# Learning the Bias Weights for Generalized Nested Rollout Policy Adaptation

Julien Sentuc[1] Farah Ellouze[1] Jean-Yves Lucas[2]
Tristan Cazenave[1]

[1]LAMSADE, Université Paris Dauphine - PSL, CNRS, Paris, France
[2]OSIRIS department, EDF Lab Paris-Saclay, Electricité de France, France

**Abstract**

Generalized Nested Rollout Policy Adaptation (GNRPA) is a Monte Carlo search algorithm for single player games and optimization problems. In this paper we propose to modify GNRPA in order to automatically learn the bias weights. The goal is both to obtain better results on sets of dissimilar instances, and also to avoid some hyperparameters settings. Experiments show that it improves the algorithm for two different optimization problems: the Vehicle Routing Problem and 3D Bin Packing.

## 1 Introduction

Monte Carlo Tree Search (MCTS) [20, 12] has been successfully applied to many games and problems [3]. It originates from the computer game of Go [2] with a method based on simulated annealing [4]. The principle underlying MCTS is learning the best move using statistics on random games.

Nested Monte Carlo Search (NMCS) [5] is a recursive algorithm which uses lower level playouts to bias its playouts, memorizing the best sequence at each level. At each stage of the search, the move with the highest score at the lower level is played by the current level. At each step, a lower-level search is launched for all possible moves and the move with the best score is memorized. At level 0, a Monte Carlo simulation is performed, random decisions are made until a terminal state is reached. At the end, the score for the position is returned. NMCS has given good results on many problems like puzzle solving, single player games [22], cooperative path finding or the inverse folding problem [23].

Based on the latter, the Nested Rollout Policy Adaptation (NRPA) algorithm was introduced [26]. NRPA combines nested search, memorizing the best sequence of moves found, and the online learning of a playout policy using this sequence. NRPA achieved world records in Morpion Solitaire and crossword puzzles and has been applied to many problems such as object wrapping [17], traveling salesman with time window [10, 15], vehicle routing problems [16, 8] or network traffic engineering [13].

GNRPA (Generalized Nested Rollout Policy Adaptation) [6] generalizes the way the probability is calculated using a temperature and a bias. It has been applied to some problems like Inverse Folding [7] and Vehicle Routing Problem (VRP) [27].

This work presents an extension of GNRPA using bias learning. The idea is to learn the parameters of the bias along with the policy. We demonstrate that learning the bias parameters improves the results of GNRPA for Solomon instances of the VRP and for 3D Bin Packing.

This paper is organized as follows. Section 2 describes the NRPA and GNRPA algorithms, as well as its extension. Section 3 presents the experimental results for the two problems studied : VRP and 3D Bin Packing. Finally, the last section concludes.

# 2 Monte Carlo Search

This section presents the NRPA algorithm as well as its generalization GNRPA. The formula for learning the bias weights is introduced. A new optimization for GNRPA, based on conventional ones, is then presented.

## 2.1 NRPA and GNRPA

The Nested Rollout Policy Adaptation (NRPA) [26] algorithm is an effective combination of NMCS and the online learning of a playout policy. NRPA holds world records for Morpion Solitaire and crosswords puzzles.

In NRPA/GNRPA each move is associated to a "move weight" stored in an array called the policy. The goal of these two algorithms is to learn these weights thanks to the solutions found during the search, thus producing a playout policy that generates good sequences of moves.

NRPA/GNRPA use nested search. In NRPA/GNRPA, each level takes a policy as input and returns a sequence and its associated score. At any level $> 0$, the algorithm makes numerous recursive calls to lower levels, adapting the policy each time with the best solution to date. It should be noted that the changes made to the policy do not affect the policy in higher levels (line 7-8 of algorithm 1). At level 0, NRPA/GNRPA return the sequence obtained by playout function as well as its associated score.

The playout function sequentially constructs a random solution biased by the weight of the moves until it reaches a terminal state. At each step, the function performs Gibbs sampling, choosing the actions with a probability given by the softmax function.

Let $w_{ic}$ be the weight associated with move $c$ in step $i$ of the sequence. In NRPA, the probability of choosing move $c$ at the index $i$ is defined by:

$$p_{ic} = \frac{e^{w_{ic}}}{\sum_k e^{w_{ik}}}$$

GNRPA [6] generalizes the way the probability is calculated using a temperature $\tau$ and a bias $\beta_{ic}$. The temperature makes it possible to vary the exploration/exploitation trade-off. The probability of choosing the move $c$ at the index $i$ then becomes:

$$p_{ic} = \frac{e^{\frac{w_{ic}}{\tau}+\beta_{ic}}}{\sum_k e^{\frac{w_{ik}}{\tau}+\beta_{ik}}}$$

By taking $\tau = 1$ and $\beta_{ik} = 0$, we find the formula for NRPA.

In NRPA, policy weights can be initialized in order to accelerate convergence towards good solutions. The original weights in the policy array are then not uniformly set to 0, but to an appropriate value according to a heuristic relevant to the problem to solve. In GNRPA, the policy initialization is replaced by the bias. Furthermore, it is sometimes more practical to use $\beta_{ij}$ biases than to initialize the weights as we will see later on.

When a new solution is found (line 8 of algorithm 1), the policy is then adapted to the best solution found at the current level (line 13 of algorithm 1). The policy is passed by reference to the Adapt function. The "move weights" in the policy are updated as in [6].

The current policy is first saved into a temporary policy array named polp before modifying it. The policy copied into polp is then modified in the Adapt function, while the current policy will be used to calculate the probabilities of possible moves. After modification of the policy, the current policy is replaced by polp. The principle of the Adapt function is to increase the weight of the chosen moves and to decrease the weight of the other possible moves by an amount proportional to their probabilities of being played (line 15 of algorithm 2).

The NRPA algorithm therefore strikes a balance between exploration and exploitation. It exploits by shifting the policy weights to the best current solution and explores by picking moves using Gibbs sampling at the lower level. NRPA is a general algorithm that has been shown to be effective for many optimization problems. The idea of adapting a simulation policy has been applied successfully for many games such as Go [18].

It should be noted that in the case of optimization problems such as the VRP, we aim at minimizing the score (consisting of a set of penalties). $bestScore$ is therefore initialized to $+\infty$ (line 5 of algorithm 1) and we update it each time we find a new $result$ such that $result \leq bestScore$ (line 9 of algorithm 1).

## 2.2 Learning the bias

The advantage of the bias over weights initialization relies on its dynamic aspect. It can therefore take into account factors related to the current state. The goal of the extension proposed in this paper is to learn the parameters of the bias. For example, if we consider a bias formula made up of several criteria, such as in [27], we obtain in the case of 2 criteria $\beta_1$ and $\beta_2$: $\beta_i c = w_1 * \beta_1 + w_2 * \beta_2$, where $\beta_1$ and $\beta_2$ describe two different characteristics of a move. For VRP, it can be the time wasted while waiting to service a customer, the distance traveled, etc.

For some instances, a criterion is a sufficient feature, while others emphasize on another. It is therefore difficult or even impossible to find a single formula that would be appropriate for all instances. To tackle this problem, we propose a simple, yet effective modification of the GNRPA Algorithm, which we name Bias Learning GNRPA (BLGNRPA). We aim at learning the parameters of the bias in order to improve the results on different instances. The idea of learning the bias parameters $w_1$ and $w_2$ lies in adapting the importance of the different criteria along with the policy to the specific instance that we are trying to solve.

**Algorithm 1** The GNRPA algorithm.

```
GNRPA (level, policy)
  if level == 0 then
    return playout (root, policy)
  else
    bestScore ← −∞
    for N iterations do
      polp ← policy
      (result,new_seq) ← GNRPA(level − 1, polp)
      if result ≥ bestScore then
        bestScore ← result
        best_seq ← new_seq
      end if
      Adapt (policy, best_seq)
    end for
    return (bestScore, seq)
  end if
```

The probability of choosing the move $c$ at the index $i$ with this bias is:

$$p_{ic} = \frac{e^{\frac{w_{ic}}{\tau}+(w_1 \times \beta_{1ic}+w_2 \times \beta_{2ic})}}{\sum_k e^{\frac{w_{ik}}{\tau}+(w_1 \times \beta_{1ik}+w_2 \times \beta_{2ik})}}$$

Let $A_{ik} = e^{\frac{w_{ik}}{\tau}+(w_1 \times \beta_{1ik}+w_2 \times \beta_{2ik})}$.
The formula for the derivative of $f(x) = \frac{g(x)}{h(x)}$ is :

$$f'(x) = \frac{g'(x) \times h(x) - g(x) \times h'(x)}{h(x)^2}$$

So the derivative of $p_{ic}$ relative to $w_1$ is:

$$\frac{\delta p_{ic}}{\delta w_1} = \frac{\beta_{1ic} A_{ic} \times \sum_k A_{ik} - A_{ic} \times \sum_k \beta_{1ik} A_{ik}}{(\sum_k A_{ik})^2}$$

$$\frac{\delta p_{ic}}{\delta w_1} = \frac{A_{ic}}{\sum_k A_{ik}} \times (\beta_{1ic} - \frac{\sum_k \beta_{1ik} A_{ik}}{\sum_k A_{ik}})$$

$$\frac{\delta p_{ic}}{\delta w_1} = p_{ic} \times (\beta_{1ic} - \frac{\sum_k \beta_{1ik} A_{ik}}{\sum_k A_{ik}})$$

The cross-entropy loss for learning to play a move is $C_i = -log(p_{ic})$. In order to apply the gradient, we calculate the partial derivative of the loss: $\frac{\delta C_i}{\delta p_{ic}} = -\frac{1}{p_{ic}}$ . We then calculate the partial derivative of the softmax with respect to the weight:

$$\nabla w_1 = \frac{\delta C_i}{\delta p_{ic}} \frac{\delta p_{ic}}{\delta w_1} = -\frac{1}{p_{ic}} \times p_{ic}(\beta_{1ic} - \frac{\sum_k \beta_{1ik} A_{ik}}{\sum_k A_{ik}}) =$$

$$\frac{\sum_k \beta_{1ik} A_{ik}}{\sum_k A_{ik}} - \beta_{1ic}$$

If we use $\alpha_1$ and $\alpha_2$ as learning rates, we update the weight with (line 16 of algorithm 2):

$$w_1 \leftarrow w_1 + \alpha_1(\beta_{1ic} - \frac{\sum_k \beta_{1ik} A_{ik}}{\sum_k A_{ik}})$$

Similarly, the formula for $w_2$ is (line 17 of algorithm 2):

$$w_2 \leftarrow w_2 + \alpha_2(\beta_{2ic} - \frac{\sum_k \beta_{2ik} A_{ik}}{\sum_k A_{ik}})$$

**Algorithm 2** The new generalized adapt algorithm

1: Adapt ($policy, sequence$)
2: $\quad polp \leftarrow policy$
3: $\quad w_{1temp} \leftarrow w_1$
4: $\quad w_{2temp} \leftarrow w_2$
5: $\quad state \leftarrow root$
6: $\quad$ **for** $move \in sequence$ **do**
7: $\qquad polp[code(move)] \leftarrow polp[code(move)] + \frac{\alpha}{\tau}$
8: $\qquad w_{1temp} \leftarrow w_{1temp} + \beta_1(move)$
9: $\qquad w_{2temp} \leftarrow w_{2temp} + \beta_2(move)$
10: $\qquad z \leftarrow 0$
11: $\qquad$ **for** $m \in$ possible moves for $state$ **do**
12: $\qquad\quad z \leftarrow z + e^{\frac{policy[code(m)]}{\tau} + w1\beta_1(m) + w2\beta_2(m)}$
13: $\qquad$ **end for**
14: $\qquad$ **for** $m \in$ possible moves for $state$ **do**
15: $\qquad\quad polp[code(m)] \leftarrow polp[code(m)] - \frac{\alpha}{\tau} \times \frac{e^{\frac{policy[code(m)]}{\tau} + w1\beta_1(m) + w2\beta_2(m)}}{z}$
16: $\qquad\quad w_{1temp} \leftarrow w_{1temp} - \alpha_1 \beta_1(m) \frac{e^{\frac{policy[code(m)]}{\tau} + w1\beta_1(m) + w2\beta_2(m)}}{z}$
17: $\qquad\quad w_{2temp} \leftarrow w_{2temp} - \alpha_2 \beta_2(m) \frac{e^{\frac{policy[code(m)]}{\tau} + w1\beta_1(m) + w2\beta_2(m)}}{z}$
18: $\qquad$ **end for**
19: $\qquad state \leftarrow play(state, b)$
20: $\quad$ **end for**
21: $\quad policy \leftarrow polp$

Optimizations for GNRPA exist and are presented in[6]. A new optimization inspired by the previous ones is presented below.

### 2.2.1 Avoid recomputing the biases

In some cases, the computation of the bias for all possible moves can be costly. In the same way as the optimization presented in [6], we avoid recomputing all the possible moves by storing the values of the bias in a $\beta$ matrix during the playout function.

The biases of the possible moves have already been calculated during the playout that found the best sequence. The optimized playout algorithm memorizes in a matrix code the biases of the possible moves during each step of the sequence construction in the playout function.

# 3 Experimental Results

We now present experiments with bias weights learning for 3D Bin Packing and Vehicle Routing.

## 3.1 3D Bin Packing

The 3D Bin Packing Problem is a combinatorial optimization problem in which we have to store a set of boxes into one or several containers. The goal is to minimize the unused space in the containers and put the greatest possible number of items into each of them, or, alternatively, to minimize the number of container used to store all the boxes.

We based our experiments on the problem modeled in the paper [30].
We kept the same capacity for the unique container and the same intervals for the items dimensions. However, as opposed to the paper, we worked on the offline variation of 3D Bin Packing, where the set of boxes are known a priori and taken into account in a given order. Also, the boxes dimensions are considered to be integers.

### 3.1.1 Heuristics

We used two heuristics proposed in the paper cited above. The first heuristic is the Least Surface Area Heuristic (LSAH) that aims to minimize the surface area of the Bin that could hold all the items that we need to pack. The candidates are selected in the structured coordinates (Empty Maximal Space-EMS). It is described by a linear program detailed in the following article [19].

The second one is the Heightmap Minimization (HM) heuristic which is described in the following article [29]. It aims at minimizing the volume increase of the object pile as observed from the loading direction.

### 3.1.2 The bias

The BLGNRPA uses these two heuristics to compute the bias using the following formula: $1/Score - of - the - heuristic$ and updating the weights of each move.

Its purpose is to adapt itself to the current situation of the problem and make it easier to choose the next legal move through learning with the bias. It enables having a priority on moves given the current state.

### 3.1.3 Modeling the problem

To represent each possible move, we use the coordinates (x, y) where the bottom-left corner of the lower side of the object will be placed. To encode the rotation, we use the

Table 1: Results of LSAH, HM, NRPA,GNRPA and BLGNRPA on the 3D Packing problem

| Method/Set | $w_1$ | $w_2$ | Set1 | | Set2 | | Set3 | | Set4 | | Set5 | | Set6 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Uti. | N | Uti. | N | Uti. | N | Uti. | N | Uti. | N | Uti. | N |
| LSAH | | | 0.502 | 39 | 0.527 | 15 | 0.623 | 27 | 0.675 | 24 | 0.431 | 15 | 0.641 | 30 |
| Heightmap | | | 0.502 | 39 | 0.463 | 14 | 0.623 | 27 | 0.738 | 27 | 0.836 | 31 | 0.565 | 27 |
| NRPA | | | 0.743 | 46 | 0.836 | 27 | 0.843 | 38 | 0.852 | 31 | **0.868** | **33** | 0.807 | 37 |
| GNRPA | 1.00 | 1.00 | 0.796 | 48 | 0.836 | 27 | **0.887** | **41** | 0.852 | 31 | **0.868** | **33** | 0.807 | 37 |
| BLGNRPA | 1.00 | 1.00 | **0.808** | **50** | **0.916** | **28** | **0.887** | **41** | **0.913** | **33** | **0.868** | **33** | 0.807 | 37 |
| GNRPA | 2.68 | 10.84 | **0.808** | **50** | 0.836 | 27 | **0.887** | **41** | 0.852 | 31 | **0.868** | **33** | 0.807 | 37 |
| BLGNRPA | 2.68 | 10.84 | **0.808** | **50** | **0.916** | **28** | **0.887** | **41** | **0.913** | **33** | **0.868** | **33** | **0.892** | **39** |

dimensions of the object along the three axes (x, y and z) for every possible rotation.

#### 3.1.4 Results

The results are shown in the Table 1. The first column of each set refers to the utilization ratio of the container and the second one to the number of boxes that were put in it. NRPA, GNRPA and BLGNRPA outperform LSAH and Heightmap heuristics across all instances. GNRPA obtained better scores than NRPA on 2 instances (Set 1 and 3) and the same score on the other 4 when using $w_1 = 1$ and $w_2 = 1$. With this initialization of the bias weights, BLGNRPA improves the results of GNRPA on 3 instances (Set 1,2 and 4) and obtains the same score on the 3 others. The final average bias weights found by BLGNRPA over all instances ($w_1 = 2.68$ and $w_2 = 10.84$) were then used to initialize GNRPA and BLGNRPA (line 6 and 7 in Table 1). First, we can see that the use of the average of the weights found by BLGNRPA improved the GNRPA score on one of the sets (Set 1).As for the initialization of the weights of the bias to 1, BLGNRPA (with $w_1 = 2.68$ and $w_2 = 10.84$) performs better than GNRPA (with $w_1 = 2.68$ and $w_2 = 10.84$) on 3 sets (Set 2,4 and 6) and obtains the same score on the 3 others. Finally, using the average bias weights for BLGNRPA improves the results for the last set (Set 6). This suggests that using better starting weights improves the results of the algorithm.

### 3.2 The Vehicle Routing Problem

The Vehicle Routing Problem is one of the most studied optimization problems. It was first introduced in 1959 by G.B. Dantzig and J.H. Ramser in ”The Truck Dispatching Problem” [14]. The goal is to find a set of optimal paths given a certain number of delivery vehicles, a list of customers or places of intervention as well as a set of constraints. We can therefore see this problem as an extension of the traveling salesman problem. In its simplest version, all vehicles leave from the same depot. The goal is then to minimize an objective function, generally defined by these 3 criteria given in order of importance: the number of customers that were not serviced, the number of vehicles used, and finally the total distance traveled by the whole set of vehicles. These 3 criteria may be assigned specific weights in the objective function, or a lexicographic order can be taken into account. The vehicle routing problem is NP-hard, so there is no known algorithm able to solve any instance of this problem in polynomial time.

Although exact methods such as Branch and Price exist, approximate methods like Monte-Carlo Search are nonetheless useful for solving difficult instances. Many companies with a fleet of vehicles find themselves faced with the vehicle routing problem [9]. Many variations of the vehicle routing problem have therefore appeared through the years. This paper focuses on the CVRPTW which adds a demand to each customer (e.g., the number of parcels they have purchased) and a limited carrying capacity for all vehicles. Each customer also have a time window in which he must be served. The depot also has a time window, thus limiting the duration of the tour.

#### 3.2.1 Solomon Instances

This work uses the 1987 Solomon instances [28] for the CVRPTW problem. Solomon instances are the main benchmark for CVRPTW to evaluate the different algorithms. The benchmark is composed of 56 instances, each of them consisting of a depot and 100 customers with coordinates included in the interval [0,100]. Vehicles start their tours with the same capacity defined in the instance. A time window is defined for each client as well as for the depot. The distances and the durations correspond to the Euclidean distances between the geometric points.

The Solomon problems are divided into six classes, each having between 8 and 12 instances. For classes C1 and C2 the coordinates are cluster based while classes R1 and R2 coordinates are generated randomly. A mixture of cluster and random structures is used for the problems of classes RC1 and RC2. The R1, C1, and RC1 problem sets have a short time horizon and only allow a few clients per tour (typically up to 10). On the other hand, the sets R2, C2 and RC2 have a long time horizon, allowing many customers (more than 30) to be serviced by the same vehicle.

#### 3.2.2 Use of the bias

In this paper, we used the dynamic bias introduced in [27]. It is made up of 3 parts. First, the distance, like previous works [1]. Second, the waiting time on arrival. Third, the ”lateness”. This consists in penalizing an arrival too early in a time window. The formula used for the bias is thus :

$$\beta_{ic} = w_1 * \beta_{distance} + w_2 * \beta_{waiting} + w_3 * \beta_{lateness},$$

with $w_1, w_2, w_3 > 0$.

$$\beta_{distance} = \frac{-d_{ij}}{max_{kl}(d_{kl})}$$

$$\beta_{lateness} = \frac{-(dd_j - max(d_ij + vt, bt_j))}{biggest\ time\ window}$$

$$\beta waiting = 0\ if\ vt + d_{ij} > bt_j$$

$$\beta waiting = \frac{-(bt_j - (d_{ij} + vt))}{biggest\ time\ window}\ if\ i \neq depot$$

$$\beta waiting = \frac{-(bt_j - max(ftw, d_{ij} + vt))}{biggest\ time\ window}\ if\ i = depot$$

where $d_{ij}$ is the distance between customer $i$ and $j$, $bt_j$ is the beginning of customer $j$ time window, $dd_j$ the end of customer $j$ time window, $vt$ is the departure instant and $ftw$ is the beginning of the earliest time window. In the previous formulas, using $-value$ instead of $\frac{1}{value}$ enables zero values for the waiting time or the lateness. To avoid too much influence from $\beta_{waiting}$ at the start of the tour, where the waiting time can be big, we used $max(ftw, d_{ij} + vt)$. The idea is to only take into account the "useful time" lost. The underlying principle behind learning the bias weights for VRP is to increase the importance of the different criteria depending on the instance. For some instances, distance is the major factor (if for example the time windows are very large). For others, the emphasis will be put on wasted time in order to reduce the number of cars needed. The idea is therefore to adapt the bias formula to make it more relevant for the corresponding instance.

It should be noted that since the bias is dynamic, it is necessary to calculate it many times. As a result, the bias must be updated quickly to reduce the impact on the running time.

#### 3.2.3 Results

In this section, the parameters used for testing NRPA, GNRPA and BLGNRPA are 3 levels, $\alpha = 1$ and 100 iterations per level. For BLGNRPA, $\alpha_1$, $\alpha_2$ and $\alpha_3$ (for $\beta_{distance}$, $\beta_{waiting}$ and $\beta_{lateness}$) are all initially set to 1 and the bias weights are learned at all level. We compare BLGNRPA with all the bias weights initially set to 0 (that we denote BLGNRPA(0)) with NRPA (GNRPA with all the weights set to 0). We also compare BLGNRPA with weights initialized to the vector of weights W (that we denote BLGNRPA(w)) and GNRPA. For both algorithm, the weights are either initialized or set to the values used in [27]. The weights used are therefore $w_1 = 15$, $w_2 = 75$ and $w_3 = 10$. The results given in table 2 are the best runs out of 10 with different seeds. The running times for NRPA, GNRPA, BLGNRPA are close to each other and smaller than 1800 seconds.

We also compare our results with the OR-Tools library. OR-Tools is a Google library for solving optimization problems. It can solve many types of VRP problems, including CVRPTW. OR-Tools offers different choices to build the $first\ solution$. In our experiments, we used "PATH_CHEAPEST_ARC" parameter. Starting from a start node, the algorithm connects it to the node which produces the cheapest route segment, and iterates the same process from the last node added. Then, OR-Tools uses local search in order to improve the solution. Several options are also possible here. We used the "GUIDED_LOCAL_SEARCH", which guides local search to escape local minima. OR-Tools is run for 1800 seconds on each problem.

The results are compared with the lexicographical approach, first taking into account the number of vehicles used $NV$ and then the total distance traveled $Km$. The best score among the different algorithms is put in bold and when the best known score is obtained an asterisk is added at the end of the vehicle number.

BLGNRPA(0) performed better than NRPA on all instances. NRPA only obtained the same result for the easiest instance c101, for which NRPA and BLGNRPA(0) got the best known solution. NRPA obtained the best known solution only on instance c101 while BLGNRPA(0) got the best known solution for 10 instances and the best results among all algorithms for 17 of the 56 instances. BLGNRPA(0) obtained similar or even better results than GNRPA and BLGNRPA(W) on some instances such as most of the C-type instances, on instances r101, r111, etc. However, it also gets much worse results for other instances (r107, rc201,...). For some instances, the initialization of the bias weights with $w_1 = 15$, $w_2 = 75$ and $w_3 = 10$, that we found manually thanks to many tries, is clearly not the most suitable. For example, on instance r101, previous experiments showed that a good set of weights for the instance is $w_1 = 20$, $w_2 = 20$ and $w_3 = 0$. Interestingly, for BLGNRPA(0) the weights at the end of the best run are $w_1 = 21.07$, $w_2 = 19.44$ and $w_3 = 1.42$. However, we could not guarantee the same favorable behavior on all instances, due to the random aspect of the algorithm, and also due to the amount of time required by the learning process in order to reach appropriate weights values.

We observe that BLGNRPA has a better score than GNRPA on 36 instances, the same score on 12 instances and a worse one on 8 instances. Therefore, learning the weights of the bias seems to improve the results of GNRPA. OR-Tools obtains a better result on the majority of the instances. However, GNRPA got a better score than OR-Tools for 12 instances and an equivalent score for 9 of them, BLGNRPA(W) got a better score than OR-Tools for 18 instances and also an equivalent score for 9 of them.

We can see that for both GNRPA and BLGNRPA the results are better on instances of class R1 and RC1 than on instances of class R2 and RC2. Instances of classes R2 and RC2 have their time window constraints weakened (larger time windows). Whenever dealing with weak constraints, local search performs better than Monte Carlo search with or without the learning of the bias weights. This observation was also made in [11] [27].

# 4 Discussion

The use of a bias in the softmax has some similarities with the formula used in Ant Colony Optimization (ACO) [21, 25, 24, 31] since a priori knowledge of a fixed bias associated to actions is also used in ACO with a kind of softmax. The originality of our approach is that we learn the parameter that multiplies the prior bias associated to actions, dynamically and on each instance. We also provide a theoretical and mathematical derivation of the way the parameters of the bias are updated.

Different kind of algorithms are used for different variations on the VRP. For example, in the recent DIMACS challenge on VRP, the number of vehicles was not taken into account to evaluate solutions. This makes difficult a fair comparison with our algorithm (the total distance of the tours might be reduced with one more vehicle).

# 5 Conclusion

In this paper, we introduced a new method to learn the bias weights for the GNRPA algorithm with BLGNRPA. This new method partially removes the need to choose hand-picked weights for GNRPA. However, GNRPA and BLGNRPA have several limitations. First they are less efficient on weakly constrained problems as we presented

Table 2: The different algorithms tested on the 56 standard instances

| | NRPA | | BLGNRPA(0) | | GNRPA | | BLGNRPA(w) | | OR-Tools | | Best Known | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instances | NV | Km | NV | Km | NV | Km | NV | Km | NV | Km | NV | Km |
| c101 | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | 10 | 828.94 |
| c102 | 10 | 1,011.40 | 10 | 843.57 | 10 | 843.57 | 10 | 843.57 | **10*** | **828.94** | 10 | 828.94 |
| c103 | 10 | 1,105.10 | 10 | 844.86 | 10 | 843.02 | 10 | 828.94 | **10*** | **828.06** | 10 | 828.06 |
| c104 | 10 | 1,112.66 | 10 | 831.88 | 10 | 839.96 | **10** | **828.94** | 10 | 846.83 | 10 | 824.78 |
| c105 | 10 | 896.93 | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | 10 | 828.94 |
| c106 | 10 | 853.76 | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | 10 | 828.94 |
| c107 | 10 | 891.22 | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | 10 | 828.94 |
| c108 | 10 | 1006.69 | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | **10*** | **828.94** | 10 | 828.94 |
| c109 | 10 | 962.35 | **10** | **834.85** | **10** | **834.85** | 10 | 836.60 | 10 | 857.34 | 10 | 828.94 |
| c201 | 4 | 709.75 | **3*** | **591.56** | **3*** | **591.56** | **3*** | **591.56** | **3*** | **591.56** | 3 | 591.56 |
| c202 | 4 | 929.93 | 3 | 609.23 | 3 | 611.08 | 3 | 611.08 | **3*** | **591.56** | 3 | 591.56 |
| c203 | 4 | 976.00 | 3 | 599.33 | 3 | 611.79 | 3 | 605.58 | **3** | **594.23** | 3 | 591.17 |
| c204 | 4 | 995.19 | 3 | 595.65 | 3 | 614.50 | 3 | 597.74 | **3** | **593.82** | 3 | 590.60 |
| c205 | 3 | 702.05 | **3*** | **588.88** | **3*** | **588.88** | **3*** | **588.88** | **3*** | **588.88** | 3 | 588.88 |
| c206 | 4 | 773.28 | **3*** | **588.49** | **3*** | **588.49** | **3*** | **588.49** | **3*** | **588.49** | 3 | 588.49 |
| c207 | 4 | 762.73 | 3 | 592.50 | 3 | 592.50 | 3 | 592.50 | **3*** | **588.29** | 3 | 588.29 |
| c208 | 3 | 741.98 | **3*** | **588.32** | **3*** | **588.32** | **3*** | **588.32** | **3*** | **588.32** | 3 | 588.32 |
| r101 | 19 | 1,660.01 | **19*** | **1,650.80** | **19*** | **1,650.80** | 19 | 1,654.67 | 19 | 1,653.15 | 19 | 1,650.80 |
| r102 | 17 | 1,593.73 | 17 | 1,499.20 | 17 | 1,508.83 | 17 | 1,501.11 | **17** | **1489.51** | 17 | 1,486.12 |
| r103 | 14 | 1,281.89 | 14 | 1,235.31 | 13 | 1,336.86 | 13 | 1,321.17 | **13** | **1,317.87** | 13 | 1,292.68 |
| r104 | 11 | 1,098.30 | 10 | 1,000.52 | 10 | 1,013.62 | **10** | **996.61** | 10 | 1,013.23 | 9 | 1,007.31 |
| r105 | 15 | 1,436.75 | 14 | 1,386.07 | **14** | **1,378.36** | 14 | 1385.76 | 14 | 1,393.14 | 14 | 1,377.11 |
| r106 | 12 | 1,364.09 | 12 | 1,269.82 | 12 | 1,274.47 | **12** | **1,265.97** | 13 | 1,243.0 | 12 | 1,252.03 |
| r107 | 11 | 1,241.15 | 11 | 1,079.96 | 10 | 1,131.19 | 10 | 1,132.95 | **10** | **1,130.97** | 10 | 1,104.66 |
| r108 | 11 | 1,106.14 | 10 | 953.15 | 10 | 990.18 | **10** | **941.74** | 10 | 963.4 | 9 | 960.88 |
| r109 | 12 | 1,271.13 | 12 | 1,173.57 | 12 | 1,180.09 | **12** | **1,171.70** | 12 | 1,175.48 | 11 | 1,194.73 |
| r110 | 12 | 1,232.03 | 11 | 1,116.64 | 11 | 1,140.22 | **11** | **1,094.84** | 11 | 1,125.13 | 10 | 1,118.84 |
| r111 | 12 | 1,200.37 | **11** | **1,071.84** | 11 | 1,104.42 | 11 | 1,073.74 | 11 | 1,088.01 | 10 | 1,096.72 |
| r112 | 10 | 1,162.47 | **10** | **965.43** | 10 | 1,013.50 | 10 | 974.56 | 10 | 974.65 | 9 | 982.14 |
| r201 | 5 | 1,449.95 | 5 | 1,250.16 | 4 | 1,316.27 | 4 | 1,293.38 | **4** | **1,260.67** | 4 | 1,252.37 |
| r202 | 4 | 1,335.96 | 4 | 1,124.91 | 4 | 1,129.89 | 4 | 1,122.80 | **4** | **1,091.66** | 3 | 1,191.70 |
| r203 | 4 | 1,255.78 | 4 | 930.58 | 3 | 1,004.49 | 3 | 970.45 | **3** | **953.85** | 3 | 939.50 |
| r204 | 3 | 1,074.37 | 3 | 765.47 | 3 | 787.69 | 3 | 772.22 | **3** | **755.01** | 2 | 852.52 |
| r205 | 4 | 1,299.84 | 3 | 1,047.53 | 3 | 1,043.81 | 3 | 1,052.15 | **3** | **1,028.6** | 3 | 994.43 |
| r206 | 3 | 1,270.89 | 3 | 982.50 | 3 | 990.88 | 3 | 959.89 | **3** | **923.1** | 3 | 906.14 |
| r207 | 3 | 1,215.47 | 3 | 871.66 | 3 | 900.17 | 3 | 878.91 | **3** | **832.82** | 2 | 890.61 |
| r208 | 3 | 1,027.12 | 3 | 726.34 | 2 | 779.25 | 2 | 737.50 | **2** | **734.08** | 2 | 726.82 |
| r209 | 4 | 1,226.67 | 3 | 954.02 | 3 | 981.82 | 3 | 960.40 | **3** | **924.07** | 3 | 909.16 |
| r210 | 4 | 1,278.61 | 3 | 970.30 | 3 | 995.50 | 3 | 991.87 | **3** | **963.4** | 3 | 939.37 |
| r211 | 3 | 1,068.35 | 3 | 821.79 | 3 | 850.33 | 3 | 798.84 | **3** | **786.28** | 2 | 885.71 |
| rc101 | 15 | 1,745.99 | 15 | 1,636.50 | **14** | **1,702.68** | 15 | 1,636.50 | 15 | 1,639.54 | 14 | 1,696.95 |
| rc102 | 14 | 1,571.50 | 13 | 1,497.11 | 13 | 1,509.86 | **13** | **1,496.16** | 13 | 1,522.89 | 12 | 1,554.75 |
| rc103 | 12 | 1,400.54 | **11** | **1,265.80** | 11 | 1,287.33 | 11 | 1,273.28 | 12 | 1,322.84 | 11 | 1,261.67 |
| rc104 | 11 | 1,264.53 | 10 | 1,147.69 | 10 | 1,160.55 | **10** | **1,146.36** | 10 | 1,155.33 | 10 | 1,135.48 |
| rc105 | 15 | 1,620.43 | **14** | **1,553.43** | 14 | 1,587.41 | 14 | 1,563.18 | 14 | 1,614.98 | 13 | 1,629.44 |
| rc106 | 13 | 1,486.81 | **12** | **1,385.21** | 12 | 1,397.55 | 12 | 1,388.80 | 13 | 1,401.73 | 11 | 1,424.73 |
| rc107 | 12 | 1,338.18 | 11 | 1,238.04 | 11 | 1,247.80 | **11** | **1,233.76** | 11 | 1,255.62 | 11 | 1,230.48 |
| rc108 | 11 | 1,286.88 | **10** | **1,150.68** | 10 | 1,213.00 | 10 | 1152.61 | 11 | 1,148.16 | 10 | 1,139.82 |
| rc201 | 5 | 1,638.08 | 5 | 1,354.84 | 4 | 1,469.50 | 4 | 1,469.16 | **4** | **1,424.01** | 4 | 1,406.94 |
| rc202 | 4 | 1,593.54 | 4 | 1,260.11 | 4 | 1,262.91 | 4 | 1,203.10 | **4** | **1,161.82** | 3 | 1,365.65 |
| rc203 | 4 | 1,431.32 | 4 | 1,010.99 | 3 | 1,123.45 | 3 | 1,141.27 | **3** | **1,095.56** | 3 | 1,049.62 |
| rc204 | 3 | 1,260.05 | 3 | 841.48 | 3 | 864.24 | 3 | 822.39 | **3** | **803.06** | 3 | 789.46 |
| rc205 | 5 | 1,578.73 | 4 | 1,359.74 | 4 | 1,347.86 | 4 | 1,333.95 | **4** | **1,315.72** | 4 | 1,297.65 |
| rc206 | 4 | 1,412.26 | 3 | 1,294.77 | 3 | 1,208.52 | 3 | 1,246.48 | **3** | **1,157.2** | 3 | 1,146.32 |
| rc207 | 4 | 1,395.02 | 4 | 1,066.06 | 3 | 1,164.99 | 3 | 1,124.15 | **3** | **1,098.61** | 3 | 1,061.14 |
| rc208 | 3 | 1,182.55 | 3 | 911.34 | 3 | 948.82 | 3 | 906.01 | **3** | **843.02** | 3 | 828.14 |

in the results section. In addition, GNRPA/BLGNRPA are designed for complete information problems. Finally, the bias must be simple to compute. Indeed, for a GNRPA/BLGNRPA search, the bias must be calculated $100^{level} \times \bar{c}$ times, where $\bar{c}$ is the average number of moves considered in the playout function. In order to have a fast and efficient search, the computation of the bias must therefore be fast.

The results we obtained show that the learning of the bias improves the solutions for GNRPA. For 3D Bin Packing, BLGNRPA got a better score on 3 out of the 6 sets and the same score on the 3 others. For VRP, BLGNRPA provided better solutions than GNRPA on 36 out of the 56 instances, the same score on 12 instances and a worse one on 8 instances. For both problems, it seems that having the bias parameters already initialized with good values for BLGNRPA improves the results compared to initializing the values to 0 or 1.

These preliminary results look promising, so in future work we plan to test some enhancements of the GNRPA on BLGNRPA such as the Stabilized GNRPA (SGNRPA). Similarly to the Stabilized NRPA, in SGNRPA the Adapt function is not systematically run after each level 1 playout, but with an appropriate periodicity. Finally, we plan to work on finding better values for the bias weights initialization, possibly by running a preliminary phase consisting of solely learning the bias weights but not the BLGNRPA policy.

# 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

[1] Ashraf Abdo, Stefan Edelkamp, and Michael Lawo. Nested rollout policy adaptation for optimizing vehicle selection in complex vrps. pages 213–221, 11 2016.

[2] Bruno Bouzy and Tristan Cazenave. Computer go: An AI oriented survey. *Artificial Intelligence*, 132(1):39–103, 2001.

[3] 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.

[4] Bernd Brügmann. Monte Carlo Go. Technical report, Max-Planke-Inst. Phys., Munich, 1993.

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

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

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

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

[9] Tristan Cazenave, Jean-Yves Lucas, Thomas Triboulet, and Hyoseok Kim. Policy adaptation for vehicle routing. *AI Communications*, 2021.

[10] 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*, pages 42–54, 2012.

[11] M. Cornu. Local search, data structures and monte carlo search for multi-objective combinatorial optimization problems. (recherche locale, structures de données et recherche monte-carlo pour les problèmes d'optimisation combinatoire multi-objectif). 2017.

[12] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. Donkers, editors, *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2006.

[13] Chen Dang, Cristina Bazgan, Tristan Cazenave, Morgan Chopin, and Pierre-Henri Wuillemin. Monte carlo search algorithms for network traffic engineering. In *ECML PKDD*, volume 12978 of *LNCS*, pages 486–501, 2021.

[14] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[15] 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.

[16] 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.

[17] 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.

[18] Tobias Graf and Marco Platzner. Adaptive playouts in monte-carlo tree search with policy-gradient reinforcement learning. In *Advances in Computer Games - 14th International Conference, ACG 2015, Leiden, The Netherlands, July 1-3, 2015, Revised Selected Papers*, pages 1–11, 2015.

[19] Haoyuan Hu, Xiaodong Zhang, Xiaowei Yan, Longfei Wang, and Yinghui Xu. Solving a new 3d bin packing problem with deep reinforcement learning method. *arXiv:1708.05930*, 7, aug 2017.

[20] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *17th European Conference on Machine Learning (ECML'06)*, volume 4212 of *LNCS*, pages 282–293. Springer, 2006.

[21] Vittorio Maniezzo, Luca Maria Gambardella, and Fabio De Luigi. Ant colony optimization. In *New optimization techniques in engineering*, pages 101–121. Springer, 2004.

[22] 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.

[23] Fernando Portela. An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem. *BioRxiv*, page 345587, 2018.

[24] Chengming Qi and Yunchuan Sun. An improved ant colony algorithm for vrptw. In *2008 International Conference on Computer Science and Software Engineering*, volume 1, pages 455–458. IEEE, 2008.

[25] Andrea Emilio Rizzoli, Roberto Montemanni, Enzo Lucibello, and Luca Maria Gambardella. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(2):135–151, 2007.

[26] Christopher D. Rosin. Nested rollout policy adaptation for Monte Carlo Tree Search. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 649–654, 2011.

[27] Julien Sentuc, Tristan Cazenave, and Jean-Yves Lucas. Generalized nested rollout policy adaptation with dynamic bias for vehicle routing. In *AI for Transportation at AAAI*, 2022.

[28] Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. In *Operations Research*, 1985.

[29] Fan Wang and Kris Hauser. Stable bin packing of non-convex 3d objects with a robot manipulator. *arXiv:1812.04093v1*, 9, dec 2018.

[30] Hang Zhao, Yang Yu, and Kai Xu. Learning efficient online 3d bin packing on packing configuration trees. In *International Conference on Learning Representations*, 2022.

[31] Tong Zhen, Qiuwen Zhang, Wenshuai Zhang, and Zhi Ma. Hybrid ant colony algorithm for the vehicle routing with time windows. In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, volume 1, pages 8–12. IEEE, 2008.