rAdam: restart Adam method to escape from local minima for bound constrained non-linear optimization problems

Thi Thoi Tran
Department of Research and Innovation, Capgemini Engineering, France
thi.tran@capgemini.com

Ismail TAYE
INSEA, France
tayee.ismail@gmail.com

Abstract

This paper presents a restart version of the Adaptive Moment Estimation (Adam) method for bound constrained nonlinear optimization problems. It aims to avoid getting trapped in a local minima and enable exploration the global optimum. The proposed method combines an adapted restart strategy coupling with barrier methodology to handle the bound constraints. Computational comparison with the standard Adam method is performed on a set of analytical problems.

Keywords Adam, Restart strategy, Restart Adam, Weight Decay, $L2$ regularization, Bound-Constrained problems, Globalization, Latin Hypercube Sampling, Design of Experiments.

Acknowledgments We thank Capgemini Engineering for supporting the work.

Adam	: Adaptive Moment Estimation Method	f	: objective function
rAdam	: Restart Adam method	θ_0	: initial point
GD	: Gradient Decent	m_0	: initial value of first moment vector
SGD	: Stochastic Gradient Decent	v_0	: initial value of second moment vector
BGD	: Batch Gradient Decent	t	: timestep
GA	: Genetic Algorithm	$\|.\|_2$	: l_2 norm
PSO	: Particle Swarm Optimization	n	: dimension of continuous variables
ACO	: Ant Colony Optimization	ϵ_c	: threshold value for constraints
GRASP	:Greedy Randomized Adaptive Search Procedure	ϵ_r	: threshold value for restart
DoE	: Design of Experiment	g_t	: gradient at iteration t
LHS	: Latin Hypercube Sampling	m_t	: first moment vector at iteration t
OF	: Objective Function	v_t	: second moment vector at iteration t
CV	: Constraint Value	β_1	: decay rate for first moment vector
CF	: Constraint Function	β_2	: decay rate for second moment vector
θ	: variable vector	m_{bt}	: bias-corrected first moment at iteration t
$\bar{\theta}$	: upper bound for θ	v_{bt}	: bias-corrected second moment at iteration t
$\underline{\theta}$	: lower bound for θ	α	: learning rate

Table 1 Abbreviations and nomenclature.

Draft paper as of August 13, 2025

1 Introduction and motivation

This paper addresses the bound constrained problems:

$$\begin{cases} \min\limits_{\theta} f(\theta) \\ \theta \in [\underline{\theta}, \bar{\theta}] \subset \mathbb{R}^n, \end{cases} \tag{1}$$

where $\theta \in \mathbb{R}^n$ is continuous variable. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is the output value.

There is a large body of works in optimization, particularly in the training of neuron network in deep learning, that regards nonlinear constrained problems. One of the simple and successfully among optimization methods is SGD method, [25]. Briefly, SGD is a variant of gradient decent method (GD), [22], which accelerates the convergence by updating parameters using single or few samples at a time. This can lead to faster training, especially for large datasets. However, the stochastic nature of sampling introduces high variance in the update steps, which may result erratic convergence behavior and oscillations near the optimal solution. To address these issues, it is critical to design an efficient mechanism to tune the learning rate schedule to promote stable and convergence. Other variants of GD includes Batch Gradient Decent (BGD), [23], mini-batch SGD, [13, 4]. The drawback of BGD is the redundant computations, as it recomputes gradients for the entire dataset before updating. In other hand, the mini-batch SGD strikes a balance between the efficiency of Gradient Descent and the stochastic nature of SGD. By updating parameters over small random subsets of data (mini-batches), it combines the advantages of both approaches. However, choosing an appropriate batch size is essential, as it can significantly impact convergence speed and memory requirements. Carefully tuning the batch size is necessary to achieve optimal performance. Some of the SGD variants propose acceleration steps to avoid being trapped in the incumbent points, e.g., the Momentum method, [3], or Nesterov Accelerated Gradient method in [18]. The two methods aim to accelerate convergence by maintaining a consistent update direction based on previous gradients. By incorporating momentum terms, these methods can navigate through flat regions and small gradients more efficiently, leading to faster convergence. However, they require careful tuning of momentum parameters to prevent overshooting and oscillations around the optimal solution. Additionally, they may struggle with saddle points or highly non-convex surfaces, where momentum can hinder convergence. Adaptive methods such as RMSprop, [10] adjust learning rates per parameter based on the average of recent gradients. They are particularly effective for training deep neural networks, where manually tuning learning rates can be challenging. By adjusting learning rates dynamically, adaptive methods can improve convergence and stability during training. However, they introduce additional hyperparameter that need to be tuned, such as the decay rates or momentum terms. Improper tuning can lead to suboptimal performance or even divergence during training.

Other approaches to solve bound constrained non-linear optimization problems rely on the nature selection such as the class of Evolutionary Algorithms, [11] and simulated annealing. The most popular methods for this class is Genetic Algorithms, [11]. They offer a unique approach to optimization by mimicking the process of natural selection and evolution. They are effective for complex, non-differentiable problems where traditional gradient-based methods fail. Genetic Algorithms excel at exploring diverse solution spaces and can find globally optimal or near-optimal solutions. However, they are computationally intensive, requiring a large number of evaluations, and lack convergence guarantees. Additionally, their performance heavily depends on the choice of genetic operators and parameters, making them challenging to tune. Another famous method in this class is Particle Swarm Optimization (PSO), [32]. PSO is a population-based optimization technique inspired by the social behavior of birds or fish. It is conceptually simple and easy to implement, making it suitable for a wide range of optimization problems. However, PSO can get trapped in local optima due to its reliance on individual and social best positions. Moreover, it struggles with high-dimensional problems where maintaining diversity among particles becomes challenging. Fine-tuning parameters such as the inertia weight and acceleration coefficients is crucial for balancing exploration and exploitation. [20] presents an Ant Colony Optimization (ACO) heuristic method. [28] proposes an exact approach with the novelty of the introduction of a dynamic aspect to the problem which is the ability to dynamically add tasks to accomplish. [26] proposes a Greedy Randomized Adaptive Search Procedure (GRASP) approach.

Among the SGD-based algorithms, Adam is one of the most popular optimization technique in deep learning. The Adam optimization algorithm, short for Adaptive moment estimation, first introduced in 2014 [14]. Adam computes adaptive learning rates for each parameter by combining first and second moment estimates of

the gradients. Adam has been widely adopted due to its effectiveness in optimizing deep neural networks, as demonstrated by empirical studies and benchmarks in various domains.

Several variants of Adam have been proposed to address specific limitations or improve performance. AdamW, introduced by [15], integrates weight decay directly into the Adam update rule to enhance stability and generalization performance. Nadam, proposed by [9], incorporates Nesterov momentum into Adam to accelerate convergence and improve generalization. AMSGrad, introduced by Reddi [21], prevents biases in the moving average of squared gradients to enhance convergence properties. Adamax, proposed by Kingma and Ba in 2014 alongside the original Adam paper, simplifies the algorithm by replacing the square of gradients with the maximum of past gradients, reducing memory requirements and computational complexity. These variants, along with the original Adam algorithm, constitute a rich landscape of optimization techniques for training deep neural networks.

This work proposes a coupling of Adam method and restart strategy to enhance the exploring phase of the method. The idea of restart strategy to escape from local minima can be found in [12, 17, 7, 27]. In [12, 17], the authors use a mechanism to restart the algorithm from completely new initial points whenever certain conditions are satisfied. Beside this *hard-restart* strategy, the authors in [7, 27] propose an adapted *soft restart* which allows one to use the information from the evaluated points and restart with the algorithm.

Another attractive track is the *multi-start* strategy, proposed for example in the method GORBIT [31]. The principle of this tactic is to start the algorithm from different starting points in parallel.

The main contribution of this paper is to integrate the restart strategy in Adam and using the soft LHS to create new starting point for each new cycle. Computational comparison with standard Adam is perform in a set of well-known analytical functions.

This paper is structured as follows. Section 2 describes Adam and AdamW algorithms. Section 3 presents the integration of restart strategy to Adam, as well as regulation method used to handle the bound constraints. Section 4 introduces the numerical results obtained with rAdam, rAdamW, rAdam with L2 reguleration, compared with the standard Adam. Then, conclusion and perspectives are given in Section 5.

2 Adam method

This section presents the Adam and AdamW algorithms. At its core, Adam maintains exponentially decaying averages of past gradients (first moments) and their squares (second moments) to adaptively adjust the learning rates during optimization. It combines the benefits of momentum optimization by incorporating a momentum term and adaptive learning rate methods by scaling the learning rates inversely proportional to the square root of the second moment estimates. This combination allows Adam to effectively navigate complex optimization landscapes and converge faster compared to traditional gradient descent methods. Additionally, Adam includes bias correction mechanisms to compensate for the initialization bias and bias introduced by the moving averages of gradients, i.e., m_t and their squares, i.e., v_t. The pseudo code of Adam method is given in Algorithm 1.

Despite its effectiveness, Adam requires careful hyperparameter tuning, particularly for the learning rate α, beta parameters, i.e., β_1, β_2 (momentum and decay rates), and ϵ (smoothing term) to ensure optimal performance across different tasks and datasets. Overall, Adam's adaptive learning rate mechanism and robust convergence properties make it a popular choice for optimizing deep neural networks, contributing to advancements in machine learning research and applications.

AdamW optimizer, i.e., an extension of Adam method with weight decay technique, is introduced in [15]. In the paper, the authors highlight that applying weight decay term in Adam effectively enhances the performance of the method. The method conducts by adding the weight decay in updating phase of the parameter, i.e., $\lambda\theta_{t-1}$. This remark brings the attention of having a restart version of AdamW can be the best version for coupling restart strategy and a variant of Adam. Another important key insight from the paper is that the L_2 regularization is not working well in adaptive setting, as the results, the weight decay is outperformed L_2 regularization. It leads to our curiosity to implement and test, in addition with rAdam, 2 other restart versions, one with weight decay,

Algorithm 1 Adam Algorithm, [14]

Require: Initial value of θ_0; Decay rates values β_1, $\beta_2 :\in [0, 1]$; Learning rate $\alpha = 0.001$;
Require: $m_0 \leftarrow 0$; $v_0 \leftarrow 0$; $t \leftarrow 0$;
1: **while** θ_t not converged **do**

Compute gradient:

$$g_t \leftarrow \nabla_\theta f_t(\theta_t) \tag{2}$$

Update parameters:

$$\begin{aligned} t &\leftarrow t + 1 \\ m_t &\leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t &\leftarrow \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &\leftarrow \frac{v_t}{1 - \beta_2^t} \\ \theta_t &\leftarrow \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned} \tag{3}$$

2: **end while**
3: **return** θ_t

i.e., rAdamW, one with L_2 regularization, i.e., rAdam$-L2-reg$. The rAdam$-L2-reg$ algorithm is similar to rAdam, with the objective function includes the $L2$ regularization term: $f(\theta_t) \leftarrow f(\theta_t) + \|CF(\theta_t, \underline{\theta}, \bar{\theta})\|_2^2$. The AdamW algorithm is summarized in Algorithm 2.

Algorithm 2 Adam weight decay Algorithm, [15]

Require: Initial value of θ_0; Decay rates values β_1, $\beta_2 :\in [0, 1]$; Learning rate $\alpha = 0.001$; $\lambda \in \mathbb{R}$
Require: $m_0 \leftarrow 0$; $v_0 \leftarrow 0$; $t \leftarrow 0$
1: **while** θ_t not converged **do**

Compute gradient:

$$g_t \leftarrow \nabla_\theta f_t(\theta_t) \tag{4}$$

Update parameters:

$$\begin{aligned} t &\leftarrow t + 1 \\ m_t &\leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t &\leftarrow \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &\leftarrow \frac{v_t}{1 - \beta_2^t} \\ \theta_t &\leftarrow \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t + \lambda \theta_{t-1} \end{aligned} \tag{5}$$

2: **end while**
3: **return** θ_t

As stated before, these algorithms can be trapped in local minima in certain cases, especially when the set of initial parameters are not as effective. To improve the exploration of methods, we propose to add a restart phase after a certain number of consecutive iterations without improving in the objective function. The technique will be described in detail in the next section.

3 rAdam method

This section presents the rAdam algorithm, a coupling of Adam, [14] and restart strategy, [7] to help Adam escaping from saddle points. The algorithm is described in the following.

The restart technique is activated once the maximum number of consecutive no-improvement in objective function, $nmax_{no-improve}$, reached. This value can be given from the users. The key point of restart strategy is to start the algorithm by a new initial point which taken into account all of the points that are exploited, as the results, the new vicinity will give more information to the algorithm. The sampling of this initial parameters is provided by the Latin Hypercube Sampling LHS technique, original proposed in [6], to have a better distribution of the newly generated parameters coordinates. Let's further discuss this Latin Hypercube Sampling technique to justify our choice.

Latin Hypercube Sampling (LHS) is a sophisticated statistical method employed in experimental design and Monte Carlo simulations to effectively sample from multi-dimensional parameter spaces, [29, 16]. The process begins by partitioning each parameter's range into equally probable intervals, constructing an $n \times n$ matrix known as a Latin hypercube. Randomization within each row and column eliminates bias, ensuring samples are spread evenly across intervals. Samples are then selected, ensuring each parameter value is sampled exactly once in each dimension. This approach offers advantages such as even coverage of the parameter space, reduced sampling error, and efficiency, making it invaluable for exploring high-dimensional parameter spaces and conducting accurate Monte Carlo simulations in diverse fields like engineering, finance, and environmental modeling. Additionally the LHS sampling technique is ensuring that the whole search space is sampled in a way that every chosen parameter is not redundant, and it also maximizes the minimal distance between all the sampled points, this way we can cover the parameters range more efficiently.

In our case, we are using this technique to sample set of parameters each time there's no improvement beyond the set threshold of the objective function, as it ensures a better random distribution of the initial set of parameters we want to sample to restart our optimization algorithm.

The proposed method, rAdam, is built based on the previously discussed Adam algorithm coupling with the restart technique, so that this algorithm can restart whenever it's trapped on a barely variant loss function value with an epsilon hyperparameter which translates to how much variability in the loss function we want to tolerate before we activate the restart strategy to generate a new set of initial points.

The proposed algorithm starts with the standard optimization using the Adam optimizer with an initial parameter which is generated by LHS. Hence, a monitoring on the objective function is set by counting the number of consecutive no-improvement in its values, in other words, the algorithm has *exploited enough* towards the local minimum. The key point is *"enough"*. Consider the case the objective function does improve, however at a very low rate, defined by a small enough ϵ_r or it's oscillating between a range of values, the counting mechanism is activated. Once the counting reaches the maximum number, the restart strategy is activated to force the algorithm to jump out and explore other neighborhoods. We now have an algorithm that has the same characteristics of its predecessor with a new restart technique exclusive to the initial parameters that can make sure it escapes local minima.

The proposed method which coupling Adam and restart strategy can address general optimization problems. However, to deal with bound constrained optimization problems, this work proposes to use barrier method, [5]. A threshold small value h_{max} is defined, if the Constraints Value at the current parameter is greater than the threshold value: $CV(\theta_t, \underline{\theta}, \bar{\theta}) > h_{max}$, then the parameter is recalculated based on a forced Constrained Function CF. The CF function's task is to ensure that θ_t is satisfied the bound constraints. The value of h_{max} is set at 10^{-3} in the numerical test. We tried to tolerate the algorithm capacity to provide us with parameters outside the boundaries mid-convergences where it gives values outside of the constraints but to a certain threshold, and that threshold is what we call the tolerance principle, in other words, it translates to how much room of error we can tolerate mid-convergence if it is beneficial in the final convergence.

The whole algorithm is summarized in Algorithm 3.

Algorithm 3 rAdam Algorithm

Require: Initial value of θ_0; Decay rates values $\beta_1 = 0.9$, $\beta_2 = 0.999$;
Require: Learning rate $\alpha = 0.001$; $\epsilon_r = 10^{-5}$
Require: $m_0 \leftarrow 0$; $v_0 \leftarrow 0$; $t \leftarrow 0$
1: **while** iteration $<$ max iterations **do**

Compute gradient:

$$g_t \leftarrow \nabla_\theta \nabla_\theta f_t(\theta_{t-1})$$

Update parameters:

$$\begin{aligned} t &\leftarrow t+1 \\ m_t &\leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &\leftarrow \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &\leftarrow \frac{v_t}{1-\beta_2^t} \\ \theta_t &\leftarrow \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned}$$

Constraints handling
if $(CV(\theta_t, \underline{\theta}, \bar{\theta}) > h\_max)$:

$$\theta_t \leftarrow CF(\theta_t, \underline{\theta}, \bar{\theta})$$

end if
Restart phase
if $(abs(OF(\theta_t) - OF(\theta_{t-1}) < \epsilon_r)$:

$$restart_{count} \leftarrow restart_{count} + 1$$

if $(restart_{count} = restart_{max})$:

$$\theta_t \leftarrow LHS$$

end if
else:

$$restart_{count} \leftarrow 0$$

end if
2: **end while**
3: **return** θ_t

As mentioned in the previous section, we apply the same principle to AdamW method to obtain rAdamW method which will be used in the numerical comparison section. The pseudo code of rAdamW is given in Algorithm 4.

Algorithm 4 rAdam Algorithm

Require: Initial value of θ_0; Decay rates values $\beta_1 = 0.9$, $\beta_2 = 0.999$;
Require: Learning rate $\alpha = 0.001$; $\epsilon_r = 10^{-5}$; $\lambda \in \mathbb{R}$
Require: $m_0 \leftarrow 0$; $v_0 \leftarrow 0$; $t \leftarrow 0$

1: **while** iteration $<$ max iterations **do**

Compute gradient:

$$g_t \leftarrow \nabla_\theta \nabla_\theta f_t(\theta_{t-1})$$

Update parameters:

$$\begin{aligned} t &\leftarrow t+1 \\ m_t &\leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &\leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &\leftarrow \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &\leftarrow \frac{v_t}{1-\beta_2^t} \\ \theta_t &\leftarrow \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t + \lambda\theta_{t-1} \end{aligned}$$

Constraints handling

if $(CV(\theta_t, \underline{\theta}, \bar{\theta}) > h\_max)$:

$$\theta_t \leftarrow CF(\theta_t, \underline{\theta}, \bar{\theta})$$

end if

Restart phase

if $(abs(OF(\theta_t) - OF(\theta_{t-1}) < \epsilon_r)$:

$$restart_{count} \leftarrow restart_{count} + 1$$

if $(restart_{count} = restart_{max})$:

$$\theta_t \leftarrow LHS$$

end if

else:

$$restart_{count} \leftarrow 0$$

end if

2: **end while**
3: **return** θ_t

4 Numerical results

This section presents comparative numerical results. After briefly presenting the comparison methodology, we conduct a test over a set of 38 well-known analytical benchmark functions and repeated 100 times, thus the test considers in total of 3800 benchmark functions. We compare the Adam method, and 3 versions of restarts: rAdam, rAdamW, rAdam$-L2-reg$.

Following the methodology proposed in [30], we compare the solvers' performances based on the number of evaluations of the objective function. This metric is a classical indicator for applications involving expensive objective-function evaluations where a solver is evaluated through its capacity to achieve a given function reduction within a limited budget of evaluations of the objective function.

In our comparisons we consider that a method solves a problem if it provides a solution $theta^*$ satisfying the following criterion on the objective-function value:

$$f(\theta_0) - f(\theta^*) \geq (1 - \tau)(f(\theta_0) - f^*), \tag{6}$$

where, in the sequel, f^* denote the best function value found by any solver (or the global-minimum value, if known), x_0 is the starting point for each solver (or the best point of the initial interpolation points), and τ is the desired accuracy, a user-defined tolerance value (in our tests, $\tau = 10^{-3}$). If a solver does not provide a solution that satisfies (6), we consider that it fails.

Performance (see [30]) is complementary tool to compare solvers on a collection of problems.

For a given a collection of test problems, the performance profile of a solver displays the fraction of the problems that are solved by the solver with respect to some performance ratio. In our comparisons, we use the ratio between:

- the number of objective-function evaluations required to reach the chosen accuracy τ in (6) for a given solver, with:
- the number of objective-function evaluations required by the most efficient of the compared solvers (to reach the same accuracy).

The performance profile of a solver depends therefore on the other solvers tested. For instance, the value of the performance profile of a given solver for a performance ratio of 2 is the number of problems solved by this solver within less than twice the number of evaluations required by the most efficient solver for each problem.

We also use the mean and median of best OF through out iterations to see how the objective functions evolute. Since the amplitude of objective function values for 38 functions are in very large range. In order to have a fair comparison, we propose normalizing the objective function values to $[0, 1]$ then calculate the mean and median of each solvers through 3800 run in total.

Table summarizes the options and main parameter values used in the comparison for the 4 solvers under study: Adam, rAdam, rAdamW, rAdam$-L2 - reg$.

Table 2 Solver parameters and options used for the benchmark

Option name	Option value
Maximal number of iteration	1000
Initial DoE	LHS (same initial point for 4 solvers)
Learning rate α	0.01
First moment decay rate β_1	0.9
Second moment decay rate β_2	0.99
Threshold for constraints ϵ_c	10^{-7}
Threshold for restart ϵ_r	10^{-5}
Weight decay λ	0.01
$max_{restart}$	10

The Table 3 lists the information of the 38 optimization problems that we selected in our test.

Table 3 Analytical necklace-optimization benchmark problems

Source instance	**Dimension**	Domain	**reference**
Branin	2	$[-5, 10] \times [0, 15]$	[8]
Camel	2	$[-3, 3] \times [-2, 2]$	[8]
Ex4-1-1	1	$[-2, 11]$	MINLPLib [1]
Ex4-1-2	1	$[1, 2]$	MINLPLib [1]
Ex8-1-1	2	$[-1, 2] \times [-1, 1]$	MINLPLib [1]
Ex8-1-4	2	$[-2, 4] \times [-5, 2]$	MINLPLib [1]
Goldstein-Price	2	$[-2, 2] \times [-2, 2]$	[8]
Hartman3	3	$[0, 1]^3$	[8]
Hartman6	6	$[0, 1]^6$	[8]
Gear	4	$[12, 60]^4$	MINLPLib2 [2]
Gear4	5	$[12, 60]^4 \times [0, 100]$	MINLPLib2 [2]
Least	3	$[0, 600] \times [-200, 200]$ $\times [-5, 5]$	MINLPLib [1]
nvs02	5	$[0, 200]^5$	MINLPLib2 [2]
nvs03	2	$[0, 200]^2$	MINLPLib2 [2]
nvs04	2	$[0, 200]^2$	MINLPLib2 [2]
nvs06	2	$[1, 200]^2$	MINLPLib2 [2]
nvs07	3	$[0, 200]^3$	MINLPLib2 [2]
nvs09	10	$[3, 9]^1 0$	MINLPLib2 [2]
nvs14	5	$[0, 200]^5$	MINLPLib2 [2]
nvs15	3	$[0, 200]^3$	MINLPLib2 [2]
nvs16	2	$[0, 200]^2$	MINLPLib2 [2]
Perm6	6	$[-6, 6]^6$	[19]
Perm8	8	$[-1, 1]^8$	[19]
Prob03	2	$[1, 5]^2$	MINLPLib2 [2]
Rbrock	2	$[-10, 5] \times [-10, 10]$	MINLPLib [1]
Schaeffer-f7-12-1	12	$[-50, 50]^{12}$	MINLPLib2 [2]
Schaeffer-f7-12-2	12	$[-50, 50]^{12}$	MINLPLib2 [2]
Schoen-6-1	6	$[0, 1]^6$	[24]
Schoen-6-2	6	$[0, 1]^6$	[24]
Schoen-10-1	10	$[0, 1]^{10}$	[24]
Schoen-10-2	10	$[0, 1]^{10}$	[24]
Shekel5	4	$[0, 10]^4$	[8]
Shekel7	4	$[0, 10]^4$	[8]
Shekel10	4	$[0, 10]^4$	[8]
Sporttournament	15	$[0, 1]^{15}$	MINLPLib2 [2]
st-mqp1	5	$[0, 1]^5$	MINLPLib2 [2]
st-mqp3	2	$[0, 3] \times [0, 50]$	MINLPLib2 [2]
st-test1	5	$[0, 1]^5$	MINLPLib2 [2]

The results obtained with Adam, rAdam, rAdamW, and rAdam$-L2-reg$ over 100 runs of 38 analytical problems are presented under the form of performance profile (Figure 1), mean best OF (Figure 2) and median best OF (Figure 3). The required accuracy for the performance profile $\tau = 10^{-3}$. For all the numerical results, Adam results are displayed in blue, rAdam results in orange, rAdamW results in green, and rAdam$-L2-reg$ results in red.

These three figures show that for this benchmark, the three versions of restart Adam, i.e., rAdam, rAdamW, rAdam$-L2-reg$ outperform the standard Adam.

The performance profiles figure 1 records Adam method succeed to solve 26.5% of the 3800 runs, whereas rAdam succeed to solve 49.6%, rAdamW 76.4% and rAdam L2 51.8%. The restart Adam with weight decay has

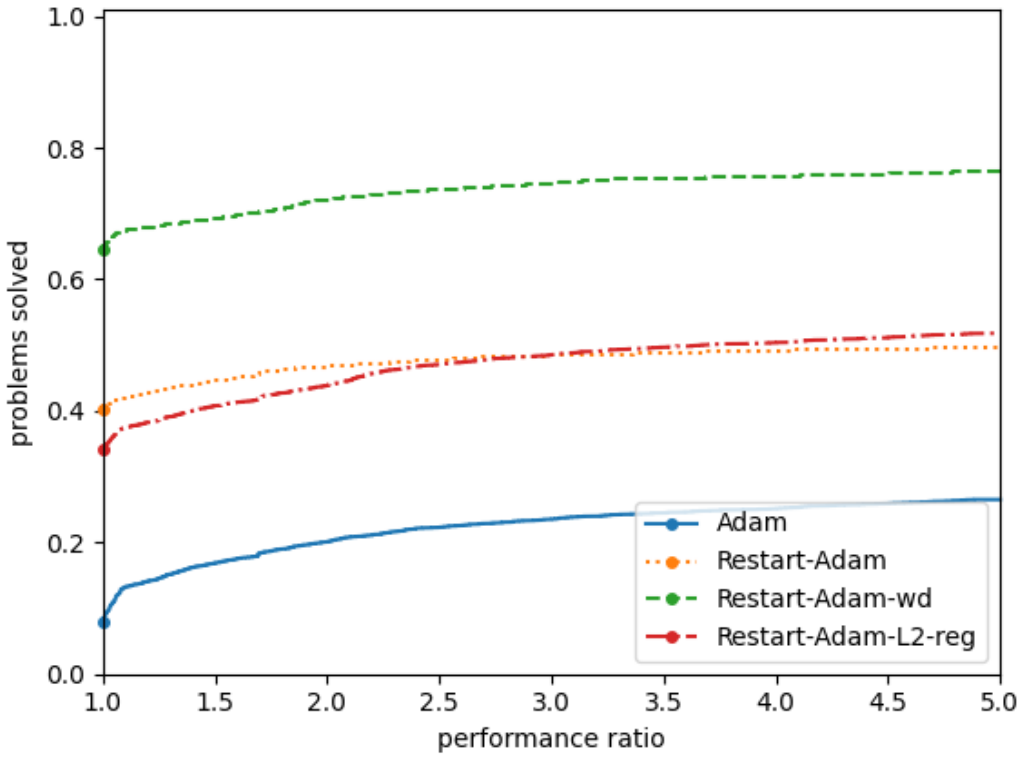

Figure 1 Performance profiles of the 4 solvers for 3800 runs.

shown the best performance among all the 4 solvers, when the L2 version does imrpove but not much compare to the restart version, which coherent with the highlight from the paper [15].

The mean and median best OF of 4 solvers in figures 2, 3 illustrate an efficient convergence to good solutions of 3 start versions.

5 Conclusion and Perspectives

In this paper we addressed bound constrained non-linear optimization problems by coupling restart strategy and Adam method. We tested the 3 versions of restart Adam: rAdam, rAdamW, and rAdam$-L2-reg$, together with the standard Adam algorithm over a set of 38 well-known benchmark functions run over 100 times. The preliminary results show a potential performances of the 3 restart versions. These 3 methods outperform the standard method in terms of both robustness and the number of iteration needed to find the optimal solution on this benchmark. In particular, the restart version with weight decay, i.e., rAdamW has illustrated the best results among 4 tested methods.

There remains several oppositeness of improvement and apply in application as future work. For the improvement, the first idea is to use variance-reducing techniques to accelerate the convergence of the method. The second idea is to use tuning hyper-parameters techniques, for instance cross validation, to find the best combination of hyper-parameters.

The application of this work is to solve the Loss function in the backpropagation of neuron-based prediction models. As the results, the restart Adam with weight decay or restart Adam will be used in the framework of our future work.

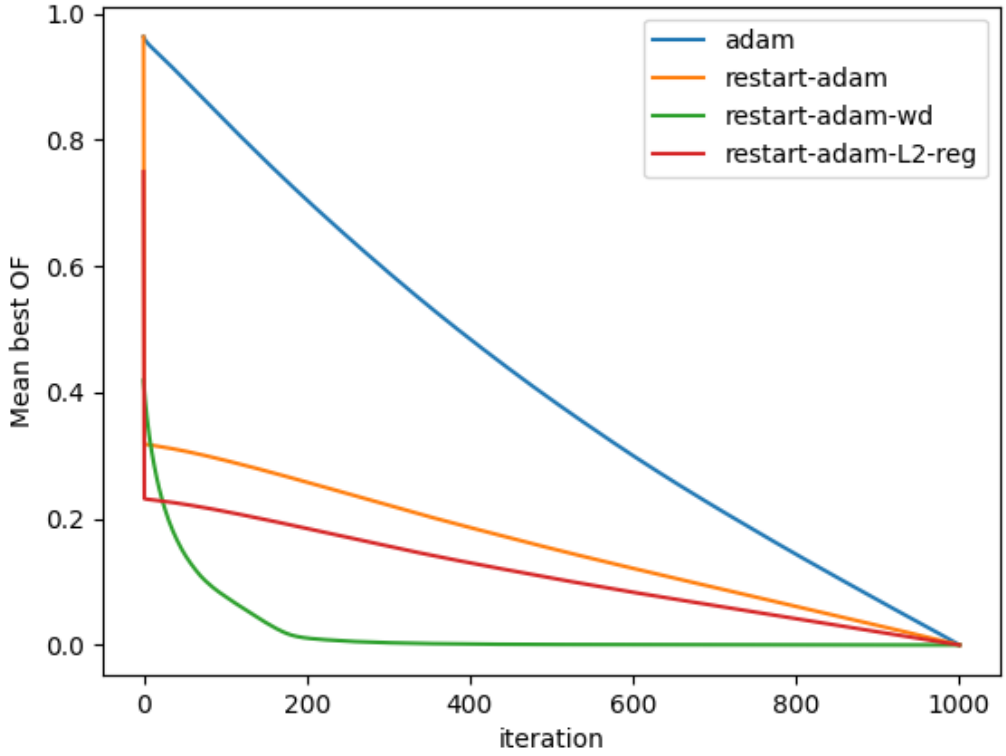

Figure 2 Mean best OF of the 4 solvers for 3800 runs.

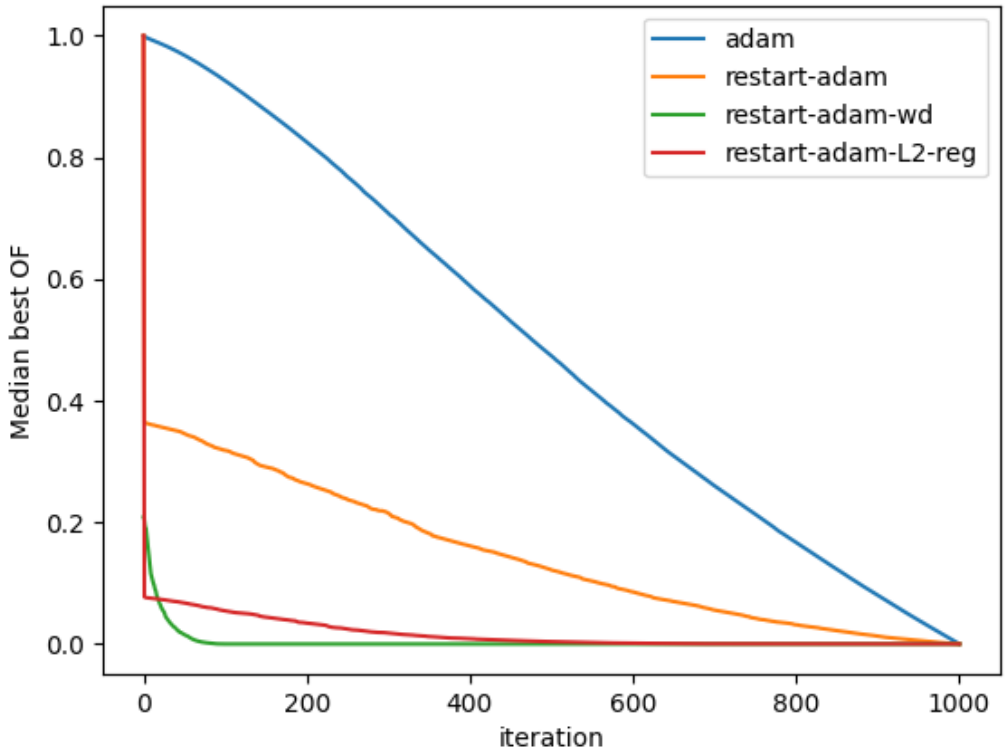

Figure 3 Median best OF of the 4 solvers for 3800 runs.

References

1 MINLPLib. http://www.minlplib.org.

2 MINLPLib2. http://www.gamsworld.org/minlp/minlplib2/html/.

3 Goodfellow I.; Bengio Y.; Courville A. *Deep Learning*. MIT Press, Cambridge, UK, 2016.

4 Li M.; Zhang T.; Chen Y.; Smola A.J. Efficient mini-batch training for stochastic optimization. In *In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 661–670, 2014.

5 Charles Audet, Andrew Conn, SÃ©bastien Le Digabel, and Mathilde Peyrega. A progressive barrier derivative-free trust-region algorithm for constrained optimization. *Computational Optimization and Applications*, 71, 11 2018.

6 M. D. McKAY; W. J. Conover; R. J. Beckman. Latin hypercube sampling. *Technometrics*, 21, 1979.

7 Coralia Cartis, Lindon Roberts, and Oliver Sheridan-Methven. Escaping local minima with derivative-free methods: A numerical investigation. *Optimization and Control (math.OC)*, 2018.

8 L.C.W Dixon and G.P Szegö. The global optimization problem: An introduction. *In: Dixon, L.C.W, Szegö, G.P (eds.) Towards Global Optimization, North Holland*, pages 1–15, 1975.

9 Timothy Dozat. Incorporating nesterov momentum into adam. In *In Proceedings of the 4th International Conference on Learning Representations*, pages 1–4, 2016.

10 Elshamy R.; Abu-Elnasr O.; Elhoseny M. et al. Improving the efficiency of rmsprop optimizer by utilizing nestrove in deep learning. *Sci Rep 13*, 2023.

11 M. Gendreau and J.Y. Potvin. *Handbook of Metaheuristics*, volume 272. International Series in Operations Research & Management Science. Springer, 2019. 3rd edition.

12 H.M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227, Mar 2001.

13 Sarit Khirirat, Hamid Reza Feyzmahdavian, and Mikael Johansson. Mini-batch gradient descent: Faster convergence under data sparsity. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2880–2887, 2017.

14 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 30, 2014. https://arxiv.org/abs/1412.6980.

15 Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv*, 2019. https://arxiv.org/abs/1711.05101.

16 W. J. Mckay M. D.; Beckman, R. J.; Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *American Statistical Association and American Society for Quality*, 42:55–61, 2010.

17 Giacomo Nannicini. On the implementation of a global optimization method for mixed-variable problems. *Open Journal of Mathematical Optimization*, 2: article no. 1, 2021.

18 Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $o(\frac{1}{k^2})$. pages 543–547, 1983. https://api.semanticscholar.org/CorpusID:202149403.

19 A. Neumaier. Neumaierś collection of test problems for global optimization. pages 1–15, Retrieved in May 2014. http://www.mat.univie.ac.at/~neum/glopt/my_problems.html.

20 Pontes Vitor Notini. *Algorithms for the Multiperiod Workforce Scheduling and Routing Problem with Dependent Task*. PhD thesis, Universidade Federal de Minas Gerais, 2020.

21 Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *ArXiv*, 30, 2018. https://arxiv.org/abs/1904.09237.

22 Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017. https://arxiv.org/abs/1609.04747.

23 Ganie A.G.; Dadvandipour S. From big data to smart data: a sample gradient descent approach for machine learning. *J Big Data 10*, 162, 2023.

24 Fabio Schoen. A wide class of test functions for global optimization. *Journal of Global Optimization*, 3(2):133–137, 1993.

25 Robbins H.; Monro S.r. A stochastic approximation method. *Ann. Math. Stat.*, 22, 1951.

26 Michel Vasquez et Christophe Wilbaut Sylvain Boussier, Hashimoto Hideki. Un algorithme grasp pour le probleme de planification de techniciens et d'interventions pour les telecommunications. *RAIRO - Operations Research 43.4*, 2009.

27 Thi Thoi Tran. *Nonlinear optimization of mixed continuous and discrete variables for blackbox simulators*. Theses, Université Paul Sabatier - Toulouse III, October 2021.

28 Christelle Gueret et Andres Medaglia Victor Pillac. On the dynamic technician routing and scheduling problem. https://hal.science/hal-00739781/document, 2022.

29 Ky Khac Vu, Claudia d'Ambrosio, Youssef Hamadi, and Leo Liberti. Surrogate−based methods for blackbox optimization. *International Transactions in Operational Research*, 2017.

30 Jorge More; Stefan Martin Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20:172–191, March 2009.

31 Stefan Martin Wild. *Derivative-Free Optimization Algorithms for Computationally Expensive Functions*. PhD thesis, Cornell University, Cornell University, Ithaca, NY, USA, 2009.

32 Feng Wang; Heng Zhang. A particle swarm optimization algorithm for mixed-variable optimization problems. *Swarm and Evolutionary Computation*, 2021.