Monte Carlo Maximum Independent Set

End of study project
LAMSADE Paris-Dauphine

Student: **Oumaima Merhbene**
Supervisors: **Mr Tristan Cazenave and Mr Florian Sikora**

Master2 IASD
Paris-Dauphine university

We have chosen to present the report as an article.

November 2020

Monte Carlo Maximum Independent Set

Abstract

There have been increasing challenges to solve combinatorial optimization graph problems with Monte Carlo search algorithms. GNRPA [Cazenave, 2020], the generalization of Nested Rollout Policy Adaptation (NRPA), shows an improvement compared to other Monte Carlo search algorithms for different domains. In this paper, we propose a combination of GNRPA and deep learning techniques to solve a basic graph optimization problem: the Maximum Independent Set (MIS) problem. We present an efficient combination of Monte Carlo Search and Deep Learning for MIS and compare the results to existing ones.

1 Introduction

Recently, researchers have made significant efforts for resolving combinatorial optimization problems that appear in various applications, e.g., sociology [Harary and Ross, 1957]), operations research [Feo *et al.*, 1994] and bioinformatics [Gardiner *et al.*, 2000]. While Reinforcement Learning algorithms has proven to be enormously successful resolving this issue [Bello *et al.*, 2016; Khalil *et al.*, 2017; Deudon *et al.*, 2018], an expanding area of interest concern solving NP-hard graph optimization problems with Monte Carlo search techniques yields to promising results.

The Nested Rollout Policy Adaptation algorithm (NRPA) is an algorithm that learns a playout policy [Rosin, 2011]. Generalized Nested Rollout Policy Adaptation GNRPA [Cazenave, 2020] is a Monte Carlo search algorithm and a generalisation of the NRPA algorithm with a temperature and a bias to analyze theoretically the algorithm. Experiments shows its improvement on NRPA for different application domains such as SameGame and the Traveling Salesman Problem with Time Windows [Cazenave, 2020]. With the right formula of the bias, GNRPA can give very good results and compete with state of the art Reinforcement Learning algorithms to solve several problems.

In this paper, we focus on a popular example of combinatorial optimization problems: the Maximum Independent Set problem. We present a new approach using the GNRPA algorithm combined with a trained neural network as an efficient way to present the bias of the algorithm. As we will show, our approach provides good performance and shows good results compared to state of the art algorithms.

The outline of this paper is as follow: the next section present the Monte Carlo search and its related algorithms, section 3 defines the Maximum Independent Set problem, section 4 presents our modeling to the problem, section 5 explains the neural network construction, section 6 presents the related work and the state of the art approaches, section 7 defines our algorithms used to compare the results and finally we give the experimental results in section 8.

2 Monte Carlo Search

Monte Carlo Tree Search (MCTS) has been successfully applied to many games and problems [Browne *et al.*, 2012].

Nested Monte Carlo Search (NMCS) [Cazenave, 2009] is an algorithm that works well for puzzles and optimization problems. It biases its playouts using lower level playouts. At level zero NMCS adopts a uniform random playout policy. Online learning of playout strategies combined with NMCS has given good results on optimization problems [Rimmel *et al.*, 2011]. Other applications of NMCS include Single Player General Game Playing [Méhat and Cazenave, 2010], Cooperative Pathfinding [Bouzy, 2013], Software testing [Poulding and Feldt, 2014], heuristic Model-Checking [Poulding and Feldt, 2015], the Pancake problem [Bouzy, 2016], Games [Cazenave *et al.*, 2016] and the RNA inverse folding problem [Portela, 2018].

Online learning of a playout policy in the context of nested searches has been further developed for puzzles and optimization with NRPA. NRPA has found new world records in Morpion Solitaire and crosswords puzzles. Stefan Edelkamp and co-workers have applied the NRPA algorithm to multiple problems. They have optimized the algorithm for the Traveling Salesman with Time Windows (TSPTW) problem [Cazenave and Teytaud, 2012; Edelkamp *et al.*, 2013]. Other applications deal with 3D Packing with Object Orientation [Edelkamp *et al.*, 2014], the physical traveling salesman problem [Edelkamp and Greulich, 2014], the Multiple Sequence Alignment problem [Edelkamp and Tang, 2015], Logistics [Edelkamp *et al.*, 2016; Cazenave *et al.*, 2020a], Graph Coloring [Cazenave *et al.*, 2020b] and RNA Design [Cazenave and Fournier, 2020]. The principle of NRPA is to adapt the playout policy so as to learn the best sequence of moves found so far at each level.

GNRPA [Cazenave, 2020] has much improved the result of NRPA for RNA Design [Cazenave and Fournier, 2020].

3 Maximum Independent Set

We define in this section the Maximum Independent Set (MIS) problem. Let $G = (V, E)$ denote a graph, where $V = \{v_1, ..., v_n\}$ is the set of vertices (or nodes) and $E \subseteq V \times V$ is the set of edges, i.e., each of the related pairs of vertices is called an edge.

The MIS problem is the following: given a graph $G = (V, E)$, find the largest subset $S \subseteq V$ in which no two vertices in S are adjacent (connected by an edge). In other words, the MIS set is the largest set that if we try to add another vertex, it would disrupt its independence.

The Maximum Independent Set problem is one of the well known NP-hard optimization problem where the exact methods become impractical to solve [Tomita *et al.*, 2010; San Segundo *et al.*, 2011]. Heuristics, on the other side, seem to find a way to successfully solve these kind of problems [Abe *et al.*, 2019]. The maximum independent set problem covers many applications including classification theory, information retrieval and computer vision [Feo *et al.*, 1994]. Independent sets are also used in efficient strategies for labeling maps [Gemsa *et al.*, 2016], computing shortest paths on road networks [Kieritz *et al.*, 2010], computing mesh edge traversal ordering for rendering [Sander *et al.*, 2008], as well as many applications in biology [Gardiner *et al.*, 2000], sociology [Harary and Ross, 1957] and e-commerce [Zaki *et al.*, 1997].

In order to capture the MIS problem as a tree search problem, we chose the following implementation for a graph and we will explain it in more detail afterwards.

A state, in the context of the MIS problem, is the current independent set of nodes, i.e. the independent set found so far. The move is the current vertex to play as we will explain it more in details later. To play a move is to add the vertex to the current independent set. The legal moves are the nodes that can be added to the current set, i.e. the nodes that are independent to each vertex in the current set. We reach the final state if we no longer have a legal node to add. Finally, to score the playout, we use the size of the independent set of the final state.

4 Modeling of the Problem

4.1 Dynamic Bias

The principle of NRPA is to adapt the playout policy by adapting weights on each action so as to learn the best sequence of moves found so far at each level [Rosin, 2011]. It starts from a uniform random policy and later improved using gradient descent steps based the best sequence discovered so far. A single move will then have the same weight in all states. In some problems, initializing the weights can be difficult since we have too many possible weights.

To generalize this algorithm and improve its performance, GNRPA proposes using a bias β that can be different for the same move according to the state which would not be possible with weight initialization.

4.2 Usual Modeling

In GNRPA each move is associated to a weight. The goal of the algorithm is to learn these weights so as to produce a playout policy that generates good sequences of moves. At each level of the algorithm the best sequence found so far is memorized. Let $s_1, ..., s_m$ be the sequence of states of the best sequence. Let n_i be the number of possible moves in a state s_i. Let $m_{i1}, ..., m_{in_i}$ be the possible moves in state s_i and m_{ib} be the move of the best sequence in state s_i. The goal is to learn to play the move m_{ib} in state s_i.

The playouts use Gibbs sampling. Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm. It is a simulation tool for obtaining samples from a non-normalized joint density function. [Gelfand, 2000]. Each move m_{ik} is associated to a weight w_{ik} and a bias β_{ik}. The probability p_{ik} of choosing the move m_{ik} in a playout is the softmax function:

$$p_{ik} = \frac{e^{w_{ik}+\beta_{ik}}}{\Sigma_j e^{w_{ij}+\beta_{ij}}}$$

If we use α as a learning rate we update the weights with:

$$w_{ij} = w_{ij} - \alpha(p_{ij} - \delta_{bj})$$

Where $\delta_{bj} = 1$ if $b = j$ and 0 otherwise.

4.3 0/1 Modeling

One way to play the moves is to chose a random order to the vertices of the graph then, at each state, the possible moves are all the legal nodes. This method can be costly since we can have thousands of possible moves at each state in case of large graphs.

Another more efficient way is 0/1 modeling as explained next.

Node order

We first order the nodes in an efficient way: the smallest degree first. Then, at each state of the sequence during the game, we update the order of the rest of the nodes. The next best node will be the one that has the fewer non played neighbors.

Modeling

We start with the node that has the lowest degree, then we have two possible moves: using or not the node i.e, add or not the node to the current independent set. Thus, given a state S with a current independent set, the legal moves are use the node or not. A move is then modeled as a pair (v, p) where v is the chosen vertex, and p is a Boolean that indicates whether to add the vertex to the solution or not.

To adapt the weights in GNRPA algorithm, we set the learning rate $\alpha = 1$ and set the number of iterations to 100 per level as in the original NRPA algorithm.

We update the weights as follow:
if the node is chosen: $w_1 = w_1 + 1 - p_1, w_0 = w_0 + p_1 - 1$
if not: $w_1 = w_1 + p_0 - 1, w_0 = w_0 + 1 - p_0$

5 Deep Learning

To improve the experimental results of Monte Carlo, we resort to deep learning techniques.

We train a neural network that given a graph $G = (V, E)$ with $V = \{v_1, ..., v_n\}$ the set of vertices and $E \subseteq V \times V$ the set of edges, the goal is to produce a binary labelling for each vertex in G such that label 1 indicates that a vertex is in the independent set and label 0 indicates that it is not.

In the next paragraphs, we explain how we prepare the dataset, how we build our model and how we train it.

5.1 Graph pre-processing

To represent a graph $G = (V, E)$ as an input for the neural network whilst maximally preserving its properties, we represent each node in the graph with an arbitrary representation vector with 3 features [Balaji *et al.*, 2010]:

$deg(v)$: the degree of a vertex: the number of edges connected to the vertex.

$s(v)$: the support of a vertex: the sum of the degrees of its neighbours.

$s_2(v)$: the sum of the supports of its neighbours.

These features can capture the properties of a node and its neighborhood in a graph. They can generate vector representations of nodes within a graph. We get then an array of shape $(n, 3)$. Each line represent a vertex and each column represent a feature. We normalize each column with its maximum to change the values to a common scale without distorting differences in the ranges of values.

5.2 The data construction

We use the following graphs available on the Github project[1] of CombOpt Zero approach defined in section 6. The number of vertices and edges in each graph are denoted by $|V|$ and $|E|$, respectively.

Graph_name	$\|V\|$	$\|E\|$
ba100_5	100	475
ba200_5	200	975
ba1000_5	1000	4975
citeseer	3327	4552
bio-SC-LC	2004	20452
bio-yeast	1458	1948
cora	2708	5429
dimacs-frb30-15-1	450	17827
dimacs-frb50-23-1	1150	80072
er20	20	34
er200_10	200	1957
er100_15	100	783
er1000_5	1000	25091
er5000_1	5000	124804

To diversify the information in the input and to prevent over-fitting, we transform each graph from the table below into a array as the Graph pre-processing explains. We concatenate them all to get our input with 18717 lines and 3 features.

As we need a labeled input to train the neural network, we create the output set using the maximum independent set that we get as a result of the greedy algorithm described later on (section 7).

For each graph, we create a binary vector. Each element of the vector represent the label of the vertex. It takes 1 if the vertex exist in the MIS of the graph, 0 otherwise.

In the same order as the input, we concatenate the vectors and we obtain the new label array of shape $(18717, 1)$.

Finally, to avoid any element of bias, we randomly permute the input rows and its labels in the same order. We can do this using the simple function "shuffle" by sklearn module.

The dataset is now preprocessed and ready to feed in the neural network.

5.3 The Neural Network Architecture

We developed a neural network that has four layers. An input layer of 200 nodes, two hidden layers of 100 and 50 nodes successively, and an output layer that contains a single neuron in order to make the prediction.

The last node uses the sigmoid activation function in order to produce a probability output in the range of 0 to 1 that can automatically be converted to class values. The other layers use ReLU as the activation function. We use dropout at each layer to prevent over-fitting.

After trying different neural network structures, this architecture seems to give the best result.

We use the loss function binary_crossentropy during training, the preferred loss function for binary classification problems. The model also uses Stochastic gradient descent optimization with a relatively low learning rate and momentum.

Finally, the accuracy metrics is collected when the model is trained.

5.4 Training Phase

In order to train the network, we split the dataset into training and testing. We use 80 % of the data as training data on which

[1]https://github.com/xuzijian629/combopt-zero/tree/master/test_graphs

we will train our neural network. The rest is used as testing data to check our trained neural network on unknown data.

We train the model for 100 epochs with batches each of size equal to 50.

To evaluate the model at the end of each epoch, we use the automatic verification dataset by setting the validation_split argument on the fit() function of keras to a percentage of 50. It automatically reserve 50% of the training data for validation.

After trying different values, these seem to give the best result.

After training, we get accuracy = 0.78 and validation accuracy = 0.81 for the training set.

5.5 Testing Phase

After fitting the model, we test it on test data, we get 0.81 as accuracy. Our constructed model, shows better result then a baseline model. With a baseline model with only two layers: an input layer and an output layer we get 0.74 accuracy. We evaluate our model on some graphs (both seen and unseen by the model). We get the following results:

Graphs	test-accuracy
er100_15	0.7600
er200_10	0.7950
er1000_5	0.8880
er5000_1	0.8982
ba100_5	0.6300
ba200_5	0.5900
ba1000_5	0.7310
ba5000_5	0.5606
cora	0.6388
citeseer	0.7280
web-edu	0.5922
web-spam	0.7004
road-minnesota	0.4992
bio-yeast	0.8779
bio-SC-LC	0.7535
rt_damascus	0.8853
soc-wiki-vote	0.7998
socfb-bowdoin47	0.8552
dimacs-frb30-15-1	0.9378
dimacs-frb50-23-1	0.9609

Our model seems to predict well the Independent Set on many graphs. Our constructed features and the representation of nodes helped to capture the properties of a node and its neighborhood in a graph.

5.6 Prediction

We finally predict on graphs using the simple method of keras: predict_proba(). It gives the probability of a vertex to be in the MIS. It takes the graph prepossessed as input and returns an array of shape (number of vertices of the graph, 1). Each row contains the probability that has the vertex to belong to the MIS.

We use each probability as the bias of its corresponding move in GNRPA.

6 State-of-art approaches for MIS problem

In this section we present state-of-the-art heuristic algorithms for the NP-hard problems. We use these methods to compare with our approach in solving the Maximum Independent Set problem.

S2V-DQN

The approach of S2V-DQN (Structure2Vec Deep Q-learning) [Khalil *et al.*, 2017] is to train a greedy algorithm to build up solutions by reinforcement learning (RL).

Khalil et al presented a combination of an RL framework with a graph embedding approach.

The framework uses a graph embedding network, called structure2vec (S2V) [Dai *et al.*, 2016] to design greedy heuristics.

The structure2vec and Q functions are implemented as neural networks, and several variants of Q learning are applied to train them.

The S2V-DQN framework receives attention for its promising results in solving graph-based combinatorial problems like the MIS problem.

CombOpt Zero

The CombOpt Zero [Abe *et al.*, 2019] ,is a novel learning strategy based on AlphaGo Zero [Silver *et al.*, 2017] that improves the S2V-DQN method. It combines reinforcement learning with a novel learning strategy inspired by AlphaGo Zero.

This strategy can be combined with several graph neural network (GNN) as well as S2V [Dai *et al.*, 2016].

Some of the GNN tested in this work are : The Graph Convolutional Network (GCN)[Kipf and Welling, 2016], The Graph Isomorphism Network (GIN) [Xu *et al.*, 2018] and The Invariant Graph Network (2-IGN+)[Maron *et al.*, 2019].

All these recently developed graph neural networks were tested on different graph combinatorial problems.

In addition to S2V-DQN, Abe and co-workers compare there strategy with some known heuristics or approximation algorithms. They used randomized algorithms as well as the state-of-the-art solver CPLEX defined later on.

State of the art solvers

As competitors, we furthermore consider the integer programming solver **CPLEX**[2] and the MIS solver based on advanced evolutionary algorithm **KaMIS v2.0**[3]. These algorithms can handle sparse graphs of millions of nodes.

KaMIS (Karlsruhe Maximum Independent Sets) integrates an advanced algorithm based on graph partitioning using the KaHIP[4] framework and reduction techniques to compute large independent sets in huge sparse networks. The framework uses an algorithm that repeatedly kernelizes[5] the graph

[2]www.cplex.com

[3]http://KarlsruheMIS.github.io

[4]https://kahip.github.io

[5]inputs to the algorithm are replaced by a smaller input, called a kernel

until a large independent set is found. We use two of KaMIS programs to compare:

ReduMIS: [Lamm *et al.*, 2016] an evolutionary algorithm based on graph partitioning and reduction techniques.

OnlineMIS: [Dahlum *et al.*, 2016] a local search algorithm that uses (online) reductions to speed up local search.

7 Our algorithms:

We use the following algorithms to compare the results :

Greedy Algorithm

For greedy algorithm, we use the same implementation of the playout. We only play one playout and we choose the node with the fewer degree at each state. We update the order of the nodes as we did in the GNRPA algorithm.

NRPA Algorithm

We use for NRPA the same algorithm as GNRPA with a bias $\beta = 0$ and a learning rate $\alpha = 1$.

GNRPA(1) Algorithm

To highlight the efficient of using the combination between GNRPA and the relevant predictions of a trained Neural Network, we test the GNRPA algorithm without the Deep learning phase. Instead, we set the bias β of each node to :

$$\beta = -\frac{neighbors\_of\_the\_node}{maximum\_neighbors}$$

GNRPA(2) Algorithm

For the bias β of each node in GNRPA(2) algorithm, we use the probabilities that we get from the Neural Network trained with different graphs described in section 5.

8 Experimental Results

For testing, we use the benchmark graph instances used in CombOpt Zero project available on GitHub[6]. We use different graphs both seen and unseen by our trained model.

The experiments for the state of the art algorithms , as indicated in their paper [Abe *et al.*, 2019], was run on Intel Xeon E5-2695 v4 with four NVIDIA Tesla P100 GPUs. Their time limits was set of 10 minutes and the best found solution was used as results . The results of the randomized algorithm were the best objective among 100 runs.

Table 1 shows a comparison between the state of the art algorithms using GNN approach on the MIS problem to various instances of benchmark graphs. Experiments shows the successful performance of CombOpt Zero if the properly GNN model is selected. Note that we used results reported in [Abe *et al.*, 2019], without reproducing it.

The experiments for CPLEX was run on MacBook Pro 2.4 GHz Quad-Core Intel Core i5 with the time limit of 10 minutes. The experiments for the KaMIS solver was run on an Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz. The time limit was set to 10 minutes (but it ended before most of the time) and other parameters to the default ones.

Table 2 shows a comparison between the state of the art solvers using operations research approach on the MIS problem . The best result is marked in bold. Experiments shows the efficiency of the evolutionary algorithm reduMIS. The local search algorithm OnlineMIS is a little less efficient.

Our algorithms was run on HP laptop Intel Core i5-7200U Processor 2.5GHz. We set $N = 100$ and level $l = 3$.

We first consider the results the best objective among 100 runs. Table 3 shows a comparison between the different Monte Carlo algorithms explained in section 7. The best result is marked in bold. The GNRPA(2) algorithm shows generally a good performance compared to both GNRPA(1) and NRPA. In particularly, GNRPA(2) shows a consistent superiority over GNRPA(1) which highlights the efficient of using the trained neural network in the algorithm. Moreover, GNRPA(2) competes with NRPA and gives generally better result.

Additionally, we set a time limit of 10 minutes for our Monte Carlo algorithms to compare the speed of the algorithms in finding the best solution as well as the performance of our algorithm compared to state of the art. Table 4 shows a comparison between Monte Carlo, the best results of the state of the art of table 1 and the best results of the state of the art MIS solvers of table 2 with time limit of 10 minutes.

A comparison between Monte Carlo algorithms shows that both GNRPA(1) and GNRPA(2) are faster than NRPA in finding the best solution. However, GNRPA(1) can be faster then GNRPA(2) for large graphs. We can also note that GNRPA(2) can be very efficient if it runs on a more powerful machine.

A comparison between GNRPA(2) and the best results of the state of the art algorithms shows that our algorithm achieves a good performance, generally superior to the discussed state of the art algorithms. However it shows a less efficient performance for large graphs.

A comparison between GNRPA(2) and the best results of the state of the art MIS solvers shows a great performance of the ReduMIS solver especially on large sparse networks. Even though, it is interesting that GNRPA(2) reached to the optimal solution of the evolutionary algorithm on some graph instances.

We additionally note that the instances used to compare the results may be too easy to evaluate the differences between the algorithms.

[6]https://github.com/xuzijian629/combopt-zero/tree/master/test_graphs

Table 1: Scores obtained by state of the art algorithms with 10 minutes time limit.

Graphs	$\lvert V\rvert$	$\lvert E\rvert$	2-IGN+	GIN	GCN	S2V	S2V-DQN	randomized
er100_15	100	783	**24**	**24**	23	**24**	**24**	23
er200_10	200	1957	40	40	39	**41**	40	37
er1000_5	1000	25091	106	107	**108**	105	106	89
er5000_1	5000	124804	-	544	538	544	**551**	435
ba100_5	100	475	**37**	**37**	**37**	**37**	**37**	**37**
ba200_5	200	975	81	81	80	**82**	**82**	79
ba1000_5	1000	4975	400	407	403	408	**409**	394
ba5000_5	5000	24975	-	2079	2062	**2085**	2078	1960
cora	2708	5429	-	1450	1448	**1451**	1448	1439
citeseer	3327	4552	-	1818	1817	1819	1817	**1860**
web-edu	3031	6474	-	**1580**	**1580**	**1580**	**1580**	**1580**
web-spam	4767	37375	-	**2464**	2456	2463	2441	2434
road-minnesota	2642	3303	-	1318	1300	1316	**1321**	1313
bio-yeast	1458	1948	990	1000	1001	**1002**	**1002**	**1002**
bio-SC-LC	2004	20452	945	959	953	**964**	948	936
rt_damascus	3052	3881	-	2673	**2683**	2679	**2683**	**2683**
soc-wiki-vote	889	2914	481	**483**	482	**483**	482	**483**
socfb-bowdoin47	2252	84387	445	456	**457**	443	426	392
dimacs-frb30-15-1	450	17827	26	26	26	**27**	26	24
dimacs-frb50-23-1	1150	80072	41	**44**	43	40	41	39

Table 2: Scores obtained by the MIS solvers with 10 minutes time limit.

Graphs	$\lvert V\rvert$	$\lvert E\rvert$	CPLEX	redumis	onlinemis
er100_15	100	783	**24**	**24**	23
er200_10	200	1957	**41**	**41**	**41**
er1000_5	1000	25091	107	**114**	112
er5000_1	5000	124804	544	**576**	556
ba100_5	100	475	**37**	**37**	**37**
ba200_5	200	975	**82**	**82**	**82**
ba1000_5	1000	4975	411	**412**	**412**
ba5000_5	5000	24975	2090	**2101**	**2101**
cora	2708	5429	**1451**	**1451**	**1451**
citeseer	3327	4552	**1867**	**1867**	**1867**
web-edu	3031	6474	**1580**	**1580**	**1580**
web-spam	4767	37375	**2470**	**2470**	**2470**
road-minnesota	2642	3323	**1323**	**1323**	**1323**
bio-yeast	1458	1948	**1002**	**1002**	**1002**
bio-SC-LC	2004	20452	**968**	**968**	**968**
rt_damascus	3052	3881	**2683**	**2683**	-
soc-wiki-vote	889	2914	**483**	**483**	**483**
socfb-bowdoin47	2252	84387	461	**466**	**466**
dimacs-frb30-15-1	450	17827	28	**30**	29
dimacs-frb50-23-1	1150	80072	43	**49**	47

Table 3: Scores obtained by our algorithms best of 100 runs.

Graphs	$\|V\|$	$\|E\|$	Greedy	NRPA	GNRPA (1)	GNRPA (2)
er100_15	100	783	**24**	**24**	**24**	**24**
er200_10	200	1957	37	**41**	**41**	**41**
er1000_5	1000	25091	99	113	112	**114**
er5000_1	5000	124804	518	**547**	544	544
ba100_5	100	475	36	**37**	**37**	**37**
ba200_5	200	975	78	**82**	**82**	**82**
ba1000_5	1000	4975	400	**412**	**412**	**412**
ba5000_5	5000	24975	2049	**2092**	2090	**2092**
cora	2708	5429	1447	**1451**	**1451**	**1451**
citeseer	3327	4552	1864	**1867**	**1867**	**1867**
web-edu	3031	6474	**1580**	**1580**	**1580**	**1580**
web-spam	4767	37375	2454	**2470**	2469	**2470**
road-minnesota	2642	3303	1311	**1323**	**1323**	**1323**
bio-yeast	1458	1948	**1002**	**1002**	**1002**	**1002**
bio-SC-LC	2004	20452	949	**968**	967	**968**
rt_damascus	3052	3881	**2683**	**2683**	**2683**	**2683**
soc-wiki-vote	889	2914	481	**483**	**483**	**483**
socfb-bowdoin47	2252	84387	440	**466**	465	465
dimacs-frb30-15-1	450	17827	23	29	28	**30**
dimacs-frb50-23-1	1150	80072	43	46	46	**48**

Table 4: Performance comparison with state of the art algorithms,10 minutes time limit.

Graphs	$\|V\|$	$\|E\|$	Best_of_ Tab1	Best_of_ Tab2	NRPA	GNRPA (1)	GNRPA (2)
er100_15	100	783	**24**	**24**	**24**	**24**	**24**
er200_10	200	1957	**41**	**41**	**41**	**41**	**41**
er1000_5	1000	25091	108	**114**	111	112	112
er5000_1	5000	124804	551	**576**	503	538	501
ba100_5	100	475	**37**	**37**	**37**	**37**	**37**
ba200_5	200	975	**82**	**82**	**82**	**82**	**82**
ba1000_5	1000	4975	409	**412**	410	411	**412**
ba5000_5	5000	24975	2085	**2101**	1992	2068	2038
cora	2708	5429	**1451**	**1451**	**1451**	**1451**	**1451**
citeseer	3327	4552	1860	**1867**	**1867**	**1867**	**1867**
web-edu	3031	6474	**1580**	**1580**	**1580**	**1580**	**1580**
web-spam	4767	37375	2464	**2470**	2441	2467	2459
road-minnesota	2642	3303	1321	**1323**	**1323**	**1323**	**1323**
bio-yeast	1458	1948	**1002**	**1002**	**1002**	**1002**	**1002**
bio-SC-LC	2004	20452	964	**968**	**968**	**968**	**968**
rt_damascus	3052	3881	**2683**	**2683**	**2683**	**2683**	**2683**
soc-wiki-vote	889	2914	**483**	**483**	**483**	**483**	**483**
socfb-bowdoin47	2252	84387	457	**466**	464	461	463
dimacs-frb30-15-1	450	17827	27	**30**	29	28	**30**
dimacs-frb50-23-1	1150	80072	44	**49**	46	46	47

9 Conclusion

In this paper, we propose Monte Carlo Search algorithms to solve the Maximum Independent Set problem. GNRPA uses a Deep Neural Network for the bias of the algorithm.

We propose an adequate implementation of the GNRPA algorithm and an efficient way to train a neural network in order to predict the bias of the generalized algorithm.

The GNRPA algorithm with the right formula of the bias, shows the best performance in solving the MIS optimisation problem compared with state-of-the-art algorithms and a competitive performance compared to the state-of-the-art solvers.

This approach can be further used in other graph problems and could be useful in several other domains. It would be also interesting to use reduction techniques on graphs to improve the GNRPA solution on large scale graphs.

Acknowledgment

This work was supported in part by the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute).

References

[Abe *et al.*, 2019] Kenshin Abe, Zijian Xu, Issei Sato, and Masashi Sugiyama. Solving np-hard problems on graphs with extended alphago zero. *arXiv e-prints*, pages arXiv–1905, 2019.

[Balaji *et al.*, 2010] S Balaji, V Swaminathan, and K Kannan. A simple algorithm to optimize maximum independent set. *Advanced Modeling and Optimization*, 12(1):107–118, 2010.

[Bello *et al.*, 2016] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[Bouzy, 2013] Bruno Bouzy. Monte-carlo fork search for cooperative path-finding. In *Computer Games - Workshop on Computer Games, CGW 2013, Held in Conjunction with the 23rd International Conference on Artificial Intelligence, IJCAI 2013, Beijing, China, August 3, 2013, Revised Selected Papers*, pages 1–15, 2013.

[Bouzy, 2016] Bruno Bouzy. Burnt pancake problem: New lower bounds on the diameter and new experimental optimality ratios. In *Proceedings of the Ninth Annual Symposium on Combinatorial Search, SOCS 2016, Tarrytown, NY, USA, July 6-8, 2016*, pages 119–120, 2016.

[Browne *et al.*, 2012] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, March 2012.

[Cazenave and Fournier, 2020] Tristan Cazenave and Thomas Fournier. Monte carlo inverse folding. In *Monte Carlo Search at IJCAI*, 2020.

[Cazenave and Teytaud, 2012] Tristan Cazenave and Fabien Teytaud. Application of the nested rollout policy adaptation algorithm to the traveling salesman problem with time windows. In *Learning and Intelligent Optimization - 6th International Conference, LION 6, Paris, France, January 16-20, 2012, Revised Selected Papers*, pages 42–54, 2012.

[Cazenave *et al.*, 2016] Tristan Cazenave, Abdallah Saffidine, Michael John Schofield, and Michael Thielscher. Nested monte carlo search for two-player games. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 687–693, 2016.

[Cazenave *et al.*, 2020a] Tristan Cazenave, Jean-Yves Lucas, Hyoseok Kim, and Thomas Triboulet. Monte carlo vehicle routing. In *ATT at ECAI*, 2020.

[Cazenave *et al.*, 2020b] Tristan Cazenave, Benjamin Negrevergne, and Florian Sikora. Monte carlo graph coloring. In *Monte Carlo Search at IJCAI*, 2020.

[Cazenave, 2009] Tristan Cazenave. Nested Monte-Carlo Search. In Craig Boutilier, editor, *IJCAI*, pages 456–461, 2009.

[Cazenave, 2020] Tristan Cazenave. Generalized nested rollout policy adaptation. In *Monte Carlo Search at IJCAI*, 2020.

[Dahlum *et al.*, 2016] Jakob Dahlum, Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F. Werneck. Accelerating local search for the maximum independent set problem. In *15th International Symposium on Experimental Algorithms SEA*, volume 9685 of *Lecture Notes in Computer Science*, pages 118–133. Springer, 2016.

[Dai *et al.*, 2016] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711, 2016.

[Deudon *et al.*, 2018] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pages 170–181. Springer, 2018.

[Edelkamp and Greulich, 2014] Stefan Edelkamp and Christoph Greulich. Solving physical traveling salesman problems with policy adaptation. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pages 1–8. IEEE, 2014.

[Edelkamp and Tang, 2015] Stefan Edelkamp and Zhihao Tang. Monte-carlo tree search for the multiple sequence alignment problem. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

[Edelkamp *et al.*, 2013] Stefan Edelkamp, Max Gath, Tristan Cazenave, and Fabien Teytaud. Algorithm and knowledge engineering for the tsptw problem. In *Computational Intelligence in Scheduling (SCIS), 2013 IEEE Symposium on*, pages 44–51. IEEE, 2013.

[Edelkamp *et al.*, 2014] Stefan Edelkamp, Max Gath, and Moritz Rohde. Monte-carlo tree search for 3d packing with object orientation. In *KI 2014: Advances in Artificial Intelligence*, pages 285–296. Springer International Publishing, 2014.

[Edelkamp *et al.*, 2016] Stefan Edelkamp, Max Gath, Christoph Greulich, Malte Humann, Otthein Herzog, and Michael Lawo. Monte-carlo tree search for logistics. In *Commercial Transport*, pages 427–440. Springer International Publishing, 2016.

[Feo *et al.*, 1994] Thomas A Feo, Mauricio GC Resende, and Stuart H Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5):860–878, 1994.

[Gardiner *et al.*, 2000] Eleanor J Gardiner, Peter Willett, and Peter J Artymiuk. Graph-theoretic techniques for macromolecular docking. *Journal of Chemical Information and Computer Sciences*, 40(2):273–279, 2000.

[Gelfand, 2000] Alan E Gelfand. Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304, 2000.

[Gemsa *et al.*, 2016] Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Evaluation of labeling strategies for rotating maps. *Journal of Experimental Algorithmics (JEA)*, 21:1–21, 2016.

[Harary and Ross, 1957] Frank Harary and Ian C Ross. A procedure for clique detection using the group matrix. *Sociometry*, 20(3):205–215, 1957.

[Khalil *et al.*, 2017] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[Kieritz *et al.*, 2010] Tim Kieritz, Dennis Luxen, Peter Sanders, and Christian Vetter. Distributed time-dependent contraction hierarchies. In *International Symposium on Experimental Algorithms*, pages 83–93. Springer, 2010.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Lamm *et al.*, 2016] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 138–150. SIAM, 2016.

[Maron *et al.*, 2019] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pages 2156–2167, 2019.

[Méhat and Cazenave, 2010] Jean Méhat and Tristan Cazenave. Combining UCT and Nested Monte Carlo Search for single-player general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):271–277, 2010.

[Portela, 2018] Fernando Portela. An unexpectedly effective monte carlo technique for the rna inverse folding problem. *BioRxiv*, page 345587, 2018.

[Poulding and Feldt, 2014] Simon M. Poulding and Robert Feldt. Generating structured test data with specific properties using nested monte-carlo search. In *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, pages 1279–1286, 2014.

[Poulding and Feldt, 2015] Simon M. Poulding and Robert Feldt. Heuristic model checking using a monte-carlo tree search algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, pages 1359–1366, 2015.

[Rimmel *et al.*, 2011] Arpad Rimmel, Fabien Teytaud, and Tristan Cazenave. Optimization of the Nested Monte-Carlo algorithm on the traveling salesman problem with time windows. In *Applications of Evolutionary Computation - EvoApplications 2011: EvoCOMNET, EvoFIN, EvoHOT, EvoMUSART, EvoSTIM, and EvoTRANSLOG, Torino, Italy, April 27-29, 2011, Proceedings, Part II*, volume 6625 of *Lecture Notes in Computer Science*, pages 501–510. Springer, 2011.

[Rosin, 2011] Christopher D Rosin. Nested rollout policy adaptation for monte carlo tree search. In *Ijcai*, pages 649–654, 2011.

[San Segundo *et al.*, 2011] Pablo San Segundo, Diego Rodríguez-Losada, and Agustín Jiménez. An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, 38(2):571–581, 2011.

[Sander *et al.*, 2008] Pedro V Sander, Diego Nehab, Eden Chlamtac, and Hugues Hoppe. Efficient traversal of mesh edges using adjacency primitives. *ACM Transactions on Graphics (TOG)*, 27(5):1–9, 2008.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[Tomita *et al.*, 2010] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, Shinya Takahashi, and Mitsuo Wakatsuki. A simple and faster branch-and-bound algorithm for finding a maximum clique. In *International Workshop on Algorithms and Computation*, pages 191–203. Springer, 2010.

[Xu *et al.*, 2018] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[Zaki *et al.*, 1997] Mohammed J Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei Li. Parallel algorithms for discovery of association rules. *Data mining and knowledge discovery*, 1(4):343–373, 1997.