Pareto-NRPA: A Novel Monte-Carlo Search Algorithm for Multi-Objective Optimization

Noé Lallouet [a, b,*], **Tristan Cazenave**[a] **and Cyrille Enderli**[b]

[a]LAMSADE, Paris Dauphine - PSL University, Paris, France
[b]Thales DMS, Elancourt, France

Abstract. We introduce Pareto-NRPA, a new Monte-Carlo algorithm designed for multi-objective optimization problems over discrete search spaces. Extending the Nested Rollout Policy Adaptation (NRPA) algorithm originally formulated for single-objective problems, Pareto-NRPA generalizes the nested search and policy update mechanism to multi-objective optimization. The algorithm uses a set of policies to concurrently explore different regions of the solution space and maintains non-dominated fronts at each level of search. Policy adaptation is performed with respect to the diversity and isolation of sequences within the Pareto front. We benchmark Pareto-NRPA on two classes of problems: a novel bi-objective variant of the Traveling Salesman Problem with Time Windows problem (MO-TSPTW), and a neural architecture search task on well-known benchmarks. Results demonstrate that Pareto-NRPA achieves competitive performance against state-of-the-art multi-objective algorithms, both in terms of convergence and diversity of solutions. Particularly, Pareto-NRPA strongly outperforms state-of-the-art evolutionary multi-objective algorithms on constrained search spaces. To our knowledge, this work constitutes the first adaptation of NRPA to the multi-objective setting.

1 Introduction

Multi-objective optimization (MOO) is a research area consisting in finding ways to optimize several, often conflicting, objective functions. A MOO problem can be formulated in the following way:

$$\begin{aligned} &\min_{x}(f_1(x), f_2(x), ..., f_p(x)) \\ &\text{subject to } x \in X \end{aligned} \tag{1}$$

where $(f_1, ..., f_p)$ are the objective functions and X is the feasible set. X can be a subset of $\mathbb{R}^n$ for continuous optimization problems or $\mathbb{N}^n$ for discrete optimization tasks, and includes possible constraints. The concept of Pareto-dominance allows one to compare two solutions x and y. It is said that a solution x *dominates* y ($x \prec y$) iff $f_i(x) \leq f_i(y), \forall i = 1, ..., k$ and there exists a $j \in 1, ..., p$ such that $f_j(x) < f_j(y)$. A solution x is said Pareto-optimal iff $\neg\exists y \in X : y \prec x$, i.e. if there is no solution y that dominates x. The set of all Pareto-optimal solutions is called the Pareto front.

In this paper, we are concerned with tackling multi-objective optimization problems over a discrete search space using Monte-Carlo methods. Monte-Carlo methods typically use random simulations to collect information about the search space and converge to a solution. We introduce Pareto-NRPA, which is an extension of the NRPA algorithm [30] to the case of multi-objective optimization. NRPA is a Monte-Carlo algorithm tailored to single-objective optimization (SOO) that uses nested searches combined with policy updates. NRPA has shown high performance on games and combinatorial optimization problems, finding world records on Morpion Solitaire and crossword puzzles [30]. The contributions of this work are the following:

- We propose Pareto-NRPA, a novel algorithm for multi-objective optimization, and discuss the motivations behind the design choices of the algorithm.
- We introduce a new benchmark dataset based on popular traveling Salesman Problem with Time Windows (TSPTW) instances adapted to MOO. This dataset (MO-TSPTW) is publicly released and available online. [1]
- We compare the results of Pareto-NRPA against state-of-the-art MOO algorithms on two different sets of problems: the bi-objective TSPTW problem defined previously and bi-objective Neural Architecture Search (NAS) on well-known NAS benchmark datasets.

We find that Pareto-NRPA exhibits strong performances on these problems. As such, Pareto-NRPA may be identified as a suitable candidate for many multi-objective optimization problems over discrete search spaces. Similarly to evolutionary multi-objective optimization algorithms, Pareto-NRPA is a non-exact method for multi-objective optimization. To the best of our knowledge, this work is the first generalization of the NRPA algorithm to the multi-objective optimization setting.

2 Related Works

Multi-objective optimization

A large part of the literature in MOO is related to evolutionary computing, and particularly genetic algorithms. Multi-objective evolutionary algorithms (MOEAs) typically generate good solutions while keeping a large degree of diversity in the solution set. NSGA-II [14] is one of such algorithms, and is popular in MOO due to its ease of use and adaptability. Other popular genetic algorithms for multi-objective optimization include SMS-EMOA [19], SPEA-2 [41], NSGA-III [13] and MOEA/D [40]. MOEAs have been used successfully for many multi-objective optimization problems, such as water management [23] or change detection in satellite images [38].

Approaches to MOO using reinforcement learning have also been proposed [34]. Multi-objective reinforcement learning (MORL) is a

* Corresponding Author. Email: noe.lallouet@dauphine.eu.

[1] https://github.com/pareto-nrpa/mo-tsptw

branch of reinforcement learning that aims to learn a set of policies for multi-objective Markov decision processes. MORL has been successfully applied to many real-world problems [26] [24] [2]. Search and planning methods such as Monte-Carlo Tree Search, on the other hand, are less popular approaches for MOO. The UCT algorithm [21], one of the foremost MCTS algorithms, has notably been adapted to MOO [10], [36]. While there are similarities between RL and MCTS techniques [35], MCTS algorithms largely ignore the discounted reward mechanism, which is crucial in RL, in favour of terminal rewards observed after a playout. MCTS methods also often keep in memory only part of the search space and collect information through simulated playouts instead of real interaction with the environment.

Monte-Carlo Tree Search and NRPA

Monte-Carlo Tree Search (MCTS) is a popular set of techniques used to explore a search tree. MCTS uses random playouts to collect information about the value of a node in the search tree and uses a selection formula to balance exploration and exploitation in the tree. The UCT algorithm [21] and RAVE [17] are some node selection methods.

Nested Monte-Carlo Search (NMCS) [3] is a different way of exploring a search tree. NMCS implements a nested structure, which means that each search recursively launches a lower-level search, until level 0 where the search returns the terminal reward of a random playout starting from the current state. A state is a point in the search space representing a partial (in which case, actions can be taken) or terminal solution. Starting from an empty sequence, NMCS chooses actions until it reaches a terminal state. In state s_i, a level ℓ NMCS search launches a level $\ell - 1$ search for each available action. The action with the highest score is played. NMCS has been successfully applied to various games and combinatorial optimization problems [31] [8].

NRPA [30] combines the nested structure of NMCS with a policy learning method. At each level, NRPA performs a lower-level search and updates the policy with respect to the best returned sequence. Each search of level ℓ launches n searches of level $\ell-1$. At level 0, a search is a playout where the actions are conditioned by a policy vector π. The policy π associates a weight $w_{s,a}$ to a $(state, action)$ couple. $(state, action)$ couples are uniquely encoded by a domain-specific $code$ method.

During a playout and at state s_i, the probability of picking action k is derived from the associated policy weight: $p_{ik} = \frac{e^{w_{i,k}}}{\sum_j e^{w_{i,j}}}$. At algorithm initialization, π is a random policy with uniform distribution over the actions. After each search, the policy π is updated using gradient ascent towards the best sequence found at the current level. Algorithms 1 shows the playout algorithm. The reader is kindly referred to the original NRPA paper [30] for pseudocodes of the NRPA algorithm and the Adapt algorithm.

The research work of [11] introduces A-NMCS, a variant of NMCS for MOO problems, using scalarization to decompose a multi-objective problem into several single-objective problems, while highlighting that no adaptation of NRPA has been proposed for multi-objective optimization. We aim to fill this gap in the literature by proposing Pareto-NRPA.

3 Contributions

Given the space of all possible sequences S, NRPA tries to find the sequence with the highest score $s^* = \arg\max_{s \in S} \text{score}(s)$. As such, NRPA is suited to single-objective optimization (SOO) problems. In

Algorithm 1 Playout

Input: $state, \pi$
1: $sequence \leftarrow []$
2: **while** $true$ **do**
3: **if** $state$ is terminal **then**
4: **return** $(\text{score}(state), sequence)$
5: **end if**
6: $z \leftarrow 0$
7: **for** $m \in$ possible moves for $state$ **do**
8: $z \leftarrow z + \exp(\pi[\text{code}(m)])$
9: **end for**
10: choose a $move$ with probability $\frac{\exp(\pi[\text{code}(move)])}{z}$
11: $state \leftarrow \text{play}(state, move)$
12: $sequence \leftarrow sequence \cup move$
13: **end while**

the context of multi-objective optimization, one aims to identify, rather than a single sequence s^*, a set S^* of Pareto-efficient sequences:

$$S^* = \{s \in S | \neg \exists u \in S : u \neq s, u \prec s\} \quad (2)$$

In order to optimize several objectives simultaneously, a number of changes have to be made to the original NRPA algorithm. First of all, instead of optimizing a single policy π, we introduce a set of policies $\Pi = \{\pi_1, ..., \pi_n\}$. The cardinality of Π is a user-defined hyperparameter. By using several policies instead of one, the algorithm is able to optimize different regions of the search space, with one policy focusing on its own region.

At the end of a search, the original NRPA algorithm returns the best score and its associated sequence. The `score()` method takes as input a solution and returns the objective function values for that solution. In Pareto-NRPA, they are replaced by the Pareto set of non-dominated scores and the associated sequences. For each playout (i.e. a level 0 search), Pareto-NRPA first samples a policy $\pi_k \in \Pi$ from a uniform distribution, then uses the policy to guide a search starting from the root where probabilities of taking actions are conditioned by policy weights. When the playout reaches a terminal state, the objective functions $f_1, ..., f_p$ are evaluated and returned. For each sequence, the policy that was followed during the playout is memorized. Storing the sequence-policy couples (s, π_k) can be done in a tabular fashion or (as implemented in this work) by storing π_k as an attribute of the evaluated sequence. Line 2 of Algorithm 2 indicates that the policy is sampled from a uniform distribution. Using a uniform distribution promotes a similar number of function evaluations for all policies. More complex policy sampling methods, such as weighting policies according to their relative performance, remain an avenue for future work.

After a level $\ell - 1$ search, instead of memorizing the best score and the associated sequence, Pareto-NRPA memorizes a non-dominated set. The algorithm uses non-dominated sorting [14] to categorize the result of a search in several fronts $F_0, F_1, ..., F_p$ and memorizes the solutions belonging to the dominating set F_0. Algorithm 3 shows that policies are adapted with respect to the solutions they have produced if and only if these solutions belong to S^* (line 9). As such, if a policy has not led to a non-dominated solution, it will not be updated. In order to update each policy at least once, a minimum of one solution per policy is kept in S^*. If a policy π_k is not represented in F_0,the next fronts $F_1, ...F_p$ are iterated through until the first solution belonging to π_k is found.

The Adapt algorithm in Pareto-NRPA differs from the original NRPA formulation by incorporating multiple sequences into the policy update, rather than a single best one. In multi-objective optimization,

a solution may outperform another with respect to one objective while being inferior with respect to another. Thus, the policy update cannot rely on a single dominant sequence but must consider a set of non-dominated sequences. For each sequence $s \in S^*$ generated under policy π_k, the Adapt algorithm performs a weighted gradient update, conditioned by α, which is a user-defined hyperparameter representing the learning rate. The weight assigned to each sequence depends on its relative isolation within the non-dominated front, which is calculated using the crowding distance (CD) [14]. This distance approximates the perimeter of the hyper-rectangle defined by the nearest neighbors of a point in objective space, thus providing a measure of how sparsely populated the region around a sequence is.

Sequences with a high crowding distance are considered more isolated and are more likely to contribute to high diversity in the solution space. To prevent instability during policy updates, particularly for sequences with extreme objective function values that yield a $+\infty$ CD, the values are clipped to a maximum of 2. This ensures that the policy is updated preferentially towards isolated, diverse sequences without causing gradient explosion. An ablation study evaluating Pareto-NRPA with and without weighting sequences using CD is presented in the supplementary material of this paper (Section 6, Table 13).

The NRPA and Adapt algorithms adapted to the multi-objective optimization setting are presented in Algorithms 2 and 3.

Algorithm 2 Pareto-NRPA

Input: $level$, policy set Π
1: **if** $level = 0$ **then**
2: Choose $\pi_k : p(\pi_k) \sim U[0, |\Pi|]$
3: **return** Playout($root$, π_k)
4: **else**
5: $S^* \leftarrow \emptyset$
6: **for** N iterations **do**
7: $set \leftarrow$Pareto-NRPA($level - 1, \Pi$)
8: $S^* \leftarrow S^* \cup set$
9: $S^* \leftarrow \{s \in S^* | \neg \exists u \in S^* : u \neq s, u \prec s\}$
10: $\Pi \leftarrow$Pareto-Adapt(Π, S^*)
11: **end for**
12: **return** S^*
13: **end if**

It is straightforward to note that Pareto-NRPA with $n\_objectives = 1$ and $|\Pi| = 1$ is equivalent to NRPA. As such, Pareto-NRPA is a generalization of NRPA. To the best of our knowledge, this work is the first generalization of NRPA to multi-objective optimization.

A large number of improvements have been proposed to NRPA over time [12] [33] [4]. Crucially, GNRPA [4] generalizes the NRPA algorithm to include a domain-specific bias term. GNRPA has been shown to improve performance on several problems, including the TSPTW. GNRPA is very easily implemented in Pareto-NRPA, as it requires minimal modifications to the algorithm. Specifically, line 10 of Algorithm 3 is replaced by $z \leftarrow z + \exp(\pi(state, m) + \beta(m))$ and line 13 is replaced by $\pi'[\text{code}(m)] \leftarrow \pi'[\text{code}(m)] - (\alpha * \frac{\exp(\pi[\text{code}(m)] + \beta(m))}{z} * D[s])$.

It may also be noted that Pareto-NRPA is parallelizable. Although more complex ways to perform parallelization will be the object of future research, a straightforward way to parallelize Pareto-NRPA is by implementing playout parallelization, where a master process runs the main Pareto-NRPA algorithm and launches several playouts simultaneously with follower processes. This way of parallelizing

Algorithm 3 Pareto-Adapt

Input: Policy vector Π, optimal set S^*
1: $D \leftarrow$ CrowdingDistance(S^*)
2: **for** $s \in S^*$ **do**
3: $\pi \leftarrow$ policy used to sample s
4: $\pi' \leftarrow \pi$
5: $state \leftarrow root$
6: **for** $move \in s$ **do**
7: $\pi'[\text{code}(move)] \leftarrow \pi'[\text{code}(move)] + (\alpha * D[s])$
8: $z \leftarrow 0$
9: **for** $m \in$ possible moves for $state$ **do**
10: $z \leftarrow z + \exp(\pi[\text{code}(m)]])$
11: **end for**
12: **for** $m \in$ possible moves for $state$ **do**
13: $\pi'[\text{code}(m)] \leftarrow \pi'[\text{code}(m)] -$
14: $(\alpha * \frac{\exp(\pi[\text{code}(m)])}{z} * D[s])$
15: **end for**
16: $state \leftarrow \text{play}(state, move)$
17: **end for**
18: $\pi \leftarrow \pi'$
19: **end for**
20: Return Π

Pareto-NRPA is reminiscent of Stabilized-NRPA [9].

4 Results

We benchmark the performances of Pareto-NRPA on two problems. The first problem is an extension of classical instances of the traveling Salesman Problem with Time Windows (TSPTW) to bi-objective optimization. The second problem relates to finding neural network architectures in a large search space (NAS).

Comparing sets of results in a MOO setting is not as straightforward as in the single-objective case. A solution set S_1 may contain elements that outperform those in another set S_2 with respect to one objective f_1, while simultaneously being dominated or inferior with respect to other objectives $f_2, ..., f_p$. We compare sets of solutions obtained by different algorithms in the following ways:

- **Qualitative:** For bi-objective optimization problems, the best solutions found by each run of the algorithm are aggregated and their global dominating front is plotted. This graphical and qualitative comparison allows one to identify performing regions, Pareto front spread, and relative dominance.
- **Quantitative:** We use the hypervolume metric [18] to compare the size of the region that the Pareto front dominates over a reference point. Let Y^* be the objective space values of the elements in the Pareto set approximation S^*. The hypervolume metric is defined as volume of the space dominated by Y^* and bounded by a reference point r. Formally, the hypervolume indicator is written:

$$HV(Y^*, r) = \lambda_m \left(\bigcup_{y \in Y^*} [y, r] \right) \tag{3}$$

where λ_m is the Lebesgue measure in m dimensions.
In order to evaluate solution diversity and Pareto front homogeneity, the overall spread (OS) and spacing metrics are used. The overall spread metric, introduced by [37], gives information on the extent

of the Pareto front. It is calculated in the following way:

$$OS(Y^*) = \prod_{i=1}^{p} \frac{\left| \max_{y \in Y^*} y_i - \min_{y \in Y^*} y_i \right|}{\left| \tilde{y}_i^l - \tilde{y}_i^M \right|} \tag{4}$$

where $\tilde{y}_i^l$ is an approximation of the ideal objective vector and $\tilde{y}_i^M$ is an approximation of the maximal objective vector.
Finally, the spacing metric [32] returns a quantity related to the variation of the Manhattan distance between the elements of Y^*. Its expression is:

$$SP(Y^*) = \sqrt{\frac{1}{|Y^* - 1|} \sum_{j=1}^{|Y^*|} \left(\bar{d} - d^1(y^j, Y^* \setminus \{y^j\})^2\right)} \tag{5}$$

where $d^1(y^j, Y^* \setminus \{y^j\})$ is the minimal L_1-distance between an element y^j and the rest of the elements of Y^* and $\bar{d}$ is the mean of all $d^1(y^j, Y^* \setminus \{y^j\})$. A lower spacing value indicates a better distribution of points over the Pareto front approximation.

The results obtained by Pareto-NRPA are compared to the following algorithms:

- NSGA-II [14] is one of the most popular multi-objective evolutionary algorithms. It uses non-dominated sorting and crowding distance assignment to manage a population of individuals. We run NSGA-II with a population size of 250 and a sample size of 25.
- SMS-EMOA [19] uses the hypervolume metric to as the selection criterion for survival. It is a state-of-the-art multi-objective evolutionary algorithm. SMS-EMOA is run with a population size of 250.
- Pareto Local Search (PLS) [27] is a very straightforward discrete multi-objective algorithm based on local search. PLS demonstrates great performance on the unconstrained multi-objective traveling salesman problem.
- MOEA/D [40] uses decomposition and genetic operators to iteratively slove single-objective decompositions of a multi-objective problem, gradually reaching a Pareto front approximation. It is widely used in multi-objective optimization problems due to its computational efficiency and high performance.
- Pareto-MCTS [10] is an adaptation of UCT to multi-objective optimization. The comparison of Pareto-NRPA to this work is relevant because both methods use random playouts to optimize a Pareto front. The two algorithms are related to Monte-Carlo Tree Search.

SMS-EMOA, MOEA/D and NSGA-II are benchmarked using their respective implementation in the PyMoo framework [1].

MO-TSPTW

In this work, we introduce a novel bi-objective optimization dataset based on the TSPTW problem. In the classical version of the TSPTW problem, one aims to find a tour (permutation) of n cities such that the tour starts and ends at city 0, each city is visited exactly once and each city is visited inside a particular time window. NRPA has shown promising results on this problem, finding state-of-the-art solutions on numerous instances [7]. For each instance of the famous Solomon-Potvin-Bengio dataset [29], we generate a new set of coordinates for each city, and set the secondary cost associated to the travel from city i to city j as the euclidean distance between i and j. Figure 1 shows the primary and secondary cost matrices for instance rc_204.2 of the MO-TSPTW dataset. The time windows associated to each city are identical to the classical TSPTW instances. The primary and secondary distances between two cities are unrelated. As such, we find that minimizing both objectives at the same time is a challenging task, suitable for evaluating Pareto-NRPA. The multi-objective optimization problem associated with MO-TSPTW is:

$$\text{minimize} \begin{cases} f_1(s) = cost_1(s) + 10^6 \times \Omega(s) \\ f_2(s) = cost_2(s) + 10^6 \times \Omega(s) \end{cases} \tag{6}$$

where :

- $cost_1(s)$ (resp. $cost_2(s)$) is the sum of the primary (resp. secondary) distances between each city and the city that follows it in the sequence s.
- $\Omega(s)$ is the number of violated constraints (i.e. the number of cities that have been visited outside of their respective time window).

The 10^6 penalty applied to violated constraints quickly forces the algorithms to find valid sequences, as any valid solution, however inefficient it might be, always dominates an invalid solution.

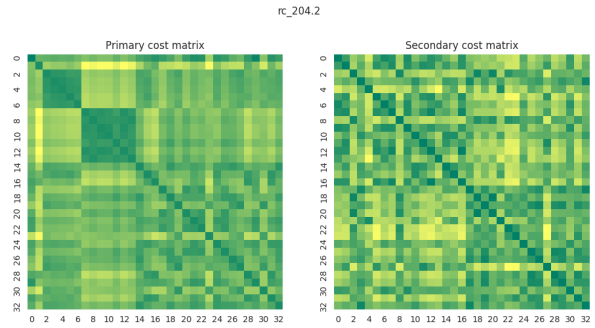


Figure 1. The two independent cost matrices for instance rc_204.2 of the TSPTW

The MO-TSPTW dataset contains 31 instances. For three of those instances, complete algorithm results (including hypervolume, overall spread, spacing and constraint violations) are presented and discussed:

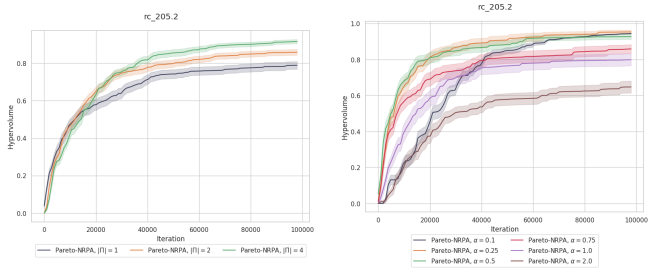
- rc_204.3 (24 cities) is the instance with the largest average time window per city. Thus, it is very easy to generate a valid solution that doesn't violate any constraints.
- rc_201.3 (32 cities) is the instance with the narrowest average time window per city. It is particularly hard to generate a valid solution for this instance.
- rc_204.1 (46 cities) is the instance with the largest number of cities to visit. While this instance has the second largest average time window in the dataset, the large number of cities strongly increases the complexity of the dataset: as such, satisfying the time window constraints for this instance is moderately hard.

Those three instances are representative of varying degrees of difficulty in the dataset and allow us to discuss different problem configurations. For all 31 instances of the dataset, the hypervolume indicator is presented in this section (Table 4). Complete results and metrics, including hypervolume, overall spread, spacing and average constraint violations for all 31 instances are shown in the supplementary material (Section 6, Tables 7-10). These tables include 95% confidence intervals : results for a metric X are shown as $\bar{X} \pm CI$, where $\bar{X} = \frac{1}{n\_runs} \sum_{i=0}^{n\_runs} X_i$ is the average value of a metric over all

runs, and $CI = 1.96 * \frac{\sigma}{\sqrt{n\_runs}}$, where σ is the standard deviation of X. For space reasons, Table 4 omits the confidence intervals, which may be found in the supplementary material.

The run configurations of each algorithm are the following:

- NSGA-II and SMS-EMOA: For both MOEAs, a member of the population is represented by a vector of size n_cities. The sampling operator samples a permutation of all cities. The mutation operator randomly inverts the order of a random subsequence in the solution [28]. Floating point values obtained after mutation and crossover are corrected using a rounding operation. The initial population size is 250.
- PLS: The initial population consists of 250 individuals.
- MOEA/D uses the same genetic parameters as NSGA-II and SMS-EMOA. The bi-objective problem is decomposed into 200 single-objective optimization subproblems.
- Pareto-MCTS: Pareto-MCTS implements the UCT algorithm [21] with leaf parallelization for improved stability [6] and speed. The UCB formula is parametrized by an exploration constant C. For optimization problems with rewards between 0 and 1, the exploration constant is often set to small values. However, as the results values of a playout in the two studied problem domains may be far greater than 1, we set C as a dynamic value computed for every node i : $C_i = \bar{r}_{\max} - \bar{r}_{\min}$, where $\bar{r}_{\max}$ (resp. $\bar{r}_{\min}$) is the maximal (resp. minimal) average node value for children of i.
- Pareto-NRPA: The same configuration of Pareto-NRPA is used for all instances in MO-TSPTW. The initial level L is set as 4 and the learning rate α as 0.5. The number of policies in Π is 4. These hyperparameters have been chosen after a succinct grid search on the rc_205.2 instance. Hypervolume evolution for different values of $|\Pi|$ and α are shown in Figures 2a and 2b. Complete hyperparameter search metrics for values of $\alpha \in \{0.1, 0.25, 0.5, 0.75, 1, 2\}$ and $|\Pi| \in \{1, 2, 4\}$ are presented in the supplementary material. Although tuning the hyperparameters for each instance would likely lead to improved performance, we intentionally use a fixed configuration to enable a fair comparison with other algorithms, which are also evaluated without instance-specific tuning.



(a) (b)

Figure 2. Different values of $|\Pi|$ and α on rc_205.2

All algorithms are run for 100 000 objective function evaluations. 30 independent runs are performed for each algorithm.

It has been shown that adding a bias to the probabilities of action selection strongly improves NRPA performance [4]. A bias value classically used for TSPTW is $-10 * d_{ij}$, with d_{ij} the normalized distance between two cities [5]. For fairness concerns, we also implement the same bias term in the sampling operators for SMS-EMOA and NSGA-II, as well as in the playout algorithm in Pareto-MCTS. An ablation study done on SMS-EMOA and NSGA-II (Section 6, Table 11) confirms that the introduction of bias significantly improves MOEA performance, as the algorithms find better solutions during the sampling step. In Tables 1, 2 and 3, the hypervolume values are calculated with respect to a vector $r = (\max_y f_1(y), \max_y f_2(y))$ for all valid solution vectors y in the aggregation of results of all 30 runs for all algorithms combined. The hypervolume indicator is then normalized using maximal hypervolume value after algorithm termination. Constraint violations (CV) refers to the average number of violated constraints for the best solution(s) over all 30 runs. The spacing metric is not computed for runs where no valid sequence has been found.

Table 1. Metrics on rc_204.3

Algorithm	Hypervolume	Overall Spread	Spacing	CV
NSGA-II	**0.97**	0.37	5.33	**0.00**
SMS-EMOA	0.96	0.33	7.06	**0.00**
Pareto-MCTS	0.61	0.16	18.08	**0.00**
PLS	0.91	0.37	**5.18**	**0.00**
MOEA/D	0.93	**0.38**	9.66	**0.00**
Pareto-NRPA	0.94	0.32	6.19	**0.00**

Table 2. Metrics on rc_201.3

Algorithm	Hypervolume	Overall Spread	Spacing	CV
NSGA-II	0.00	0.00	-	7.33
SMS-EMOA	0.00	0.00	-	5.97
Pareto-MCTS	0.00	0.00	-	9.63
PLS	0.00	0.00	-	11.80
MOEA/D	0.00	0.00	-	5.53
Pareto-NRPA	**0.91**	**0.35**	**4.38**	**0.00**

Table 3. Metrics on rc_204.1

Algorithm	Hypervolume	Overall Spread	Spacing	CV
NSGA-II	0.12	0.06	12.30	1.90
SMS-EMOA	0.00	0.00	-	2.17
Pareto-MCTS	0.00	0.00	-	20.47
PLS	0.00	0.00	-	5.67
MOEA/D	0.01	0.07	**4.98**	2.23
Pareto-NRPA	**0.26**	**0.08**	10.12	**0.00**

Table 1 indicates that NSGA-II and SMS-EMOA converge to a slightly better solution set than Pareto-NRPA on a very easy instance. Every algorithm succeeds in finding valid solutions for all 30 runs. However, Tables 2 and 3 show that, on the two harder instances, Pareto-NRPA converges to the best solutions, illustrated by the superior hypervolume values. The evolutionary algorithms, PLS and Pareto-MCTS fail to produce valid solutions for almost every run, while Pareto-NRPA converges quickly to solutions that do not violate time constraints. A hypervolume value of 0 indicates that the solution set generated by the algorithm is dominated by the reference point, which means that no valid solution has been found.

The results in Table 4 show that Pareto-NRPA strongly outperforms the state-of-the-art evolutionary MOO algorithms on most instances of the multi-objective TSPTW dataset. Precisely, Pareto-NRPA returns

Table 4. Normalized hypervolume on all instances of MO-TSPTW

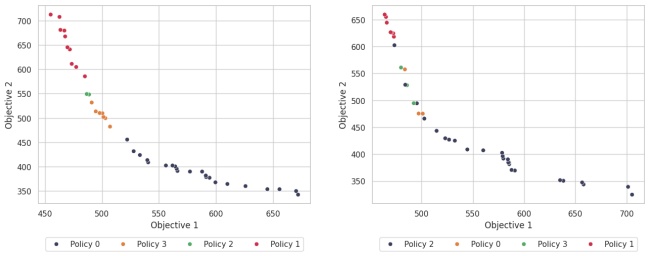
Instance	Cities	NSGA-II	SMS-EMOA	Pareto-UCT	PLS	MOEA/D	Pareto-NRPA
rc_206.1	4	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
rc_207.4	6	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
rc_203.4	15	**1.00**	**1.00**	0.65	0.97	0.99	0.95
rc_202.2	14	**1.00**	**1.00**	0.84	0.91	0.99	0.98
rc_204.3	24	**0.97**	0.96	0.61	0.91	0.93	0.94
rc_203.1	19	0.91	0.87	0.44	0.72	0.88	**0.97**
rc_204.2	33	0.80	0.79	0.00	0.80	0.57	**0.82**
rc_208.2	29	**0.94**	0.93	0.00	0.89	0.84	0.86
rc_204.4	14	**1.00**	**1.00**	0.60	0.89	0.99	0.99
rc_205.1	14	**1.00**	**1.00**	0.63	0.98	0.99	**1.00**
rc_204.1	46	0.12	0.00	0.00	0.00	0.01	**0.29**
rc_203.2	33	0.11	0.27	0.00	0.05	0.21	**0.86**
rc_203.3	37	0.08	0.04	0.00	0.03	0.04	**0.90**
rc_208.3	36	0.92	**0.93**	0.00	0.57	0.63	0.81
rc_202.4	28	0.01	0.07	0.00	0.00	0.00	**0.77**
rc_208.1	38	0.06	0.06	0.00	0.02	0.01	**0.46**
rc_207.3	33	0.24	0.28	0.00	0.05	0.37	**0.92**
rc_207.2	31	0.00	0.08	0.00	0.00	0.02	**0.33**
rc_206.3	25	0.56	0.63	0.01	0.03	0.79	**0.94**
rc_207.1	34	0.42	0.44	0.00	0.00	0.34	**0.79**
rc_202.1	33	0.00	0.08	0.00	0.02	0.04	**0.91**
rc_205.3	35	0.27	0.25	0.00	0.00	0.00	**0.77**
rc_201.1	20	0.77	0.73	0.61	0.28	0.93	**0.97**
rc_205.2	27	0.00	0.00	0.00	0.00	0.01	**0.92**
rc_205.4	28	0.25	0.28	0.00	0.00	0.22	**0.80**
rc_202.3	29	0.00	0.00	0.00	0.00	0.00	**0.94**
rc_206.2	37	0.00	0.00	0.00	0.00	0.00	**0.76**
rc_206.4	38	0.00	0.00	0.00	0.00	0.00	**0.51**
rc_201.2	26	0.06	0.25	0.00	0.00	0.14	**0.84**
rc_201.4	26	0.00	0.08	0.00	0.00	0.03	**0.33**
rc_201.3	32	0.00	0.00	0.00	0.00	0.00	**0.93**

a greatly superior hypervolume metric to the other state-of-the-art algorithms on 22 out of the 31 problem instances of MO-TSPTW. Evolutionary algorithms match or slightly outperform (in 5 cases) Pareto-NRPA in 8 of the 10 easiest instances in the dataset, while Pareto-NRPA is the superior algorithm in 20 of the 21 remaining hardest instances of the dataset. We note that Pareto-MCTS performs poorly on most hard instances of the dataset. Indeed, the UCB formula [21] is not suited to constraint handling. Setting the reward with a large penalty for violated constraints prevents efficient node value estimation, and simply aborting a playout where constraints are violated often leads to suboptimal, over-conservative policies [22]. While Pareto-NRPA and Pareto-MCTS are both based on Monte-Carlo playouts, these results show that Pareto-NRPA is much more suited to constrained search spaces.

NSGA-II, MOEA/D and SMS-EMOA converge to good and diverse solution sets on easy instances (such as rc_202.2 or rc_205.1, even slightly outperforming Pareto-NRPA), but generally fail to converge to a valid solution for harder instances, while Pareto-NRPA succeeds in finding solutions that don't violate any time constraints. Indeed, the policy adaptation mechanism implemented in Pareto-NRPA allows the algorithm to learn sequences of moves which leads to faster convergence. Pareto-NRPA thus emerges as an efficient algorithm for constraint handling in sequential discrete multi-objective optimization problems, strongly outperforming MOEAs on problems with constraints that are hard to satisfy.

We note that Pareto Local Search, despite being very popular on the unconstrained multi-objective TSP, doesn't yield good results on MO-TSPTW. This can be explained by the fact that the algorithms are capped at 100 000 function evaluations, a relatively low value for PLS which requires more iterations to efficiently evaluate diverse regions of the search space. Further experiments comparing the algorithms under equal search times show that PLS reaches good performances with more function evaluations (Section 6, Tables 21-23).

The diversity of the solutions produced by Pareto-NRPA is not always as good as the MOEAs. This is due to the fact that MOEAs maintain a large population, which retains diversity, while Pareto-NRPA only maintains a front of non-dominated sequences. Pareto-NRPA does not explicitely enforce diversity of solutions and their associated policy weights: the policy update is merely performed with respect to the crowding distance of sequences in S^*. As such, the overall distribution of policies in the Pareto front approximation may be uneven. Figure 3 shows the distribution of policies over the generated solutions for two different runs on rc_204.3. It may be appreciated that one run has converged to policies focusing on different regions of the search space, leading to varied solutions, while the other has a less distinct separation of policies in the solution space. Improving Pareto-NRPA's distribution of policies over different regions of the search space remains an axis for future work.



(a) (b)

Figure 3. Policy distribution for two different runs on rc_204.3. Each color represents the policy from which the solution has been sampled.

It is worth noting that Pareto-NRPA is computationally heavier than the compared state-of-the-art algorithms, especially the extremely fast MOEA/D and PLS. When the size of the search space increases, the complexity of Pareto-NRPA grows. Experimental results comparing the algorithms under equal CPU time are shown in the supplementary material of this paper, and demonstrate that Pareto-NRPA can be outperformed under strict search time conditions. Improving the computational complexity of Pareto-NRPA is a very clear research avenue.

Neural Architecture Search

Neural Architecture Search (NAS) refers to the task of automatically exploring a search space to find a neural network architecture minimizing one, or several, metrics. A classical goal for single-objective NAS is finding the architecture according to:

$$a^* = \arg\min_{a \in S} \mathcal{L}(X, Y, W_a) \tag{7}$$

where S is the space of all architectures, W_a are the neural network weights associated to architecture a and $\mathcal{L}(X, Y, W_a)$ is the loss computed on the validation dataset. The search space S is, in practical applications, typically large ($|S| > 10^{10}$). Recently, tabular benchmarks have been proposed to facilitate NAS research. These tabular benchmarks contain training and validation accuracies for every neural network in a restrained search space. Some of these benchmarks are NAS-Bench-201 [15], which contains 15625 architectures, and NAS-Bench-101 [39], with 423 000 neural network architectures. We use these two benchmarks to quickly evaluate the metrics associated to neural networks in a multi-objective setting. The two objective functions we aim to optimize are :

- $f_1 = 100 - Acc(a)$: the classification error of network a computed over the CIFAR-10 validation set.

- $f_2 = \#params$: the number of parameters of the neural network.

Minimizing these two metrics simultaneously amounts to finding an efficient trade-off between validation accuracy and neural network computational complexity. Indeed, it may be of interest to deep learning practicioners to identify high-performing neural networks with a limited resource budget in terms of inference speed of memory constraints. Multi-objective NAS approaches have been proposed using evolutionary algorithms [16] [25] and reinforcement learning [20]. Moreover, we intentionally limit the number of objective function evaluation to a low value (2000 function evaluations) as a single function evaluation is often very costly in NAS [42].

The first NAS benchmark dataset is NAS-Bench-201 [15]. It is a cell-based search space where a cell is represented as a directed acyclic graph with 4 vertices. Each edge (u, v) in the DAG is associated with one of the following operations applied to vertex u, transforming it to vertex v: skip-connection, 1x1 convolution, 3x3 convolution, 3x3 average pooling, or no operation at all. Searching a cell thus consists in assigning one of the available operations to each edge. The searched cell is finally used to create a neural network according to a pre-defined skeleton. We refer the reader to [15] for more information concering NAS-Bench-201.

NSGA-II, SMS-EMOA, Pareto-MCTS and Pareto-NRPA are independently run 30 times on the NAS-Bench-201 dataset. Each run consists in 2000 objective function evaluations. Figure 4a plots the accuracy of the neural networks found during search $(100 - f_1)$. It indicates that all algorithms have succeeded in finding the true Pareto front for NAS-Bench-201 after 30 runs. The Pareto fronts are superposed, meaning that all the algorithms have found the same set of points. Data points plotted in grey refer to the available architectures in the search space. It is expected that all algorithms succeed in finding the optimal Pareto front since the search space is small ($|S| = 15625$). Computing the average overall spread and hypervolume metrics however indicates that some runs converge to slightly suboptimal Pareto front approximations (Table 5). While all algorithms show very good performance, NSGA-II and SMS-EMOA obtain a hypervolume value 0.01 larger than Pareto-NRPA.

Table 5. Metrics on NAS-Bench-201

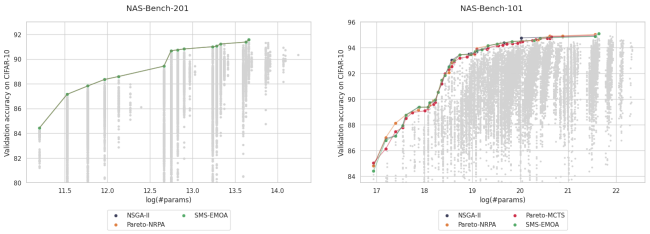
Algorithm	Hypervolume	Overall Spread
NSGA-II	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.72 \pm 0.02}$
SMS-EMOA	$\mathbf{0.99 \pm 0.00}$	0.70 ± 0.03
Pareto-MCTS	0.96 ± 0.00	0.68 ± 0.05
Pareto-NRPA	0.98 ± 0.00	0.70 ± 0.03

The second NAS benchmark dataset is NAS-Bench-101 [39]. NAS-Bench-101 is also a cell-based search space, where each cell is encoded as a DAG with 7 vertices. While NAS-Bench-201 associates operations to graph edges and feature maps to vertices, NAS-Bench-101 associates operations to vertices. The available operations are 1x1 convolution, 3x3 convolution and 3x3 max pooling. There are a total of 423 000 valid architectures in the NAS-Bench-101 search space: as such, finding a good Pareto front approximation on this dataset is slightly harder than on NAS-Bench-201. Each algorithm is run for 2000 objective function evaluations for 30 runs.

Results in Table 6 and Figure 4b show that, once again, all algorithms succeed in finding good solutions sets for NAS-Bench-101. This time, Pareto-NRPA slightly outperforms the state-of-the-art MOEAs. The two NAS search spaces presented are both unconstrained, which means that no neural network architecture is considered invalid. As such, MOEAs are well-suited to the task and Pareto-NRPA does not outperform them as strongly as in constrained search spaces such as harder instances of MO-TSPTW.

Table 6. Metrics on NAS-Bench-101

Algorithm	Hypervolume	Overall Spread
NSGA-II	0.98 ± 0.00	0.64 ± 0.04
SMS-EMOA	0.97 ± 0.00	0.62 ± 0.03
Pareto-MCTS	0.97 ± 0.00	0.56 ± 0.03
Pareto-NRPA	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.72 \pm 0.04}$



(a) (b)

Figure 4. Aggregated Pareto fronts on NAS-Bench-201 and NAS-Bench-101

5 Conclusion

We have proposed Pareto-NRPA, a novel multi-objective Monte-Carlo search algorithm using nested rollouts and policy adaptation. Pareto-NRPA uses a set of policies to guide a recursive search over several nested levels, and adapts the policy weights based on non-dominated solutions in the objective space. We benchmark Pareto-NRPA on a novel dataset extending classical instances of the Traveling Salesman Problem with Time Windows to multi-objective optimization (MO-TSPTW), as well as a neural architecture search task (NAS). We report that Pareto-NRPA outperforms state-of-the-art multi-objective evolutionary algorithms on MO-TSPTW and leads to competitive results on the NAS task. Pareto-NRPA is identified as a strong algorithm for constraint handling in sequential discrete multi-objective optimization problems. Future research directions include:

- Extending Pareto-NRPA to continuous search spaces.
- Benchmarking Pareto-NRPA on multi-objective optimization problems with 3 and more objectives.
- Improving policy diversity in the solution set.
- Accelerating Pareto-NRPA by reducing the computational complexity of the algorithm.

References

[1] J. Blank and K. Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2990567. URL https://ieeexplore.ieee.org/document/9078759.

[2] A. Castelletti, F. Pianosi, and M. Restelli. Tree-based Fitted Q-iteration for Multi-Objective Markov Decision problems. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, June 2012. doi: 10.1109/IJCNN.2012.6252759. URL https://ieeexplore.ieee.org/document/6252759. ISSN: 2161-4407.

[3] T. Cazenave. Nested Monte-Carlo Search. 2009.

[4] T. Cazenave. Generalized Nested Rollout Policy Adaptation, Mar. 2020. URL http://arxiv.org/abs/2003.10024. arXiv:2003.10024 [cs].

[5] T. Cazenave. Generalized Nested Rollout Policy Adaptation with Limited Repetitions, Jan. 2024. URL http://arxiv.org/abs/2401.10420. arXiv:2401.10420 [cs].

[6] T. Cazenave and N. Jouandeau. On the Parallelization of UCT.

[7] T. Cazenave and F. Teytaud. Application of the Nested Rollout Policy Adaptation Algorithm to the Traveling Salesman Problem with Time Windows. In Y. Hamadi and M. Schoenauer, editors, *Learning and Intelligent Optimization*, pages 42–54, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34413-8.

[8] T. Cazenave, A. Saffidine, M. J. Schofield, and M. Thielscher. Nested Monte Carlo Search for Two-Player Games. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 687–693, Phoenix, Arizona, United States, Feb. 2016. URL https://hal.science/hal-01436286.

[9] T. Cazenave, J. Sevestre, and M. Toulemont. Stabilized nested rollout policy adaptation. *CoRR*, abs/2101.03563, 2021. URL https://arxiv.org/abs/2101.03563.

[10] W. Chen and L. Liu. Pareto Monte Carlo Tree Search for Multi-Objective Informative Planning. In *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN 978-0-9923747-5-4. doi: 10.15607/RSS.2019.XV.072. URL http://www.roboticsproceedings.org/rss15/p72.pdf.

[11] M. Cornu. *Local Search, data structures and Monte Carlo Search for Multi-Objective Combinatorial Optimization Problems*. PhD Thesis, 2017. URL http://www.theses.fr/2017PSLED043/document.

[12] C. Dang, C. Bazgan, T. Cazenave, M. Chopin, and P.-H. Wuillemin. Warm-Starting Nested Rollout Policy Adaptation with Optimal Stopping. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10): 12381–12389, June 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i10.26459. URL https://ojs.aaai.org/index.php/AAAI/article/view/26459. Number: 10.

[13] K. Deb and H. Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, Aug. 2014. ISSN 1941-0026. doi: 10.1109/TEVC.2013.2281535. URL https://ieeexplore.ieee.org/document/6600851.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

[15] X. Dong and Y. Yang. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. Technical Report arXiv:2001.00326, arXiv, Jan. 2020. URL http://arxiv.org/abs/2001.00326. arXiv:2001.00326 [cs] type: article.

[16] T. Elsken, J. H. Metzen, and F. Hutter. Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. Technical Report arXiv:1804.09081, arXiv, Feb. 2019. URL http://arxiv.org/abs/1804.09081. arXiv:1804.09081 [cs, stat] type: article.

[17] S. Gelly and D. Silver. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*, 175:1856–1875, 2011. URL http://www.sciencedirect.com/science/article/pii/S000437021100052X.

[18] A. P. Guerreiro, C. M. Fonseca, and L. Paquete. The Hypervolume Indicator: Problems and Algorithms. *ACM Computing Surveys*, 54(6): 1–42, July 2022. ISSN 0360-0300, 1557-7341. doi: 10.1145/3453474. URL http://arxiv.org/abs/2005.00515. arXiv:2005.00515 [cs].

[19] N. Hochstrate, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181:1653–1669, Feb. 2007. doi: 10.1016/j.ejor.2006.08.008.

[20] C.-H. Hsu, S.-H. Chang, J.-H. Liang, H.-P. Chou, C.-H. Liu, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, and D.-C. Juan. MONAS: Multi-Objective Neural Architecture Search using Reinforcement Learning. Technical Report arXiv:1806.10332, arXiv, Dec. 2018. URL http://arxiv.org/abs/1806.10332. arXiv:1806.10332 [cs, stat] type: article.

[21] L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

[22] J. Lee, G.-h. Kim, P. Poupart, and K.-E. Kim. Monte-Carlo Tree Search for Constrained POMDPs. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/54c3d58c5efcf59ddeb7486b7061ea5a-Abstract.html.

[23] A. Lewis and M. Randall. Solving multi-objective water management problems using evolutionary computation. *Journal of Environmental Management*, 204(Pt 1):179–188, Dec. 2017. ISSN 1095-8630. doi: 10.1016/j.jenvman.2017.08.044.

[24] C. Li and K. Czarnecki. Urban Driving with Multi-Objective Deep Reinforcement Learning, Feb. 2019. URL http://arxiv.org/abs/1811.08586. arXiv:1811.08586 [cs].

[25] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf. NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm. Technical Report arXiv:1810.03522, arXiv, Apr. 2019. URL http://arxiv.org/abs/1810.03522. arXiv:1810.03522 [cs] type: article.

[26] Y. Oh, A. Ullah, and W. Choi. Multi-Objective Reinforcement Learning for Power Allocation in Massive MIMO Networks: A Solution to Spectral and Energy Trade-Offs. *IEEE Access*, 12:1172–1188, 2024. doi: 10.1109/ACCESS.2023.3347788. URL https://ieeexplore.ieee.org/document/10375483.

[27] L. Paquete, M. Chiarandini, and T. Stützle. Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 177–199, Berlin, Heidelberg, 2004. Springer. ISBN 978-3-642-17144-4. doi: 10.1007/978-3-642-17144-4_7.

[28] Y. M. Peng. Traveling salesman problem implementation with pymoo. Technical report, 2020. URL https://github.com/Peng-YM/pymoo/blob/master/pymoo/operators/mutation/inversion_mutation.py.

[29] J.-Y. Potvin and S. Bengio. The Vehicle Routing Problem with Time Windows Part II: Genetic Search. *INFORMS Journal on Computing*, 8(2): 165–172, May 1996. ISSN 1091-9856. doi: 10.1287/ijoc.8.2.165. URL https://pubsonline.informs.org/doi/abs/10.1287/ijoc.8.2.165. Publisher: INFORMS.

[30] C. Rosin. Nested Rollout Policy Adaptation for Monte Carlo Tree Search. pages 649–654, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-115.

[31] M. Roucairol and T. Cazenave. *Solving the HP model with Nested Monte Carlo Search*. Jan. 2023. doi: 10.48550/arXiv.2301.09533.

[32] J. Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. PhD thesis, Massachussets Institute of Technology, 1995.

[33] J. Sentuc, F. Ellouze, J.-Y. Lucas, and T. Cazenave. Learning the Bias Weights for Generalized Nested Rollout Policy Adaptation. In *Learning and Intelligent Optimization: 17th International Conference, LION 17*, pages 194–207. Springer-Verlag, Oct. 2023. doi: 10.1007/978-3-031-44505-7_14. URL https://doi.org/10.1007/978-3-031-44505-7_14.

[34] K. Van Moffaert, M. M. Drugan, and A. Nowé. Hypervolume-Based Multi-Objective Reinforcement Learning. In R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, and J. Shaw, editors, *Evolutionary Multi-Criterion Optimization*, pages 352–366, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37140-0.

[35] T. Vodopivec, S. Samothrakis, and B. Ster. On Monte Carlo Tree Search and Reinforcement Learning. *Journal of Artificial Intelligence Research*, 60:881–936, Dec. 2017. ISSN 1076-9757. doi: 10.1613/jair.5507. URL https://jair.org/index.php/jair/article/view/11099.

[36] W. Wang and M. Sebag. Multi-objective Monte-Carlo Tree Search.

[37] J. Wu and S. Azarm. Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set. *Journal of Mechanical Design*, 123(1):18–25, Mar. 2001. ISSN 1050-0472, 1528-9001. doi: 10.1115/1.1329875. URL https://asmedigitalcollection.asme.org/mechanicaldesign/article/123/1/18/471851/Metrics-for-Quality-Assessment-of-a-Multiobjective.

[38] A. Yavariabdi and H. Kusetogullari. Change Detection in Multispectral Landsat Images Using Multiobjective Evolutionary Algorithm. *IEEE Geoscience and Remote Sensing Letters*, 14(3):414–418, Mar. 2017. ISSN 1558-0571. doi: 10.1109/LGRS.2016.2645742. URL https://ieeexplore.ieee.org/document/7828040.

[39] C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy, and F. Hutter. NAS-Bench-101: Towards Reproducible Neural Architecture Search. Technical Report arXiv:1902.09635, arXiv, May 2019. URL http://arxiv.org/abs/1902.09635. arXiv:1902.09635 [cs, stat] type: article.

[40] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec. 2007. ISSN 1941-0026. doi: 10.1109/TEVC.2007.892759. URL https://ieeexplore.ieee.org/document/4358754.

[41] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Report, ETH Zurich, May 2001. URL https://www.research-collection.ethz.ch/handle/20.500.11850/145755. Accepted: 2022-08-12T12:06:45Z Publication Title: TIK Report Volume: 103.

[42] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. 2016. doi: 10.48550/ARXIV.1611.01578. URL https://arxiv.org/abs/1611.01578.

6 Supplementary material

6.1 Full results on MO-TSPTW

The full results on TSPTW are shown in Tables 7, 9, 8 and 10. Specifically,

- Table 7 shows hypervolumes for all instances of MO-TSPTW
- Table 8 shows the overall spread (OS) metric
- Table 9 shows the spacing metric
- Table 10 shows the average constraint violations

These tables include 95% confidence intervals : results for a metric X are shown as $\bar{X} \pm CI$, where $\bar{X} = \frac{1}{n\_runs} \sum_{i=0}^{n\_runs} X_i$ is the average value of a metric over all runs, and $CI = 1.96 * \frac{\sigma}{\sqrt{n\_runs}}$, where σ is the standard deviation of X. The instances are ordered by ascending ratio between average time window width of cities and number of cities. While not exactly representative of instance hardness, this ordering gives an idea of the relative ease of satisfying the constraints.

Instance	Cities	NSGA-II	SMS-EMOA	Pareto-UCT	Pareto-NRPA
rc_206.1	4	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_207.4	6	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_203.4	15	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.65 ± 0.01	0.95 ± 0.01
rc_202.2	14	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.84 ± 0.01	0.98 ± 0.00
rc_204.3	24	$\mathbf{0.97 \pm 0.01}$	0.96 ± 0.01	0.61 ± 0.02	0.94 ± 0.01
rc_203.1	19	0.91 ± 0.02	0.87 ± 0.03	0.44 ± 0.04	$\mathbf{0.97 \pm 0.01}$
rc_204.2	33	0.80 ± 0.03	0.79 ± 0.04	0.00 ± 0.00	$\mathbf{0.82 \pm 0.02}$
rc_208.2	29	$\mathbf{0.94 \pm 0.01}$	0.93 ± 0.01	0.00 ± 0.00	0.86 ± 0.02
rc_204.4	14	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.60 ± 0.04	0.99 ± 0.01
rc_205.1	14	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.63 ± 0.04	$\mathbf{1.00 \pm 0.01}$
rc_204.1	46	0.12 ± 0.11	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.29 \pm 0.05}$
rc_203.2	33	0.11 ± 0.08	0.27 ± 0.12	0.00 ± 0.00	$\mathbf{0.86 \pm 0.02}$
rc_203.3	37	0.08 ± 0.07	0.04 ± 0.05	0.00 ± 0.00	$\mathbf{0.90 \pm 0.02}$
rc_208.3	36	0.92 ± 0.06	$\mathbf{0.93 \pm 0.01}$	0.00 ± 0.00	0.81 ± 0.02
rc_202.4	28	0.01 ± 0.02	0.07 ± 0.06	0.00 ± 0.00	$\mathbf{0.77 \pm 0.08}$
rc_208.1	38	0.06 ± 0.07	0.06 ± 0.07	0.00 ± 0.00	$\mathbf{0.46 \pm 0.06}$
rc_207.3	33	0.24 ± 0.12	0.28 ± 0.14	0.00 ± 0.00	$\mathbf{0.92 \pm 0.02}$
rc_207.2	31	0.00 ± 0.00	0.08 ± 0.09	0.00 ± 0.00	$\mathbf{0.33 \pm 0.11}$
rc_206.3	25	0.56 ± 0.17	0.63 ± 0.16	0.01 ± 0.01	$\mathbf{0.94 \pm 0.02}$
rc_207.1	34	0.42 ± 0.15	0.44 ± 0.14	0.00 ± 0.00	$\mathbf{0.79 \pm 0.03}$
rc_202.1	33	0.00 ± 0.00	0.08 ± 0.09	0.00 ± 0.00	$\mathbf{0.91 \pm 0.02}$
rc_205.3	35	0.27 ± 0.14	0.25 ± 0.13	0.00 ± 0.00	$\mathbf{0.77 \pm 0.05}$
rc_201.1	20	0.77 ± 0.15	0.73 ± 0.16	0.61 ± 0.05	$\mathbf{0.97 \pm 0.00}$
rc_205.2	27	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.92 \pm 0.05}$
rc_205.4	28	0.25 ± 0.13	0.28 ± 0.14	0.00 ± 0.00	$\mathbf{0.80 \pm 0.03}$
rc_202.3	29	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.94 \pm 0.01}$
rc_206.2	37	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.76 \pm 0.07}$
rc_206.4	38	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.51 \pm 0.08}$
rc_201.2	26	0.06 ± 0.08	0.25 ± 0.15	0.00 ± 0.00	$\mathbf{0.84 \pm 0.03}$
rc_201.4	26	0.00 ± 0.00	0.08 ± 0.09	0.00 ± 0.00	$\mathbf{0.33 \pm 0.11}$
rc_201.3	32	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.93 \pm 0.02}$

Table 7. Normalized hypervolume on all instances of MO-TSPTW

Table 8. Pareto front spread on all instances of MO-TSPTW

Instance	Cities	NSGA-II	SMS-EMOA	Pareto-MCTS	Pareto-NRPA
rc_206.1	4	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_207.4	6	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_203.4	15	$\mathbf{0.58 \pm 0.01}$	0.58 ± 0.03	0.29 ± 0.07	0.30 ± 0.05
rc_202.2	14	$\mathbf{0.79 \pm 0.00}$	$\mathbf{0.79 \pm 0.00}$	0.40 ± 0.04	0.26 ± 0.03
rc_204.3	24	$\mathbf{0.37 \pm 0.03}$	0.33 ± 0.02	0.16 ± 0.02	0.32 ± 0.02
rc_203.1	19	0.28 ± 0.04	0.28 ± 0.03	0.05 ± 0.02	$\mathbf{0.35 \pm 0.03}$
rc_204.2	33	0.41 ± 0.04	0.35 ± 0.04	0.00 ± 0.00	$\mathbf{0.44 \pm 0.04}$
rc_208.2	29	$\mathbf{0.40 \pm 0.04}$	0.25 ± 0.03	0.00 ± 0.00	0.25 ± 0.03
rc_204.4	14	$\mathbf{0.55 \pm 0.00}$	$\mathbf{0.55 \pm 0.00}$	0.34 ± 0.03	0.54 ± 0.01
rc_205.1	14	0.70 ± 0.02	$\mathbf{0.71 \pm 0.00}$	0.35 ± 0.04	0.67 ± 0.03
rc_204.1	46	0.06 ± 0.06	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.10 \pm 0.03}$
rc_203.2	33	0.03 ± 0.03	0.09 ± 0.04	0.00 ± 0.00	$\mathbf{0.52 \pm 0.04}$
rc_203.3	37	0.04 ± 0.03	0.03 ± 0.03	0.00 ± 0.00	$\mathbf{0.37 \pm 0.04}$
rc_208.3	36	$\mathbf{0.37 \pm 0.05}$	0.29 ± 0.03	0.00 ± 0.00	0.23 ± 0.04
rc_202.4	28	0.00 ± 0.01	0.02 ± 0.02	0.00 ± 0.00	$\mathbf{0.40 \pm 0.06}$
rc_208.1	38	0.02 ± 0.02	0.02 ± 0.02	0.00 ± 0.00	$\mathbf{0.13 \pm 0.04}$
rc_207.3	33	0.11 ± 0.06	0.10 ± 0.05	0.00 ± 0.00	$\mathbf{0.44 \pm 0.05}$
rc_207.2	31	0.00 ± 0.00	0.03 ± 0.03	0.00 ± 0.00	$\mathbf{0.10 \pm 0.04}$
rc_206.3	25	0.24 ± 0.08	0.27 ± 0.09	0.00 ± 0.00	$\mathbf{0.44 \pm 0.05}$
rc_207.1	34	0.18 ± 0.07	0.22 ± 0.08	0.00 ± 0.00	$\mathbf{0.39 \pm 0.04}$
rc_202.1	33	0.00 ± 0.00	0.03 ± 0.04	0.00 ± 0.00	$\mathbf{0.45 \pm 0.06}$
rc_205.3	35	0.07 ± 0.04	0.12 ± 0.07	0.00 ± 0.00	$\mathbf{0.16 \pm 0.04}$
rc_201.1	20	$\mathbf{0.63 \pm 0.12}$	0.60 ± 0.13	0.21 ± 0.03	0.42 ± 0.06
rc_205.2	27	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.65 \pm 0.06}$
rc_205.4	28	0.11 ± 0.07	0.06 ± 0.03	0.00 ± 0.00	$\mathbf{0.45 \pm 0.04}$
rc_202.3	29	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.71 \pm 0.06}$
rc_206.2	37	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.23 \pm 0.06}$
rc_206.4	38	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.12 \pm 0.03}$
rc_201.2	26	0.05 ± 0.07	0.17 ± 0.10	0.00 ± 0.00	$\mathbf{0.44 \pm 0.08}$
rc_201.4	26	0.00 ± 0.00	0.04 ± 0.06	0.00 ± 0.00	$\mathbf{0.48 \pm 0.02}$
rc_201.3	32	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.35 \pm 0.03}$

Table 9. Spacing on all instances of MO-TSPTW (lower is better)

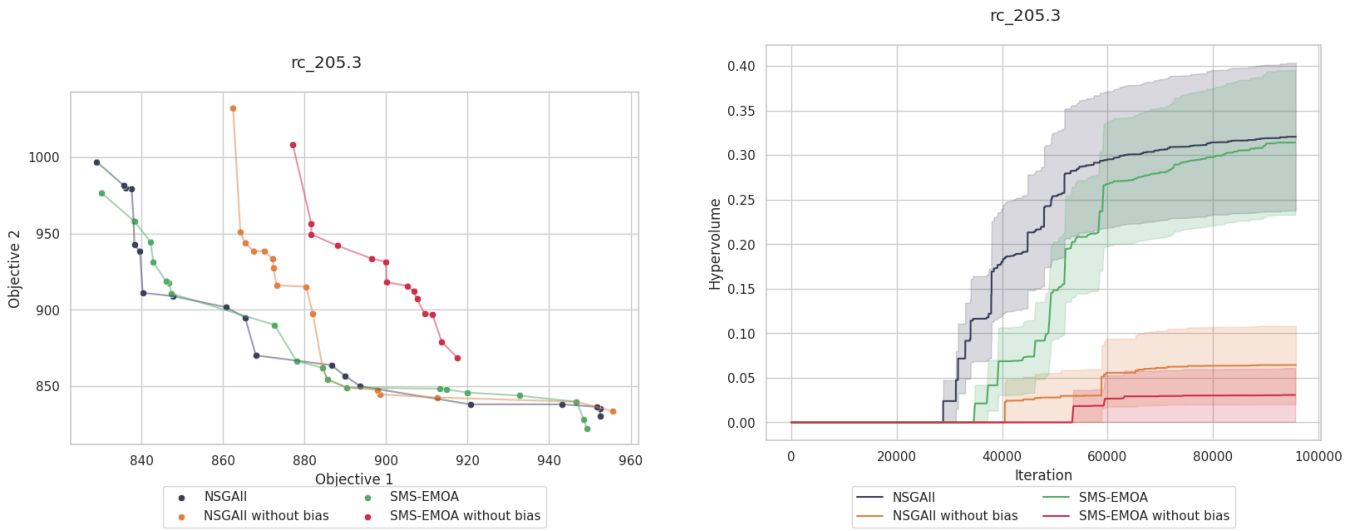
Instance	Cities	NSGA-II	SMS-EMOA	Pareto-MCTS	Pareto-NRPA
rc_206.1	4	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_207.4	6	$\mathbf{6.01 \pm 0.00}$	$\mathbf{6.01 \pm 0.00}$	$\mathbf{6.01 \pm 0.00}$	$\mathbf{6.01 \pm 0.00}$
rc_203.4	15	7.32 ± 0.18	7.51 ± 0.42	19.07 ± 2.48	$\mathbf{6.17 \pm 0.92}$
rc_202.2	14	$\mathbf{11.91 \pm 0.01}$	11.96 ± 0.07	15.62 ± 2.36	13.56 ± 0.13
rc_204.3	24	$\mathbf{5.33 \pm 0.61}$	7.06 ± 1.19	18.08 ± 3.26	6.19 ± 0.56
rc_203.1	19	7.34 ± 1.07	$\mathbf{7.28 \pm 1.39}$	15.67 ± 7.11	8.35 ± 0.90
rc_204.2	33	11.54 ± 1.86	14.69 ± 2.71	-	$\mathbf{10.43 \pm 1.21}$
rc_208.2	29	$\mathbf{7.90 \pm 1.06}$	10.26 ± 2.63	-	10.36 ± 1.15
rc_204.4	14	$\mathbf{7.23 \pm 0.16}$	7.28 ± 0.07	9.26 ± 1.61	7.25 ± 0.24
rc_205.1	14	8.04 ± 0.12	8.02 ± 0.2	9.13 ± 2.26	$\mathbf{7.75 \pm 0.26}$
rc_204.1	46	12.30 ± 2.54	-	-	$\mathbf{10.68 \pm 2.20}$
rc_203.2	33	11.80 ± 7.34	9.74 ± 5.21	-	$\mathbf{7.13 \pm 1.12}$
rc_203.3	37	8.21 ± 2.19	7.04 ± 2.43	-	$\mathbf{4.51 \pm 0.74}$
rc_208.3	36	$\mathbf{10.61 \pm 1.61}$	11.13 ± 1.66	-	11.29 ± 1.74
rc_202.4	28	15.34 ± 21.26	12.20 ± 5.45	-	$\mathbf{11.10 \pm 1.81}$
rc_208.1	38	11.92 ± 5.45	$\mathbf{8.45 \pm 2.09}$	-	13.16 ± 3.83
rc_207.3	33	$\mathbf{9.63 \pm 0.81}$	12.52 ± 3.18	-	9.94 ± 0.78
rc_207.2	31	-	11.74 ± 2.48	-	$\mathbf{8.65 \pm 2.18}$
rc_206.3	25	9.07 ± 1.83	9.63 ± 2.02	-	$\mathbf{7.58 \pm 1.84}$
rc_207.1	34	7.73 ± 1.11	11.28 ± 1.72	-	$\mathbf{7.68 \pm 0.93}$
rc_202.1	33	-	8.73 ± 6.31	-	$\mathbf{8.12 \pm 1.06}$
rc_205.3	35	13.46 ± 4.78	13.09 ± 2.69	-	$\mathbf{11.99 \pm 2.43}$
rc_201.1	20	$\mathbf{4.44 \pm 0.01}$	$\mathbf{4.44 \pm 0.01}$	13.65 ± 2.17	$\mathbf{4.57 \pm 0.08}$
rc_205.2	27	-	-	-	$\mathbf{9.07 \pm 0.69}$
rc_205.4	28	9.50 ± 4.52	8.49 ± 2.26	-	$\mathbf{7.11 \pm 0.76}$
rc_202.3	29	-	-	-	$\mathbf{6.51 \pm 0.63}$
rc_206.2	37	-	-	-	$\mathbf{12.56 \pm 2.56}$
rc_206.4	38	-	-	-	$\mathbf{7.20 \pm 2.16}$
rc_201.2	26	6.79 ± 0.30	8.54 ± 2.17	-	$\mathbf{6.57 \pm 0.44}$
rc_201.4	26	-	$\mathbf{13.48 \pm 11.13}$	-	19.73 ± 0.10
rc_201.3	32	-	-	-	$\mathbf{4.38 \pm 0.48}$

Table 10. Average constraint violations after 100000 objective function evaluations on all instances of MO-TSPTW

Instance	Cities	NSGA-II	SMS-EMOA	Pareto-MCTS	Pareto-NRPA
rc_206.1	4	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_207.4	6	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_203.4	15	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$
rc_202.2	14	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_204.3	24	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.00 ± 0.00	$\mathbf{0.00 \pm 0.00}$
rc_203.1	19	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_204.2	33	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	4.07 ± 0.28	$\mathbf{0.00 \pm 0.00}$
rc_208.2	29	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	7.10 ± 0.41	$\mathbf{0.00 \pm 0.00}$
rc_204.4	14	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_205.1	14	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_204.1	46	1.90 ± 0.47	2.17 ± 0.43	20.47 ± 0.32	$\mathbf{0.00 \pm 0.00}$
rc_203.2	33	1.90 ± 0.54	1.17 ± 0.43	10.63 ± 0.45	$\mathbf{0.00 \pm 0.00}$
rc_203.3	37	2.40 ± 0.52	2.63 ± 0.43	10.53 ± 0.26	$\mathbf{0.00 \pm 0.00}$
rc_208.3	36	0.03 ± 0.06	$\mathbf{0.00 \pm 0.00}$	13.43 ± 0.42	$\mathbf{0.00 \pm 0.00}$
rc_202.4	28	1.57 ± 0.30	2.10 ± 0.55	6.60 ± 0.35	$\mathbf{0.07 \pm 0.09}$
rc_208.1	38	0.97 ± 0.15	1.03 ± 0.22	23.97 ± 0.22	$\mathbf{0.13 \pm 0.26}$
rc_207.3	33	1.23 ± 0.41	1.13 ± 0.35	16.03 ± 0.35	$\mathbf{0.00 \pm 0.00}$
rc_207.2	31	1.17 ± 0.16	1.03 ± 0.17	18.50 ± 0.20	$\mathbf{0.40 \pm 0.18}$
rc_206.3	25	0.67 ± 0.35	0.50 ± 0.33	1.00 ± 0.13	$\mathbf{0.00 \pm 0.00}$
rc_207.1	34	0.73 ± 0.33	0.70 ± 0.34	4.60 ± 0.46	$\mathbf{0.00 \pm 0.00}$
rc_202.1	33	1.93 ± 0.32	1.70 ± 0.44	9.00 ± 0.49	$\mathbf{0.00 \pm 0.00}$
rc_205.3	35	1.90 ± 0.55	1.80 ± 0.53	10.80 ± 0.33	$\mathbf{0.00 \pm 0.00}$
rc_201.1	20	0.20 ± 0.14	0.33 ± 0.21	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
rc_205.2	27	4.73 ± 0.51	3.90 ± 0.44	3.83 ± 0.31	$\mathbf{0.00 \pm 0.00}$
rc_205.4	28	0.93 ± 0.32	0.87 ± 0.35	5.13 ± 0.29	$\mathbf{0.00 \pm 0.00}$
rc_202.3	29	2.10 ± 0.47	2.63 ± 0.60	10.23 ± 0.27	$\mathbf{0.00 \pm 0.00}$
rc_206.2	37	4.70 ± 0.63	3.70 ± 0.55	22.17 ± 0.29	$\mathbf{0.03 \pm 0.06}$
rc_206.4	38	5.13 ± 0.47	4.57 ± 0.49	19.13 ± 0.27	$\mathbf{0.10 \pm 0.14}$
rc_201.2	26	3.63 ± 0.54	1.93 ± 0.73	4.37 ± 0.24	$\mathbf{0.00 \pm 0.00}$
rc_201.4	26	2.97 ± 0.54	1.93 ± 0.36	3.40 ± 0.25	$\mathbf{0.00 \pm 0.00}$
rc_201.3	32	7.33 ± 0.50	5.97 ± 0.43	9.63 ± 0.52	$\mathbf{0.00 \pm 0.00}$

6.2 *Impact of bias on evolutionary multi-objective optimization algorithms*

Section 4 indicates that implementing a bias term in the rollout policy strongly improves NRPA performance. In order to provide a fair comparison between Pareto-NRPA and state-of-the-art MOO algorithms, the same bias term is implemented in the sampling operators for NSGA-II and SMS-EMOA. Figures 5a and 5b as well as Table 11 show that implementing a bias term improves the results of NSGA-II as well. The initialization part of the algorithm benefits from higher-quality samples thanks to the bias, which gives an important head start to algorithm convergence. Results in Table 11 demonstrate that adding a bias to NSGA-II gives an equal or higher hypervolume in 23 out of 31 instances, and adding a bias to SMS-EMOA gives equal or higher hypervolume in 25 out of 31 instances.



(a) Aggregated Pareto front

(b) Normalized hypervolume evolution

Figure 5. Impact of bias on EMOA convergence on rc_205.3

Table 11. Impact of bias on EMOA hypervolume

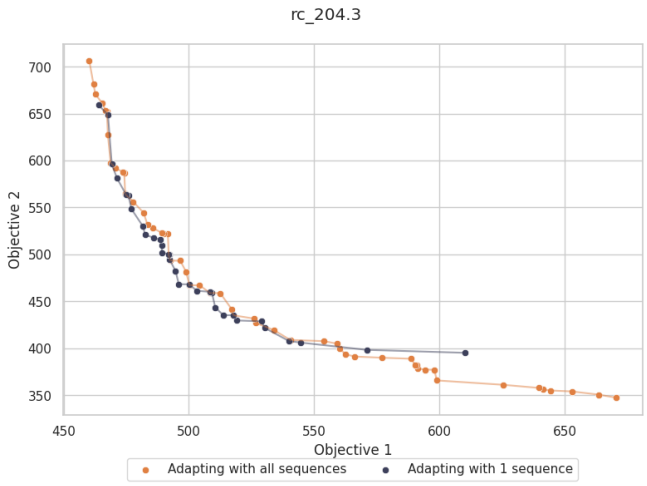
Instance	Cities	NSGA-II	NSGA-II (no bias)	SMS-EMOA	SMS-EMOA (no bias)
rc_206.1	4	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_207.4	6	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_203.4	15	$\mathbf{1.00 \pm 0.00}$	0.97 ± 0.01	$\mathbf{1.00 \pm 0.00}$	0.96 ± 0.02
rc_202.2	14	$\mathbf{1.00 \pm 0.00}$	0.97 ± 0.02	$\mathbf{1.00 \pm 0.00}$	0.98 ± 0.00
rc_204.3	24	$\mathbf{0.97 \pm 0.01}$	0.96 ± 0.01	0.96 ± 0.01	0.95 ± 0.01
rc_203.1	19	$\mathbf{0.91 \pm 0.02}$	0.78 ± 0.03	0.87 ± 0.03	0.77 ± 0.02
rc_204.2	33	$\mathbf{0.80 \pm 0.03}$	0.70 ± 0.06	$\mathbf{0.80 \pm 0.04}$	0.68 ± 0.05
rc_208.2	29	$\mathbf{0.94 \pm 0.01}$	$\mathbf{0.94 \pm 0.01}$	0.93 ± 0.01	0.92 ± 0.01
rc_204.4	14	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_205.1	14	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_204.1	46	$\mathbf{0.12 \pm 0.11}$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
rc_203.2	33	0.11 ± 0.08	0.24 ± 0.12	$\mathbf{0.27 \pm 0.12}$	0.19 ± 0.11
rc_203.3	37	$\mathbf{0.09 \pm 0.07}$	0.01 ± 0.02	0.05 ± 0.05	0.00 ± 0.00
rc_208.3	36	0.85 ± 0.06	0.79 ± 0.06	$\mathbf{0.86 \pm 0.02}$	0.80 ± 0.03
rc_202.4	28	0.02 ± 0.02	0.10 ± 0.07	0.07 ± 0.07	$\mathbf{0.18 \pm 0.10}$
rc_208.1	38	0.06 ± 0.07	$\mathbf{0.10 \pm 0.09}$	0.06 ± 0.07	0.10 ± 0.08
rc_207.3	33	0.42 ± 0.15	0.29 ± 0.14	$\mathbf{0.44 \pm 0.14}$	$\mathbf{0.44 \pm 0.14}$
rc_207.2	31	0.00 ± 0.00	0.11 ± 0.10	0.08 ± 0.09	$\mathbf{0.13 \pm 0.10}$
rc_206.3	25	0.56 ± 0.17	0.70 ± 0.15	0.63 ± 0.16	$\mathbf{0.81 \pm 0.11}$
rc_207.1	34	0.42 ± 0.15	0.29 ± 0.14	$\mathbf{0.44 \pm 0.14}$	$\mathbf{0.44 \pm 0.14}$
rc_202.1	33	0.00 ± 0.00	0.03 ± 0.05	$\mathbf{0.08 \pm 0.09}$	0.00 ± 0.00
rc_205.3	35	$\mathbf{0.27 \pm 0.14}$	0.05 ± 0.07	0.25 ± 0.13	0.02 ± 0.04
rc_201.1	20	$\mathbf{0.77 \pm 0.15}$	0.60 ± 0.18	0.73 ± 0.16	0.60 ± 0.18
rc_205.2	27	0.00 ± 0.00	0.03 ± 0.06	0.00 ± 0.00	$\mathbf{0.09 \pm 0.09}$
rc_205.4	28	0.26 ± 0.13	0.16 ± 0.11	$\mathbf{0.28 \pm 0.14}$	0.16 ± 0.11
rc_202.3	29	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.03 \pm 0.06}$
rc_206.2	37	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
rc_206.4	38	0.00 ± 0.00	$\mathbf{0.02 \pm 0.04}$	0.00 ± 0.00	0.00 ± 0.00
rc_201.2	26	0.06 ± 0.08	0.09 ± 0.10	$\mathbf{0.25 \pm 0.15}$	0.03 ± 0.06
rc_201.4	26	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.07 \pm 0.09}$	$\mathbf{0.07 \pm 0.09}$
rc_201.3	32	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00

6.3 *Ablation study: Adapt algorithm*

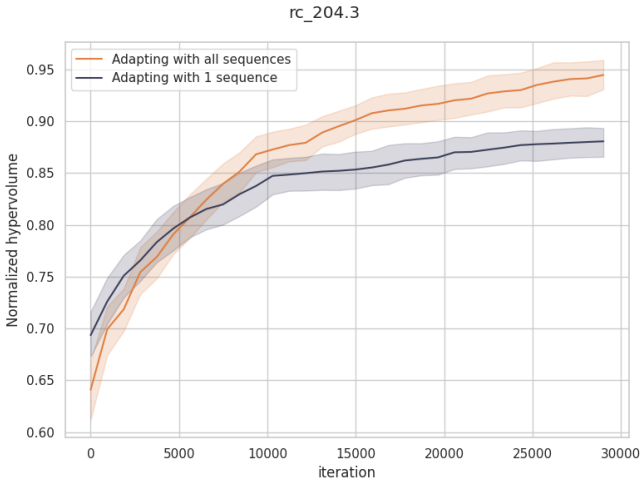
We are interested in finding out if adapting the policy with respect to all sequences in the Pareto front performs better than with only one sequence. In the *one sequence* setting, the solutions of the Pareto front are sorted according to their crowding distance, and each policy π_k updates with respect to the sequence originating from π_k that maximizes the crowding distance. The two algorithms are run on the instance `rc_204.3`. An easy instance such as this one is chosen because NRPA quickly finds multiple valid solutions, which highlights the difference between the two adapt methods.

Table 12. Metrics for the two adapt strategies

Adapt strategy	Normalized hypervolume	Overall Spread	Spacing
One sequence	0.50 ± 0.10	0.25 ± 0.04	10.48 ± 2.16
All sequences	$\mathbf{0.70 \pm 0.09}$	$\mathbf{0.56 \pm 0.07}$	$\mathbf{7.63 \pm 1.20}$



(a) Aggregated Pareto front for the two adapt strategies



(b) Normalized hypervolume evolution for the two adapt strategies

Figure 6. One-vs-all policy adaptation method

Figures 8a, 8b and Table 12 show the differences between adapting the policy with respect to a single sequence compared to all sequences. The metrics are shown with a 95% confidence interval. It is clear that adapting with all sequences gives the algorithm a significant performance boost, not only in solution quality but also diversity.

6.4 Ablation study: Crowding distance weighting

Section 3 mentions the use of crowding distance weighting during policy adaptation. Indeed, Algorithm 3 shows that the gradient ascent step for policy π_k and sequence $s \in S^* | s.policy = \pi_k$ is weighted by a quantity related to the crowding distance (CD) [14] of the sequence compared to other individuals in S^*. This design choice is made to promote policy adaptation towards more isolated sequences, with the aim to maximize the coverage of the Pareto front approximation in the solution space. In this ablation study, we compare Pareto-NRPA as presented in Section 3 to Pareto-NRPA where each sequence has the same weight during policy adaptation (Pareto-NRPA without CD weighting). The relevant metric for this comparison is the overall spread (OS) metric, which indicates the extent of the Pareto front approximation. Table 13 clearly shows that using crowding distance weighting during the Pareto-Adapt algorithm significantly improves the spread of S^* in the solution space.

Table 13. Overall spread for Pareto-NRPA with and without CD weighting

Instance	Cities	Pareto-NRPA	Pareto-NRPA without CD weighting
rc_206.1	4	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_207.4	6	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
rc_203.4	15	0.51 ± 0.05	$\mathbf{0.56 \pm 0.04}$
rc_202.2	14	0.53 ± 0.00	$\mathbf{0.56 \pm 0.04}$
rc_204.3	24	$\mathbf{0.67 \pm 0.03}$	0.53 ± 0.07
rc_203.1	19	$\mathbf{0.65 \pm 0.06}$	0.58 ± 0.08
rc_204.2	33	$\mathbf{0.47 \pm 0.04}$	0.28 ± 0.04
rc_208.2	29	$\mathbf{0.38 \pm 0.05}$	0.21 ± 0.03
rc_204.4	14	0.88 ± 0.02	$\mathbf{0.90 \pm 0.01}$
rc_205.1	14	$\mathbf{0.93 \pm 0.04}$	0.90 ± 0.04
rc_204.1	46	$\mathbf{0.17 \pm 0.06}$	0.00 ± 0.00
rc_203.2	33	$\mathbf{0.50 \pm 0.04}$	0.38 ± 0.04
rc_203.3	37	$\mathbf{0.38 \pm 0.05}$	0.22 ± 0.04
rc_208.3	36	$\mathbf{0.26 \pm 0.04}$	0.09 ± 0.02
rc_202.4	28	$\mathbf{0.40 \pm 0.06}$	0.35 ± 0.06
rc_208.1	38	$\mathbf{0.15 \pm 0.04}$	0.04 ± 0.02
rc_207.3	33	$\mathbf{0.58 \pm 0.06}$	0.37 ± 0.04
rc_207.2	31	$\mathbf{0.10 \pm 0.04}$	0.04 ± 0.02
rc_206.3	25	$\mathbf{0.53 \pm 0.07}$	0.42 ± 0.05
rc_207.1	34	$\mathbf{0.43 \pm 0.05}$	0.29 ± 0.06
rc_202.1	33	$\mathbf{0.44 \pm 0.06}$	0.21 ± 0.04
rc_205.3	35	$\mathbf{0.19 \pm 0.04}$	0.17 ± 0.04
rc_201.1	20	$\mathbf{0.57 \pm 0.08}$	0.42 ± 0.07
rc_205.2	27	$\mathbf{0.65 \pm 0.06}$	0.58 ± 0.05
rc_205.4	28	0.49 ± 0.05	$\mathbf{0.55 \pm 0.05}$
rc_202.3	29	$\mathbf{0.71 \pm 0.06}$	0.65 ± 0.06
rc_206.2	37	$\mathbf{0.21 \pm 0.05}$	0.16 ± 0.04
rc_206.4	38	$\mathbf{0.11 \pm 0.03}$	0.03 ± 0.01
rc_201.2	26	$\mathbf{0.49 \pm 0.09}$	0.47 ± 0.08
rc_201.4	26	$\mathbf{0.48 \pm 0.02}$	0.44 ± 0.05
rc_201.3	32	$\mathbf{0.34 \pm 0.03}$	0.26 ± 0.03

6.5 *Hyperparameter sensitivity analysis*

In order to choose the hyperparameters for Pareto-NRPA, we run a concise grid search on the rc_205.2 instance. The goal is not to find the best hyperparameters for each instance, but rather to identify a suitable hyperparameter configuration for all instances. As such, finding the optimal parameter combination for this particular instance is not the aim of this grid search. The hyperparameter search is conducted over values of $\alpha = \{0.1, 0.25, 0.5, 0.75, 1, 2\}$ and values of $|\Pi| = \{1, 2, 4\}$. Once hyperparameter values have been identified ($\alpha = 0.5$ and $|\Pi| = 4$), the $level$ parameter is studied for values in $\{3, 4\}$.

Table 14. Normalized hypervolume for each hyperparameter configuration.

$\Pi \backslash \alpha$	0.1	0.25	0.5	0.75	1	2
1	0.92 ± 0.04	0.83 ± 0.11	0.67 ± 0.13	0.46 ± 0.13	0.50 ± 0.14	0.38 ± 0.13
2	0.88 ± 0.04	0.92 ± 0.04	0.92 ± 0.04	0.87 ± 0.07	0.66 ± 0.13	0.40 ± 0.13
4	0.84 ± 0.02	$\mathbf{0.93 \pm 0.02}$	$\mathbf{0.93 \pm 0.02}$	0.82 ± 0.10	0.83 ± 0.07	0.65 ± 0.10

Table 15. Overall spread for each hyperparameter configuration.

$\Pi \backslash \alpha$	0.1	0.25	0.5	0.75	1	2
1	0.49 ± 0.05	0.42 ± 0.07	0.32 ± 0.08	0.20 ± 0.08	0.23 ± 0.08	0.14 ± 0.06
2	0.40 ± 0.05	0.53 ± 0.05	0.53 ± 0.05	0.43 ± 0.06	0.31 ± 0.08	0.16 ± 0.06
4	0.28 ± 0.05	$\mathbf{0.54 \pm 0.05}$	$\mathbf{0.54 \pm 0.05}$	0.45 ± 0.08	0.40 ± 0.05	0.28 ± 0.06

Table 16. Spacing for each hyperparameter configuration.

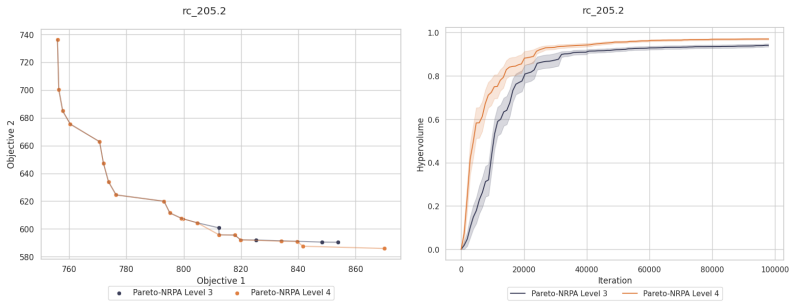
$\Pi \backslash \alpha$	0.1	0.25	0.5	0.75	1	2
1	8.65 ± 0.58	7.64 ± 0.69	6.66 ± 0.93	6.17 ± 1.26	7.62 ± 1.53	$\mathbf{5.54 \pm 1.53}$
2	8.40 ± 0.73	8.42 ± 0.60	8.09 ± 0.60	8.09 ± 0.88	7.70 ± 1.97	6.81 ± 1.44
4	8.71 ± 0.68	8.94 ± 0.57	8.85 ± 0.56	8.24 ± 0.64	8.18 ± 0.85	8.84 ± 1.60

Table 17. Average constraint violations for each hyperparameter configuration.

$\Pi \backslash \alpha$	0.1	0.25	0.5	0.75	1	2
1	$\mathbf{0.00 \pm 0.00}$	0.03 ± 0.06	0.07 ± 0.09	0.20 ± 0.08	0.23 ± 0.08	0.14 ± 0.06
2	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.13 ± 0.12	0.43 ± 0.20
4	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	0.04 ± 0.08	0.03 ± 0.06	0.04 ± 0.08

Table 18. Metrics on rc_205.2 for different values of NRPA level

Algorithm	Hypervolume	Overall Spread	Spacing	CV
Pareto-NRPA level 3	0.86 ± 0.05	0.48 ± 0.06	$\mathbf{8.82 \pm 0.79}$	$\mathbf{0.00 \pm 0.00}$
Pareto-NRPA level 4	$\mathbf{0.94 \pm 0.03}$	$\mathbf{0.63 \pm 0.05}$	8.95 ± 0.57	$\mathbf{0.00 \pm 0.00}$



(a) (b)

Figure 7. Aggregated Pareto front and hypervolume evolution for different values of $level$

6.6 *Results on the unconstrained multi-objective traveling salesman problem*

This paper introduces a novel benchmark dataset, MO-TSPTW, based on classical instances of the traveling salesman problem with time windows. As such, the main experiments in the paper reflect performance on this new dataset. However, one might argue that benchmarking Pareto-NRPA on the better-known MO-TSP problem is of interest. MO-TSP associates two independent costs matrices to a number n of cities, which can be visited in any order as there aren't any time windows to respect.

We benchmark Pareto-NRPA on two MO-TSP problem sizes: 50 cities and 100 cities. It may be appreciated that both of these sizes are larger than the largest MO-TSPTW instance (46 cities) but remain relatively smal. Indeed, Pareto-NRPA becomes computationally inefficient when the number of cities is larger than 500, due to the policy vector exploding in size. Improving the computational efficiency of Pareto-NRPA remains an avenue for future research.

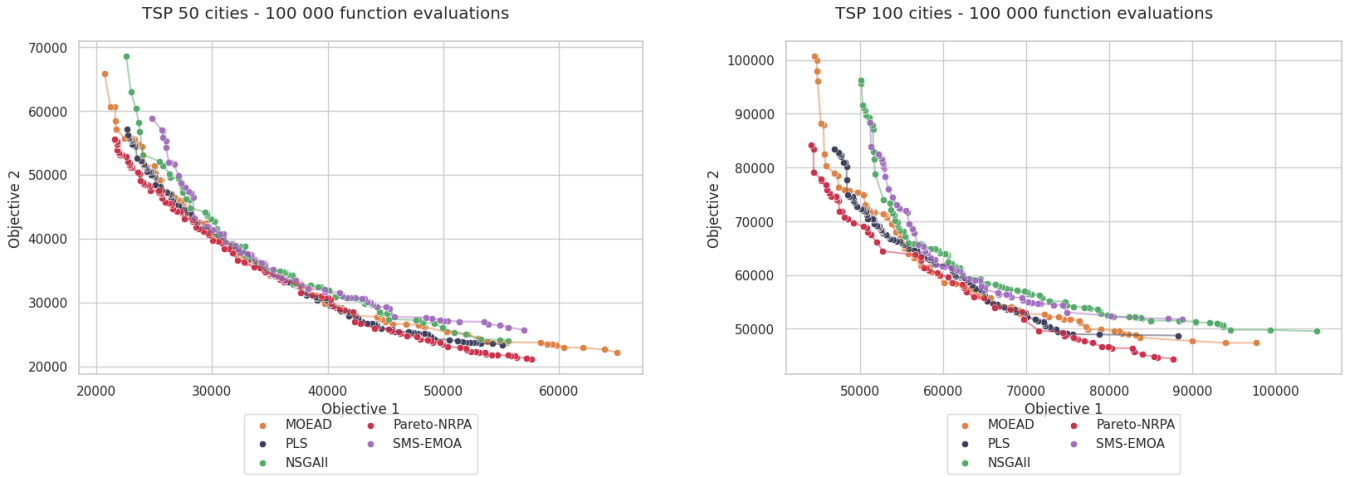
Table 19. MO-TSP : 50 cities

Algorithm	Hypervolume	Overall Spread	Spacing
NSGA-II	0.88 ± 0.00	0.51 ± 0.03	682.08 ± 87.39
SMS-EMOA	0.85 ± 0.00	0.35 ± 0.03	828.07 ± 115.47
PLS	0.88 ± 0.01	0.22 ± 0.02	$\mathbf{429.07 \pm 34.41}$
MOEA/D	0.93 ± 0.00	$\mathbf{0.55 \pm 0.03}$	589.00 ± 68.94
Pareto-NRPA	$\mathbf{0.98 \pm 0.00}$	0.41 ± 0.02	828.07 ± 115.47

Table 20. MO-TSP : 100 cities

Algorithm	Hypervolume	Overall Spread	Spacing
NSGA-II	0.79 ± 0.01	0.25 ± 0.10	$\mathbf{711.67 \pm 86.60}$
SMS-EMOA	0.77 ± 0.01	0.34 ± 0.05	853.00 ± 94.88
PLS	0.80 ± 0.01	0.16 ± 0.02	561.86 ± 56.71
MOEA/D	0.85 ± 0.01	0.32 ± 0.11	783.72 ± 136.09
Pareto-NRPA	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.42 \pm 0.04}$	849.45 ± 114.69

On the two MO-TSP datasets, Pareto-NRPA yields the best hypervolume values after 30 runs of 100000 iterations. We note that the performance of Pareto-NRPA is much closer to the other state-of-the-art algorithms on MO-TSP. This shows that MO-TSPTW exhibits higher complexity due to the time window constraints. As such, we believe that MO-TSPTW may be used for future research.



(a) MO-TSP with 50 cities

(b) MO-TSP with 100 cities

Figure 8. Pareto front approximations on MO-TSP

6.7 On algorithm compexity and CPU time

Despite its remarkable sample effiency and fast convergence speed, it may be observed that Pareto-NRPA's algorithmic complexity increases with the size of the search space. Indeed, the Adapt algorithm requires copying the policy vector to a temporary value. When the search space is large, the number of stat-action couples greatly increases and, as such, so does the cost of copying the policy. For example, copying the policy vector for MO-TSP with 500 cities exhibits a redhibitory computational cost.

Given the fact that Pareto-NRPA is less time-efficient than competing state-of-the-art algorithms, we benchmark the performances under the constraint of equal CPU-time for all algorithms, instead of equal number of function evaluations. This scenario is more representative of problems where evaluating the objectives is quick, while fixing the number of function evaluations represents cases (such as neural architecture search) where evaluating a solution is costly.

Table 21. Metrics on rc_204.3

Algorithm	Hypervolume	Overall Spread	Spacing	CV
NSGA-II	0.97 ± 0.02	0.43 ± 0.04	6.13 ± 2.25	$\mathbf{0.00 \pm 0.00}$
SMS-EMOA	0.75 ± 0.35	0.27 ± 0.15	11.08 ± 12.35	$\mathbf{0.00 \pm 0.00}$
Pareto-MCTS	0.41 ± 0.03	0.23 ± 0.08	44.94 ± 24.52	$\mathbf{0.00 \pm 0.00}$
PLS	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.68 \pm 0.18}$	$\mathbf{3.69 \pm 1.41}$	$\mathbf{0.00 \pm 0.00}$
MOEA/D	0.96 ± 0.03	0.35 ± 0.07	4.57 ± 2.06	$\mathbf{0.00 \pm 0.00}$
Pareto-NRPA	0.65 ± 0.15	0.15 ± 0.07	8.82 ± 2.11	$\mathbf{0.00 \pm 0.00}$

Table 22. Metrics on rc_201.3

Algorithm	Hypervolume	Overall Spread	Spacing	CV
NSGA-II	0.00 ± 0.00	0.00 ± 0.00	-	6.88 ± 0.88
SMS-EMOA	0.00 ± 0.00	0.00 ± 0.00	-	7.62 ± 0.84
Pareto-MCTS	0.00 ± 0.00	0.00 ± 0.00	-	11.62 ± 1.34
PLS	0.00 ± 0.00	0.00 ± 0.00	-	10.62 ± 3.23
MOEA/D	0.00 ± 0.00	0.00 ± 0.00	-	12.62 ± 1.25
Pareto-NRPA	$\mathbf{0.39 \pm 0.23}$	$\mathbf{0.15 \pm 0.10}$	$\mathbf{7.22 \pm 5.82}$	$\mathbf{0.12 \pm 0.23}$

Table 23. Metrics on rc_204.1

Algorithm	Hypervolume	Overall Spread	Spacing	CV
NSGA-II	0.00 ± 0.00	0.00 ± 0.00	-	2.62 ± 0.98
SMS-EMOA	$\mathbf{0.12 \pm 0.23}$	$\mathbf{0.12 \pm 0.23}$	$\mathbf{3.31 \pm 0.00}$	$\mathbf{2.25 \pm 0.83}$
Pareto-MCTS	0.00 ± 0.00	0.00 ± 0.00	-	22.12 ± 0.23
PLS	0.00 ± 0.00	0.00 ± 0.00	-	4.75 ± 1.46
MOEA/D	0.00 ± 0.00	0.00 ± 0.00	-	4.50 ± 0.92
Pareto-NRPA	0.00 ± 0.00	0.00 ± 0.00	-	10.25 ± 1.08

Tables 21, 22 and 23 display metrics for all algorithms with a CPU runtime limited to 240 seconds. Naturally, the number of function evaluations differs greatly under this scenario. For rc_204.3, which is a very easy and small search space, Pareto Local Search gives the best performances, due to the algorithm's exceptional speed. However, rc_201.3, which is the hardest instance but with a moderate number of cities, gives back the advantage to Pareto-NRPA. Finally, rc_204.1, which is the largest instance, is best solved by SMS-EMOA. Remarkably, Pareto-NRPA performs poorly on rc_204.1 when the search time is constrained: indeed, the large search space increases the complexity of the algorithm, leading to the extremely small number of approximately 1600 function evaluations performed during the time frame. We believe that reducing the computational complexity of Pareto-NRPA could have a very positive effect on performances under a constrained search time.