AI Communications 0 (2020) 1
IOS Press

Policy Adaptation for Vehicle Routing

Tristan Cazenave [a,*], Jean-Yves Lucas [b], Thomas Triboulet [b] and Hyoseok Kim [b]
[a] *LAMSADE, Université Paris-Dauphine, PSL, CNRS, France*
E-mail: Tristan.Cazenave@dauphine.psl.eu
[b] *OSIRIS department, EDF Lab Paris-Saclay, Electricité de France, France*
E-mails: jean-yves.lucas@edf.fr, thomas.triboulet@edf.fr, hyoseok.kim@ensiie.fr

Abstract. Nested Rollout Policy Adaptation (NRPA) is a Monte Carlo search algorithm that learns a playout policy in order to solve a single player game. In this paper we apply NRPA to the vehicle routing problem. This problem is important for large companies that have to manage a fleet of vehicles on a daily basis. Real problems are often too large to be solved exactly. The algorithm is applied to standard problem of the literature and to the specific problems of EDF (Electricité De France, the main French electric utility company). These specific problems have peculiar constraints. NRPA gives better result than the algorithm previously used by EDF.

Keywords: Vehicle Routing Problems, Capacitated Vehicle Routing with Time Windows, Nested Rollout Policy Adaptation

1. Introduction

Monte Carlo Tree Search (MCTS) has been successfully applied to many games and problems [1]. Nested Monte Carlo Search (NMCS) [2] is an algorithm that works well for puzzles. 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 [3]. Other applications of NMCS include Single Player General Game Playing [4], Cooperative Pathfinding [5], Software testing [6], heuristic Model-Checking [7], the Pancake problem [8], Games [9], Cryptography [10] and the RNA inverse folding problem [11].

Online learning of a playout policy in the context of nested searches has been further developed for puzzles and optimization with Nested Rollout Policy Adaptation (NRPA) [12]. NRPA has found new world records in Morpion Solitaire and crosswords puzzles. Edelkamp, Cazenave 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 [13, 14]. Other applications deal with 3D Packing with Object Orientation [15], the physical traveling salesman problem [16], the Multiple Sequence Alignment problem [17], Logistics [18, 19], Graph Coloring [20] and Inverse Folding [21]. 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.

Electricité De France (EDF) is the main French electric utility company. Each year, eleven millions of services have to be carried out by EDF technicians at customers places (problem fixing, meter replacement, energetic diagnosis, business prospects). A technician drive by car more than 22,000 kilometers per year. There are about 10,000 EDF technicians servicing customers, thus the total distance travelled by these technicians on a yearly basis is around 220,000,000 kilometers. Each service is accomplished by a technician having the required skills within a time window determined during the appointment booking. As a result, numerous Vehicle Routing Problems with Time Windows (VRPTW) have to be solved on a daily basis (one problem per catchment area). Any of these VRPTW involves quite a big search space, since EDF technicians of a single catchment area have to achieve every day hundreds of visits at customers places. The objective of the work described in this paper was to adapt the NRPA algorithm to the specifics of EDF problem.

Related approaches that apply Artificial Intelligence techniques to transportation include agent based approaches [22], Monte Carlo Search approaches [18, 23–25] and learning of simulations of urban traffic [26]. Our work is close to the Monte-Carlo Search approach applied to the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW).

[*]Corresponding author. E-mail: Tristan.Cazenave@dauphine.psl.eu.

We now give the outline of the paper. The next section briefly describes the state of the art of Vehicle Routing. The third section describe the problems of EDF and the current solver. The fourth section details nrpaDQ, the NRPA algorithm that we implemented, focusing on the proposed improvements from standard NRPA. The fifth section gives experimental results. In subsection 5.1 we provide numerical results obtained with nrpaDQ for various instances of Vehicle Routing, both from literature (Solomon instances) and from real-life problems. In subsections 5.2 to 5.5 we present the results we obtained for some variations of nrpaDQ that we tested on the same set of instances. Finally, the last section concludes.

2. The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one of the most studied optimization problems. It was first stated in 1959 in [27]. Basically, it consists of finding an optimal route for a number of vehicles that are used to deliver goods or services to a set of customers, taking into account a set of constraints. In the simpler variant of the problem, all vehicles start from a single depot and end their route in this depot. The objective function to be minimized may combine up to three criteria (namely the number of customers that are not serviced, the number of vehicles used, the total distance travelled by the vehicles), each criterion having an associated weight. The VRP can be formalized as a graph problem: let G = (V, A) be a directed graph where V is the set of vertices, and A is the set of arcs. Each arc is labelled with a non-negative number. One vertex represents the depot, the others vertices represent the customers locations. The arcs connect vertices, thus they represent the roads between two locations, and the label of the arcs give the associated distance (or the travel duration, or the travel cost). Lets us remind that G is a directed graph, and thus its associated distance matrix is not necessarily symmetric. Now the problem consists of finding the minimum number of routes so that each vertex except the depot belongs to one and only one route, and the depot belongs once to each of them. Figure 1 shows the example of a VRP problem with 3 customers and a depot. In order to simplify the graphic illustration, there is only one arc between two nodes, indicating the travel duration.

The VRP is of the non deterministic polynomial time hard type (NP-Hard), which implies that so far we do not know of a general method which is able to optimally solve any instance of the problem in polynomial time. As a result, the VRP is considered very difficult. Nowadays exact methods can solve to optimality quite challenging instances of the problem (for instance more than one hundred visits with Branch and Price methods).

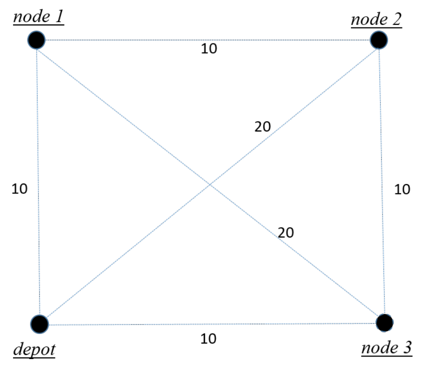


Fig. 1. Example of VRP graph

2.1. VRP variations

The VRP is a real challenge for delivery companies operating a fleet of vehicles. It has given rise to a number of variations. The CVRP (Capacitated VRP) is defined as a VRP with a demand associated to each customers (e.g., the number of parcels they have purchased) and each vehicle has a limited capacity. The VRP with time windows (VRPTW) implies to serve each customer within a given time window (possibly different for each customer). The capacitated VRP with time windows (CVRPTW) combines the characteristics of the two previous variants. Figure 2 gives an illustration of CVRPTW graph with 4 vertices: one depot and 3 customers. Each customer both consumes a quantity q and has a specific duration of service, which must be scheduled in a given time-window.

In this example we assume the that the capacity of a vehicle is 100.

- Figure 3 gives an example of feasible route.
- Figure 4 gives an example of unfeasible route with the same visited customers but with a different order: vehicle arrives at node 1 after a 50 duration including travel and first appointment duration (20+20+10, in bold characters in the figure) which is too late to schedule a duration of 30 for the second service within the [0,50] time window of the customer (in bold characters in the figure).

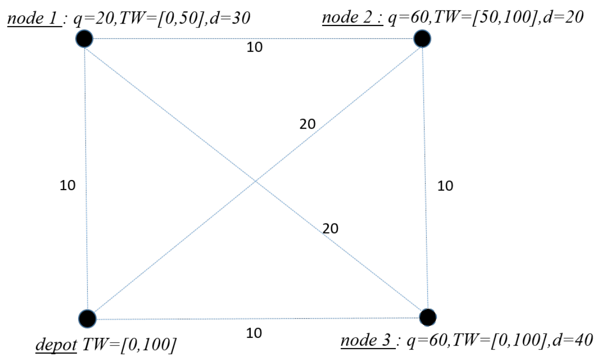

node 1 : q=20,TW=[0,50],d=30
node 2 : q=60,TW=[50,100],d=20
10
20
10
10
20
depot TW=[0,100]
10
node 3 : q=60,TW=[0,100],d=40

Fig. 2. Example of CVRPTW graph

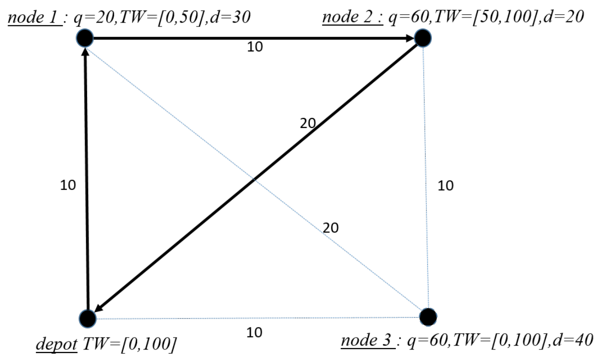

node 1 : q=20,TW=[0,50],d=30
node 2 : q=60,TW=[50,100],d=20
10
20
10
10
20
depot TW=[0,100]
10
node 3 : q=60,TW=[0,100],d=40

Fig. 3. Example of feasible route in a CVRPTW problem

- Figure 5 gives an example of unfeasible route because of capacity reason. With the hypothesis of 100 capacity vehicle, the sum of the quantity needed for the visited customers (60+60, in bold characters in the figure) outsizes the 100 capacity of the vehicle.
- Figure 6 gives an example of unfeasible route because of the time window of the depot. The duration of the route is the sum of service duration and travel duration (110=10+30+20+40+10, in bold characters in the figure) can not be scheduled within the [0,100] time window of the depot (in bold characters in the figure).

Although the previous variations are the most widely studied because of their great practical importance, many other extensions of the basic VRP have also been proposed. Among them, let us just mention two of them. First, the dynamic VRP (DVRP) where vehicles may be dynamically re-routed during their route in order to fulfill new customers orders. Second, the VRP with Pickup and Delivery (VRPPD) where a fleet of vehicles must satisfy transportation requests (e.g., picking parcels up at some places and delivering them at given locations).

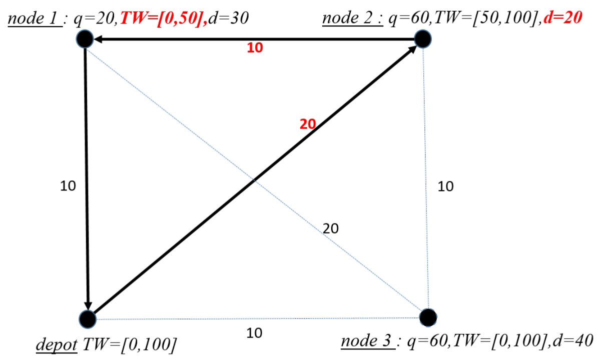

node 1 : q=20,TW=[0,50],d=30
node 2 : q=60,TW=[50,100],d=20
10
20
10
10
20
depot TW=[0,100]
10
node 3 : q=60,TW=[0,100],d=40

Fig. 4. Example of unfeasible route for customer time window reason

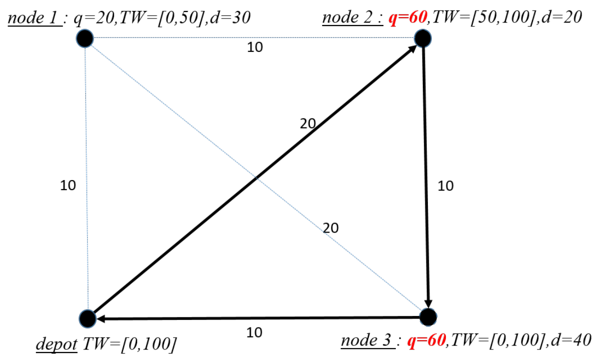

node 1 : q=20,TW=[0,50],d=30
node 2 : q=60,TW=[50,100],d=20
10
20
10
10
20
depot TW=[0,100]
10
node 3 : q=60,TW=[0,100],d=40

Fig. 5. Example of unfeasible route for capacity reason (capacity of vehicle=100)

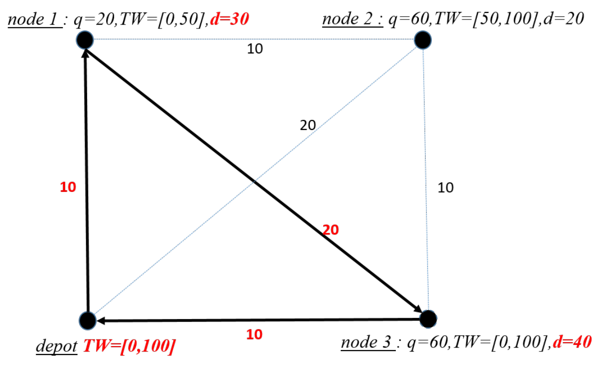

node 1 : q=20,TW=[0,50],d=30
node 2 : q=60,TW=[50,100],d=20
10
20
10
10
20
depot TW=[0,100]
10
node 3 : q=60,TW=[0,100],d=40

Fig. 6. Example of unfeasible route for depot time window reason

2.2. *Principal methods used for solving VRP*

A wide range of methods have been used to solve the VRP. These methods may be broken down into three sub-groups, namely exact methods, heuristic methods and metaheuristic methods. These methods are presented hereafter.

Exact methods. These methods tend to find an optimal solution for the VRP. As mentioned above, they are thus used to solve instances of the VRP of consequent size. Among the most studied exact methods for the VRP and variations we may mention the Branch-and-Cut and the Branch-and-Price algorithms [28], the column-generation algorithm [29] and the set-partitioning method [30].

These methods can also provide a bound of optimum, especially in relaxing some of the most complicated constraints. For example, in suppressing the conventional subtour elimination constraint.

Heuristic methods. Not in a comprehensive manner, let us mention two interesting approaches. First the cluster-first, route-second heuristic [31]. Second, the savings heuristic [32] [33]. For the local search approaches we see in [34] a heuristic projection pool with powerful insertion and guided local search strategies. The local search described in [35] gives reference solutions on several instances of the benchmark VRP Solomon instances tested in this work.

Metaheuristic methods. The most commonly used metaheuristics for solving VRP and variations are the particle swarm optimization [36], simulated annealing [37], genetic algorithm [38] [39] and Tabu search [40] [41]. Among the best algorithms for solving the VRP Solomon instances tested in this work, we can cite genetic algorithm described in [42], evolutionary algorithm detailed in [43] or Tabu search proposed in [44].

3. The EDF Capacitated Vehicle Routing Problem with Time Windows

In this section we first explain the optimization problem encountered at EDF and then describe the current EDF approach to the problem.

3.1. Description of the EDF problems

The Vehicle Routing Problem modeled here includes time windows, which is a classical feature of VRP problems. It means that:

- Each technician has an availability time window. Each route starts at the beginning of this time window and must end before the end of this time window.
- Each appointment has a time window and a duration. A route visiting the appointment must start after the beginning of the time window and end (including the appointment duration) before the end of the time window
- If a route arrives at an appointment location before the beginning of its time windows, it is possible to add waiting time before starting the service.

The problem also includes capacities, another classical feature of VRP problems:

- Each vehicle has several stock capacities (6 per vehicle for the EDF problem). Vehicle stocks are initialized with initial capacity at the beginning of each route.
- For any stock, each operation will consume a stock quantity.
- For each route, for each capacity, the sum of the quantity used by the operation of the route must not be greater than the capacity of the stock.

Several peculiar features are taken into account in EDF modelization.

- Taking into account an off-duty time window for lunch break. No place is imposed for it. It can interrupt a trip but not a service to a customer.
- Taking into account specific skills for visits. For each visit, only a subset of technicians is skilled to carry out the technical operations.
- Journey distance and journey duration are not proportional, because vehicle mean speed depends on the journey itself. Consequently, 2 different distance matrices are provided.
- Distance matrices are not symmetric: the distance between two points depends on the direction of the route.

Another difference with academic data is that on real problems, there is not always enough people to carry out the whole set of visits. Consequently, we must first maximize the number of visits that will be done. Some of the appointments have a high priority, and because of customers satisfaction, it is not possible to cancel it. Other appointments have less priority, because they consist for example in a technical operation that can be postponed, with no corresponding customer appointment. Moreover, network preventive maintenance visits have less priority than troubleshooting activities. In order to evaluate a solution, a lexicographic objective function is taken into account:

- Maximization of the number of achieved visits of high priority.

- Maximization of the number of achieved visits of low priority.
- Minimization of the economic function, taking into account number of technicians used (ponderated with a proportional daily wage) and number of kms (ponderated with a proportional km cost).

Thus, when comparing two solutions, we first compare the number of visits of high priority that are achieved in each solution.if these numbers are not equal then we know which solution is best. If they are equal, then we compare the number of visits of low priority, if again these numbers are equal, then we compare the third criterion of the two solutions, that is the economic function. The problem is solved each day for determining the technicians routes of the next day, with a computing time which must not last more than an few hours.

3.2. Description of current EDF approach

Current approach for the problem is based on stochastic greedy algorithm and variable neighborhood search. The stochastic greedy algorithm is based on [45] algorithm. First phase consist in creating routes and insert in each route the visits that increases the less the travelled distance, until there is no room for new visits. For each insertion, the place in the route is chosen to minimize the increase of kilometers. If no insertion is possible, a new route is created. Then the process is repeated, until no visit and technician is left.

The local search consist of 2 kinds of movements:

- Small movements consist in choosing randomly an appointment, and finding the best place to move the appointment, i.e., the best place in all the existing routes to reinsert the appointment.
- Large movements consist in choosing randomly a route, then entirely destroying it and trying to reinsert all its appointments in the other routes. If no improvement is possible, the movement is canceled.

Advantages of such a method is that it computes rapidly good solutions, very often acceptable by operators because greedy algorithm choices, based on distance, are humanly intuitive.

4. Nested Rollout Policy Adaptation for Vehicle Routing

In this section we start with explaining the NRPA algorithm. Then we give our modelling of the CVRPTW problem for NRPA. We finish the section describing how weight are heuristically initialized in order to speed-up convergence of the algorithm.

4.1. Description of NRPA

An effective combination of nested levels of search [2] and of policy learning has been proposed with the NRPA algorithm [12]. NRPA holds world records for Morpion Solitaire and crosswords puzzles.

NRPA is given in algorithm 3. The principle is to learn weights for the possible actions so as to bias the playouts. The playout algorithm is given in algorithm 1. It performs Gibbs sampling, choosing the actions with a probability proportional to the exponential of their weights.

The adaptive rollout policy is a policy parameterized by weights on each action. During the playout phase, action is sampled according to this weights. The playout algorithm is given in algorithm 1. It uses Gibbs sampling, each move is associated to a weight. A move is coded as an integer that gives the index of its weight in the policy array of floats. The algorithm starts with initializing the sequence of moves that it will play (line 2). Then it performs a loop until it reaches a terminal states (lines 3-6). At each step of the playout it calculates and stores in the z variable the sum of all the exponentials of the weights of the possible moves (lines 7-10) and chooses a move proportional to its probability given by the softmax function (line 11). Then it plays the chosen move and adds it to the sequence of moves (lines 12-13). Then, the policy is adapted on the best current sequence found, by increasing the weight of the best actions.

The Adapt algorithm is given in algorithm 2. Basically, a policy is an array of floats which associates each move (coded by an integer) to its weight (coded by a float). The Adapt algorithm starts with saving in the local variable *polp* the current policy before modifying it. The policy saved in *polp* will be modified by the Adapt function while the current policy will be used to calculate the probabilities of the moves. The global array *code* stores the integers that are the coding associated to each move. After the modifications the policy in *polp* is copied back to the current policy. The weights of the actions are updated at each step of the algorithm so as to favor moves of the best sequence found so far at each level. The principle of the adaptation is to add α to the action of the best sequence for each state encountered in the best sequence (lines 3-5) and to decrease the weights of the other possible actions by an amount proportional to their probabilities of being played (lines 6-12).

In NRPA, each nested level takes as input a policy, and returns a sequence. Inside the level, the algorithm makes many recursive calls to lower levels, providing weights, getting sequences and adapting the weights on those sequences. In the end, the algorithm returns the best sequence found in that level. At the lowest level, the algorithm simply makes a rollout.

The NRPA algorithm is given in algorithm 3. At level zero it simply performs a playout (lines 2-3). At greater levels it performs N iterations and for each iteration it calls itself recursively to get a score and a sequence (lines 4-7). If it finds a new best sequence for the level it keeps it as the best sequence (lines 8-11). Then it adapts the policy using the best sequence found so far at the current level (line 12).

NRPA balances exploitation by adapting the probabilities of playing moves toward the best sequence of the level, and exploration by using Gibbs sampling at the lowest level. It is a general algorithm that has proven to work well for many optimization problems.

Playout policy adaptation has also been used for games such as Go [46] or various other games with success [47].

Algorithm 1 The playout algorithm

```
1:  playout ($state$, $policy$)
2:      $sequence \leftarrow []$
3:      while true do
4:          if $state$ is terminal then
5:              return (score ($state$), $sequence$)
6:          end if
7:          $z \leftarrow 0.0$
8:          for $m$ in possible moves for $state$ do
9:              $z \leftarrow z + \exp(policy[\mathrm{code}(m)])$
10:         end for
11:         choose a $move$ with probability $\frac{exp(policy[code(move)])}{z}$
12:         $state \leftarrow$ play ($state$, $move$)
13:         $sequence \leftarrow sequence + move$
14:     end while
```

4.2. Modeling the problem

There are several design choices when implementing NRPA. The first choice is to decide how to code the possible moves. For the vehicle routing problem we choose to code a move as a starting node, an arrival node and a vehicle. More precisely, a solution to the problem is an ordered sequence of visits, including special visits (SV) that represent the fact that the corresponding technician is at the depot. So a sequence solution always starts and ends with a SV. If there is a SV between two visits within the sequence, that means the end of a route for a technician, and the beginning of a route for another technician (with no chronological ordering of these two routes that will be carried out simultaneously).

Another design choice is to define the score of a playout. Score includes number of non visited customers multiplied by a great penalization number, number of used vehicles multiplied by an rather great weight, and number of kilometers. We reject move-

Algorithm 2 The Adapt algorithm

```
1:  Adapt ($policy$, $sequence$)
2:      $polp \leftarrow policy$
3:      $state \leftarrow root$
4:      for $move$ in $sequence$ do
5:          $polp$ [code($move$)] $\leftarrow$ $polp$ [code($move$)] + $\alpha$
6:          $z \leftarrow 0.0$
7:          for $m$ in possible moves for $state$ do
8:              $z \leftarrow z + \exp(policy[\mathrm{code}(m)])$
9:          end for
10:         for $m$ in possible moves for $state$ do
11:             $polp$ [code($m$)] $\leftarrow$ $polp$ [code($m$)] - $\alpha * \frac{exp(policy[code(m)])}{z}$
12:         end for
13:         $state \leftarrow$ play ($state$, $move$)
14:     end for
15:     $policy \leftarrow polp$
```

Algorithm 3 The NRPA algorithm.

```
1:  NRPA ($level$, $policy$)
2:      if level == 0 then
3:          return playout (root, $policy$)
4:      else
5:          $bestScore \leftarrow -\infty$
6:          for N iterations do
7:              (result,new) $\leftarrow$ NRPA($level - 1$, $policy$)
8:              if result $\geqslant$ bestScore then
9:                  bestScore $\leftarrow$ result
10:                 seq $\leftarrow$ new
11:             end if
12:             policy $\leftarrow$ Adapt (policy, seq)
13:         end for
14:         return (bestScore, seq)
15:     end if
```

ments that do not respect the time windows in the playout algorithm: all solutions respect both customers time windows and vehicle time windows.

If there were more objectives, a lexicographic ordering of the objectives would be the best way to represent the scores of a playout. Playouts could then be compared simply and easily.

4.3. Initialization of the Weights

In NRPA weights are uniformly initialized to 0.0. We propose to heuristically initialize the weights of NRPA in order to speedup convergence. The greater the distance from the current city to another city, the less likely the other city is a good choice. Standard NRPA starts with an uniform policy with all weights set to 0.0 and does not distinguishes between close and far cities. The heuristic initialization of the weights initializes each weight with a value proportional to the inverse of the distance. This initialization is only used for the first iteration of NRPA.

4.4. The Quantile Heuristic

The objective of the Quantile Heuristic is to penalize movements from bad solutions. Solution scores for all playouts need to be stored. When a new playout is computed, if its score is worse than a defined quantile, then its movements weights are decreased. Efficient implementation is needed to limit the increase of computation time needed to sort the solution scores and compute the quantile. Sequence of algorithm is described in 4. The adapt function is the same as for good solutions, with negative coefficient α, as described in 5. The coefficient α used for bad solutions can be different from the one used to adapt policy to good solutions. In the rest of the paper we call nrpaDQ our implementation of NRPA with both the Distance heuristic and the Quantile heuristic.

Algorithm 4 The NRPA algorithm with quantiles.

```
1:  NRPA_with_quantile ($level$, $policy$)
2:    allScores ← []
3:    quantile ← $-\infty$
4:    if level == 0 then
5:      (result,new) ← playout(root,policy)
6:      allScores ← allScores + result
7:      quantile ←updateQuantile(allScores)
8:      return (result,new)
9:    else
10:     $bestScore \leftarrow -\infty$
11:     for N iterations do
12:       (result,new) ← NRPA($level - 1$, $policy$)
13:       if result $\geqslant$ bestScore then
14:         bestScore ← result
15:         seq ← new
16:       else
17:         if result $\leqslant$ quantile then
18:           policy ← Adapt_Bad(policy, new)
19:         end if
20:       end if
21:       policy ← Adapt(policy, seq)
22:     end for
23:     return (bestScore, seq)
24:   end if
```

Algorithm 5 The adapt algorithm to bad solutions

```
1:  Adapt_Bad ($policy$, $sequence$)
2:    $polp \leftarrow policy$
3:    $state \leftarrow root$
4:    for $move$ in $sequence$ do
5:      $polp$ [code($move$)] ← $polp$ [code($move$)] - $\alpha$
6:      $z \leftarrow 0.0$
7:      for $m$ in possible moves for $state$ do
8:        $z \leftarrow z$ + exp ($policy$ [code($m$)])
9:      end for
10:     for $m$ in possible moves for $state$ do
11:       $polp$ [code($m$)] ← $polp$ [code($m$)] + $\alpha * \frac{exp(policy[code(m)])}{z}$
12:     end for
13:     $state$ ← play ($state$, $move$)
14:   end for
15:   $policy \leftarrow polp$
```

5. Experimental Results

The operational process of EDF consists in providing a solution to the CVRPTW within around two hours. Each problem is solved during the night so as to provide solutions for the next day. In this section, the parameters used when testing the 3 variations of NRPA are 3 levels, $\alpha = 1$ and 100 iterations per level (N=100). The standard algorithm is called NRPA in our tables. The nrpaD (NRPA with distance heuristic) is tested and the initialization of the weights is only applied at the first iteration. In the nrpaDQ (NRPA with Distance and Quantile heuristic), the Quantile heuristic is applied on 80% of the worst cases and uses a specific $\alpha = 0.5$. These values have been set thanks to a whole set of simulations done previously. The 3 variations of NRPA have benefited of a root parallelization. Each instance was run 11 times with 11 different random seeds, each time retaining the best of the 11 results.

The current solver of EDF, Opturn first perform a greedy algorithm, then goes on with a local search. These procedures are both parallelized on 6 threads. At the first stage, each thread involves 3,000 iterations of greedy search. The best among the 6 solutions is used as a starting point for the 6 simultaneous local searches, each performing 3,000 iterations. These values were chosen during the tuning of Opturn after many experiments that showed that no improvements are obtained with bigger values. As a result, the running time of Opturn is smaller than the running time of the NRPA.

The running time for NRPA and nrpaD is approximately 7,000 seconds and up to 8,000 seconds when using the Quantile heuristic. The running time of Opturn is approximately 5,000 seconds. Importantly, these run times are compatible with the operational process as they last approximately two hours.

5.1. Main Results

Table 1 gives results on the literature instances. It compares 4 algorithms:

- NRPA: Standard NRPA (3 levels) without heuristics
- nrpaD: NRPA (3 levels) with Distance initialization heuristic
- nrpaDQ: NRPA (3 levels) with Distance initialization heuristic and Quantile heuristic
- Opturn: current EDF solver, based on Solomon and LNS heuristic.

To compare the results of the 4 variants, the lexicographical approach takes into account first the number of vehicles used and then the kilometers.

- V: number of needed vehicles
- Km: sum of the Kms of the routes

Table 2 gives a summary of runs on the standard problems. The Km and the Vehicles (V) columns give the average of objective function values on the Solomon instances. The δ value is the number of problems that are solved better than with standard NRPA minus the number of problems that are solved worse. Comparison is lexicographic, based on number of vehicles, and if equals, on the number of Kms. Logically, the δ value is 0 for the reference (that is, the NRPA variant).

We observe that nrpaDQ scores 55 while Opturn scores -50. This is a great improvement over the current solver. However, it should be stressed that the current solver was optimized on the EDF problems. The Distance heuristic improves a lot over standard NRPA. The Distance + Quantile heuristic only improves slightly over the Distance heuristic alone.

Table 3 and Table 4 give the results of the 4 algorithms on the EDF instances. For each algorithm, the 3 components of objective function are displayed:

- Missed: number of missed appointments
- V: number of needed vehicles
- Km: sum of the Kms of the routes

Table 3 compares the results of runs of the different algorithms on the EDF problems. For each instance we compare the number of realized technical operations because on real EDF instances there are not necessarily enough vehicles to achieve all the operations. We also measure the number of vehicles and the number of kilometers, as in classical VRP problem instances. To compare the results of the 4 variants, the lexicographical approach takes into account first the achieved operations, then the number of vehicles used and finally the kilometers.

Table 4 gives the average values for the whole set of EDF instances.

The average number of missed visits is smaller with NRPA than with Opturn. It is the same for all NRPA variants. The number of vehicles used for the operations is also slightly smaller and this is the same for all NRPA variants. The average number of kilometers is greater with standard NRPA than with Opturn, however it is smaller with nrpaD and nrpaDQ than with Opturn. With respect to the lexicographic objective function described in subsection 4.1, we can conclude that nrpaD and nrpaDQ improves on the current solver for the real-life EDF instances.

Table 1

The different algorithms tested on the 56 standard instances.

	NRPA		nrpaD		nrpaDQ		Opturn	
instance	V	Km	V	Km	V	Km	V	Km
c101	**10**	**828.94**	**10**	**828.94**	**10**	**828.94**	10	852.95
c102	10	986.77	10	858.18	**10**	**843.57**	11	1,458.24
c103	10	1,117.30	10	872.45	**10**	**857.14**	11	1,674.76
c104	10	1,120.67	**10**	**894.83**	10	901.79	11	1,640.93
c105	11	900.69	**10**	**828.94**	**10**	**828.94**	11	1,102.17
c106	11	940.90	**10**	**828.94**	**10**	**828.94**	11	1,465.60
c107	10	1,071.41	**10**	**828.94**	**10**	**828.94**	10	994.84
c108	10	1,104.49	**10**	**830.16**	10	831.04	11	1674.53
c109	10	1,061.88	**10**	**842.43**	10	849.09	11	1,688.91
c201	4	723.69	**3**	**591.56**	**3**	**591.56**	4	974.46
c202	4	870.43	3	611.08	**3**	**609.23**	4	952.33
c203	4	1,108.55	**3**	**617.06**	3	628.40	4	1,058.52
c204	4	1,043.12	3	773	**3**	**629.67**	4	1,265.41
c205	4	720.12	3	591.17	**3**	**588.88**	4	1,007.44
c206	4	753.90	**3**	**590.79**	3	600.88	4	1,252.91
c207	4	733.45	**3**	**597.53**	**3**	**597.53**	4	1,032.26
c208	4	717.03	**3**	**601.05**	3	604.36	4	1,189.83
r101	19	1,665.61	19	1,656.54	**19**	**1,655.36**	22	2,187.97
r102	18	1,520.63	17	1,546.12	**17**	**1,523.75**	19	1,967.61
r103	14	1,331.52	14	**1,287.66**	**13**	1,379.44	17	1,887.31
r104	11	1,113.42	11	**1,074.57**	**10**	1,112.86	13	1,524.18
r105	15	1,467.54	**14**	**1,439.11**	14	1,456.76	17	1,931.07
r106	13	1,346.79	**12**	1,313.12	13	**1,301.35**	15	1,687.18
r107	12	1,222.77	11	1,163.35	**11**	**1,158.45**	14	1,621.52
r108	11	1,142.77	10	1,008.88	**10**	**1,004.15**	12	1,414.54
r109	13	1,290.98	**12**	**1,218.86**	12	1,220.98	16	1,866.01
r110	12	1,236.98	11	1,159.06	**11**	**1,158.85**	13	1,625.52
r111	12	1,181.52	**11**	**1,125.77**	11	1,189.42	14	1,659.14
r112	11	1,105.92	**10**	**1,018.92**	10	1,023.14	12	1,513.96
r201	4	1,528.10	**4**	**1,419.93**	4	1,445.68	5	1,889.49
r202	4	1,369.36	**4**	**1,264.17**	4	1,305.25	6	1,871.68
r203	4	1,276.17	3	1,198.43	**3**	**1,137.19**	4	1,632.11
r204	3	1,086.88	**3**	**879.22**	3	890.48	4	1,127.19
r205	4	1,226.37	3	1,182.85	**3**	**1,174.74**	4	1,475.82
r206	3	1,230.22	3	1,114.39	**3**	**1,098.25**	4	1,357.13
r207	3	1,243.13	3	1,017.76	**3**	**975.49**	3	1,327.14
r208	3	1,048.79	2	860.49	**2**	**859.89**	3	1,076.75
r209	4	1,191.46	3	1,136.35	**3**	**1,055.18**	4	1,495.34
r210	4	1,238.99	**3**	**1,117.78**	3	1,151.20	4	1,588.28
r211	3	1,059.63	3	919.90	**3**	**895.23**	3	1,239.46
rc101	16	1,701.31	**15**	**1,625.45**	15	1,640.51	19	2,483.55
rc102	14	1,560.96	**13**	1,549.49	14	**1,511.67**	16	2,261.19
rc103	12	1,426.39	**11**	**1,321.51**	11	1,356.68	14	1,952
rc104	11	1,264.83	10	1,202.19	**10**	**1,165.98**	13	1,686.93
rc105	15	1,610.40	14	1,652.48	**14**	**1,634.49**	19	2,343.03
rc106	12	1,451.27	13	**1,417.10**	**12**	1,428.08	14	1,952.11
rc107	12	1,404.08	11	1,312.12	**11**	**1,289.70**	14	1,755.26
rc108	11	1,380.20	11	1,197.96	**11**	**1,188.93**	14	1,888.36
rc201	5	1,695.44	4	1,675.35	**4**	**1,644.26**	5	2,537.02
rc202	5	1,439.66	4	1,423.27	**4**	**1,378.31**	4	2,037.64
rc203	4	1,384.52	**3**	**1,265.06**	3	1,295.67	4	1,571.54
rc204	3	1,185.91	**3**	**960.21**	3	962.46	4	1,447.34
rc205	5	1,601.08	**4**	**1,497.85**	4	1,508.59	5	2,313.72
rc206	4	1,447.54	**3**	**1,341.10**	3	1,348.50	4	1,856.01
rc207	4	1,374.81	3	1,299.20	**3**	**1,294.41**	4	1,773.09
rc208	3	1,279.30	**3**	**1,007.62**	3	1,016.23	3	1,494.40

Table 2
Summary of the different algorithms tested on the 56 standard instances.

NRPA			nrpaD			nrpaDQ			Opturn		
V (mean)	Km (mean)	δ	V (mean)	Km (mean)	δ	V (mean)	Km (mean)	δ	V (mean)	Km (mean)	δ
8.21	1,216.72	0	7.59	1,097.47	53	**7.57**	**1,094.40**	**55**	9.16	1,600.10	-50

Table 3
Results for the different algorithms tested on the EDF instances.

	NRPA			nrpaD			nrpaDQ			opturn		
instance	Missed	V	Km	Missed	V	Km	Missed	V	Km	Missed	V	Km
EDF1	8	6	649.17	8	6	464.50	**8**	**6**	**433.97**	9	6	613.41
EDF2	0	6	454.90	**0**	**6**	**448.92**	0	6	456.63	1	6	575.99
EDF4	1	5	172.90	**1**	**5**	**160.99**	1	5	164.98	1	5	222.15
EDF5	**0**	**2**	**224.06**	**0**	**2**	**224.06**	**0**	**2**	**224.06**	0	2	271.24
EDF6	**0**	**1**	**85.27**	**0**	**1**	**85.27**	**0**	**1**	**85.27**	0	1	88.71
EDF7	0	4	302.72	**0**	**4**	**302.7**	0	4	302.8	2	4	370.73
EDF8	2	3	156.12	2	3	158.23	**2**	**3**	**152.40**	2	3	115.63
EDF9	4	11	861.30	**4**	**11**	**767.63**	4	11	777.35	4	12	498.19

Table 4
Summary for the different algorithms tested on the EDF instances.

	NRPA			nrpaD			nrpaDQ			opturn		
	Missed	V	Km	Missed	V	Km	Missed	V	Km	Missed	V	Km
Summary (mean)	1.88	4.75	363.30	1.88	4.75	326.54	**1.88**	**4.75**	**324.68**	2.13	4.88	344.51

The nrpaDQ with Distance and Quantile heuristics provides solutions with the same values for the 3 components of the objective function. These two heuristics perform better in average than the standard NRPA variant as for the number of traveled kilometers: 324 vs 363, that is a 10 percents improvement. If we compare with Opturn results, the improvement account for around 5 percents. This is still significant from a practical point of view because of the huge amount of kilometers travelled by the technicians on a yearly basis: about 220 millions of kilometers.

5.2. *nrpaDQ with different numbers of levels*

As described at the beginning of section 5, the operational process implies finding a solution to the CVRPTW within about two hours. This is why the experiments described above used number of levels L=3 and number of playouts at each level N=100. Beside meeting this time constraint, it would be of interest to determine whether modifying the number of levels could significantly improve the quality of the provided solutions. This is what we detail in this section. In the experiments we tried various numbers of levels L for nrpaDQ, while varying the number N of playouts at levels greater or equal to one, so that the total amount of playouts at level 0 would remain approximately the same, around 1,000,000. Thus we set N=1,000,000 for L=1, N=1,000 for L=2, N=100 for L=3, N=32 for L=4, N=16 for L=5, N=10 for L=6, and finally L=7 for N=7.

In this section as well as in the 3 next sections, we are looking for an overall improvement of the algorithm behavior on all the tested instances. Therefore, from now on, for Solomon instances, we will focus on the average value of the objective functions of the whole set of instances. In contrast, for EDF instances we will give results for each instance along with a summarized view through the mean values of the objective functions. The lexicographic objective function is summed up with a one million weight for missed appointments, a one thousand weight for the number of used vehicles, and a unitary weight for kilometers. Figures 7, 8 and table 5 give the results of the 7 different runs with the parameter settings described above (L=1 up to L=7).

The choice of nrpaDQ with 4 levels appears to be the best setting, although the 3-level variation performs also very well as compared to the others. For L=5 and upper, there is no significant difference between the mean values of objective functions. This suggests that

Fig. 7. Mean value of objective functions on Solomon instances for different numbers of levels.

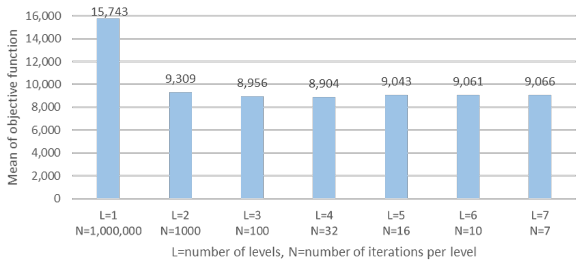
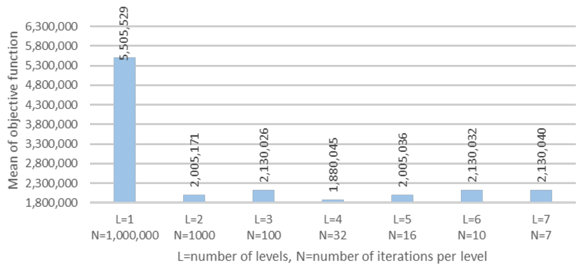


Table 5

Value of objective functions on EDF instances for different numbers of levels.

	L=1, N=1,000,000	L=2, N=1000	L=3, N=100	L=4, N=32	L=5, N=16	L=6, N=10	L=7, N=7
EDF1	17,006,682.5	**8,006 521.3**	9,006,483.0	8,006,544.4	9,006,479.2	9,006,486.4	9,006,530.5
EDF2	7,006,584.8	1,006,446.0	**6,459.7**	6,470.0	6,459.7	6,496.2	6,494.6
EDF4	5,005,236.4	1,005,181.9	**1,005,168.1**	1,005,185.1	1,005,211.9	2,005,152.8	2,005,151.7
EDF5	4,002,279.8	2,235.8	1,002,209.9	**2,224.1**	**2,224.1**	**2,224.1**	2,224.5
EDF6	2,001,091.6	1,088.7	**1,085.3**	**1,085.3**	1,088.5	1,088.5	**1,085.3**
EDF7	3,004,395.5	4,321.7	4,304.6	4,303.0	4,304.5	4,304.6	**4,302.8**
EDF8	2,004,198.1	2,003,103.7	**2,003,084.0**	2,003,085.4	2,003,088.9	2,003,092.1	2,003,092.2
EDF9	4,013,760.2	4,012,468.9	4,011,413.3	4,011,460.1	4,011,434.6	**4,011,407.5**	4,011,434.0

Fig. 8. Mean value of objective functions on EDF instances for different numbers of levels.



further modifying the trade-off between intensification and diversification in favor of the latter is unnecessary.

5.3. *nrpaDQ with more playouts*

Another way of trying to improve the quality of the solutions lies in increasing N, the number of iterations at each level. Of course this has a significant impact on the running time. We ran our experiment with L=3 (nrpaDQ with 3 levels), so the running time increases as the cube of N (again, in this section we relax the operational time constraint). We tested 6 values for N, namely N=75, N=100, N=125, N=150, N=175 and finally N=200. Beyond this value, running times become prohibitive (more than 20 hours per instance).

Figure 9, table 6 and figure 10 give the results of the 6 different runs described above (N=75 up to L=200).

Not surprisingly, the quality of results clearly improves when increasing the number of playouts up to 175. However, N=200 does not show significant improvements over N=175.

5.4. *nrpaDQ with extensive parallelization*

In section 5 we ran our experiments taking into account the common computer resources that can be accessed at an operational level. However, it is interesting to check whether being able to use more threads during the search would improve the quality of the provided solutions. Thus in this section we use root parallelization [48, 49]. The principle of root parallelization is to

Fig. 9. Mean value of objective functions on Solomon instances for different numbers of playouts at each level (L=3).

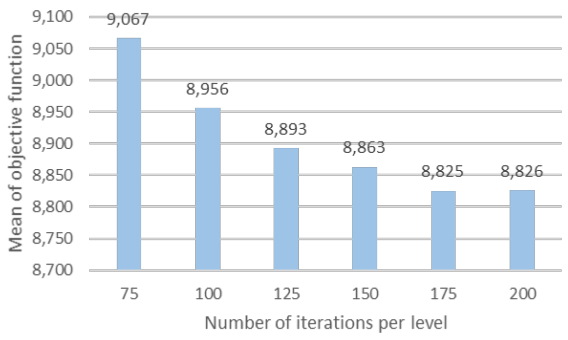
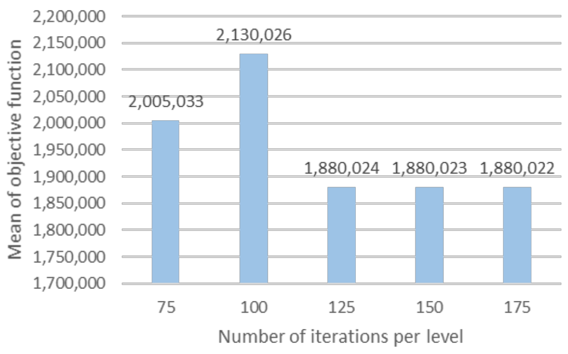


Table 6

Value of objective functions one EDF instances for different numbers of playouts at each level (L=3).

	N=75	N=100	N=125	N=150	N=175	N=200
EDF1	8,006,491.3	9,006,483.0	8,006,469.2	8,006,474.3	8,006,475.5	**8,006,446.5**
EDF2	**6,454.8**	6,459.7	6,458.1	6,459.7	6,462.6	6,470.4
EDF4	1,005,180.7	1,005,168.1	1,005,168.7	**1,005,155.8**	1,005,164.4	1,005,164.3
EDF5	1,002,214.4	1,002,209.9	2,228.7	2,226.9	2,225.4	**2,224.1**
EDF6	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**
EDF7	4,309.3	**4,304.6**	4,313.3	4,305.0	4,305.0	4,304.9
EDF8	2,003,097.6	2,003,084.0	2,003,090.5	2,003,082.2	2,003,080.9	**2,003,080.2**
EDF9	4,011,434.0	4,011,413.3	4,011,376.5	4,011,392.0	4,011,378.4	**4,011,375.1**

Fig. 10. Mean value of objective functions on EDF instances for different numbers of playouts at each level (L=3).



run several threads independently with different seeds and to join the corresponding results when the thinking time is elapsed. We run nrpaDQ on a number of threads varying from 1 to 12, while keeping L=3 and N=100. Each thread is launched using a different seed for the random function.

In figure 11, table 7, and figure 12 we present the results with the number of threads T varying from 1 to 12.

Logically, the more threads are used, the better the results.

5.5. *Stabilized nrpaDQ*

Stabilized NRPA [50] is a simple improvement of NRPA. The principle is to play P playouts at level 1 before each call to the adapt function. The number of calls to the adapt function at level 1 is still N, the number of iteration of upper levels. So at level 1, $P \times N$ playouts are performed.

Fig. 11. Mean value of objective functions on Solomon instances for different numbers of threads (L=3, N=100).

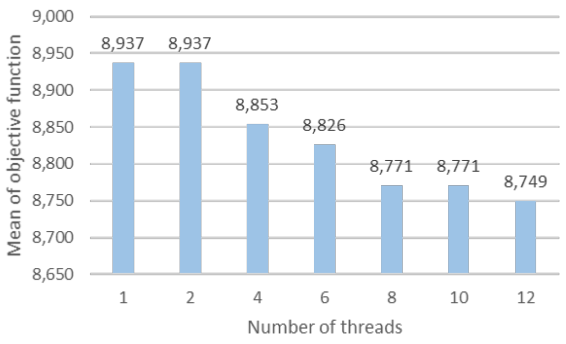
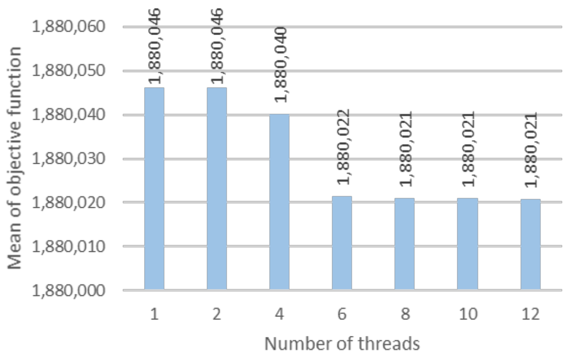


Table 7

Value of objective functions on EDF instances for different numbers of threads (L=3, N=100).

	T=1	T=2	T=4	T=6	T=8	T=10	T=12
EDF1	8,006,601.7	8,006,601.7	8,006,601.7	**8,006,456.8**	**8,006,456.8**	**8,006,456.8**	**8,006,456.8**
EDF2	6,501.0	6,501.0	6,459.7	6,459.7	6,459.7	6,459.7	**6,457.3**
EDF4	1,005,174.6	1,005,174.6	1,005,168.1	1,005,165.3	**1,005,162.0**	**1,005,162.0**	**1,005,162.0**
EDF5	2,225.3	2,225.3	2,225.3	2,225.3	**2,225.1**	**2,225.1**	**2,225.1**
EDF6	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**	**1,085.3**
EDF7	4,304.8	4,304.8	4,304.6	4,303.5	4,303.5	4,303.5	**4,303.0**
EDF8	2,003,084.4	2,003,084.4	**2,003,084.0**	**2,003,084.0**	**2,003,084.0**	**2,003,084.0**	**2,003,084.0**
EDF9	**4,011,392.3**	**4,011,392.3**	**4,011,392.3**	**4,011,392.3**	**4,011,392.3**	**4,011,392.3**	**4,011,392.3**

Fig. 12. Mean value of objective functions on EDF instances for different numbers of threads (L=3, N=100).



In figure 13, table 8, figure 14 we present the results of a nrpaDQ with 1 thread (1,000,000 playouts), the results of a stabilized nrpaDQ with a period of P=4 (4,000,000 playouts) and a nrpaDQ with 4 threads (that is 4 $\times$ 1,000,000 playouts). On EDF instances, the stabilization improves the results but not as much as adding more threads: with a same number of playouts, root parallelization on 4 threads has better results than the stabilized approach (actually, a better solution is only found on instances EDF1 and EDF6, with one less missed visit each). On Solomon instances, the stabilized approach outperforms the 4 threads approach.

6. Conclusion

We have presented the NRPA algorithm and its application to the CVRPTW problems. This problem is important for EDF, a company that plans numerous operations every day on the French electrical network.

Fig. 13. Mean value of objective functions on Solomon instances for the stabilized approach (L=3, N=100)

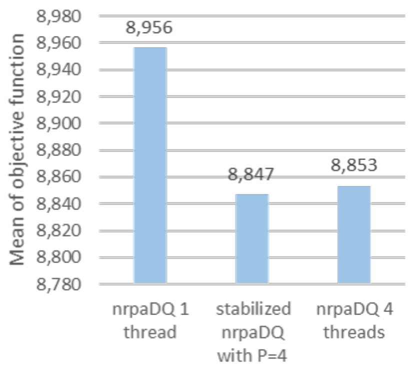
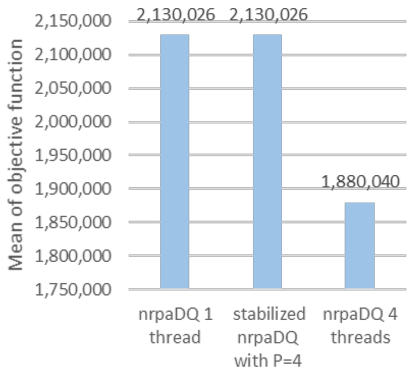


Table 8
Value of objective functions on EDF instances for the stabilized approach (L=3, N=100).

	nrpaDQ 1 thread	stabilized nrpaDQ with P=4	nrpaDQ 4 threads
EDF1	9,006,483.0	9,006,454.9	**8,006,601.7**
EDF2	6,459.7	**6,452.9**	6,459.7
EDF4	1,005,168.1	**1,005,161.1**	1,005,168.1
EDF5	1,002,209.9	**2,224.5**	2,225.3
EDF6	**1,085.3**	1,001,086.4	**1,085.3**
EDF7	**4,304.6**	4,308.4	**4,304.6**
EDF8	2,003,084.0	**2,003,080.9**	2,003,084.0
EDF9	4,011,413.3	**4,011,362.1**	4,011,392.3

Fig. 14. Mean value of objective functions on EDF instances for the stabilized approach (L=3, N=100)



We have given the modelization used to address the CVRPTW problem and two heuristics to improve on standard NRPA: the Distance heuristic and the Quantile heuristic. The Distance heuristic improves a lot on standard NRPA. The Quantile heuristic in combination with Distance heuristic is a slight improvement over NRPA with only Distance heuristic. We also compared nrpaDQ, our implementation of NRPA with the heuristics, to the current EDF solver Opturn. On standard instances nrpaDQ is much better. On the operational EDF instances it is still better even though Opturn was tuned for these instances, while nrpaDQ is a general algorithm that uses a Distance heuristic that works for all kinds of VRP problems.

As we have seen from the examples of EDF instances, NRPA with heuristics performs better than the current solver. The percentage of kilometers saved by heuristic NRPA is greater than 5% of the total number of kilometers. EDF agents drive hundreds of millions of kilometers each year. The use of nrpaDQ could save millions of kilometers each year and reduce the carbon footprint of EDF by hundreds of tons of CO_2.

Complementary experiments have also been carried out in order to test alternative parameter settings. The purpose of these tests was to check whether, regardless of operational constraints (computer resources and running time limitations) the nrpaDQ was capable of providing us with better solutions. It turns out that the answer is yes. Though the corresponding variations could not be operationally implemented yet for practical reasons, they give promising leads for future improvements.

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] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis and S. Colton, A Survey of Monte Carlo Tree Search Methods, *IEEE Transactions on Computational Intelligence and AI in Games* **4**(1) (2012), 1–43. doi:10.1109/TCIAIG.2012.2186810.

[2] T. Cazenave, Nested Monte-Carlo Search, in: *IJCAI*, C. Boutilier, ed., 2009, pp. 456–461.

[3] A. Rimmel, F. Teytaud and T. Cazenave, Optimization of the Nested Monte-Carlo Algorithm on the Traveling Salesman Problem with Time Windows, in: *Applications of Evolutionary Computation*, Lecture Notes in Computer Science, Vol. 6625, Springer, 2011, pp. 501–510.

[4] J. Méhat and T. 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) (2010), 271–277.

[5] B. 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*, 2013, pp. 1–15.

[6] S.M. Poulding and R. 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*, 2014, pp. 1279–1286.

[7] S.M. Poulding and R. 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*, 2015, pp. 1359–1366.

[8] B. 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*, 2016, pp. 119–120.

[9] T. Cazenave, A. Saffidine, M.J. Schofield and M. 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*, 2016, pp. 687–693.

[10] A.D. Dwivedi, P. Morawiecki and S. Wójtowicz, Finding differential paths in arx ciphers through nested monte-carlo search, *International Journal of electronics and telecommunications* **64**(2) (2018), 147–150.

[11] F. Portela, An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem, *bioRxiv* (2018), 345587.

[12] C.D. Rosin, Nested Rollout Policy Adaptation for Monte Carlo Tree Search, in: *IJCAI*, 2011, pp. 649–654.

[13] T. Cazenave and F. 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*, 2012, pp. 42–54.

[14] S. Edelkamp, M. Gath, T. Cazenave and F. Teytaud, Algorithm and knowledge engineering for the TSPTW problem, in: *Computational Intelligence in Scheduling (SCIS), 2013 IEEE Symposium on*, IEEE, 2013, pp. 44–51.

[15] S. Edelkamp, M. Gath and M. Rohde, Monte-Carlo Tree Search for 3D Packing with Object Orientation, in: *KI 2014: Advances in Artificial Intelligence*, Springer International Publishing, 2014, pp. 285–296.

[16] S. Edelkamp and C. Greulich, Solving physical traveling salesman problems with policy adaptation, in: *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, IEEE, 2014, pp. 1–8.

[17] S. Edelkamp and Z. Tang, Monte-Carlo Tree Search for the Multiple Sequence Alignment Problem, in: *Eighth Annual Symposium on Combinatorial Search*, 2015.

[18] S. Edelkamp, M. Gath, C. Greulich, M. Humann, O. Herzog and M. Lawo, Monte-Carlo tree search for logistics, in: *Commercial Transport*, Springer, 2016, pp. 427–440.

[19] T. Cazenave, J. Lucas, H. Kim and T. Triboulet, Monte Carlo Vehicle Routing, in: *Eleventh International Workshop on Agents in Traffic and Transportation co-located with the 24th European Conference on Artificial Intelligence (ECAI 2020), Santiago de Compostela, Spain, September 4, 2020*, I. Dusparic, F. Klügl, M. Lujak and G. Vizzari, eds, CEUR Workshop Proceedings, Vol. 2701, CEUR-WS.org, 2020, pp. 1–8.

[20] T. Cazenave, B. Negrevergne and F. Sikora, Monte Carlo Graph Coloring, in: *Monte Search at IJCAI*, 2020.

[21] T. Cazenave and T. Fournier, Monte Carlo Inverse Folding, in: *Monte Search at IJCAI*, 2020.

[22] A.L. Bazzan and F. Klügl, A review on agent-based technology for traffic and transportation, *The Knowledge Engineering Review* **29**(3) (2014), 375–403.

[23] K. Sörensen and M. Sevaux, A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling, *Journal of Mathematical Modelling and Algorithms* **8**(4) (2009), 387.

[24] J. Mańdziuk and C. Nejman, Uct-based approach to capacitated vehicle routing problem, in: *International Conference on Artificial Intelligence and Soft Computing*, Springer, 2015, pp. 679–690.

[25] A. Abdo, S. Edelkamp and M. Lawo, Nested rollout policy adaptation for optimizing vehicle selection in complex VRPs, in: *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*, IEEE, 2016, pp. 213–221.

[26] L. Crociani, G. Lämmel, G. Vizzari and S. Bandini, Learning Obervables of a Multi-scale Simulation System of Urban Traffic., in: *ATT@ IJCAI*, 2018, pp. 40–48.

[27] G.B. Dantzig and J.H. Ramser, The Truck Dispatching Problem, *Management Science* **6**(1) (1959), 80–91.

[28] G. Gutierrez-Jarpa, G. Desaulniers, G. Laporte and M. V., A Branch-And-Price Algorithm for the Vehicle Routing Problem with Deliveries, Selective Pickups and Time Windows, *European Journal of Operational Research* **206**(12) (2010), 341–349.

[29] N. Azi, M. Gendreau and J.-Y. Potvin, An Exact Algorithm for a Vehicle Routing Problem with Time Windows and Multiple Use of Vehicles, *European Journal of Operational Research* **202**(3) (2010), 756–763.

[30] Y. Agrawal, K. Mathur and H.M. Salkin, A Set-Partitioning-Based Algorithm for the Vehicle Routing Problem, *Networks* **19**(7) (1989), 731–749.

[31] M. Fisher and R. Jaikumar, A Generalized Assignment Heuristic for Vehicle Routing, *Networks* **11**(2) (1981), 109–124.

[32] G. Clarke and J. Wright, Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, *Operations Research* **12** (1964), 171–183.

[33] S.P. Anbuudayasankar, K. Ganesh, S.C. Lenny Koh and Y. Ducq, Modified Savings Heuristics and Genetic Algorithm for Bi-Objective Vehicle Routing Problem with Forced Backhauls, *Expert Systems and Applications* **30** (2012), 2296–2305.

[34] N. Yuichi and B. Olli, A Powerful Route Minimization Heuristic for the Vehicle Routing Problem with Time Windows, in: *Operations Research Letters, Vol. 37, No. 5*, 2009, pp. 333–338.

[35] Y. Rochat and E.D. Taillard, Probabilistic Diversification and Intensification in Local Search for Vehicle Routing, in: *Journal of Heuristics 1*, 1995, pp. 147–167.

[36] Y. Marinakis, G.-R. Iordanidou and M. Marinaki, Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands, *Applied Soft Computing* **13** (2013), 1693–1704.

[37] W.C. Chiang and R. Russel, Simulated Annealing Metaheuristics for the Vehicle Routing Problem with Time Windows, *Annals of Operations Research* **13**(1) (1996), 3–27.

[38] J.-Y. Potvin and S. Bengio, The Vehicule Routing Problem with Time Windows Part II : Genetic Search, *INFORMS Journal on Computing* **8**(2) (1996), 165–172.

[39] S.R. Tangiah, K.E. Nygard and J. P.L., GIDEON : A Genetic Algorithm System for Vehicule Routing with Time Window, in: *IEEE CAIA 1991 - Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications, 24-28 February 1991, Miami beach, Florida, USA*, 1991, pp. 322–328.

[40] J.-F. Cordeau, M. Gendreau and G. Laporte, A Tabu Search heuristic for the Periodic and Multi-Depot Vehicle Routing Problems, *Networks* **30**(2) (1997), 105–119.

[41] J.-Y. Potvin, T. Kervahut, B.-L. Garcia and J.-M. Rousseau, The Vehicule Routing Problem with Time Windows Part I : Tabu Search, *INFORMS Journal on Computing* **8**(2) (1996), 158–164.

[42] J. Berger, M. Barkaoui and O. Bräysy, A Parallel Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows, in: *Working paper, Defense Research Establishment Valcartier, Canada*, 2001.

[43] D. Mester, O. Bräysy and W. Dullaer, A Multi-parametric Evolution Strategies Algorithm for Vehicle Routing Problems, in: *Working Paper, Institute of Evolution, University of Haifa, Israel*, 2005.

[44] E. Taillard, P. Badeau, M. Gendreau, F. Geurtin and J.Y. Potvin, A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows, in: *Transportation Science 31*, 1997, pp. 170–186.

[45] Solomon, Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints, in: *Operations Research*, 1985.

[46] T. Graf and M. Platzner, Adaptive playouts for online learning of policies during Monte Carlo Tree Search, *Theor. Comput. Sci.* **644** (2016), 53–62.

[47] T. Cazenave, Playout policy adaptation with move features, *Theor. Comput. Sci.* **644** (2016), 43–52.

[48] T. Cazenave and N. Jouandeau, On the parallelization of UCT, in: *Computer Games Workshop*, 2007.

[49] G.M.-B. Chaslot, M.H. Winands and H.J. van Den Herik, Parallel monte-carlo tree search, in: *International Conference on Computers and Games*, Springer, 2008, pp. 60–71.

[50] T. Cazenave, J.-B. Sevestre and M. Toulemont, Stabilized Nested Rollout Policy Adaptation, in: *Monte Search at IJCAI*, 2020.