Hierarchical Spiral Discovery Networks for Multi-Layered Exploration-Exploitation Tradeoffs

Adam B. Csapo

Department of Informatics, Széchenyi István University
Győr, Hungary
csapo.adam@sze.hu

Abstract: The Spiral Discovery Network (SDN) was recently proposed as a tool for automated parametric optimization based on the Spiral Discovery Method. SDN can be seen as a heuristic optimization approach that offers tradeoffs between exploration and exploitation without having recourse to explicit gradient-based feedback information (unlike classical neural networks) and without requiring hand-coded representations of metaheuristic constructs such as genotypes (unlike genetic algorithms). In this paper, the properties of the SDN model are further explored, and two extensions to the model are proposed. The first extension corrects a shortcoming of the original model and has to do with the assignment of credit among different output components based on the most recent performance of the model at any given time. The second extension consists of using multiple SDN cells in a hierarchical architecture, which enables a fuller and more effective exploration of the parametric space. The improvements provided by the two extensions are validated on the same set of simulations discussed in earlier work.

Keywords: Spiral Discovery Network; Spiral Discovery Method; Non-convex optimization; Exploration versus exploitation

1 Introduction

The debate on whether evolutionary methods or neural networks are better suited to tackle problems in artificial intelligence is strongly related to the classical debate on nature versus nurture [5]. Clearly, both classes of approach are important components of human intelligence, and both have their pros and cons from a computational point of view. Evolutionary methods are well suited to finding good candidate solutions in large parametric spaces, however they often rely on the availability of hand-coded genotypes that are expected to yield useful results as the classical operations of recombination and genetic mutation are applied to them – a constraint that can be considered as arbitrary and limiting given that it is not directly relevant to the problem domain itself. At the same time, neural networks are known to be capable of automatically generating high-dimensional input encodings that are often useful for the problem at hand, thus alleviating the need for hand-coded inputs; however, their applicability rests on the assumption that their performance can be evaluated using a loss function that is effectively computable and differentiable.

In cases where the loss function associated with a problem is unknown or difficult to compute and / or differentiate, the problem of assigning credit to the many individual components involved in the functionality of a neural network – or other parametric model – becomes intractable (for more on the credit assignment problem see e.g. [14, 9]). In such cases, methods based on coarser-grained feedback, such as reinforcement learning or direct search in weight space are often used [14]. The Spiral Discovery Network (SDN) model can be considered as an alternative approach with the following properties [3, 4]:

- Instead of relying on evolutionary operations such as recombination and mutation, SDN operates directly within the search space, and so the requirement of creating useful, hand-coded input encodings is alleviated;
- To compensate for the lack of (genetically-motivated) operations for generating new candidate solutions, the model explores the search space along a parametric hyper-spiral structure;
- The application of this hyper-spiral structure in turn implicitly generates differential feedback information (besides the coarse-grained feedback obtained via each candidate solution), allowing the model to adapt its behavior in subsequent cycles of exploration.

The hierarchical SDN model proposed in this paper goes a step further by searching both in the parametric space of the problem domain and the space of the model parameter domain, while allowing for the performance of the latter to be informed by the performance of the former. Generally speaking, hierarchical models offer improvements over flat architectures by enabling different parts of a problem to be tackled at different levels, and for partial solutions found in lower levels to be reused higher up in the architecture. As will be shown through a simulation example, information gleaned through individual cycles of SDN exploration can be used to find better hyper-parameters for the model in the hierarchical extension proposed in this paper. This in turn allows the model to keep finding improvements to earlier candidate solutions, without getting stuck in local minima.

The paper is structured as follows. Section 2 gives an overview of the theoretical background behind parametric optimization, in order to further highlight the motivations behind SDN. Section 3 briefly recapitulates the simplest (non-hierarchical) version of SDN proposed in [3]. Section 4 provides a brief analysis on the operation of the original model based on a simulation introduced in an earlier publication. Based on the analysis, two extensions to the model are proposed in Sections 5 and 6. Finally, the results of the paper are summarized in the Conclusions section.

2 Theoretical Motivations behind SDN

Non-convex optimization is a broad field of mathematics that has applications in engineering tasks where the goal is to find sufficiently good solutions on high-dimensional parametric manifolds. One of the most relevant application examples for non-convex optimization today is finding useful architectures for (deep) neural networks or other kinds of graphical models. The usual way to solve such challenges is to search in iterations based on the gradient of a globally defined loss function – an approach referred to as gradient descent [11].

The general idea of gradient descent can be highly successful on parametric landscapes that are associated with a clearly defined cost function, and contain no more than a small number local minima in terms of that function. However, as soon as the value of a cost function becomes difficult to interpret, or the cost function becomes so intractable that it is computationally difficult to determine its gradients, and / or it produces an intractably large number of local minima, the naive solution of gradient-based iterative optimization often starts to break down.

The problem of dealing with local minima can be addressed to some degree by finding good trade-offs between exploration and exploitation, i.e. by modifying the gradient descent approach slightly to counteract situations where the optimization process might slow down or stop. This approach is reflected in a host of existing solutions. One fruitful idea was to experiment with the scaling factor of the gradient – for example by making it adaptive to changes in sign via the concept of "momentum" [13, 12, 15], or by making it specific to the different dimensions in the parameter space [6, 8]. Other ideas include the normalization of inputs across layers and batches (specifically in training neural network models) [7], or simply adding noise to the gradients [10].

Despite the availability of such ingenious solutions, the requirement of a loss function that is defined globally, easy to compute and also differentiate may be too limiting in certain applications. The original motivation behind the spiral discovery approach that is further extended in this paper came from the problem domain of designing useful multimodal user interfaces [2, 1]. In applications where the performance of a system has to be tuned based on user feedback, it is obviously difficult to obtain feedback at a high input resolution (this would be tedious work for users providing the feedback) as well as at high output resolution (providing consistent evaluations over a period of time is difficult for human subjects). The original Spiral Discovery Method and its various extensions – including the ones proposed in this paper – render such processes of optimization more tractable by limiting the number and resolution of feedbacks necessary, and by compensating for the lack of detailed information through the structure of parametric discovery. It is worth noting that this approach fits well into the framework of interactive evolutionary computation [16, 17].

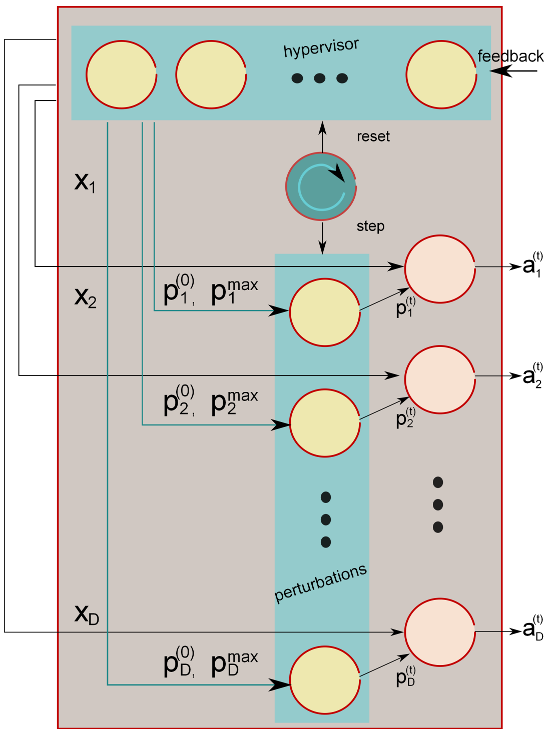

Figure 1
Neural network inspired formulation of an SDN cell.

3 The SDN Cell

As introduced earlier in [3], SDM can be formulated in simpler terms than the original tensor algebra based formulation (i.e. in [2]) using a recurrent model referred to as the Spiral Discovery Network cell. Such a cell consists of:

- A **timer** that functions as a modulo counter for updating the state of the cell at discrete time steps
- A **perturbation module** that determines the direction in which, and the extent to which the slope of exploration is to be modified at each time step
- A **hypervisor module** that refreshes the hyperparameters of the perturbation module based on feedback signals

A graphical representation of an SDN cell and its modules is shown in Figure 1. The updated activation at time t is:

$$\mathbf{a}^{(t)} = (t * step\_sz + 1)\mathbf{x} + \mathbf{p}^{(t)} \tag{1}$$

where:

$$\begin{aligned} \mathbf{p}^{(t)} &= \mathbf{p}^{(t-1)} + sgn(cycle\_dir^{(t)}) \cdot \frac{\mathbf{p}^{max} - \mathbf{p}^{(0)}}{cycle\_len} \\ cycle\_dir^{(t)} &= \begin{cases} 1 & \text{, if } \frac{cycle\_len}{4} \leq t < \frac{3*cycle\_len}{4} \\ -1 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

Generally speaking, the state of the SDN cell is updated in a series of timesteps which together constitute optimization cycles. In the update equations, $\mathbf{x}$ refers to the (normalized) principal component vector – the general direction in the parametric space that is being explored by the cell, while $\mathbf{p}$ refers to the perturbation vector that is added to the principal component. The relationship between the two is governed by the hyperparameter $step\_sz$. The contribution of $\mathbf{x}$ is incremented by $step\_sz$ at each timestep to ensure that the path of parametric discovery expands in the general direction of the principal component (hence, $step\_sz$ represents the degree of *exploitation* in the optimization process). The direction and norm of $\mathbf{p}^{(t)}$, by contrast, which ultimately depends on the relationship between $\mathbf{p}^{(0)}$ and $\mathbf{p}^{(max)}$, determines how far from the principal component the exploration will deviate (therefore, it is directly related to the concept of degree of *exploration* in the optimization process). $cycle\_dir$ governs the direction in which the perturbations are changed, and is dependent on the length of the cycle as well as the current phase within the cycle. The values of $\mathbf{p}^{(0)}$, $\mathbf{p}^{max}$ and $\mathbf{x}$ are dependent on the cycle (or more precisely, on the discoveries made during the previous cycle), and are updated as follows:

$$\begin{aligned} p^{(0)}_{i,unnormed}[c] &= p_i^{(\arg\min_t h_i^t[c-1])}[c-1] \\ p^{max}_{i,unnormed}[c] &= p_i^{(0)}[c] + softmax(\sigma_{h_i}[c-1])(\sigma_{h_i}[c-1]+1) = \\ &= p_i^{(0)}[c] + \frac{\exp \sigma_{h_i}[c-1]}{\sum_I \exp \sigma_{h_I}[c-1]}\,[\sigma_{h_i}[c-1+1] \\ \mathbf{p}^{(0)}[c] &= ||\mathbf{p}^{(0)}_{unnormed}[c]|| \\ \mathbf{p}^{max}[c] &= ||\mathbf{p}^{max}_{unnormed}[c]|| \\ x_i[c] &= ||x_i[c-1] + \frac{p^{(0)}_{i\ unnormed}[c]}{||\mathbf{p}^{(0)}[c]||}|| \end{aligned} \tag{3}$$

Here, the value of a parameter within a cycle c is represented using square brackets, so that for example $h_i^t[c-1]$ refers to the value of the i-th hypervisor cell at time t of cycle $c-1$. σ_{h_i} denotes the standard deviation of value the i-th hypervisor cell. The update equations ensure that:

- the perturbations in the new cycle are centered, in each dimension, around the perturbation that was associated with the lowest cost function value in the previous cycle (note that h_i refers to the i-th hypervisor cell)

- the maximum values of the perturbations are set to their starting value, plus a value that depends on the standard deviation of the corresponding hypervisor cell in the previous cycle, as well as its relation to the standard deviations of other hypervisor cells. In general, the larger the deviation in a given dimension in an absolute sense, the greater the distance will be between the initial and maximal perturbation in the following cycle (in which case the network will be more explorative in that dimension). Similarly, exploration in dimensions that are characterized by large relative deviations will also be higher in the following cycle.
- the principal component, $\mathbf{x}$ is set to the initial principal component plus the normalized value of the initial perturbation.

4 Analysis of the SDN Cell

In order to get a general glimpse into the operation of an SDN cell, we consider its performance on a simulation example introduced earlier in [3] (see also Figure 2):

$$z = \begin{cases} 500 & \text{if } x, y \notin (0,10] \times (0,10] \\ 70 & \text{if } x, y \in [1,1.5] \times [2.75,4.5] \\ -10 & \text{if } x, y \in [3.25,3.5] \times [3.5,4.25] \\ (x-5)^2 + \dots & \\ -2(y-2) + \dots & \text{otherwise} \\ x + e^{\frac{1}{(x+y)}} & \end{cases} \tag{4}$$

Figure 3 shows the points within the two-dimensional parameter space that are visited by the cell in each cycle for the first 15 cycles. Further, the plot for each cycle shows the principal component vector, initial perturbation vector and maximal perturbation vector computed for the following cycle. Based on the figure, the following conclusions can be drawn:

- Generally speaking, the initial perturbation vector (blue) for the next cycle will point towards the direction of the perturbation that was applied when the point with the smallest loss value was generated (largest point on the plot);
- The maximal perturbation vector (red) is obtained by adding to the initial perturbation vector (blue) a vector that characterizes, in each dimension, the variance of the feedback, both in absolute terms and in relative terms (i.e., as compared with other dimensions). Note that in the simulation, the output of the loss function is a scalar value and there is no credit assignment among

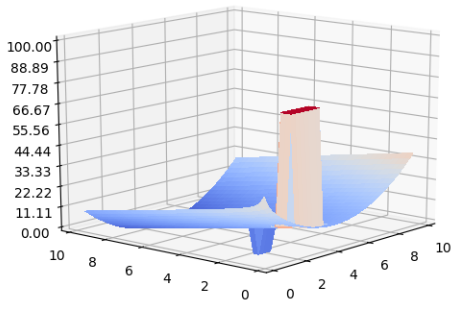

Figure 2
Plot of the loss function to be minimized in the simulation example.

the dimensions of the parameter space (the SDN model described in equation contains no such credit assignment), therefore both x and y components of the initial and maximal perturbation vectors are the same. This needs to be solved in future versions of SDN (including the hierarchical model proposed in this paper).

- As a result of the previous observation, the angle between the initial (blue) and maximal (red) perturbation vector largely depends on the orientation of the former (their difference is always a vector that points in the direction $[1,1]$). This in turn influences the angle between $\mathbf{p}^{(0)} - \mathbf{x}$, and $\mathbf{p}^{max} - \mathbf{x}$, which is what determines the path of exploration.

5 Modified SDN Cell Model with Hypervisor Credit Assignment

Based on the above, a modified version of the SDN cell is proposed with hypervisor credit assignment. The approach introduces a feedback path from the output of the cell to the hypervisor cell, through the mediation of a time delay component called the **delta module** (Figure 4). The role of this module is to model the influence of changes in each of the output dimensions on the value of the loss function. Given the fact that the SDN model makes no assumptions on the loss function, it cannot be assumed that this influence can be modeled effectively over long periods of time. Therefore, changes through longer periods of time are discounted based on the following update equation:

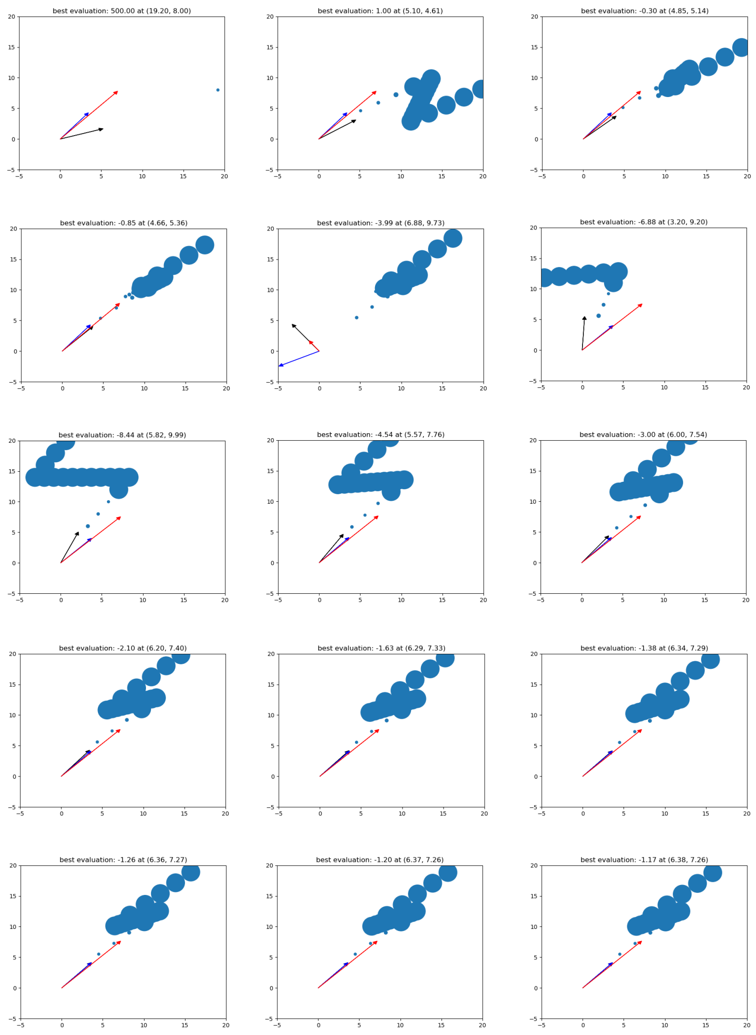

Figure 3
Plots of simulation in the first 15 cycles (left to right, top to bottom). The plot for each cycle shows the points within the parameter space that were visited in the cycle (with the sizes of the points being inversely proportional to the value of the loss function), as well as the unnormalized principal component vector (black), unnormalized initial perturbation vector (blue) and unnormalized maximal perturbation vector (red) computed for the next cycle (the length of all three vectors is scaled up for visibility).

$$\begin{aligned} h_i[t] &= loss * \gamma_{h_i}[t] = \\ &= loss * softmax\left(\sum_{i=1}^{\infty} i^{-1}(a_i[t] - a_i[t-i])^2\right) \end{aligned} \quad (5)$$

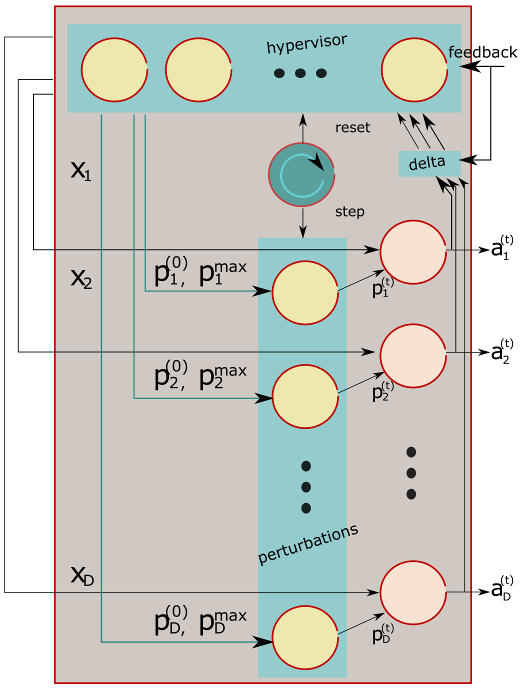

Figure 4
Neural network inspired formulation of an SDN cell with hypervisor credit assignment.

where γ is a scaling factor applied to the loss function value that is calculated through a softmax function and depends on all dimensions of the output activity.

Figure 5 shows the same simulation as earlier, this time carried out using the modified version of the SDN cell with hypervisor credit assignment. It is clear – as it should be – that it is no longer the case that the difference between the maximal and initial perturbation vectors is always a vector pointing in the direction $[1,1]$ – hence, this model is better at achieving higher degrees of exploration when useful.

However, upon closer inspection it can be noticed that the evolution of the model seems to involve cycles and does not further adapt after a certain period of time. For example, the path of the outputs and the directions of the principal and perturbation vectors are very similar in the 15th cycle (fifth row, last plot in Figure 5) and in the 5th cycle (second row, middle plot in Figure 5). If the model is run longer, indeed it becomes clear that after a time, the same paths of exploration are revisited. Thus, the idea of adding further cells to the model – i.e., turning the model into a hierarchical network of SDN cells – seems to be worth exploring.

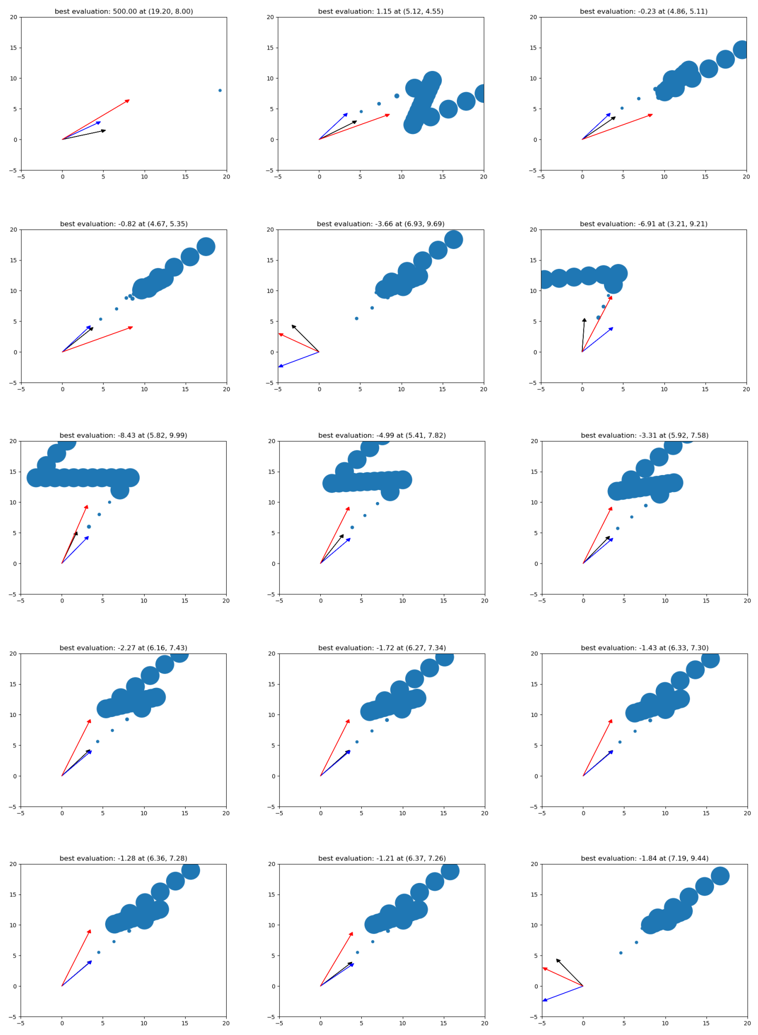

Figure 5

Plots of the second simulation including hypervisor credit assignment in the first 15 cycles (left to right, top to bottom). The plot for each cycle shows the points within the parameter space that were visited in the cycle (with the sizes of the points being inversely proportional to the value of the loss function), as well as the unnormalized principal component vector (black), unnormalized initial perturbation vector (blue) and unnormalized maximal perturbation vector (red) computed for the next cycle (the length of all three vectors is scaled up for visibility).

6 The Hierarchical SDN Model

Based on the above, a hierarchical SDN model can be proposed in which a top-level SDN cell computes the $step\_sz$ and principal component of the output layer

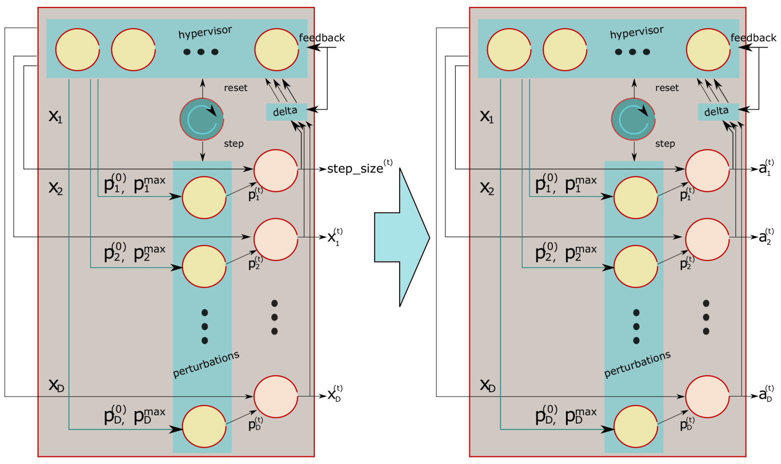

Figure 6
Hierarchical SDN comprised of two SDN cells.

(Figure 6). Following the *temporal slowness principle*, the lower-level SDN cell has a shorter period and produces a relatively large variety of outputs per input received from the top-level cell. Therefore, each output configuration of the top-level cell is associated with a single aggregated feedback from a multitude of outputs provided by the lower-level cell.

Accordingly, the third simulation presented in this paper was comprised of two SDN cells, both with a cycle of 20 steps and a period ratio of 20 lower level cycles per each cycle at the higher level. The smallest loss function value obtained over the 20 low-level cycles were fed back to the higher-level SDN cell and used as feedback for the configuration that yielded the corresponding $step\_sz$ and principal component vector ($\mathbf{x}$). Key results from the simulation are shown on Figure 7. The figure shows that the global minimum was found as early as in the 3rd high-level cycle. This is a qualitative improvement over all previous simulations, in which the search became stuck in cycles that didn't include the global minimum. A further observation that seems to underline the superiority of the hierarchical model in its modeling capabilities is the observation that no actual feedback was necessary from the lower-level layer – i.e. that it was sufficient to try 3 different higher-level output configurations to find the global minimum. This may of course also reflect the simplicity of the simulation problem.

Conclusions Evolutionary and neural network based parametric optimization approaches are different approaches, each with their own set of assumptions and limi-

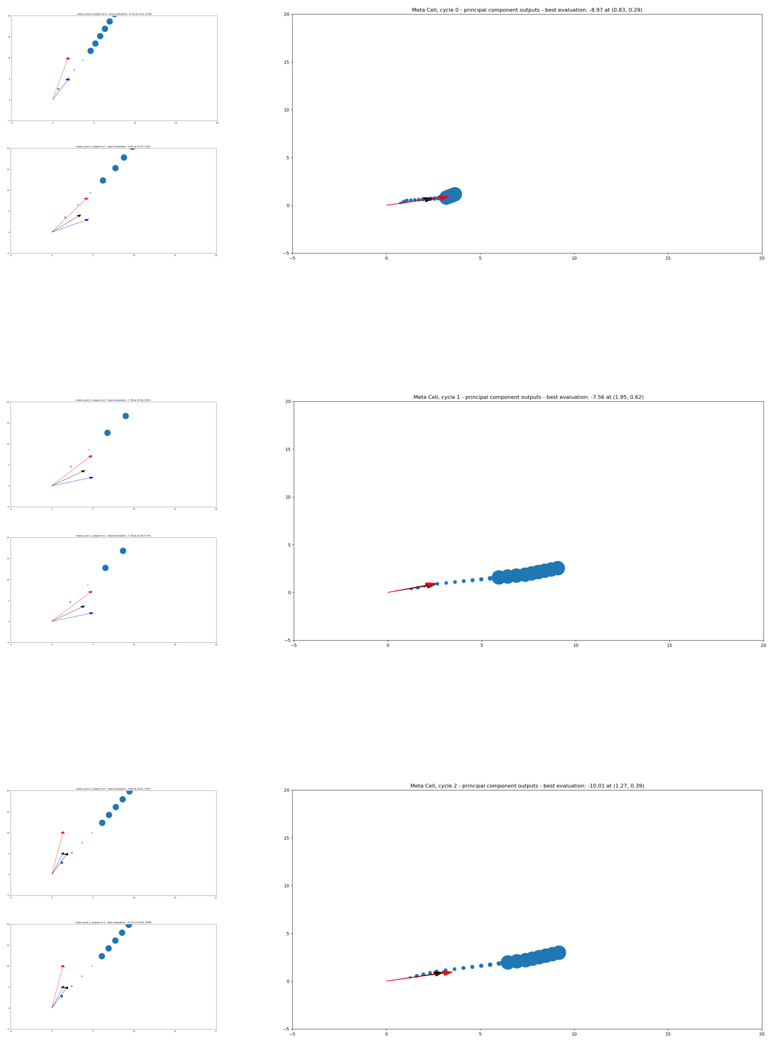

Figure 7

Plots of the third simulation – in this case using the hierarchical SDN model including cells with hypervisor credit assignment. The figure shows the top two results (left column) associated with the first three high-level (meta) cycles (right hand column). Because each meta cycle provides 3-dimensional outputs ($step\_sz$ and a two-dimensional principal component vector), only the elements of the principal component vector are shown in the right-hand column.

tations. Problem domains exist where neither approach is effective. First, it may be difficult to encode candidate solutions as evolutionary genotypes that behave well (or provide useful results) under the operations of mutation and recombination. Second, the performance of candidate solutions sometimes cannot be evaluated using a globally defined and differentiable loss function. In such cases, the Spiral Discovery Network model and its extensions proposed in this paper can be used as an alternative approach: one that works even with feedback evaluations obtained less frequently, and without any explicit gradient information. The SDN approach can be characterized as follows: Instead of relying on evolutionary operations such as recombination and mutation, it operates directly within the search space in an auto-regressive fashion, and so the requirement of creating useful, hand-coded input encodings is alleviated; To compensate for the lack of a generally computable and differentiable loss function, it explores the search space along a parametric hyper-spiral structure that implicitly generates differential feedback information, allowing the model to adapt its behavior in subsequent cycles of exploration.

References

[1] Peter Baranyi, Adam Csapo, and Gyula Sallai. *Cognitive Infocommunications (CogInfoCom)*. Springer International Publishing, 2015.

[2] A. Csapo and P. Baranyi. The Spiral Discovery Method: an Interpretable Tuning Model for CogInfoCom Channels. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 16(2):358–367, 2012.

[3] Adam B Csapo. The spiral discovery network as an automated general-purpose optimization tool. *Complexity*, 2018:1–8, 2018.

[4] Adam B Csapo. The spiral discovery network as an evolutionary model for gradient-free non-convex optimization. In *IEEE International Conference on Cognitive Infocommunications*, pages 1–6, 2018.

[5] Pedro Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[10] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[11] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[12] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[13] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[14] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[15] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[16] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001.

[17] Hideyuki Takagi and Hitoshi Iba. Interactive evolutionary computation. *New Generation Computing*, 23(2):113–114, 2005.