Ising Evolution and the Benefits of Optimization Diversity

Anonymous Authors

***Abstract*—In this paper, we study the problem of finding the global minima of a given function. Specifically, we consider complicated functions with many local minima, as is often true for real-world data mining losses.**

We do so by applying a model from theoretical physics to create an ising model-based evolutionary optimization algorithm.

Our algorithm creates stable regions of local optima and a high potential for improvement between these regions. This allows finding global minima more accurately than comparable methods and has promising applications to ensembles.

***Index Terms*—Evolutionary Optimization, Evolutionary Data Mining, Optimization Algorithms, Physics inspired algorithms, Ensembles**

I. Introduction

The development of gradient descent [1] made it possible to create and improve many data mining algorithms by for example allowing methods based on neural networks to be trained efficiently. Still gradient descent methods have their limits. Most importantly, gradient descent requires a continuous loss function and thus can not be applied to every type of model: While historically we preferred to describe complicated relations by intricate mathematical formulas. But using gradient descent, we have to hide these precise mathematical objects in huge neural network matrices. This is not always possible [2] and made it almost impossible to understand the underlying relation [3].

A potential solution to improve this, would be to employ more specialized models, and optimize them using a different paradigm like for example evolutionary optimization [4], [5]. While evolutionary optimization is often much more time costly, variants are applicable to almost every minimization task. Neither the loss function, nor its inputs need to be continous, and it can even allow optimizing more complicated structures, like lists of variable length, or the tree describing a mathematical function.

However because of the increased time cost, evolutionary optimization is usually specialized only to find a good enough optimum. While it is less able to get stuck in a local optima than gradient descent methods, we will see later that they are usually not able to even come close to the global optima. While this is a good approach, for example in hyperparameter optimization [6], where function evaluation are very costly, and the performance increase might often be tiny, the situation is arguably different in data mining. When searching for an explainable model, only locally optimal solutions also represent a wrong explanation of the underlying process, as there is an explanation that describes it better. This also implies that a locally optimal solution generalizes less well to a new dataset.

To test algorithms in this problem situation, we suggest a test function in this paper, which has many local minimas of different qualities. These are also distributed in a way, that makes it easy to evaluate how many local minimas were found.

Using this test loss, we suggest a new evolutionary optimization paradigm, which is especially good at finding global minima. This algorithm is inspired by the Ising model [7] from statistical physics, which is often studied because it exhibits the emergence of regions with slightly different behaviour. We observe that when extending this algorithm to evolutionary optimization, the emergence of regions remains. This allows our optimization algorithm to consider a much wider range of possible solutions, making it significantly more likely to find the global minima.

Additionally, we notice that our optimization can be very helpful for ensemble models [8]. While finding the global optimum of a loss function usually implies a valuable data mining model, combining different models into an ensemble can be even more effective. This is because while a single model might make mistakes, these mistakes are often canceled by averaging multiple ones [9].

But to do so, we must find multiple significantly different solutions to a given data mining task, as repeating the same errors does not allow them to cancel out. This is not necessarily an easy task, as it often relies on different initializations finding different local minima. And even if it works, it requires running a potentially very slow evolutionary optimization algorithm many times.

The solution diversity our algorithm provides can solve both problems. Instead of requiring multiple optimizations, we can benefit from different solutions in different regions.

Concretely our contributions in this paper are threefold:

- Definition of an easy-to-study function for the problem of evolutionary global minima optimization
- Suggestion of a new physics-based optimization algorithm
- Evaluation of its properties and benefits.

To help with further research, our code is available at https://anonymous.4open.science/r/IsingEvo-E835.

II. Related Work

A. Evolutionary Optimization

The problem of optimization [10] is central to machine learning. Because of this there are many ways to solve this. A common method is Gradient Descent [1], which efficiently trains complicated neural networks. But Gradient Descent requires a continuous loss function and does not handle randomness well. Both are conditions that restrict the type of problems that can be optimized. Additionally, Gradient Descent can often not find the global minimum of a loss function, but only a local minimum.

Instead, evolutionary optimization solves some of these problems. Instead of requiring a continuous loss function differential, it alters the current input randomly and either accepts or rejects the changed value depending on whether it decreases the loss function. This allows applying it to both functions of non-numeric values, as well as to loss functions that are not differentiable [4].

Besides this, there are many variants of evolutionary optimization algorithms. Notably are Genetic Algorithms [11], Particle Swarm Optimization [12], and Differentiable Evolution [13]. A few specific algorithms are described in Sec. III-A.

When searching for the global optimum, the optimization algorithm needs to be able to make a locally suboptimal choice. While this is easy to implement for an evolutionary algorithm, by adding a probability to create a suboptimal choice, it is not optimal. Theoretically, it might be possible to go from a local optimum to a different one, but this is still very unlikely, as it often requires choosing the suboptimal choice many times in a row. Instead we suggest a method to increase the variance of inputs tried, which makes it considerably more likely to find the global minima.

B. *Ising Model*

We do so by using an Ising model inspired algorithm. This model, studied 1924 by Ernest Ising in his dissertation [7], was initially used to model spin interactions: Two neighboring spins are either aligned ($s_i \cdot s_j = 1$) or opposite to each other ($s_i \cdot s_j = -1$), and the energy of the system (similar to a loss function) depends on this interaction $E = -\sum_{i\ neighbour\ of\ j} s_i \cdot s_j$.

It is one of the most commonly studied models in statistical physics, because it is one of the simplest models to show a phase transition [14]. This means its behaviour is drastically different based on one control parameter (the temperature T or the more commonly used inverse temperature $\beta = \frac{1}{T}$): For low temperatures spins are aligned and stable regions form. But for high temperatures the spins become chaotic.

Notable is also that this effect depends on the dimension in which it is studied. In one dimension, every spin has only two neighbours, and the system behaves unremarkable. But it shows the aforementioned phase transitions in two or more dimensions (four or more neighbors).

While the Ising model has originally been used to model magnetism, by now there are many more appiclications. From protein folding [15] to social sciences [16] and modeling the stock market [17] , by now there are many more problems, that can be modeled by an Ising model.

C. *Science inspired Machine learning*

There is a deep link between machine learning and natural sciences like physics. Not only are machine learning methods often used to conduct scientific research [18]–[20], but knowledge from science is also often used to improve machine learning methods, invent new ones, or explain existing ones.

Statistical physics can be used to explain the behavior of neural networks [21] and directly inspired Boltzmann machines [22]. Particle swarm optimization [12] and Hopfield networks [23], are inspired by the behavior of animals and neurons, respectively.

Notably is Simulated Annealing [24], a commonly used evolutionary optimization strategy inspired by metallurgy research, which uses a temperature variable to control the randomness of a system and guide it to an optimal state. We will use this algorithm as a competitor.

And also Ising models have been used to understand the training of neural networks [25] and have inspired markov random fields [26], which are used to model joint probability distributions. Still to the best of our knowledge, they have not been used as a basis for an evolutionary algorithm.

D. *Ensemble methods*

Replacing a complicated model by an ensemble of models can often be a very effective way of increasing the model performance and making it more reliable. While there are many approaches to ensembles (like stacking or boosting [27], [28]), the most commonly studied task combines multiple independent models using a combination function [29]. For this to help, we require the errors of ensemble submodels to differ from each other [9].

This difference can be achieved in one of two ways. Either by changing the setup of each submodel, by for example training different types of models. But this makes the resulting combination task harder, and can often (especially for unsupervised tasks) limit the resulting performance [30], [31].

So instead, submodels are often identically, except to a random seed [32]. But combining this with an optimization algorithm like gradient descent [1] or evolutionary optimization [4] means we can only achieve the variance needed for the ensemble to work when there are many different equally easily reachable local minima. Since this assumption is often not fulfilled and requires retraining many submodels many times, we instead suggest a new optimization method that can directly find a set of local minima and thus a whole ensemble during training.

III. Ising based Evolution

We describe our approach in Alg. 1. We show here the two-dimensional version, but this could easily be adapted by mostly changing what it means to be neighboring indices. While Ising based evolution is similar to cellular evolution [33] as updates happen locally, the likelihood of changing a value is given by Eq. 1

$$p(old, new) = \begin{cases} 1 & \text{if } old > new \\ \exp(\beta \cdot (old - new)) & \text{else} \end{cases} \quad (1)$$

When minimizing the existing value, the probability is $p = 1$, and such the update step happens. But if not improving it, there is a still a chance that the value gets updated. This is necessary for the method to be able to not get stuck in local minima. Here this probability is bigger, the smaller β is. This is the same probability used in the usual Ising model, to go to a state of different energy.

We chose here a mutation of our value that is either sexual or asexual: Either two values are averaged, or one value is changed according to a normal distribution. Still, this mutation is not part of our algorithm and can be changed at will. We will also use a similar mutation step in our competitor algorithms.

Algorithm 1 Ising based evolution

```
Require: $f(x)$
Require: $Region$, $wid$, $hei$, $\beta$, $n_{steps} > wid \cdot hei$
Require: $uniform(Region)$, $normal(mean, std)$
  for $i < wid$ do
    for $j < hei$ do
      $population_{i,j} \leftarrow uniform(Region)$
      $qual_{i,j} \leftarrow f(population_{i,j})$
      $n_{steps} \leftarrow n_{steps} - 1$
    end for
  end for
  $solution \leftarrow population_{argmin(qual)}$
  $qual_{min} \leftarrow min(qual)$
  while $n_{steps} > 0$ do
    $i \leftarrow uniform(0, wid)$
    $j \leftarrow uniform(0, hei)$
    $\bar{i}, \bar{j} \leftarrow neighbour(i, j)$
    $test \leftarrow \frac{population_{i,j} + population_{\bar{i},\bar{j}}}{2}$
    if $uniform(0,1) < 0.5$ then
      $test \leftarrow normal(mean = population_{i,j}, std = 100)$
    end if
    $quality_{test} \leftarrow f(test)$
    if $uniform(0,1) < \exp(\beta \cdot (qual_{i,j} - qual_{test}))$ then
      $population_{i,j} \leftarrow test$
      $qual_{i,j} \leftarrow qual_{test}$
      if $qual_{test} < qual_{min}$ then
        $solution \leftarrow test$
        $qual_{min} \leftarrow qual_{test}$
      end if
    end if
    $n_{steps} \leftarrow n_{steps} - 1$
  end while
  return $solution$ =0
```

A. Comparison algorithms

We chose five different evolutionary optimization strategies to compare our algorithm to.

Cellular evolution [33] is a similar approach to Alg. 1. The main difference is that the probability of choosing a suboptimal solution is fixed at 10%. While this change seems small, it would already be enough to disable the emergence of a phase transition in the physical model.

Simulated Annealing [24] is also a physics-inspired algorithm in which one solution is updated based on mutations, with a probability given by Eq. 1. As it is inspired by the cooling of a metal, the value of β increases with the number of update steps i. A common way is to parameterize this is $\beta = \beta_0 \cdot i$. But to allow more variance with a high number of update steps, we choose here $\beta = \beta_0 \cdot \sqrt{i}$

Mutation evolution is a much simpler algorithm, which generates a new value through a standard distribution $newvalue = normal(mean = currentValue, std = 100)$. The value is updated if the new value has a lower score than the old one. We use this algorithm here similar to an ablation study, since we use a similar update step in our algorithm. Still, Mutation evaluation is a case of differential evolution [13].

Mixture evolution is also a case of differential evolution [13], which we use like an ablation study. Here we are given a population of 100 random values, and we randomly average the value of two instances to generate a new one. If this generated value reaches higher performance than one of its parents, or if not also in 10% of cases, we keep it. While Mutation evolution could be seen as inspired by asexual reproduction, Mixture evolution is more similar to sexual reproduction.

Random Search is our final and simplest algorithm. Instead of using a procedure to update the current value, we randomly guess it. While this algorithm cannot get stuck in local minima, it can also not benefit from any local feedback. The likelihood of reaching the optimal state is the same, whether in the second-best state or in the worst one.

IV. Test Description

To test how good a particular algorithm is at finding the global minima of a function, we need to test it on a function with many local optima. For this we suggest minimizing Eq. 2.

$$f(x) = \|sin(\frac{x+1}{100})\|, \; x \in \mathbb{Z} \tag{2}$$

We visualize this function in Fig. 1.

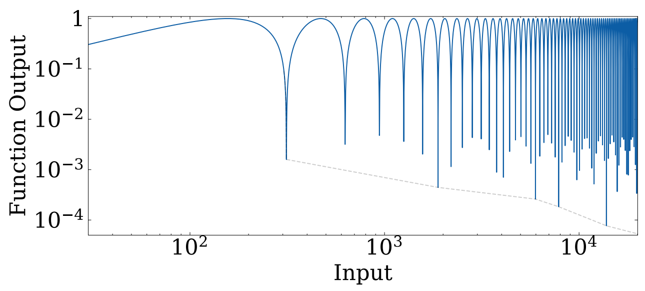

Fig. 1. Visualization of Eq. 2. Because of the double logarithmic axes, the sinus looks unusual. But this highlights that the local minima are not randomly distributed.

While Eq. 2 is almost a simple sinus, the evaluation over the range of integers means that most minima are only local minima (See Theorem 1). Still, the periodicity and improving approximation of a real number give the function some general properties. For example $f(x) \approx 0 \rightarrow f(n \cdot x) \approx 0 \; \forall n \in \mathbb{N}$.

Theorem 1. *The only global optima to function 2 is $x = -1$.*

Proof.

$$f(-1) = \|sin(0)\| = 0; \; f(x) \geq 0 \; \forall x$$

$$f(x) = 0, x \neq -1 \rightarrow \frac{x+1}{100} = k \cdot \pi, k \in \mathbb{Z} \setminus 0$$

$$\rightarrow \pi = \frac{x+1}{100 \cdot k} \in \mathbb{Q}$$

□

Thus we can exclude the global minima by only considering $x \in \mathbb{N}$. We additionally only evaluate $x < 10^5$ for there to be an optimal result and to put evaluation costs in perspective. In total, this means there are 318 local minima of our function.

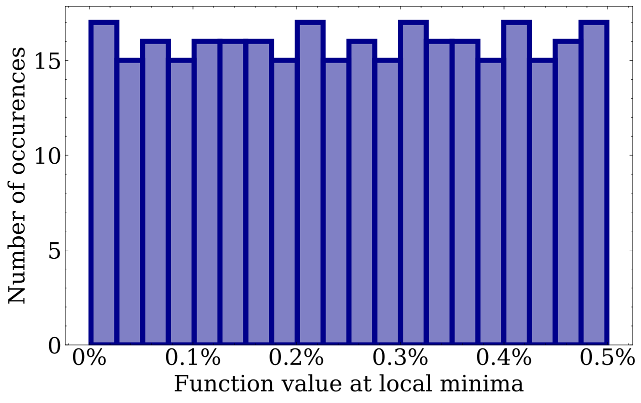

Fig. 2. Distribution of locally minimal function values.

Further, as shown in Fig. 2, the quality of minima is equally distributed. This means, we can use the average found function value, to evaluate an optimization algorithm: When an algorithm only finds a local minima, its average performance will be about $0.25\% = 2.5 \cdot 10^{-3}$. A perfect algorithm instead would find the lowest point of our range $f(84822) \approx 1.64 \cdot 10^{-5}$.

V. Results

Using this setup, we randomly initialize and run every algorithm 100 times and measure the average found result. Each algorithm can evaluate the function up to 10^5 times to find the lowest function value. In theory, every function value in our range could be evaluated, guaranteeing that the lowest value is found. Still, no algorithm reliably finds the lowest performance all the time since we allow our algorithm to make the same function call multiple times. While repetitions should usually be cached in an actual application, this implicitly allows us to punish algorithms that do not efficiently explore the entire search space and guarantees that algorithms run in finite time.

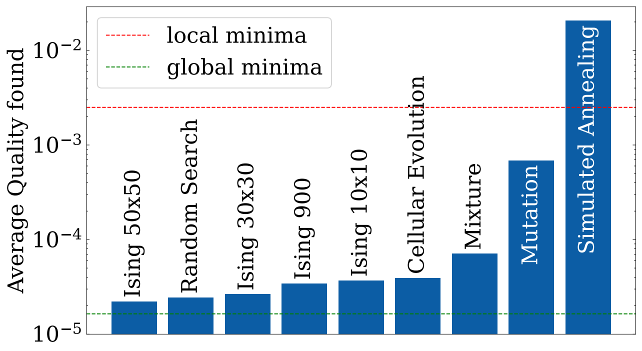

Fig. 3. Average minimal value found for different evolutionary optimization algorithms.

The resulting average function values are shown in Fig. 3.

The Mutation algorithm does not improve much upon finding a local minimum. And Simulated Annealing also freezes quickly. Interestingly, the Mixture algorithm does much better than both. This is likely, because if $f(x) \approx 0 \approx f(y)$ this implies that $f(\frac{x+y}{2}) \approx 0$ or $f(\frac{x+y}{2}) \approx 1$, which is a property we do not expect to generalize to different tasks.

Each of our Ising based models performs much better, with the best reaching an average performance of $\approx 2.2 \cdot 10^{-}5$, close the the theoretical optimum. We will spend the rest of this section understanding this further. Importantly, our Ising based optimization works better than the related Cellular Evolution. We used here $\beta = 100$, but choosing a different value does not affect the resulting quality much, up to the limit studies in Section V-A.

The Random Search model seems to be the most robust competitor algorithm. This is a result of our construction: The likelihood that it does not find the optimal value is $(1 - \frac{1}{10^5})^{10^5} \approx 0.3679 \approx \frac{1}{e}$, and we could improve it further, by removing duplicate function calls. Still while it can find absolute minima relatively commonly, it cannot benefit from local feedback. This means its practical applications are limited, as most practical functions do not contain as many local minima, and a higher number of optimizable parameters exponentially increases the number of samples to be tried.

We study this further by comparing the performance of our Ising based approach and the Random Search as a function of the number of function evaluations allowed. This is shown in Fig. 4.

It seems that for the first function evaluations, each algorithm shown performs very similarly. This is because the population is first initialized randomly, making the Ising based algorithm equivalent to the Random Search for 10^2 to $25^2 \approx 10^{2.8}$ function evaluations. Afterward, each Ising based algorithm decreases first faster than the Random Search algorithm, which means that they are faster reaching a better solution than the Random Search and showing the benefit of local feedback in our algorithms. The change might look small, but because of the double logarithmic axes, even a small change can mean a tenfold speed increase. But the model performance converges at some point and does not

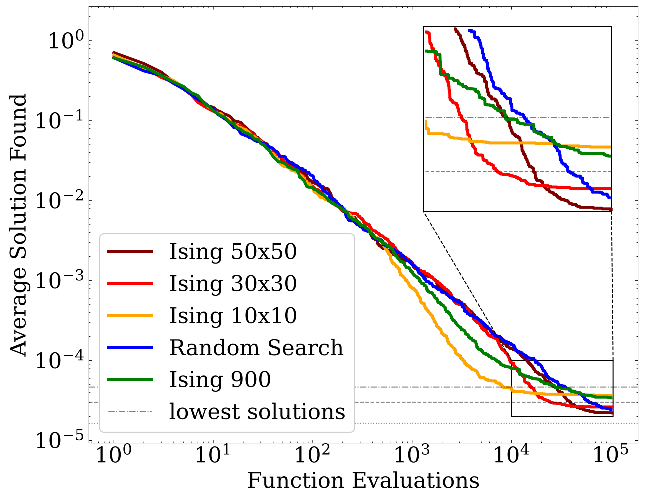

Fig. 4. Average lowest value found as a function of the number of function evaluations used. We zoom in on this region because the most interesting changes happen in the lower right corner. We also add the lowest three possible solutions of Eq. 2 to be found as horizontal lines.

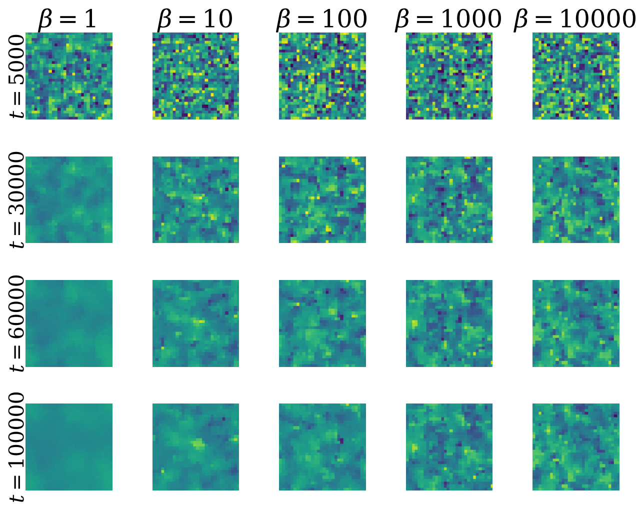

Fig. 5. Current state (color) over the population, as a temperature and update step function.

improve further. Both the resulting quality and the cost of evaluations seem to be a function of the size of the population of our algorithm: The bigger the board, the longer we follow the Random Search performance, but the lower is also the resulting quality. And ultimately, there is a point where a 50x50 population converges at a quality below Random Search in $\approx 50\%$ less function evaluations.

As an ablation study, we also show a one-dimensional Ising based model with the same population size of the 30x30 two-dimensional model. But it seems to converge slower and at a worse performance than its two-dimensional alternative.

A. *Phase transitions*

The Ising model is often studied in physics because it is one of the simplest models that can show phase transitions. While for high temperatures (low β), the system is chaotic and almost random, for low temperatures (high β), relatively stable regions of aligned spins are formed. Interestingly, this effect does not appear for the one-dimensional Ising model but only in at least two dimensions. We want to study if a similar transition also appears here.

For this, we show the current state of our population for different temperatures after different numbers of update steps for the 30x30 sized algorithm in Fig. 5.

Similar to the physical effect, we see a relatively stable distribution. After the almost random initialization, each model shows unique structures, which still appear after further update steps, even though they change a bit (For example, the yellow spot in the center of $\beta = 10$).

Also similarly to the physical effect, does this distribution depend on the temperature value. For a high temperature (low β) the resulting structures seem to disappear the most with increasing update steps. But for low temperatures (high β) the results are more chaotic, and we see more stable regions form.

But after 10^5 update steps even the almost convergent distribution of $\beta = 1$ still differs by about $\approx 15\%$ of the available range in its solutions.

To characterize further how the distribution depends on the temperature and update step, we visualize the standard deviation of these images as a function of both. We show a two-dimensional model in Fig. 6 and a one-dimensional one in Fig. 7.

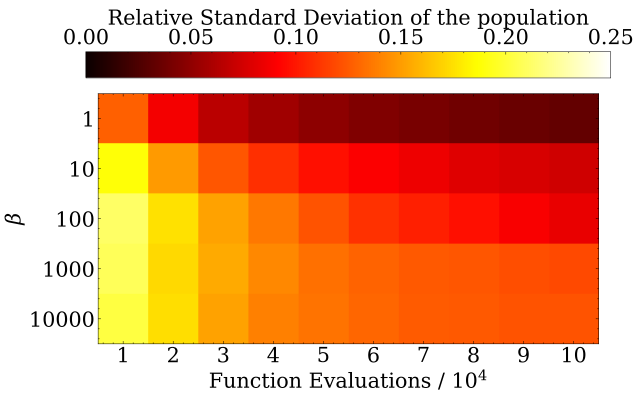

Fig. 6. Relative standard deviation (standard deviation divided by 10^5) of models with different β value during the optimization process for a two-dimensional ising model of population size 30x30.

As we have seen before, in the two-dimensional case the system becomes more chaotic with a higher value of β, while also needing a longer time to converge. We also almost see phases: For a low number of evaluations and a high value of β, the variance is high, while for a high number of evaluations and a low β it is significantly smaller. But importantly, the population variance is not dropping to zero, as the other evolution algorithms show. This is not only an effect of continuing random mutation steps, as these should at most create a relative standard deviation of $\frac{100}{10^5} = 0.001$ (See Alg. 1).

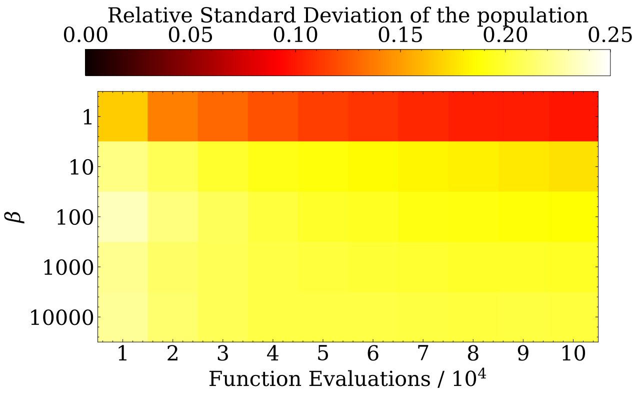

Fig. 7. Relative standard deviation (standard deviation divided by 10^5) of models with different β value during the optimization process for a one-dimensional ising model of population size 900.

Interestingly and similar to the physical effect, the one-dimensional case seems to perform quite differently from the two-dimensional one. While the two-dimensional model shows a gradual transition from the low to the high variance phase, the one-dimensional one freezes out almost completely for every $\beta > 1$.

B. Ensembles

Finally, we want to study how well our Ising based approach can be used to find not only the global minima of a function but multiple different ones that could be combined into an ensemble. To do this, we test not how often the global minima is found but how many of the lowest ten solutions are found. This is shown in Fig. 8.

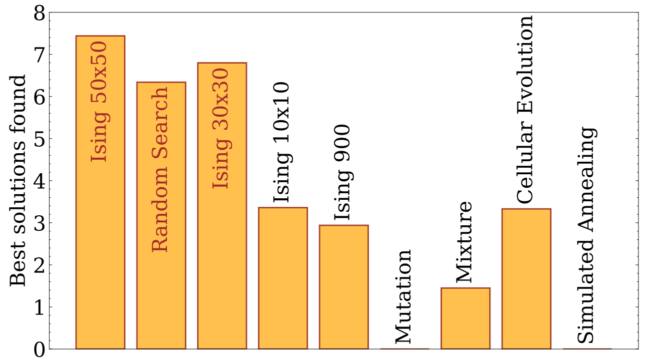

Fig. 8. When not only searching for the best solution but also for the 9 best local optima, how many of these are found on average by a given method.

Our Ising based approach finds the most of these minimas, with on average $7.44/10$ valuable minimas being found. Similar to our analysis in Fig. 3, a higher size of the Ising population seems to help, but here the improvement over the Random Search algorithm grows, as even a 30x30 population is enough to outperform it.

This shows that using submodels from different regions can create ensembles that are better than randomly initializing them. And we can create an ensemble from our results by only requiring one optimization loop instead of multiple ones (one per submodel). Additionally, we could link the number of different solution regions to the number of ensemble submodels needed. But this still requires further study.

VI. Conclusion

This paper introduced an evolutionary optimization algorithm based on Ising models. Our method is able to find global minima even in very complicated loss functions. It does this by leveraging knowledge from theoretical physics, adapting a model that shows the emergence of semi-stable regions, and adjusting these to evolutionary optimization.

We can use these regions to do what we call high variance optimization. This kind of optimization allows us not only to find better optima but also has interesting applications for ensembles.

In further studies, we would like to characterize other shapes of our population (higher dimensionality, nonsquare shapes) and see if we can improve the tradeoff between quality and time cost by using a region-based initialization. Applying this method to various other loss functions and mutation procedures would also be interesting.

Still, we are most interested in the possibility of learning complicated and inherently explainable models (like a tree representing a function) using an algorithm like ours.

References

[1] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[2] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 3839–3848.

[3] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *ArXiv*, vol. abs/2011.07876, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:226964842

[4] C. Blum, R. Chiong, M. Clerc, K. De Jong, Z. Michalewicz, F. Neri, and T. Weise, *Evolutionary Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–29. [Online]. Available: https://doi.org/10.1007/978-3-642-23424-8_1

[5] J. A. Vrugt and B. A. Robinson, "Improved evolutionary optimization from genetically adaptive multimethod search," *Proceedings of the National Academy of Sciences*, vol. 104, no. 3, pp. 708–711, 2007. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0610471104

[6] C. Wang, Q. Wu, M. Weimer, and E. E. Zhu, "Flaml: A fast and lightweight automl library," in *Fourth Conference on Machine Learning and Systems (MLSys 2021)*, April 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/flaml-a-fast-and-lightweight-automl-library/

[7] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift fur Physik*, vol. 31, no. 1, pp. 253–258, Feb. 1925.

[8] A. Zimek, R. J. Campello, and J. Sander, "Ensembles for unsupervised outlier detection: Challenges and research questions a position paper," *SIGKDD Explor. Newsl.*, vol. 15, no. 1, p. 11–22, mar 2014. [Online]. Available: https://doi.org/10.1145/2594473.2594476

[9] K. Ali and M. Pazzani, "Error reduction through learning multiple descriptions," *Machine Learning*, vol. 24, 11 1997.

[10] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[11] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb 2021. [Online]. Available: https://doi.org/10.1007/s11042-020-10139-6

[12] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[13] R. Storn and K. Price, “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997. [Online]. Available: https://doi.org/10.1023/A:1008202821328

[14] A. Taroni, “90 years of the ising model,” *Nature Physics*, vol. 11, no. 12, pp. 997–997, Dec 2015. [Online]. Available: https://doi.org/10.1038/nphys3595

[15] A. Bakk and J. S. Høye, “One-dimensional ising model applied to protein folding,” *Physica A: Statistical Mechanics and its Applications*, vol. 323, pp. 504–518, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437103000189

[16] C. Li, F. Liu, and P. Li, “Ising model of user behavior decision in network rumor propagation,” *Discrete Dynamics in Nature and Society*, vol. 2018, p. 5207475, Aug 2018. [Online]. Available: https://doi.org/10.1155/2018/5207475

[17] L. Zhao, W. Bao, and W. Li, “The stock market learned as ising model,” *Journal of Physics: Conference Series*, vol. 1113, no. 1, p. 012009, nov 2018. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1113/1/012009

[18] C. Xu and S. A. Jackson, “Machine learning and complex biological data,” *Genome Biology*, vol. 20, no. 1, p. 76, Apr 2019. [Online]. Available: https://doi.org/10.1186/s13059-019-1689-0

[19] Y.-H. He, E. Heyes, and E. Hirst, “Machine learning in physics and geometry,” 03 2023.

[20] J. A. Keith, V. Vassilev-Galindo, B. Cheng, S. Chmiela, M. Gastegger, K.-R. Müller, and A. Tkatchenko, “Combining machine learning and computational chemistry for predictive insights into chemical systems,” *Chemical Reviews*, vol. 121, no. 16, pp. 9816–9872, Aug 2021. [Online]. Available: https://doi.org/10.1021/acs.chemrev.1c00107

[21] K. Weng, A. Cheng, Z. Zhang, P. Sun, and Y. Tian, “Statistical physics of deep neural networks: Initialization toward optimal channels,” *Phys. Rev. Res.*, vol. 5, p. 023023, Apr 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.5.023023

[22] A. Patel and R. K. Rama, “An overview of boltzmann machine and its special class,” 09 2020.

[23] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, “Hopfield networks is all you need,” 07 2020.

[24] S. Kirkpatrick, C. Gelatt, and M. Vecchi, “Optimization by simulated annealing,” *Science (New York, N.Y.)*, vol. 220, pp. 671–80, 06 1983.

[25] G. Tkacik, E. Schneidman, M. Berry II, and W. Bialek, “Ising models for networks of real neurons,” 12 2006.

[26] R. Kindermann and L. Snell, *Markov random fields and their applications*. American Mathematical Society, 1980, vol. 1.

[27] M. O. Sandim, “Using stacked generalization for anomaly detection,” Ph.D. dissertation, 2017.

[28] Y. Freund and R. E. Schapire, “A short introduction to boosting,” in *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1999, pp. 1401–1406.

[29] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug 1996.

[30] S. Klüttermann and E. Müller, “Evaluating and comparing heterogeneous ensemble methods for unsupervised anomaly detection,” 06 2023, pp. 1–8.

[31] A. Lazarevic and V. Kumar, “Feature bagging for outlier detection,” vol. 21, 01 2005, pp. 157–166.

[32] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/a:1010933404324

[33] N. Noman and H. Iba, “Cellular differential evolution algorithm,” in *AI 2010: Advances in Artificial Intelligence*, J. Li, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 293–302.