure is by looking at a transportation network, for example, an underground network, where there are some big central stations which are connected to many more stations via several lines than the ones that are usually far away and only receive one line.

Now, two observations should be made. First, we do not need the unweighted and undirected conditions, but we made these assumptions for simplicity reasons and in order to be able to work upon other authors like Ma et al. (1). Secondly, core-periphery structure need not be limited to just two groups, and one could have three or more groups ranging from the innermost core with the highest density to the outermost one with the lowest one. These networks are usually said to have an *onion* structure. Also, one could have several cores not directly related to each other. Again, these two cases will not be considered for the sake of simplicity, but we refer to (2) for more details.

## 3. Algorithms for detecting core-periphery structures

Now, it is time we introduce our algorithms to detect the core-periphery structure in networks in this section. The first one will be the very simple degree-based algorithm, which just classifies the nodes based on their degree. Although this algorithm looks too simple, we will see that it actually provides a very good division into our two groups. Secondly, we will introduce the algorithm by (1), where they define a centrality measure and work upon it. We will call this last algorithm the $MRZJ$ algorithm, based on the initials of its original authors.

Finally, we are going to introduce a method for comparing how good a division is and how it can be improved based on the work in (3) and as used by (1).

**A. Degree-based algorithm.** This algorithm takes a very simple approach, which basically consists on ordering all nodes by their degree and choosing the ones with higher degree to be on the core, while the rest are on the periphery

group. However, there is a very important and not so trivial question to ask which is: where do we put the threshold on the degree necessary to be in the core group. In order to answer that question, we take a look at the approach taken by Zhang et al. in (4) and we are going to quickly summarize their work.

They first define $\gamma_r$ to be the probability for a random node to be in the group $r = 1, 2$, which fixes $\gamma_1 = 1-\gamma_2$. This parameter gives us the expected sizes of the groups $n_r = n\gamma_r$ where $n$ is the size of the network. Their approach is to model the network using a stochastic block model in order to deduce what these values should be. They also define $q_r^i = \sum_g q(g)\delta_{g_i,r}$ to be the *marginal probability* within the chosen distribution $q(g)$ that vertex $i$ belongs to group $r$ where the sum goes over all assignments of the vertices to groups.

Now, we look at the average degrees in the code and periphery, which are, respectively,

$$\bar{d}_1 = \gamma_1 c_{11} + \gamma_2 c_{12}, \quad \bar{d}_2 = \gamma_1 c_{12} + \gamma_2 c_{22},$$

where

$$p_{rs} = \frac{c_{rs}}{n}$$

and from which we deduce that $\bar{d}_1 > \bar{d}_2$ from the fact that $c_{11} > c_{12} > c_{22}$.

Now, after several computations, they derive the main equations and relations to be used to compute $\gamma_r$. The first ones are

$$\gamma_r = \frac{1}{n}\sum_i q_r^i, \quad R = \frac{\kappa_1}{\kappa_2},$$

where $\kappa_r$ is the expected degree in group $r$:

$$\kappa_r = \frac{\sum_i k_i q_r^i}{\sum_i q_r^i} \qquad [1]$$

(here $k_i$ means the degree of vertex $i$) and $R > 1$ is the ratio between expected degree in groups 1 and 2, defined as follows,

$$R = \frac{c_{12}}{c_{22}}.$$

Also they deduce the equations

$$q_1^i = \frac{\gamma_1 e^{-\bar{d}_1} R^{k_i}}{\gamma_2 e^{-\bar{d}_2} + \gamma_1 e^{-\bar{d}_1} R^{k_i}}, \quad q_2^i = 1 - q_1^i.$$

Now, as they indicate on their paper, the only last thing to do is fix different starting values for the parameters $\gamma_r$ and $p_{rs}$, and iterate the equations until finding a stable solution for each starting value. Only then, we choose the best one among all obtained results and choose the core as the $n_1 = \gamma_1 n$ vertices with highest degree, with the periphery consisting of the rest. Note that some of these initial values can explode towards a "trivial" partition, i.e. a partition in which the core has $0, 1, n-1$ or $n$ elements. Therefore, whenever someone desires to implement this method, they should consider whether these solutions should be accepted or not.

Again, this is just the naive algorithm for computing the core and, for more details on the deduction of the equations, please check the work by Zang et al. (4).

**B. MRZJ algorithm.** This algorithm was introduced by Ma et al. in (1). The notation we use is as follows. We fix an unweighted and undirected network $G = (V, E)$, with $|V| = n$ and with an adjacency matrix $A = (a_{ij})_{i,j}$ where $a_{ij} = 1$ if nodes $i$ and $j$ are connected and $a_{ij} = 0$ otherwise. In this section, again, the degree of node $i$ is defined as $k_i = \sum_j a_{ij}$.

Now, the idea they had is to introduce a new centrality measure that we are going to call *MRZJ centrality*. This centrality $C(i)$ has the property that nodes in the core tend to have the highest centralities while the nodes in the periphery have the lowest centralities. The MRZJ centrality is defined

$$C(i) = \frac{1}{k_i}\sum_{j=1}^{n} \frac{a_{ij}(k_i - k_j)}{k_i + k_j - 2},$$

and they set that a node $i$ will belong to the core if and only if $C(i) \geq 0$ and, thus, node $i$ will be in the periphery if and only if $C(i) < 0$.

There is one characteristic about this algorithm that is worth to note that Ma et al. already presented, and that is that the centrality measure for a given node $i$ only depends on the neighbors of $i$. However, they did not explain how this dependency works. We think the best way to describe this dependency is: a node will have a positive centrality if it has a higher degree than its neighbors, as one can derive from the formula. This can lead to a different result from the DB algorithm as MRZJ will include in the core nodes with lower degree but with neighbors with an even smaller degree, but leave out nodes with high degree that have lower degree than its neighbors. In fact, we will see a case in which this happens in A.

The rest of their algorithm is a very general step. So much so that we will study it as a general method in the next subsection.

**Improvement by objective function.** Now, the next step of the algorithm consists of improving or refining the core-periphery partition obtained so far. However, as we will see, this refinement is independent from the previous steps, which is why it deserves another subsection.

For this improvement they take a look at the work by Cucuringu et al. (3) where they define a function which assigns a value to (almost) every core-periphery partition of a given network. In the work of Ma et al. this function $\Phi$ is called the objective function, which is defined as follows in (3):

$$\Phi = \frac{2E_{11}}{n_1(n_1-1)} + \frac{E_{12}}{n_1 \cdot n_2} - \frac{2E_{22}}{n_2(n_2-1)}, \qquad [2]$$

where $n_1$ and $n_2$ are the number of nodes in the core and the periphery groups, respectively, $E_{11}$ is the number of edges in the core, $E_{22}$ is the number of edges in the periphery, and $E_{12}$ is the number of edges between core and periphery nodes. By definition, $|E| = E_{11} + E_{12} + E_{22}$.

The first term of the sum in Eq. (2) is the edge density of the core, the second term is the edge density between the core and the periphery, and the third one is the edge density between nodes in the periphery. This means that, if chosen correctly, the objective function $\Phi$ will maximize the edge density among core nodes and between core and periphery nodes, while minimizing the edge density among periphery nodes. Thus, the additional step they take in the algorithm is, once we have computed the core using any given algorithm

(in particular, the MRZJ algorithm), to check if adding one of the periphery nodes gives a higher value for $\Phi$ and, in such case, proceed the exact same way from the new core until there is not any additional node in the periphery with such property.

Lastly, this improving algorithm needs a few observations we thought of. First of all, it follows from Eq. (2) that the function is not defined for trivial partitions. This is clearly a problem whenever we want to evaluate one of these partitions, which can arise for example, as we mentioned before, in the DB algorithm A. Problems can arise when we try to use the $\Phi$ function as a way of comparing different results for different choices of $\gamma$ and $p$, if the partition they produce are trivial. This will be the case in our implementation, as we will see in the next section.

Furthermore, we speculate this algorithm could also be used "backwards" and the check could consist in whether eliminating one element of the core, we obtain a higher value for $\Phi$, iterating until getting to a local maximum. One could also consider combining the *additive* and *shrinking* to find a more precise maximum. However, we think this could lead to a very costly algorithm with cost up to $O(2^n)$, as one could end up checking (asymptotically) all combinations of nodes. However, in all implementations we have done, only the adding version is considered. Further research could be done to clarify these points.

## 4. Results

For this section we implemented both the DB algorithm and MRZJ algorithm, as well as the Improving algorithm. Then we tested it on three synthetic graphs, designed as limit cases to better understand the behavior of our algorithms. Lastly, we applied the algorithms to our LOTR dataset.

**A. Some synthetic and extreme examples.** As we mentioned before, we have studied our algorithms over three particular synthetic graphs. The first one is the classical doble-centered *star* graph of cardinality 22, i.e. a graph with two central nodes we called 0 and 11, which are each connected to ten different nodes. On the other hand, these twenty nodes are not connected with each other.

The natural core-periphery structure we expect from this network is that in which the core is conformed by the nodes 0 and 11, while the rest of the nodes form the periphery. This partition would clearly have a core-periphery structure because $p_{11} = 1$, as for any pair of nodes in the core (there is only one), they are trivially connected. Also, $p_{12} = \frac{1}{2}$, as any given node in the periphery will be connected to only one of the two core nodes, and there is the same amount of periphery neighbors for both center nodes. Lastly, $p_{22} = 0$ as all the periphery nodes are disconnected from each other.

After this claim, we applied both the DB and MRZJ algorithms to our star graph first, and then we applied the Improving algorithm to each of them. The four results were exactly the same, which is the predicted core-periphery structure from last paragraph. One can see this partition in Figure 1. The objective function value for this partition is $\Phi = 1.5$, which is a fairly high number. As any other non-trivial combination gives a lower $\Phi$ value, we conclude this is the best partition.

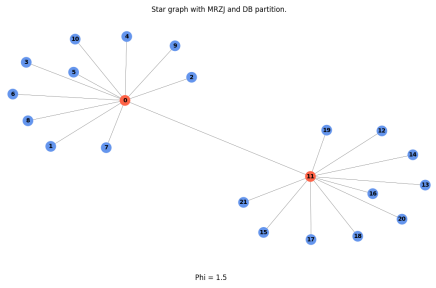


**Fig. 1.** Double-centered star graph with core-periphery partition.

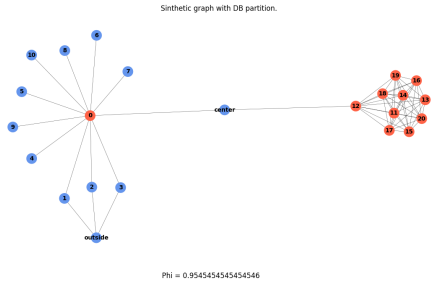


**Fig. 2.** Synthetic graph with DB-produced partition.

The next synthetic example is that of a complete graph of 10 nodes. In this case, the MRZJ algorithm produced the trivial core-periphery structure in which the core contains all nodes. In this case $p_{11} = p_{12} = p_{22} = 1$ trivially, as the periphery is empty. This is not a pretty result, but it should not be surprising. In the end, all nodes have a MRZJ centrality of 0.

However, when we applied the DB algorithm to this limit case, we found out that, no matter the choice of $\gamma$ and $p$, the solutions always exploded towards the trivial ones in which the core was either empty or full. This is perfectly coherent with the MRZJ algorithm as one could choose for the core the nodes with centralities $\geq 0$ or $> 0$, which is a very small difference that determines one of the two trivial solutions. Observe that, in this graph and its produced partitions, it does not make sense to apply the Improving algorithm, as well as the $\Phi$ function. Also, please note that working with solutions that tend towards 0 can be generally dangerous many times. In the implementation of the DB algorithm, when applied to the complete graph, sometimes the values for $q_r^i$ could be so small as to go beyond the accuracy of the computer and be identified as 0, when they really were not. This lead to divisions by zero when computing the $\kappa$ values at Eq. (1) that had to be fixed.

The last example is the graph shown in Figure 2. This graph consists of a simple-centered star graph connected to a complete graph through a "center" node, plus an additional "outside" node connected to three non-centered nodes from the star graph. In this graph, our two algorithms worked differently.

The application of the DB algorithm produced the partition shown in Figure 2. As we can see, the partition consists of the 0 node, central to the star graph, plus all the nodes in

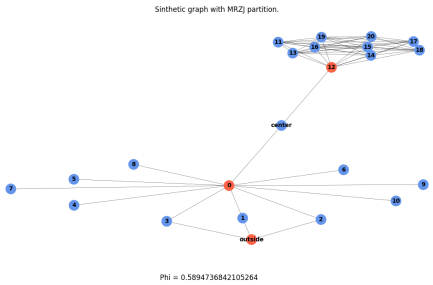


**Fig. 3.** Synthetic graph with MRZJ-produced C-P structure.

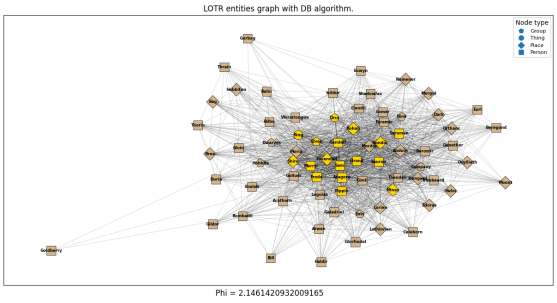


**Fig. 4.** LOTR network with DB partition.

the complete graph part. The objective value is $\Phi \approx 0.955$. The key to this partition is that the $\gamma_1$ value produced is around 0.5, so the core will contain the upper half of the nodes ranked by degree, which are these special nodes already mentioned.

However, the MRZJ algorithm produces a weaker result that can be seen in figure 3. We note that the objective value of this partition is $\Phi \approx 0.599$, a lower value than before. As specified in B, we find some very different results from the DB result. This is because this time the partition contains the node "outside" which has a higher degree than its neighbors, although it does not seem to be as naturally a part of the core. Also, as node 12 has one more neighbor than nodes 11 and $13 - 20$, it is contained in the core while its neighbors from the connected graph are not. Again, this would seem as a poorer choice than that of the DB algorithm.

However, there is room for improvement, and that is because the Improving (adding) algorithm adds the nodes 3 and "center". This addition takes the $\Phi$ function to a value of around 0.858, which is better, but still worse than the one produced by the DB algorithm.

This last graph is the first example in which the characteristic of choosing nodes with a higher degree than its neighbors rather than nodes with the highest degree, makes the partition by the MRZJ algorithm a poorer option than that of the DB algorithm with our chosen criterion of the $\Phi$ function. However, we will see this is not the only case in which DB performs better than MRZJ.

**B. Real World Example.** Since the invention of writing, and especially since the invention of the printing press, literature has been one of the most tools inventions humanity has ever produced. Books contain all kinds of information, art and stories and they can be transmitted to almost every person in the planet.

One of the most popular literary pieces ever produced is J.R.R. Tolkien's *The Lord of the Rings*. Clearly, this story is one of the best ever written and, since it was published in 1954-55, it settled the basis for all modern epic and fantasy novels to be written afterwards. That is the reason why we think many authors and literary experts will be interested in understanding more about the technical structure and characteristics of this trilogy. In order to do so we decided to use the power of community detection in complex networks, particularly through the use of our two algorithms: DB and MRZJ. We think data science is a very powerful tool that has not been applied enough to literature yet and we hope to contribute in this direction.

The network we use is based on the compilation work done by José Calvo (5), who published a dataset with information about the books in Github, which has served as our main dataset in this article.

Our network contains 73 nodes which consist of entities appearing throughout the three books. These entities are classified in four different types: people, groups, places and things. Although one expect characters to be more important than the rest, the reader that is familiar with the book will understand how the Ring, a thing, is one of the most central figures of the books, how the different countries and places in the books are determinant to what is happening at that time, and how Orcs or other groups mean more than each of its members individually.

Secondly, our network contains 1444 undirected and unweighted edges. There is an edge between two nodes if and only if the two entities they represent appear together in at least one paragraph throughout the book. This is as to emphasize when two entities have a special relation together or are closely connected in the story. That way, Goldberry should not be related to Gollum, but Frodo should.

Firstly, we applied the DB algorithm. The result it produced can be seen in Figure 4. We observe very natural results, where Sam, Frodo, the Ring or Gandalf are core nodes, while Durin, Bill or Glorfindel are not. Also, although a particular entity may appear very often in the book, that does not mean that it reaches many other entities, of which Gollum is an example and is in the periphery in our case. We also observe a very high objective value of $\Phi \approx 2.146$. The $\gamma_1$ value provided by the DB algorithm in this network is $\approx 0.23$, which translated into the core having 17 elements, while the periphery contains 56 elements. It is worth noting that the Improving algorithm did not improve this C-P partition.

Secondly, we applied the MRZJ algorithm to our network and we got the core-periphery partition seen in Figure 5. In this case we can see how the core is bigger in size than before. We note how the DB core is a strict subset of the MRZJ core. In this case, the new core contains the 17 DB core nodes plus 12 additional nodes which contain nodes like Theoden, Gimli or Mordor. This leaves the periphery with 44 nodes.

For this partition, the objective value falls down to $\Phi \approx 1.846$. Additionally, the Improving algorithm does not modify this partition, so we can conclude that, based on our

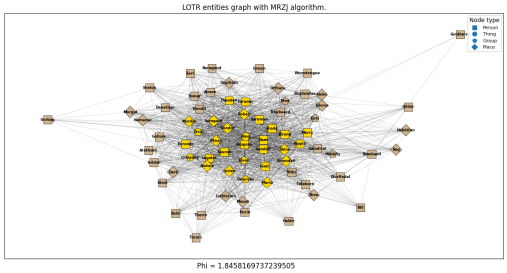


**Fig. 5.** LOTR network with MRZJ partition.

established criterion, the DB algorithm produces a better result than the MRZJ algorithm.

**Discussion.** This project's purpose was to better understand what core-periphery structures are and, specially, how to compute them, being able to compare algorithms for this purpose based on their result performance. For that we established the criterion of the $\Phi$ function by Cucuringu et al. (3) and applied it to the naive Degree Based algorithm based on the work by Zhang et al. (4) and to the MRZJ algorithm by Ma et al. (1). For the application we chose some synthetic datasets and a Lord of the Rings dataset. After applying the two algorithms, we observed how in no case DB was worse than MRZJ and, in some cases MRZJ was worse than DB. This leads us to prefer DB over MRZJ. Further improvement for MRZJ could come from the authors publishing more mathematical theoretical details about how their algorithm works and where their new centrality comes from. That way, other researchers will be able to add to the refining of said tool.

We have learned that the naive Db algorithm to compute the C-P structure is still a very reasonable algorithm, while the MZRJ algorithm, although fairly good (and often good enough), could be revised and improved, which opens room for future work. We have also learned that, although some characters are very present in different books, they may not be as vertebrating in the social relations with other characters and, in general, in the universe of the book.

Lastly, we think we helped open a fairly unexplored topic, which is the use of computing for the study of literary structures. Some interesting results will come from studying how characters relate to each other in different pieces. Maybe new criteria could be developed which allows us to better understand what distinguishes a good book. Future work should also involve more complicated networks, using weighted or directed networks, maybe counting how many times two entities are connected throughout the book (weight) or perhaps which entity approaches which (direction).

## Conclusion

These results demonstrate that the study of core-periphery structure is critical to developing a strong understanding of networks that arise from the real world. Without proper tools and algorithms for analysis, we are left not only without a strong foundation, but also without opportunity for further study. Universal and objective criteria results very necessary for comparing said tools.

As we saw in our results, analyzing a network without a proper and precise tool can skew our understanding of it, so implementing new tools with a new perspective is necessary, as well as improving the ones already existing. Future research should continue to create tools for this, especially devising algorithms that cover a wide array of networks across various fields and comparative tools that dilucidate what results are better than others in a topic where defining things precisely is usually not easy and there is room for different definitions that refer to the same intuitive idea.

We have used one criterion, namely the $\Phi$ function, to compare two different algorithms. We conclude that the DB algorithm is a priori preferable to the MRZJ algorithm, at least for the networks we have chosen, and hope that this last algorithm can be refined for future application.

## Materials and Methods

For the application of the theory, we implemented the following 6 algorithms into python:
(1) 'MRZJ': This is the basic implementation of the MRZJ B algorithm. The input is a graph and the outpu is the C-P partition.
(2) 'DB': As the implementation of the DB algorithm is a little more complicated, we created three functions for this algorithm: to compute $q$ from $\gamma, p$, to compute $\gamma, p$ from $q$ and to call the other two as many times as necessary. A restriction on the values of $\gamma$ was implemented in order to exclude values that explode towards 0 or 1. Additionally, it outputs the $\Phi$ and $\gamma$ values of the partition.
(3) 'Phis': this function takes as input a graph and a C-P partition and outputs the $\Phi$ value.
(4) 'Improving algorithm by adding': This is the implementation for the Improving algorithm by adding. Takes as input a graph and a C-P partition and outputs a new C'-P' output.
(5) 'Plot': It has one variation for the LOTR graph and another one for the synthetic graphs, but they work very similarly. They plot the input graph. They also take some setting inputs in order to write the title and other details in the graphics.
(6) 'Comparing': There are two auxiliary functions which work very similarly for comparing different C-P partitions. One of them compares four different given $\Phi$ values and plugs them into a plot, while the other one computes how many nodes overlap for two given C-P partitions and how many do not, for later plotting as well.

## Supplementary material

The code used for this project can be found at https://github.com/JaviLobillo/ComplexNetworks.

## AI Writing Tool Declaration.

No AI tool was used in the writing of this paper whatsoever. AI tools where used for the coding and implementation of the algorithms as to give them a more clear and efficient writing.

1. P Ma, X Ren, J Zhu, Y Jiang, Detecting the core of a network by the centralities of the nodes. *Chin. Phys. B* **33**, 088903 (2024).
2. M Newman, *Networks.* (Oxford University Press), (2018).
3. M CUCURINGU, P ROMBACH, SH LEE, MA PORTER, Detection of core–periphery structure in networks using spectral methods and geodesic paths. *Eur. J. Appl. Math.* **27**, 846–887 (2016).
4. X Zhang, T Martin, MEJ Newman, Identification of core-periphery structure in networks. *Phys. Rev. E* **91**, 032803 (2015).
5. J Calvo, Projects/lotr at master · morethanbooks/projects (2017).