Monte Carlo Tree Search for Community Detection MCTS-CD

Imane Hamzaoui, Thanina Hamitouche, Nour El Houda Benazzoug, Fatmazohra Rezkellah, Alaa Dania Adimi, Maissa Irmouli, Malika Bessedik, Fatima Benbouzid-Sitayeb

Monte Carlo Tree Search for Community Detection MCTS-CD

Imane Hamzaoui*, Thanina Hamitouche, Nourelhouda Benazzoug,
Fatma Zohra Rezkellah, Alaa Dania Adimi, Irmouli Maissa
Ecole Nationale Supérieure d'Informatique
Algiers, Algeria
Emails: {ji_hamzaoui, jt_hamitouhce, jn_benazzoug, jf_rezkellah, ja_adimi, jm_irmouli}@esi.dz

***Abstract*—Real-world graph analysis and the study of complex networks encounter the common challenge of community detection(CD), encompassing diverse domains such as transportation networks, cyber-security, animal, and social media networks. This problem can be modeled as an NP-hard optimization problem. Inspired from [24] [23] that combined Monte Carlo Tree Search (MCTS) and Reinforcement Learning on large action-space environments, we adapt MCTS for the problem of Community Detection in Social Networks and propose MCTS-CD. MCTS-CD is an arm-bandit [2] anytime algorithm. We show that when compared to reference algorithms for CD such as Louvain [1], with allocating the same amount of resources, MCTS-CD acheives better results on many benchmark datasets such as Zachary Karate Club, Dolphin and the Pol Books datasets.**

***Index Terms*—community detection, monte carlo, reinforcement learning, arm-bandit**

I. Introduction

From social media networks to biological networks, community detection emerged as the identification of internal communities inside these structures. It is an NP-hard optimization problem that requires a lot of computational resources in order to find the optimal community partitioning. Community detection can be modeled as an optimization problem with an objective function that measures the quality of the solution. Modularity [19] and conductance [11] are among the most used metrics. These measures aim to maximize the connectivity inter-community and minimize the connectivity intra-community, that is to say, it uses the graph structure to find the best community partitioning and doesn't use features such as the graph's weights and labels: it only takes the community structure in consideration. Few works have used tree search structures for the problem of community detection, such as [17] on dynamic social networks. Recent advances in the applications of reinforcement learning methods such as Monte Carlo Tree Search showed promising results in games like [24], [23], [10] and in combinatorial optimization problems. Monte Carlo Tree Search is an iterative algorithm that searches the state space and builds statistical evidence about the decisions available in particular states and it is applicable to MDPs [5]. The search is guided by Monte Carlo simulations, the selection of a node in the tree is viewed as arm-bandit problem [22], it allows the expansion of the tree while balancing between the exploration and exploitation of tree nodes. Monte Carlo simulations were used previously for the problem of community detection on weighted brain graphs [7], which used Monte Carlo simulations to simulate the data. In MCTS-CD we use these simulations to search for the solution.

*Corresponding author

II. Definitions

A. Monte Carlo Tree Search

Monte Carlo Tree Search is one of the most popular tree search methods in recent years, the search is guided by Monte Carlo simulations. The selection of a node in the tree is viewed as an arm-bandit problem [25], which means that we allow resources, time and space, and keep the best solution reached in that scope. The selection allows the expansion of the tree while balancing between the exploration and exploitation of tree nodes. It keeps in the memory the explored tree T, and for each arc e issued from a node, it keeps a triplet $(N(v,e), Q(v,e))$.

- $N(v,e)$:is the number of times an arc e has been visited. We also store the number of visits of the child.
- $Q(v,e)$: is the estimation of the arc value, it is the mean of all the results of all the simulations launched from the sub-tree rooted at v.

B. Community Detection

Community detection is the process of identifying groups of nodes in a network that are more densely connected to each other than to the rest of the network. It is a key aspect of understanding the structure and functionality of complex networks, and it can be used to extract useful information from them [12]. A large number of techniques have been suggested to find optimal communities in a reasonable time. Most of these techniques are based on the optimization of objective functions. Modularity optimization so far is one of the most widely used techniques among them. We will define modularity in another section (V-A).

III. Related works

A. MCTS

MCTS is characterized by 4 phases:

- Selection: starting from the root node, the algorithm explores the search space according to a policy until

visiting all the nodes represented in the memory or reaching to a leaf node which represents a final state in games.
- Expansion: adding a child node that represents a state reached after performing an action, if a terminal state is reached then we can't add a child node.
- Simulation: starting from a node, a random simulation of the game/problem is launched to estimate the gain of the final state or the result of the game. A score is attributed to a node based on how promising it is.
- Backpropagation: The score from the last visited node in the tree is propagated to the parents until reaching the root node.

MCTS is mainly used in games [10] [23] [24] but recent works have used it in combinatory optimization problems such as [22]

1) Tree Policy: The epsilon greedy algorithm follows a greedy arm selection policy, selecting the best-performing arm at each time step. However, ϵ percent of the time, it will go off-policy and choose an arm at random. The value of ϵ determines the fraction of the time when the algorithm explores available arms, and exploits the ones that have performed the best historically the rest of the time.
Unlike epsilon greedy algorithms, Upper Confidence Bound algorithms construct a confidence interval of what each arm's true performance might be, factoring in the uncertainty caused by variance in the data and the fact that we're only able to observe a limited sample of pulls for any given arm. The algorithms then optimistically assume that each arm will perform as well as its upper confidence bound (UCB), selecting the arm with the highest UCB.
The UCB formulas can be expressed as follows :

2) UCB1:

$$\text{UCB1}(i) = \frac{Q(i)}{N(i)} + C\sqrt{\frac{\ln N(p)}{N(i)}} \tag{1}$$

in other words:

$$\text{UCB1} = \text{average\_reward} + \sqrt{\frac{2\ln(\text{total\_selections})}{\text{arm\_selections}}}$$

3) UCB2:

$$\text{UCB2}(i) = \frac{Q(i)}{N(i)} + C\sqrt{\frac{\ln N(p)}{N(i)+1}} \tag{2}$$

or in other words

$$\text{UCB2} = \text{average\_reward} + \sqrt{\frac{2\ln(\text{total\_selections})}{\text{arm\_selections}}} + \sqrt{\frac{\ln(\text{total\_selections})}{\text{arm\_selections}} \cdot \min\left\{1, \frac{\text{variance}}{\text{arm\_selections}}\right\}}$$

Where :

- i represents the child node or action being considered.
- $Q(i)$ represents the total reward accumulated by selecting action i.
- $N(i)$ represents the number of times action i has been selected.
- $N(p)$ represents the number of times the parent node has been visited.
- C is a constant that balances the exploration and exploitation trade-off.

B. Community detection

1) Modularity maximization: Among the classical algorithms that rely on modularity maximization we find the work of Neumann [20], the famous Louvain [1], [6].
Other modularity based methods are label propagation [21] and overlapping community detection [9].

2) Meta-heuristics: Methods using genetic algorithms have been proposed such as [16], [14].
We also find algorithms based on random walks such as [26] and the results were competitive.

3) Deep Learning : In recent years, deep learning techniques emerged, [28] used Node embedding techniques such as Walktrap and Node2Vec. The downside of this kind of method is the scalability and the enormous time to generate the embeddings of each graph.
[18] proposed Ucode that uses graph convolutional neural networks, this method showed interesting results on both overlapping and non overlapping community structures.

IV. Our method

We define a Monte Carlo Tree Node as follows:

- $community\_list$ is the partitioning of the communities on the nodes of the graph, where $community\_list[i]$ represents the community of node i.
- $gain$ is the parent's reward minus the current node's reward.
- $parent$ is the parent node of the current node.
- $children$ represents the children of the current node.
- $visits$ represents the number of visits of the node
- $wins$

Note that a node in the tree doesn't represent a node in the community. Each node carries a community list as described above.
We start by considering each node as a community and launch the simulation for a number of iterations, interrupting it when the stopping criterion is met. The criteria are either to reach a solution that improves the simulated node's modularity with at least τ and if τ isn't reached, the simulation continues for a defined number of iterations that we'll call cpt.
During this phase of simulation, at each iteration, we will assign a node to one of its neighbors randomly with a probability of $p = 1/2$ and to the best neighbor with a probability $1 - p = 1/2$ which is more greedy. This step is different from the Louvain algorithm [1] that can disconnect a central node in a community and cause the problem of disconnected communities [27]. This problem is often escaped due to the selection policy and the other parameters that lead to more

diversification in the search space
The τ parameter decays with the following equation,

$$\tau = \tau * \exp -it/\lambda \tag{3}$$

because as long as we move through the search tree we will get better solutions, thus reaching an improvement of τ will be harder. The learning rate for this equation is $1/\lambda$, further details on these parameters will be given in the experiments and results section [VI].

Algorithm: MCTS-CD

1. Affect each node to its own community
2. Create a root node where each community is assigned to a node.

for $nbIter$ **do**

a. Select the node to simulate according to the policy
b. Simulate the node and estimate its gain:
c. Pick 0.2 of random nodes and assign the best community to it.
Update the learning rate τ.
d. Append children nodes with the resulting communities.
e. Backpropagate and update the reward.

end for

return $bestcommunity$

The MCTS-CD phases for the Selection and Backpropagation are the same as the classical MCTS, our modification lays in the alternation between the simulation phase and the expansion phase. The simulation phase's results are appended as children nodes to the simulated node. And this is what makes up the expansion.

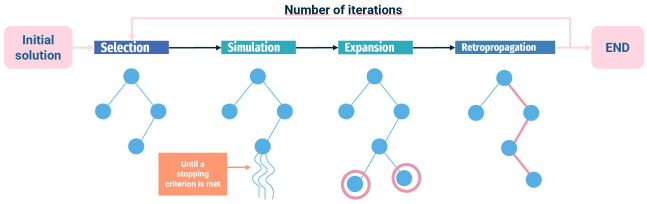

Fig. 1. Steps of the MCTS-CD.

V. Metrics

For the problem of community detection, there are a lot of metrics to decide on the quality of a solution, among them modularity, Normalized mutual info score(NMI), conductance [11], similarity [29]... For this work, we use modularity as an objective function to maximize and we will evaluate the quality of our solutions compared to the ground truth community assignment using the NMI metric.

A. Modularity

Modularity measures the strength of the division of a graph into communities [19]. We aim to maximize the connectivity between the nodes of the same community and minimize the connections between separate communities. The modularity takes values between $-1/2$ and 1. Many argue that modularity isn't the best metric for deciding on the quality structures but it remains a good metric, especially on

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

where:

- Q is the modularity value,
- m is the total number of edges in the network,
- A_{ij} is the element in the adjacency matrix corresponding to nodes i and j,
- k_i and k_j are the degrees of nodes i and j, respectively,
- c_i and c_j are the communities to which nodes i and j belong, respectively, and
- $\delta(c_i, c_j)$is the Kronecker delta function, which is equal to 1 if c_i is equal to c_j, and 0 otherwise.

B. Normalized Mutual Information NMI

the Normalized Mutual Information(NMI) is a measure that assesses the similarity between two partitions based on the information mutually contained on them [4]. The values of the NMI can vary from 0 (no mutual information) to 1 (perfect correlation).

$$\text{NMI}(C, K) = \frac{-2 \sum_{c \in C} \sum_{k \in K} \frac{n_{ck}}{N} \log \left(\frac{n_{ck} N}{n_c n_k} \right)}{\sum_{c \in C} \frac{n_c}{N} \log \left(\frac{n_c}{N} \right) + \sum_{k \in K} \frac{n_k}{N} \log \left(\frac{n_k}{N} \right)}$$

- C represents the true community assignment,
- K represents the predicted community assignment,
- n_{ck} is the number of nodes that are assigned to both community c and community k,
- n_c is the number of nodes assigned to community c,
- n_k is the number of nodes assigned to community k, and
- N is the total number of nodes in the network.

VI. Experiments and Results

A. Benchmark datasets

We test our method on benchmark real-world and synthetic datasets : we test on the Girvan–Newman (GN) [8] and Lancichinetti–Fortunato–Radicchi (LFR) [13] which are synthetic graphs generated according to some parameters such as μ(Fraction of inter-community edges incident to each node), node degree. The first real-world benchmark we test on is the Zachary Karate Club's network [8]. The second one is the Dolphin [15] and the Books dataset. The results of MCTS-CD are computed and then compared to the results produced by Louvain [1], and HGT [3]. The comparaison is made through the NMI in TABLE I and through modularity in TABLE II.

B. Calibrating hyperparameters

1) τ: According to equation (3), the τ parameter will be decreased with the iterations. We attempted to leave τ as the difference between 0.9 or 1 (which is the maximum modularity that could be achieved) and the current solution's τ, it wasn't performing well. This is due to the fact that such an improvement can almost never be achieved with stochastic

permutations, therefore the simulation won't improve the solution quickly. It is always repeated a cpt amount of times which harms the execution time and the quality of the solution returned. Note that we keep the best solution encountered during those iterations but our experiments showed that doing so leads the algorithm to be stuck at a local minima. With that said, we stick to the formula given by equation (3).

2) λ: The λ parameter in equation (3) plays some sort of learning rate and depends on how fast we want the τ parameter to decay. The range in which it is taken will depend on the resources allocated (number of iterations). We picked an interval suitable for the range of iterations that have been experimented.

3) p: p stands for the probability of picking a greedy community affectation and a random affectation of the node to its neighbors. It has been discussed earlier in section [IV].

4) $nbIter$: $nbIter$ stated in the algorithm (IV) refers to the number of global iterations, it represents the number of arms and the resources allocated in this case. This parameter heavily depends on the size of the graph and its complexity. Since we're using small benchmark graphs we wanted to see what can be achieved during that fixed number of iterations.

5) C: C represents the confidence parameter and it controls the level of exploration, it is present in the UCB formula in equations (1) and (2).

TABLE I
NMI COMPARAISON

Benchmark	Louvain	HGT	*MCTS UCB2*	*MCTS UCB1*
GN* $\mu = 0.2$	**0.9748**	**0.9748**	**0.9748**	**0.9748**
GN* $\mu = 0.4$	0.41	0.56	**0.6**	0.64
Zachary Karate	0.56	0.69	**1**	**1**
Football	0.69	**0.917**	0.81	0.87
Dolphin	0.57	0.497	0.58	**0.63**
Books	0.53	0.47	**0.64**	0.552

* GN stands for Girvann Neumann benchmark graph

TABLE II
MODULARITY COMPARAISON

Benchmark	Louvain	HGT	*MCTS UCB2*	*MCTS UCB1*
GN $\mu = 0.2$	**0.65**	**0.65**	**0.65**	**0.65**
Zachary Karate	0.41	**0.42**	0.36	0.4
Football	0.55	0.58	0.55	**0.59**
Dolphin	**0.51**	**0.51**	0.46	0.44
Books	0.5	0.5	0.46	**0.52**

C. Scalability

Since the described method is a tree search, and in our tree, each node contains the community structure of a solution, we had to find a way to decrease the size of a node. We start with an initial clustering of the nodes and launch the algorithm on the formed communities to make convergence faster, we tested using Kmeans and DBscan for the initial clustering and the latter showed better results. That's due to the poor clustering quality returned by Kmeans, while DBSCAN creates real communities with outliers. The number of outliers can vary from one graph to another but if there are a lot of outliers. MCTS-CD can be used for the affectation of the outliers but without changing the community clustering offered by DBSCAN, which is judged to be close from the ground truth in most cases.

However, MCTS-CD is still able to perform well on big graphs, the following figure compares our algorithm to Louvain without an initial clustering, note that the method took some time to converge but we expect to get better results and less time with an initial solution that went through a clustering phase.

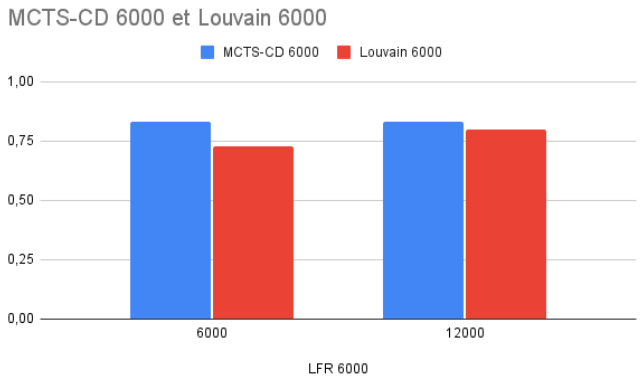

Fig. 2. MCTS-CD and Louvain [1] NMI on LFR benchmark [13] with 6000 and 12000 nodes

CONCLUSION

This paper presented a novel method to adapt Monte Carlo Tree Search for the problem of community detection. Through this hybridisation, we made use of the power of reinforcement learning in order to make the search more efficient, locating through the simulations of Monte Carlo Tree Search interesting regions of the tree, and learning through the backpropagation how to guide the search to these regions. The diversification that the simulations allow, and the intensification the expansion and the learning provided made MCTS-CD produce good NMIs for the benchmarks tested on, and this, in very reasonable computational time. It is suggested that to refine MCTS-CD, another layer of Machine Learning should be introduced : a clustering layer that would modify the original structure of the network in order for the MCTS-CD to be able to act on what would be considered as big networks, always in reasonable times.

REFERENCES

[1] 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, oct 2008.

[2] Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and contextual bandits. *CoRR*, abs/1904.10040, 2019.

[3] Salmi Cheikh, Bouchema Sara, and Zaoui Sara. A hybrid heuristic community detection approach. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–7, 2020.

[4] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, sep 2005.

[5] G. Eason, B. Noble, and I. N. Sneddon. On certain integrals of lipschitz-hankel type involving products of bessel functions. *Phil. Trans. Roy. Soc. London*, A247:529–551, April 1955.

[6] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, feb 2010.

[7] Kathleen M. Gates, Teague Henry, Doug Steinley, and Damien A. Fair. A monte carlo evaluation of weighted community detection algorithms. *Frontiers in Neuroinformatics*, 10, 2016.

[8] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[9] Prem K. Gopalan and David M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013.

[10] Ercüment Ilhan and A. Sima Etaner-Uyar. Monte carlo tree search with temporal-difference learning for general video game playing. *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 317–324, 2017.

[11] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, may 2004.

[12] Bisma S. Khan and Muaz A. Niazi. Network community detection: A review and visual survey. *CoRR*, abs/1708.00977, 2017.

[13] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110, Oct 2008.

[14] Zhangtao Li and Jing Liu. A multi-agent genetic algorithm for community detection in complex networks. *Physica A: Statistical Mechanics and its Applications*, 449:336–347, 2016.

[15] David Lusseau. The emergent properties of a dolphin social network, 2003.

[16] Mehdi Rezapoor Mirsaleh and Mohammad Reza Meybodi. A michigan memetic algorithm for solving the community detection problem in complex network. *Neurocomputing*, 214:535–545, 2016.

[17] Sneha Mishra, Shashank Sheshar Singh, Shivansh Mishra, and Bhaskar Biswas. Tcd2: Tree-based community detection in dynamic social networks. *Expert Systems with Applications*, 169:114493, 2021.

[18] Atefeh Moradan, Andrew Draganov, Davide Mottin, and Ira Assent. Ucode: Unified community detection with graph convolutional networks, 2021.

[19] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004.

[20] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), jun 2004.

[21] Usha Nandini Raghavan, Ré ka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), sep 2007.

[22] SA Rashkovskiy. Monte carlo solution of combinatorial optimization problems. In *Doklady Mathematics*, volume 94, pages 720–724. Springer, 2016.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[24] David Silver, Julian Schrittwieser, Karen Simonyan, and et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[25] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

[26] Christian Toth, Denis Helic, and Bernhard C. Geiger. Synwalk: community detection via random walk modelling. *Data Mining and Knowledge Discovery*, 36(2):739–780, jan 2022.

[27] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233, 2019.

[28] Yichi Zhang and Minh Tang. Exact recovery of community structures using deepwalk and node2vec, 2022.

[29] Łukasz Brzozowski, Grzegorz Siudem, and Marek Gagolewski. Community detection in complex networks via node similarity, graph representation learning, and hierarchical clustering, 2023.