# Monte Carlo Tree Search for Community Detection MCTS-CD

Imane Hamzaoui, Thanina Hamitouche, Nour El Houda Benazzoug, Maissa Irmouli, Fatmazohra Rezkellah, Alaa Dania Adimi

# Monte Carlo Tree Search for Community Detection MCTS-CD

Imane Hamzaoui.
ESI Algiers *
ji_hamzaoui@esi.dz

Thanina Hamitouche
ESI Algiers
jt_hamitouche@esi.dz

Nour El Houda Benazzoug
ESI Algiers
jn_benazzoug@esi.dz

Fatma Zohra Rezkellah
ESI Algiers
jf_rezkellah@esi.dz

Alaa Dania Adimi
ESI Algiers
ja_adimi@esi.dz

Maissa Irmouli
ESI Algiers
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 [25] [24] 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.

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

## 1 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[20] and conductance[12] 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 [18] 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 [25],[24], [10] and in combinatorial optimization problems[11]. 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 [23], 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.

*Ecole Nationale Supérieure d'Informatique, Algiers, Algeria

5th Deep Learning Indaba Conference (DLI 2023).

## 2 Definitions

### 2.1 Monte Carlo Tree Search

The search in a Monte Carlo Tree is guided by Monte Carlo simulations and a policy. The selection of a node in the tree is viewed as a arm-bandit problem[26]. The selection allows the expansion of the tree while balancing between the exploration and exploitation of tree nodes. 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.

The exploration in MCTS is done via a tree policy like Upper Confidence Bound (UCB), and unlike epsilon greedy algorithms, UCB 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 UCB, selecting the arm with the highest UCB.
The UCB formulas can be expressed as follows :

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

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

- $i$ 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.

### 2.2 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 [13]. 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. Modularity is defined in section (5.1).

## 3 Related works

MCTS is used in games [10] [24] [25] and other works have used it in combinatory optimization problems [23]. Among the classical algorithms that rely on modularity maximization we find the work of Neumann[21], the famous Louvain[1], [6]. Other known methods are label propagation [22] and overlapping community detection [9].
Different methods using genetic algorithms have been proposed such as [17], [15]. We also find algorithms based on random walks such as [27] and the results were competitive.
Recent trends went towards deep learning based techniques. [29] used Node embedding such as Walktrap and Node2Vec. [19] proposed Ucode that uses graph convolutional neural networks, this method showed interesting results on both overlapping and non overlapping community structures.

## 4 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

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 stopping criteria is either to reach a solution that improves the simulated node's modularity with at least $\tau$ and if $\tau$ 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 [28]. This problem is often escaped due to the selection policy and the other parameters that lead to more diversification in the search space
The $\tau$ 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 $\tau$ will be harder. $1/\lambda$ is the learning rate for $tau$ .

**Algorithm 1** Community Partition Algorithm

**Require:** Graph
**Ensure:** Best community parition of the graph
*Initialisation*: Affect each node to its own community
Create MCTS root node where each graph node is assigned to a community.
**for** $nbIter$ **do**
    Select the node to simulate according to the policy
    Simulate the node and estimate its gain:
        Assign 1/2 of random nodes to the best neighbour's community to them and assign 1/2 of the remaining nodes to a random neighbour's community.
        Update the learning rate $\tau$.
    Append children nodes with the resulting communities.
    Backpropagate and update the reward.
**end for**
**return** best community partition

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.

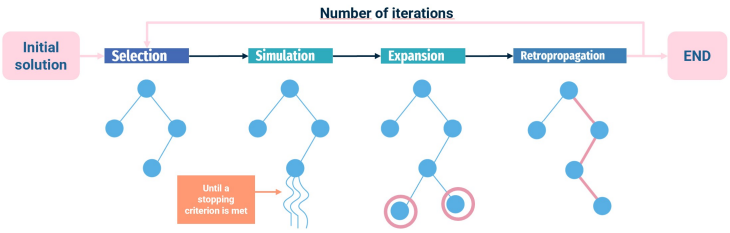


Figure 1: Steps of the MCTS-CD.

# 5 Metrics

## 5.1 Modularity

Modularity measures the strength of the division of a graph into communities [20]. 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.

## 5.2 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).

# 6 Experiments and results

We test our method on benchmark real-world and synthetic datasets : we test on the Girvan–Newman (GN) [8] and Lancichinetti–Fortunato–Radicchi (LFR)[14] graphs which are synthetic graphs generated according to some parameters such as $\mu$ (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[16] and the Pol 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. We did a calibration of the following parameters through a random search and used the best combination: $\tau$ and $\lambda$ in equation (3), $p$ in section [4], $nbIter$ in algorithm (1), $C$ in equations (1) and (2).

Table 1: 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 2: 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** |

We also tested it on a LFR benchmark graph with a $\mu = 0.2$ of 6000 and 12000 nodes, and acheived an NMI of 0.83 and 0.76 respectively with MCTS-CD compared to 0.7 and 0.6 for Louvain [1] respectively.

# Conclusion

This paper presented a 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. pages 1–7, 08 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] Jorik Jooken, Pieter Leyman, Patrick De Causmaecker, and Tony Wauters. Exploring search space trees using an adapted version of monte carlo tree search for a combinatorial optimization problem. *CoRR*, abs/2010.11523, 2020.

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

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

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

[15] 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.

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

[17] 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.

[18] 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.

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

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

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

[22] 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.

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

[24] 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.

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

[26] 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.

[27] 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.

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

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