A Q-learning based genetic algorithm : a novel hybrid approach to solve community detection problem

Abstract— **Community detection in social networks is an NP-hard search problem, and many metaheuristics have been proposed to solve it. One of them is to apply population-based metaheuristics, such as genetic algorithms. The problem with these methods, which evolve a population through generations, is the impact of the initial population on the quality of the results and the execution time. Indeed, the initial population must be generated in an intelligent way so as to ensure that it contains good quality solutions and that it is diverse. In this paper, we propose a new hybrid method using the k-means algorithm to generate a part of the initial population with the help of reinforcement learning to determine the best K-parameters and the distance used for the k-means algorithm. The newly proposed hybrid approach performs well on synthetic and real instances.**

Keywords— Community detection, Machine learning, Meta-heuristics, Genetic Algorithms, K-means, Reinforcement learning, Q-Learning.

I. INTRODUCTION

A social network is defined as a set of actors with interaction patterns or "links" between them. It can be represented by a graph where the nodes represent actors that can be people or organizations, and the connections between them represent social relationships such as friendships and common interests. Nowadays, social networks are a very interesting source of information. Social network analysis aims to understand the behavior of network participants and their evolution. It is therefore interested in the members of the network and the relationships between them in a specific social context. This leads us to the notion of community, which corresponds to groups of nodes that are largely connected to each other and weakly connected to the rest of the network. One of the major problems in social network analysis is that of community detection, which has received a lot of attention recently because of its importance. For example, in a criminology context, community detection can help investigators find criminal communities, and identify their criminal activity.

The problem of identifying network communities is NP-Hard [1], Therefore, instead of trying to find an optimal solution, many efforts had been devoted to design approached methods to find high quality solutions within a reasonable computation time.

Many heuristics have therefore been proposed to solve the problem. In the first generation of community detection algorithms, the problem is formulated as a graph partitioning problem. Essentially, there are splitting algorithms that detect links between communities and remove them from the network, and agglomeration algorithms that merge similar nodes/communities recursively [2]. More recently, Newman formulated the community detection problem as an optimization problem by introducing a new metric "modularity" to assess the quality of a solution. Several algorithms have been proposed afterwards whose objective is to maximize the modularity such as the Louvain method [4] and Leiden algorithm [5].

Among the metaheuristics used for the community detection problem, the GA-Net algorithm [6]. It is based on genetic algorithms, which have been widely used to solve complex combinatorial optimization problems [7]. Authors in [14] proposed a parallel implementation of the well-known k-means clustering algorithm to detect overlapping communities in complex networks.

However, all these previous methods have important shortcomings due to their dependence on the combination of parameters used in the algorithm or the considered instance, which considerably reduces their efficacy in terms of quality and run-time. Alternatively, a number of algorithms have enhanced the performance of metaheuristics by hybridization with machine learning. An approach combining the " Cuckoo Search " metaheuristic with K-Means has been proposed for solving the sentiment analysis problem [8]. The K-Means algorithm allows to generate good quality initial solutions. Similarly, the Particle Swarm Intelligence algorithm has been enhanced in [9]. An improvement of the Hill Climbing method using Q-learning was proposed in 2014 [10]. Q-learning has also been used to generate initial solutions for the Greedy randomized adaptive search procedure [11]. Another proposal for the vehicle routing problem was to predict the quality of a solution using supervised learning to aid the decision of the metaheuristic (Tabu search) [12]. In a more recent work [13] (2021), a solution combining two metaheuristics: one for exploration (Minimal Population Search) and another for exploitation (Covariance Matrix Adaptation Evolution Strategy) has been proposed. The transition point between the two (in order to decide to explore or exploit) was learned using supervised learning methods (SVM, KNN, decision trees).

We propose in this work a hybrid solution combining a genetic algorithm and Q-Learning for the problem of community detection in social networks. We combine both unsupervised learning and reinforcement learning. Q-Leaning selects the best distance metric and the number of clusters for K-means, to generate high quality solutions. These solutions are then fed to the initial population of the genetic algorithm.

The rest of the paper is organized as follows: The second section will state the problem. The third section recall briefly the background information about K-means and Q-Learning. The fourth section presents the general architecture of the proposed method, followed by The encoding of the solution and the adaptation of K-means and Q-Learning. Experimental Results

are discussed in section five. Finally, the last section concludes the paper and gives some perspectives.

II. Problem statement

We can define a social network as graph G (V, E) where V is the ensemble of nodes and E the ensemble of edges (a,b) with a,b elements of V. For further explanations, we set A as the adjacency matrix, which is a square matrix used to represent a finite graph.

$$A_{ij} = \begin{cases} 1 \text{ if nodes i and j are connected} \\ 0 \text{ otherwise} \end{cases}$$

The modularity is a scalar value between -1 and 1 that measures the density of links within communities compared to the links between communities:

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$

A_{ij}: weight between i, j

k_i: sum of weights attached to i, $K_i = \sum_j A_{i,j}$

c_i: community of vertex i

$\delta(u, v)$: 1 if u=v, 0 otherwise

Formula 1: Mathematic equation of modularity

III. BACKGROUNDER Information

A. *Kmeans*

K-means is an unsupervised learning algorithm. It's an iterative clustering method used to analyze a dataset characterized by a set of features, in order to group "similar" data points into groups (or clusters). The similarity between two data can be inferred from the "distance" between their features; thus, two very similar data are two data whose features are very close. First, we initialize the centroids by taking random data from the dataset. K-means alternates several times between these two steps to optimize the centroids and their groups:

- Group each object around the nearest centroid.
- Replace each centroid according to the average of the descriptors in its group.

After a few iterations, the algorithm converges to a stable clustering of the data set.

B. *Q-Learning*

Reinforcement learning (RL) is a machine learning method whose aim is to allow an agent placed in an interactive environment to choose actions maximizing quantitative rewards. Q-learning is a reinforcement learning algorithm that seeks to find the best action to take given the current state.

To design a Q-Learning algorithm, three concepts need to be defined: a reward (Q-value function), a set of states and a set of actions. After defining these elements, the Q-learning algorithm starts by initializing a matrix $Q(s, a)$ of s rows and a lines where s is the number of possible states and a is the number of possible actions. Then, the algorithm chooses an action to perform according to a ϵ-greedy rule which is a criterion of exploration-exploitation, that consists of choosing the action with the highest value of Q with probability $1 - \epsilon$, and choosing a random action with probability ϵ. Lastly, the q-value is updated with the following expression:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha[r_t + \gamma \max_A Q(s_{t+1}, a)]$$

Formula 2: Bellman equation for Q-Learning

IV. The Proposed approach

Many experiments showed that GA-Net [6] tends to stuck in local optimums and has a slow convergence in comparison with other state of the art algorithms such as Louvain method [4] and Leiden algorithm [5]. This is one of the main drawbacks of GA-Net. This is due to the fact that GA-Net uses an initial population that is generated mostly randomly or with the help of some heuristics. However, this initialization step is a crucial part of any optimization algorithm because it can lead to rapid convergence or divergence if the initial population is not diverse enough, or if it does not contain interesting potential solutions that can guide the search in future generations.

In this paper, we propose a novel approach to address this problem using k-means algorithm that generates good solutions to inject in the initial population using the alpha + beta model in order to improve the initial population quality. The motivation behind this choice is that k-means produces good communities if we know a priori the value of K which is the number of communities and the metric D to use for the calculations of distances. As a consequence, we are facing a new challenge, how do we know which value of K and D to use?

To tackle this challenge, we propose in this work to train an agent whose role is to choose the best combination of K and distance D each time a solution is generated. The task of Q-Learning is therefore to determine these parameters. Our proposal defines actions as the set of parameters of the k-means (k, distance), states as the number of solutions we attempt to find. However, the range of the values of K to use must be known. To deal with this issue, we propose to generate a solution using a heuristic, then we generate an interval of values whose center corresponds to the number of 2 communities produced by this solution. For example, the selected heuristic has produced a solution with 4 communities, so we can create an interval where the center is 4 with a fixed length and a fixed incremental step like [3, 4, 5, 6,7].

In this section, we first describe the general architecture of the proposed approach, then we give details of the hybrid solution combining K-means and Q-learning for the community detection problem.

A. General Architecture

The newly proposed approach consists of 2 principal phases: an offline learning phase and an iterative phase.

- The offline learning phase:

It is the core of the algorithm. It aims to generate the best possible population using Q-learning and k-means. Q-learning agent is trained for n numbers of episodes. The solution for every episode gets generated with k-means, evaluated by the fitness function (modularity in our case) and stored in a list of generated solutions. After the training is completed, we take the best m generated solutions and inject them as the initial population into the GA-NeT algorithm.

- The iterative phase:

It consists in genetically modifying the initial population using genetic operators such as crossover and mutation for a predefined number of iterations "i". It involves the evolution of the initial population through several generations in order to converge to a good solution.

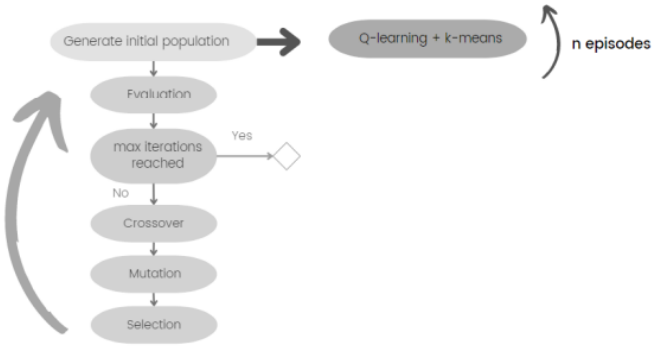


Figure 1: General architecture Flowchart

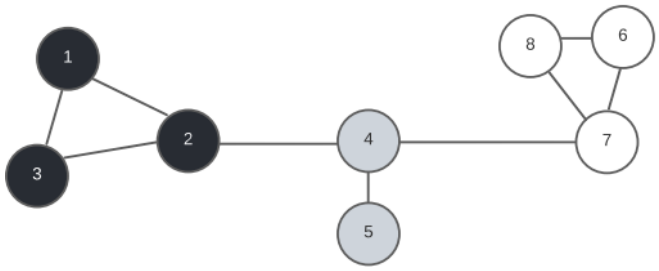
Algorithm: Proposed method
1: Select best parameters for k-means (distance + k) with Q-Learning
2: Generate a % initial solutions with k-means
3: Generate (100-a) % random initial solutions
4: While i < Maxiter
5: Parent selection
6: Crossover
7: Mutation
8: Update population
9: end While

B. Genetic Algorithm for Community detection

Genetic Algorithm is a metaheuristic inspired from the process of natural selection and genetic evolution mechanism, introduced by John Holland from the University of Michigan in 1975 It is an iterative algorithm based on pseudo-random exploration of the search space. It repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population evolves toward an optimal solution.

In order to apply genetic algorithms for the community detection problem, we mainly need to define the encoding of a solution. A population in the context of community detection represents a set of generated individuals. We use the locus-based encoding [15], where an individual is a candidate solution that assigns each node to its respective community. A community is represented as a list of node labels and a solution or an individual is a list of communities as illustrated in the locus-based adjacency representation [15].

The following figure is an example of an encoding solution:



Position	1	2	3	4	5	6	7	8
Genotype	2	3	2	5	4	8	6	6

Figure 2: Solution encoding example

C. K- means for community detection

In the proposed adaptation of k-means to community detection, the data points are the nodes of the graph. Each node is represented by a vector of n elements; n is the number of nodes in the graph. The $i\ th$ element of the vector is equal to one if an edge exists between the actual node and the i th node of the graph. Else, it's equal to zero. To calculate the distance between two different nodes, we use one of these two metrics: The Euclidean distance and Manhattan distance [Formulas 3,4], that are applied to the two vectors representing the nodes.

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Formula 3: Euclidean distance formula

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Formula 4: Manhattan distance formula

K-Means is based on the minimization of the squared error. This method partitions the data into K classes (clusters) represented by their centroids (μ1, μ2..., μk). The method starts by randomly generating, or using a heuristic, k centroids. At each iteration, each object is assigned to the class whose centroid is the closest using the chosen distance. Then, the new cluster centers are recalculated. The search stops when a predefined number of iterations is reached. The pseudo K-means algorithm is given in the following:

Algorithm: K-means
1: Choose the number of clusters K.
2: Generate Randomly the centroids μ1, μ2,..., μk
3: While i < Maxiter do:
4: Assign each node to the class whose centroid μj is closest to it
5: Compute the new centroids of the different clusters
6: End while
7: End

At the end, each cluster found by k means represents a community.

D. Q-Learing for community detection

Our approach aims to find the best initial population using k-means to start the GaNet algorithm. However, k-means needs a pre-defined number of clusters and a distance metric, this is very problematic when the number of communities in the graph is unknown. To solve this issue, we use the Q-learning algorithm to determine the most suitable number of clusters and distance metric to our graph. The algorithm is as follows:

- The set of actions is a set of k-means parameters (k, distance), k is the number of clusters and varies in a pre-defined interval. As for the distance, we use the Euclidean and Manhattan distance.
- The set of states are the solutions. For example, if we want to inject 10 good solutions to our initial genetic algorithm population, these solutions represent the state of our algorithm.
- In this case, we use the modularity measure as a Q-value function, which is updated by the expression above. [Formula 1]

At each step of the algorithm a set of k-means parameters are chosen using the ϵ–greedy rule, these parameters are used to perform k-means and obtain one of our solutions. Lastly, the Q-table is updated and the process is reiterated until the number of episodes is reached. At the end we get the best number of clusters and distance metric that maximizes modularity for each one of the wanted solutions.

V. EXPERIMENTAL RESULTS

In this section, we first start by presenting the benchmarks used to verify the proposed algorithm, then we analyze the performances of our algorithm on these benchmarks, finally we compare the performance with the Louvain [4] and Ga-Net [6] algorithms.

Table 1: A brief description of the real-world benchmarks used in our experimental studies

Network	Nodes	Edges	Description
Karate	34	78	It contains social ties among the members of a university karate club.
Dolphins	62	159	Social network of dolphins, Doubtful Sound, New Zealand.
Football	35	118	It contains the network of American football. games between Division IA colleges during regular season Fall 2000.

Table 2: A brief description of the synthetic benchmarks used in our experimental studies

Mixing parameter	Nodes	Edges
μ = 0.1	128	2048
μ = 0.2	128	2048
μ = 0.3	128	2048
μ = 0.4	128	2048
μ = 0.5	128	2048
μ = 0.6	128	2048
μ = 0.7	128	2048
μ = 0.8	128	2048
μ = 0.9	128	2048

A. *Benchmarks presentation*

In order to measure the performances of our method, we tested it using several networks, both real networks [Table1] and synthetic networks [Table 2]. Real world networks include the Karate, Dolphins and football networks. Synthetic networks have been generated using an approach based on the mixing parameter μ that determines the difficulty to identify good communities.

B. *Performance assessment*

We ran QL-GA 30 times on each benchmark, we evaluated its performances based on the modularity and the execution time. All tests were performed with 1500 generations and a population of 50 individuals and 50 episodes.

Table 3: Performances of QL-GA in synthetic benchmark: mean, max and min of the obtained modularity, average execution time of QL-GA in seconds, and the average training time of Q-Learning in seconds.

μ	mean	Max	min	Execution time	Training time
0.1	0.227	0.26	0.17	46.9	15.18
0.2	0.175	0.23	0.15	47.24	16.33
0.3	0.157	0.18	0.12	46.32	18.41
0.4	0.125	0.15	0.11	45.78	21.31
0.5	0.118	0.15	0.1	46.03	16.36
0.6	0.111	0.13	0.09	45.63	16.38
0.7	0.119	0.13	0.09	45.74	16.11
0.8	0.124	0.14	0.1	45.44	16.24
0.9	0.127	0.14	0.12	45.49	15.96

Table 4: Performances of QL-GA in real benchmarks: mean, best and worst of the obtained modularity, average execution time of the algorithm in seconds, and the average training time of Q-Learning in seconds.

Network	mean	best	worst	Execution time	Training time
Dolphins	0.527	0.528	0.525	30.18	8.05
Football	0.604	0.604	0.602	41.25	15.41
Karate	0.419	0.419	0.419	26.59	4.17

Table 3 shows for each synthetic benchmark the best, worst and the mean of the resulted modularity, and also provides the training time of the Q-learning and the execution time of the rest of the algorithm, and table 4 showcases the same type of results but for the real benchmarks.

As we can see from the tables, QL-GA achieves good solutions, especially on real networks, it finds partitions with a high modularity in a very reasonable amount of time.

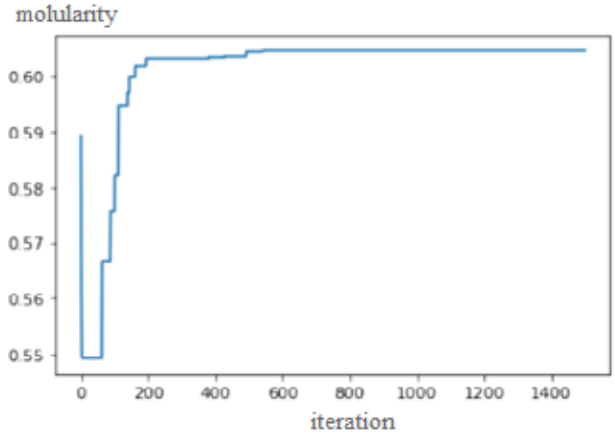


Figure 3: Evolution of the modularity score based on number of iterations, in the case of the football data-set

To highlight the efficiency of our proposed hybrid Learning GA, we study the convergence of this method through the generations. The graph in Figure 3 represents the evolution of the modularity score based on the number of the iterations in the football data-set.

As we can see, the algorithm converged towards the optimal solution in only 200 iterations. And even though the initial solution was so good but it didn't stick in the local optimum, it was able to avoid it and go to find the best solution.

C. *Comparative analysis*

We compare in this section QL-GA with the genetic algorithm Ga-Net [6] and the Louvain algorithm [4] which is a landmark algorithm for modularity maximization. We used the same test scenarios, and ran our algorithm and Ga-Net 30 times for each benchmark. Box-plot of Figure 4 shows some statistical results for the two methods on synthetic instances. In the case of the real data, both our method and Ga-Net give the same results during the 30 executions, thus for better insights we plot a bar-plot [Figure 5] in this case in order to compare them with Louvain.

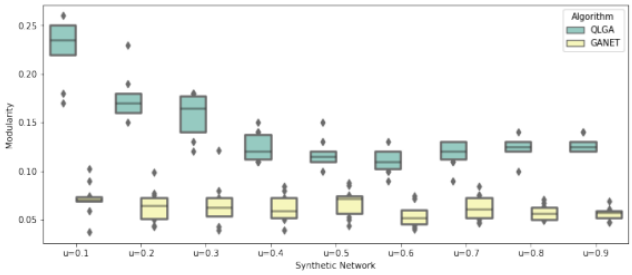


Figure 3: A box-plot showing statistical results [Min, Max, Median and the confidence interval] after running the algorithm 30 times for each benchmark.

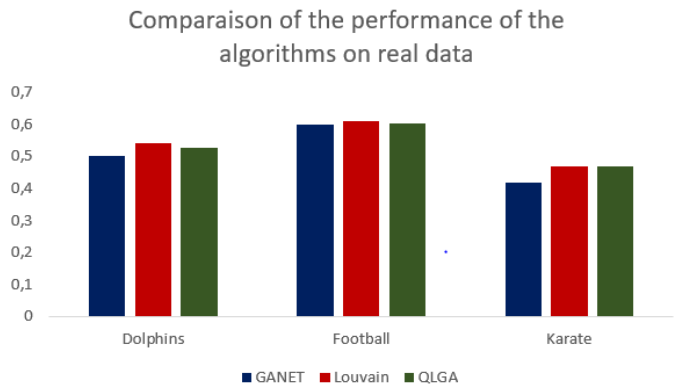


Figure 4: A bar-plot showing the modularity scores on the real data, and comparing the results with Louvain and GA-NET.

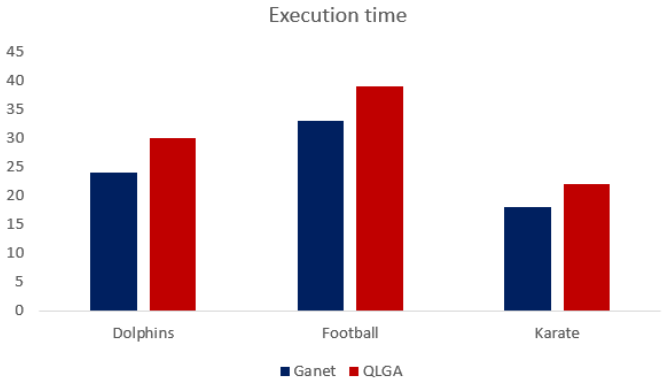


Figure 5: A bar-plot showing the execution time on the real data, and comparing the results with GA-NET's execution time

We notice from figure 5 that the proposed hybrid algorithm QL-GA outperforms Ga-Net in terms of modularity in both real and synthetic instances, and it is equivalent to Louvain on real data. QL-GA achieves good results and especially better than Ga-Net. We also notice that Ga-Net is slightly faster than our algorithm, and that's because both algorithms ran with 1500 iterations and our approach uses Q-Learning, but as shown in previous figure, we were able to converge in only 200 iterations.

CONCLUSION

We present in this study a novel approach to solve the community detection problem in social networks using the hybridization of machine learning: K-means, Reinforcement learning and meta-heuristics: Genetic Algorithm. Our method was able to achieve up to 5 times better results than basic Ga-NET with less computational time and resources, and close results to state of the art algorithms: Louvain, Leiden. Our work is a new technique to the domain of community detection and we hope it can inspire insights towards further more researches: one way to improve QL-GA would be using Machine Learning techniques in order to improve the operations of crossover and mutation of the genetic algorithm part.

REFERENCES

[1] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," Physical review E, vol. 72, no. 2, p.027104, 2005.

[2] Punam Bedi and Chhavi Sharma. Community detection in social networks. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 6(3):115–135, 2016.

[3] A Attea Bara'a, Amenah D Abbood, Ammar A Hasan, Clara Pizzuti, Mayyadah Al-Ani, Suat Ozdemir, and Rawaa Dawoud ¨ Al-Dabbagh. A review of heuristics and metaheuristics for community detection in complex networks: Current usage, emerging development and future directions. Swarm and Evolutionary Computation, 63:100885, 2021

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10):P10008, 2008.

[5] Traag, V.A., Waltman, L. & van Eck, N.J. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 9, 5233 (2019).

[6] Clara Pizzuti. Ga-net: A genetic algorithm for community detection in social networks. In International conference on parallel problem solving from nature, pages 1081–1090. Springer, 2008

[7] Manoj Kumar, Mohammad Husain, Naveen Upreti, and Deepti Gupta. Genetic algorithm: Review and application. Available at SSRN 3529843, 2010.

[8] Avinash Chandra Pandey, Dharmveer Singh Rajpoot, and Mukesh Saraswat. Twitter sentiment analysis using hybrid cuckoo search method. Information Processing & Management, 53(4):764–779, 2017.

[9] Sayed, Awny & Abdallah, Mohamed & Zaki, Alaa & Ali, Ali. (2020). Big Data analysis using a metaheuristic algorithm: Twitter as Case Study. 20-26. 10.1109/ITCE48509.2020.9047790.

[10] 'Guided' Restarts Hill-Climbing David Catteeuw, Madalina Drugan, and Bernard Manderick Artificial Intelligence Lab, Vrije Universiteit

[11] F. C. de Lima, J. D. de Melo and A. D. Doria Neto, "Using the Q-learning algorithm in the constructive phase of the GRASP and reactive GRASP metaheuristics," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 4169-4176.

[12] Calvet, L., Ferrer, A., Gomes, M. I., Juan, A. A., & Masip, D. (2016). Combining statistical learning with metaheuristics for the Multi-Depot Vehicle Routing Problem with market segmentation. Computers & Industrial Engineering, 94, 93–104.

[13] A. Bolufé-Röhler and Y. Yuan, "Machine Learning for Determining the Transition Point in Hybrid Metaheuristics," 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 1115-1122

[14] Andr´as B´ota, Mikl´os Kr´esz, and Bogd´an Zav´alnij. Adaptations of the k-means algorithm to community detection in parallel environments. In 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pages 299–302, 2015.

[15] Zohra, Beldi & Bessedik, Malika. (2019). A New Brainstorming Based Algorithm for the Community Detection Problem. 2958-2965. 10.1109/CEC.2019.8789897.