Q-learning and K-means in GaNet : a novel hybrid approach to solve community detection problem

Mohamed Islem KARABERNOU, Adel BOULAOUAD
Sarah ABCHICHE, Mustapha Ayoub BELOUADAH
Akli YALAOUI ,Mohamed DJILANI

June 2022

Abstract

Community detection is a common research problem, and many approaches were proposed to solve it in the most efficient way. One of them is applying a genetic based approach to discover communities in social networks (GA-Net). The problem with this kind of approaches is that the initial population affects the quality of the results and the execution time. In this paper, we propose a new method using the k-means algorithm to generate a part of the initial population with the help of reinforcement learning to determine the best parameters K and the used distance for the k-means algorithm. Using this approach, we got better results in a shorter amount of time.

Keywords: Community detection, Machine learning, Meta-heuristics, GA-Net, K-means, Reinforcement learning, Q-Learning.

1 Introduction

Community detection discovers groups which have similar behaviors among themselves rather than other communities in the network [2]. Modularity is used to evaluate the quality of partitioning, it has a value between 1 and -1. Exact modularity optimization is a problem that is computationally hard [3],various algorithms have been triggered to overcome this problem [1].
The algorithm named GA-Net identify communities in networks by employing genetic algorithms [8],which have been used for solving optimization problems based on natural selection [6].
In this work, we propose an improvement of GA-Net. 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 the best m solutions, these solutions are then fed to the initial population of the genetic algorithm.

Previous work has explored the adaptation of K-means [4] to community detection and the optimization of metaheuristics with machine learning methods [5, 7].
The paper is organized as follows. The second section describes briefly the background information about K-means,Q-Leaning,and GA-Net.The third 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 and compared to the ordinary GA-Net in section four. Conclusion section concludes the paper.

2 Background Information

2.1 Genetic Algorithm

Genetic Algorithm is a metaheuristic inspired from the process of natural selection and genetic evolution mechanisme, 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. The genetic algorithm is built on the following principles :

- a chromosomic representation (encoding) of a solution
- a fitness function that is used as an evaluation of a solution , solutions are selected based on their fitness
- a reproductive mechanism that consists of genetic operators (crossover , mutation) that define how the characteristics of the parents are passed on to descendants.

Genetic operators :
Crossover : it is the mechanisme of producing a new individual by combining the characteristics of its parents.

Mutation : it is applied after the crossover , it consists of randomly modifying one or more genes of the individual for diversification purposes.

Algorithm 1 GA

```
Initialize random population
while i < MaxIterations do
    Parent selection
    Crossover
    Mutation
    Update population
end while
```

2.2 K-means Algorithm

K-means is an unsupervised clustering algorithm. It is used to analyse a dataset characterised 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. This definition allows us to formulate the data partitioning problem as the search for K "prototype data", around which the other data can be grouped. These prototype data are called centroids; in practice the algorithm associates each data with its closest centroid, in order to create clusters. On the other hand, the averages of the features of the data in a cluster define the position of their centroid in the feature space: this is the origin of the name of this algorithm. After initializing its centroids by taking random data from the dataset, K-means alternates several times between these two steps to optimize the centroids and their groups:

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

After a few iterations, the algorithm finds a stable division of the dataset: we say that the algorithm has converged.

2.3 Q-Learning Algorithm

Reinforcement learning is a machine learning method whose aim is to allow an agent (virtual entity: robot, program, etc.), placed in an interactive environment (its actions modify the state of the environment), to choose actions maximising quantitative rewards. The

agent tries out and improves its action strategy according to the rewards provided by the environment. There are many reinforcement learning algorithms categorised into several sub-families. The Q-learning algorithm is relatively simple and allows us to understand learning mechanisms common to many other models. To do this, there are three central elements to define : a reward (defining a Q-value function), a set of states and a set of actions.

After defining our 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. Note that an element of the matrix represents the reward gained after choosing an action while being in a specific state. 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 :

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

where s_t is the current state, a_t is the action performed in the state s_t, r_t is the received reward for executing a_t in s_t, and s_{t+1} is the new state; γ is a discount factor ($0 \leq \gamma < 1$), while α ($0 < \alpha < 1$) is the learning rate.

The two last steps define an episode of the q-learning algorithm. A certain number of episodes is executed until we reach convergence. The pseudo code of the q-learning algorithm is presented in the following :

3 Proposed Method

3.1 General Architecture

Our approach consists of 2 principal phases : an **offline learning** phase and an **iterative** phase

The offline learning phase is the core of our approach , it aims to generate the best possible population using Q-learning and k-means,

Algorithm 2 Q-Learning

```
Initialize Q(s,a)
for each episode do
    Initialize s
    for each step of episode do
        Choose a according to the ϵ-greedy rule
        Perform action a, observe r, s'
        Update Q(s,a)
        s ← s'
    end for
end for
```

the Q-learning agent is trained for an n number 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 consists of genetically modifying the initial population using genetic operators such as the crossover and the mutation for an i number of iterations.

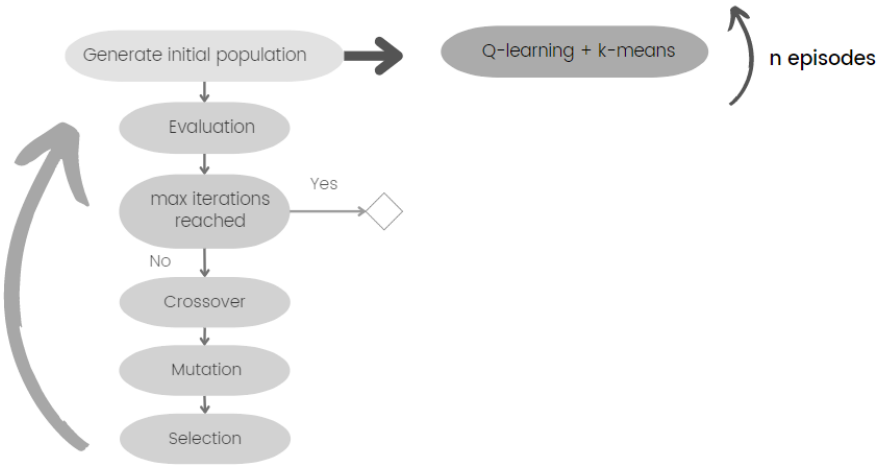

Figure 1: General architecture Flowchart

3.2 Solution encoding

A population in the context of community detection represents a set of generated individuals , 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.

3.3 Adaptation of K-means

In our 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, that are applied to the two vectors representing the nodes. The algorithm itself is similar to the one described in the previous section. At the end, each cluster found by k means represents a community.

3.4 Adaptation of Q-learning

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 use, we use the q-learning algorithm to determine the most suitable number of clusters and distance metric to our graph. Our adaptation of the algorithm is as follows :

- Our 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.
- Our set of states are the solutions we are trying to find. For example, if we want to inject 10 good solutions to our initial genetic algorithm population, these solutions represent the state of our algorithm.

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

- In this case, we use the modularity measure as our q-value function, which is updated by the expression above.

The algorithm is similar to the one presented in the previous section. 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.

4 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 obtained results with the Louvain and Ga-Net algorithms.

4.1 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 has been generated using an approach based on the mixing parameter u that determines the difficulty to identify good communities.

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

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

4.2 Obtained results

We ran our algorithm 10 times on each benchmark, we evaluated its performances based on the modularity and the execution time. The table below [Table 3] presents for each synthetic benchmark the best, worst and the mean of the resulted modularities, 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 . All tests were performed with 1500 generations and a population of 50 individuals and 50 episodes.

mu	mean	max	min	Execution Time	Training Time
0.1	0.227	0.26	0.17	46.9	15.184
0.2	0.175	0.23	0.15	47.242	16.338
0.3	0.157	0.18	0.12	46.322	18.41
0.4	0.125	0.15	0.11	45.787	21.314
0.5	0.118	0.15	0.1	46.03	16.346
0.6	0.111	0.13	0.09	45.632	16.381
0.7	0.119	0.13	0.09	45.742	16.119
0.8	0.124	0.14	0.1	45.446	16.249
0.9	0.127	0.14	0.12	45.492	15.965

Table 3: Performances of our algorithm in synthetic benchmarks, this table is show the mean, max and the min of the obtained modularities, also the average execution time of the algorithm in seconds, and the average training time of Q-Learning in seconds.

As we can see from the tables, our method did a very good job, especially on real networks, it finds partitions with a high modularity in a very reasonable amount of time.

Network	mean	max	min	Execution Time	Training Time
dolphins	0.527	0.528	0.525	30.181	8.056
football	0.604	0.604	0.602	41.254	15.416
karate	0.419	0.419	0.419	26.599	4.172

Table 4: Performances of our algorithm in real benchmarks, this table is show the mean, max and the min of the obtained modularities, also the average execution time of the algorithm in seconds, and the average training time of Q-Learning in seconds.

Now we are going to study the convergence of this method. The below graph [Figure 2] represents the evolution of the modularity score based on the number of the iterations in the football data-set.

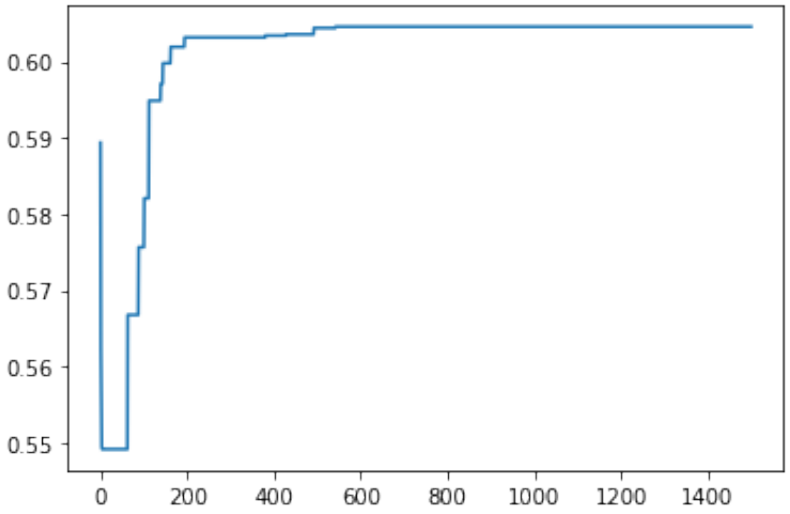

Figure 2: Evolution of the modularity score based on number of iteration, in the case of the football data-set

In fact 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 stuck in the local optimum, it were able to avoid it and go to find the best solution.

4.3 Comparative analysis

Here we compare our algorithm with the ordinary Ga-Net and the Louvain algorithm. We used the same test scenarios, we ran our

algorithm and Ga-Net 10 times each for each benchmark, and the box-plot below [Figure 3] shows some statistical results for the two methods in synthetic data, In the case of the real data, both our method and Ga-Net give the same results during the 10 executions,thus for better insights we plot a bar-plot [Figure 4] in this case in order to compare them with Louvain.

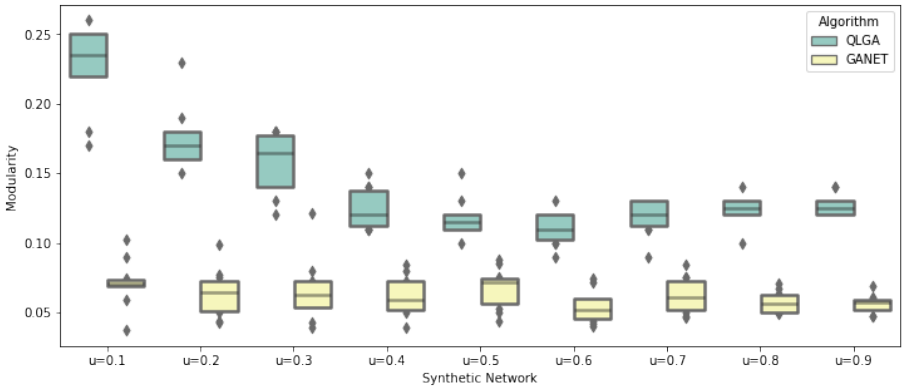

Figure 3: A box-plot showing statistical results [Min, Max, Median and the confidence interval] after running the algorithm 10 times for each benchmark.

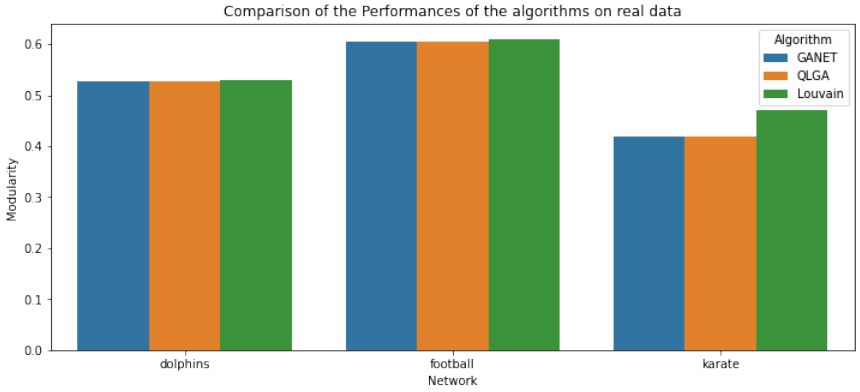

Figure 4: A bar-plot showing the modularity scores on the real data, and comparing the results with Louvain and GA-NET.

Our method outperforms Ga-Net in terms of modularity in both real and synthetic data, and it is equivalent to Louvain on real data. It doesn't do a great job on synthetic data as Louvain but it gives acceptable 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.

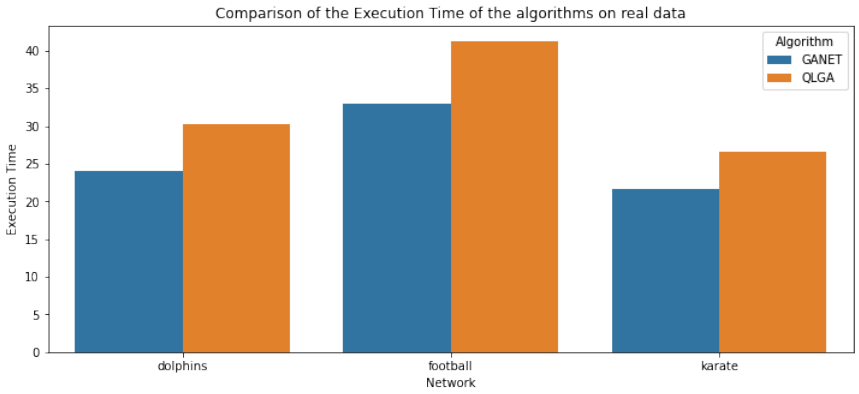

Figure 5: A bar-plot showing the execution time on the real data, and comparing the results with GA-NET's execution time.

5 Conclusion

This paper presents 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, with a good balance of exploration and exploitation, 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.

References

[1] A Attea Bara'a, Amenah D Abbood, Ammar A Hasan, Clara Pizzuti, Mayyadah Al-Ani, Suat Özdemir, and Rawaa Dawoud Al-Dabbagh. A review of heuristics and metaheuristics for community detection in complex networks: Current usage, emerg-

ing development and future directions. *Swarm and Evolutionary Computation*, 63:100885, 2021.

[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] 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.

[4] András Bóta, Miklós Krész, and Bogdán Zaválnij. 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.

[5] Francisco Chagas De Lima, Jorge Dantas De Melo, and Adriao Duarte Doria Neto. Using the q-learning algorithm in the constructive phase of the grasp and reactive grasp metaheuristics. In *2008 ieee international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 4169–4176. IEEE, 2008.

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

[7] 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.

[8] 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.