Hindawi
Complexity
Volume 2018, Article ID 1947250, 8 pages
https://doi.org/10.1155/2018/1947250

Research Article

The Spiral Discovery Network as an Automated General-Purpose Optimization Tool

Adam B. Csapo

Department of Informatics, Széchenyi István University, Győr, Hungary

Correspondence should be addressed to Adam B. Csapo; csapo.adam@sze.hu

Received 29 September 2017; Accepted 22 January 2018; Published 12 March 2018

Academic Editor: Kevin Wong

The Spiral Discovery Method (SDM) was originally proposed as a cognitive artifact for dealing with black-box models that are dependent on multiple inputs with nonlinear and/or multiplicative interaction effects. Besides directly helping to identify functional patterns in such systems, SDM also simplifies their control through its characteristic spiral structure. In this paper, a neural network-based formulation of SDM is proposed together with a set of automatic update rules that makes it suitable for both semiautomated and automated forms of optimization. The behavior of the generalized SDM model, referred to as the Spiral Discovery Network (SDN), and its applicability to nondifferentiable nonconvex optimization problems are elucidated through simulation. Based on the simulation, the case is made that its applicability would be worth investigating in all areas where the default approach of gradient-based backpropagation is used today.

1. Introduction

The question of how to gain an understanding of the operation of a system arises naturally in a wide range of application areas. However, this question is not always easy to answer, in part because different use cases favor different approaches. While a set of closed formulae might be useful when it comes to predicting exactly how the system will operate under specific conditions, they may be difficult to formulate when the conditions themselves and/or their effects are hard to characterize. In such cases, black-box identification and heuristic modelling approaches are often used.

The neural network presented in this paper, referred to as the Spiral Discovery Network (SDN), is a generalized version of the Spiral Discovery Method, which is a semiautomated cognitive artifact [1, 2]. SDM originally served the purpose of helping users to discover systematic relationships between multiple inputs to a system and the system's output behavior, even when the inputs have nonlinear effects and multiplicative cross-effects on the output. The goal in extending the SDM model is to extend its applicability to automated settings in which neural networks (or other parametric black-box models) tune their behavior based on a set of functional constraints, such as requirements on the structure of their output or other external error feedback signals.

Through the formulation proposed in this paper, it turns out that SDM is applicable whenever a data-driven approach is available to the identification of a system and whenever the effects of various changes in its inputs can be evaluated in a reasonable amount of time. When the evaluations are performed by humans, SDM shows motivations and characteristics similar to those of the paradigm of interactive evolutionary computation [3, 4]; however, it shows differences in terms of the logic through which it helps to discover parametric spaces. Its extended version, SDN, is also more generally applicable by allowing for automated evaluations. As discussed in the conclusions of the paper, SDN is noteworthy in that it does not rely on gradient information, a feature that can be seen to reduce the complexity of the required computations, as well as being potentially helpful in cases where the performance of gradient-based solutions is far from optimal (for a detailed discussion on such cases, the reader is referred to [5]).

The paper is structured as follows. Section 2 provides a short overview of the literature on nonconvex optimization in order to position the relevance of this work with respect to earlier results. Section 3 then briefly reviews the background of the original Spiral Discovery Method (SDM). Section 4 introduces the tensor-algebra based numerical structures behind the original SDM formulation. In Section 5, the neural network-based Spiral Discovery Network (SDN) is introduced. A simulation example is provided in Section 6 in order to demonstrate the viability of the model in handling nonconvex and nondifferentiable optimization problems. Finally, Section 7 concludes the paper.

2. Historical Overview

Nonconvex optimization is a broad field of mathematics that finds many applications in engineering tasks where the goal is to find sufficiently good solutions on high-dimensional parametric manifolds. One of the most relevant examples today is finding useful architectures for (deep) neural networks or other kinds of graphical models, as well as finding the right set of parameters with which to operate them. The common approach in solving such problems is to iteratively refine a candidate solution in a way that incrementally improves upon it in terms of a globally defined loss function: this is known as gradient descent [6].

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 of 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, that is, 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" [7–9] or by making it specific to the different dimensions in the parameter space [10, 11]. Other ideas include the normalization of inputs across layers and batches (specifically in training neural network models) [12] or by simply adding noise to the gradients [13].

The above solutions notwithstanding, the general idea of modifying a candidate solution in the direction of the negative gradient of a loss function has largely remained unchallenged. Only recently have the remarks of G. Hinton and other highly regarded researchers become widely publicized, which suggest that gradient descent, at least based on backpropagation, may prove not to be the ultimate solution for training neural networks (see, e.g., the article entitled "Why We Should Be Deeply Suspicious of BackPropagation" by C. E. Perez on https://medium.com/intuitionmachine/the-deeply-suspicious-nature-of-backpropagation-9bed5e2b085e).

In this paper, the earlier idea of the Spiral Discovery Method is extended to the domain of automatic training in neural networks through a neural architecture. Instead of relying on gradients to update its search location, the method follows a hierarchical hyperspiral structure within the parametric space, thus gaining insight into search directions that may be fruitful.

3. Original Problem Formulation Behind SDM

In this section, we consider a generic formulation of the class of problems to which the original Spiral Discovery Method (SDM) can be applied. To this end, we will make use of the following concepts and notations:

(i) A vector of *generation parameters* $\mathbf{g} \in \mathbb{R}^G$

(ii) A perceptually accessible *output* $o \in \mathbb{R}$

(iii) A *system transfer function* $S : \mathbb{R}^G \rightarrow \mathbb{R}$, which evaluates generation parameter vectors to produce perceptually accessible outputs

(iv) An *evaluation function* $E : \mathbb{R} \rightarrow \mathbb{R}$, which associates perceptually accessible outputs with a real number referred to as the *perceptual value* of a given output

(v) A set $\mathcal{D} = \{(\mathbf{g}_1, \mathbf{v}_1), (\mathbf{g}_m, \mathbf{v}_M)\}$ referred to as the *data set*, which contains tuples of generation parameter vectors and perceptual values.

In the original problem formulation, the goal is to find a set of generation parameter vectors that are suitable for the generation of a controlled set of outputs, controlled, that is, from the perspective of the perceptually driven evaluation function. Most often, the problem would present itself in such a form that a user is given a perceptual value, v', and the goal is to find a generation parameter vector, $\mathbf{g}'$, suitable for the generation of an output that yields $\mathbf{v}'$ as its perceptual value. In general, solving this problem amounts to more than just inverting the system transfer function (if such an inversion were even possible to begin with), as the relationship between system output and its perception value, which is usually much too complex to be formulated analytically, also must be taken into account.

Application areas in which the above formulation is of interest include the following:

(i) Tuning a set of parameters to a uni- or multimodal synthesis algorithm for perceptual continuity: for example, in a virtual reality with object-to-sound and object-to-vibration mappings, given a set of parameters used to generate audio signals and vibration patterns for spherical and block-like objects, the goal might be to find an appropriate set of generation parameters for certain kinds of polyhedra, conceptually situated "somewhere between" spheres and blocks.

(ii) Controlling inputs to complex black-box models based on derived quantifications of success: for example, inputs to a multispeaker system or a distributed

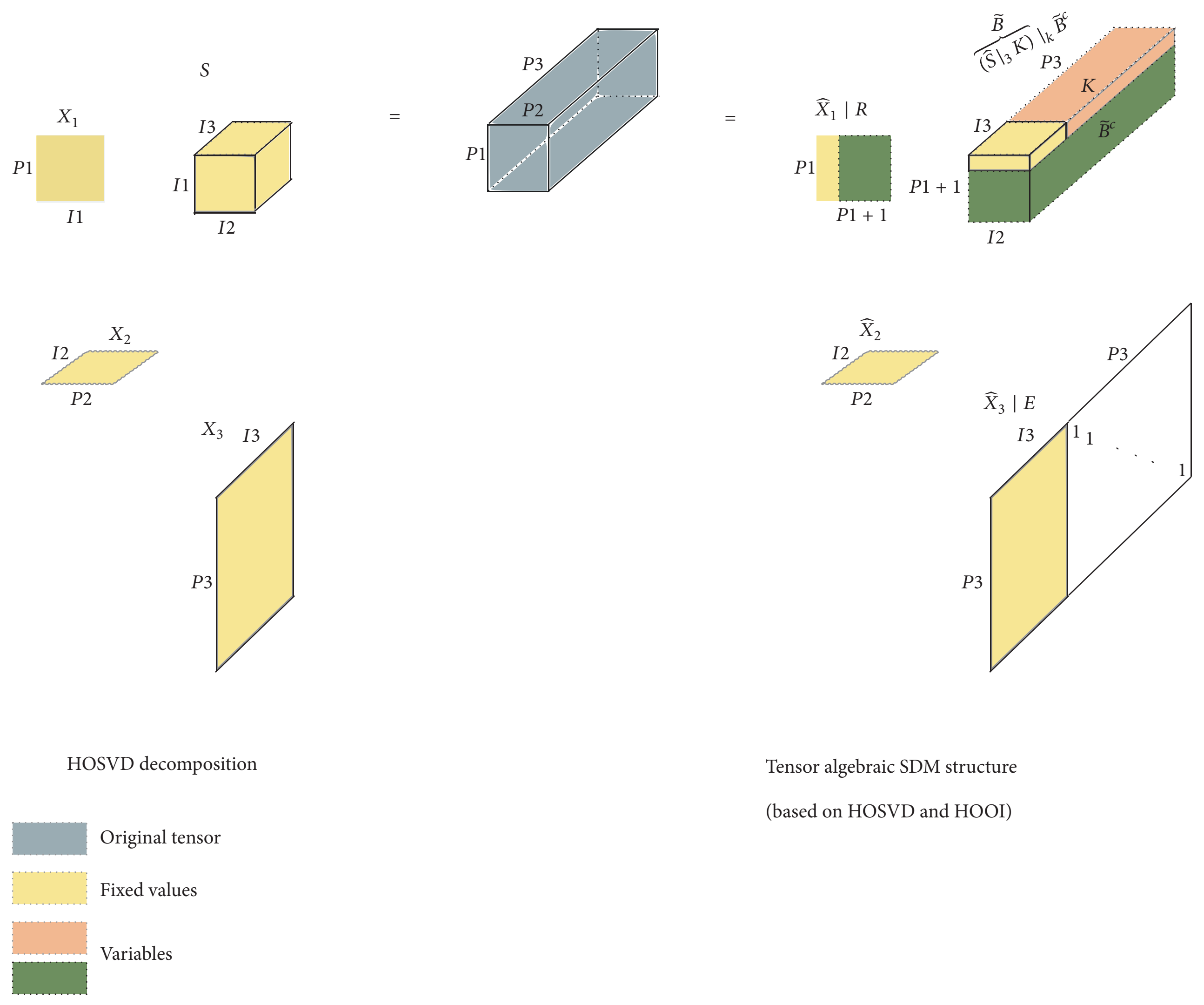

Figure 1: Tensor algebraic formulation of the Spiral Discovery Method based on higher-order singular value decomposition and higher-order orthogonal iteration (figure adapted from [2]).

heating system in a large auditorium might be fine-tuned in order to accommodate extrinsic requirements of comfort and cost-effectiveness.

The overall characteristic of the problem formulation is that it encompasses problems where a set of parameters can be used to control a model, usually a black-box model, whose functionality can best be evaluated indirectly through effects that are not well understood, for example, perceptual effects, qualitative measures such as comfort, or aggregated measures such as cost-effectiveness.

It is clear that such formulation can be easily generalized to cases where the evaluation is performed not by humans, but by any kind of automatic process extrinsic to the system. Such processes might still involve a weaker link to human perception or more generally to qualitative cognitive measures but would nevertheless be directly or indirectly measureable and interpretable.

4. Tensor Algebraic Formulation of the Spiral Discovery Method

The original formulation of SDM is in a tensor algebraic form, shown in Figure 1. It is based on the discretization of a hypothetical function that maps vectors of perceptual values $\mathbf{v}_i$ to generation parameters $\mathbf{g}_i$. In most cases, this function cannot be expressed analytically and might even be different depending on various circumstances, such as the user performing the evaluation. At the same time, a discretized form of the function can often be sampled through experiments (this idea is inspired by the Tensor Product model [14–16]). The discretization is stored in a tensor, $\mathcal{F}$, such that all dimensions, save for the last one, correspond to discrete gradations along perceptual scales (e.g., "roughness," "softness," "degree of comfort," or "cost-effectiveness"), while the last dimension stores G-dimensional generation parameter vectors corresponding to the perceptual configurations.

The above described tensor, $\mathcal{F}$, is first decomposed into a core tensor and a set of weighting matrices based on the higher-order singular value decomposition (HOSVD) [17]. This is followed by an iterative rank-reduction step, known as higher-order orthogonal iteration (HOOI) [18], which creates a rank-reduced approximation of the complete system, such that its outputs are controlled by only a single parameter in the perceptual dimension of interest. The twist in the approach is that the "meaning" of this parameter, in other words, the hyperplane along which it influences the system, is cyclically changed through a numerical reconstruction of the system and the systematic manipulation of the core tensor.

The conceptual background of SDM can be well described through a 2-dimensional numerical example. Consider the function described by $\mathbf{F}$:

$$\mathbf{F} = \begin{pmatrix} 5 & 2 \\ 3 & 3 \\ 10 & 5 \end{pmatrix} \tag{1}$$

in which there are 2 generative parameters for 3 different perceptual gradations. Using singular value decomposition (SVD, instead of HOSVD because we are in case of two dimensions), we obtain

$$\begin{aligned} \mathbf{F} &= \mathbf{S} \times_1 \mathbf{U} \times_2 \mathbf{V} \\ &= \begin{pmatrix} 13.04 & 0 \\ 0 & 1.4 \end{pmatrix} \times_1 \begin{pmatrix} 0.41 & -0.39 \\ 0.31 & 0.91 \\ 0.86 & -0.14 \end{pmatrix} \\ &\quad \times_2 \begin{pmatrix} 0.89 & -0.46 \\ 0.46 & 0.89 \end{pmatrix}. \end{aligned} \tag{2}$$

Optimal rank-reduction in the 2-dimensional case consists simply of removing the second column of $\mathbf{S}$ and the second row of $\mathbf{V}$ or setting $\mathbf{S}(2,2) = 0$ (thus, in this simple case of two dimensions, HOOI needs not be used). Once $\mathbf{S}(2,2) = 0$, the second row of the core tensor consists of all zeros and can be removed (as a result, the second column of $\mathbf{U}$ is also removed).

After augmenting the matrix of singular values and the weighting matrices as specified by SDM, we obtain

$$\begin{aligned} \widetilde{\mathbf{S}} &= \begin{pmatrix} 13.04 & 0 & a & b \\ d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \end{pmatrix} \\ \widetilde{\mathbf{U}} &= \begin{pmatrix} 0.41 & r_{11} & r_{12} & r_{13} \\ 0.31 & r_{21} & r_{22} & r_{23} \\ 0.86 & r_{31} & r_{32} & r_{33} \end{pmatrix} \\ \widetilde{\mathbf{V}} &= \begin{pmatrix} 0.89 & -0.46 & 1 & 0 \\ 0.46 & 0.89 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{3}$$

If a and b and the random values in the second, third, and fourth columns of $\widetilde{\mathbf{U}}$ are specified, the second, third, and fourth rows of $\widetilde{\mathbf{S}}$ can be calculated such that the original system is reconstructed. Then, by modifying just the first column of weighting matrix $\widetilde{\mathbf{U}}$, a linear subspace of the original 2-dimensional space can be explored, starting from any of the three perceptual gradations. By separating what is constant from the parts of the equation that are changed, we obtain

$$\widetilde{\mathbf{F}} = \begin{bmatrix} \tilde{u}_{11} \\ \tilde{u}_{21} \\ \tilde{u}_{31} \end{bmatrix} \begin{bmatrix} 13.04 & 0 & a & b \end{bmatrix} \widetilde{\mathbf{V}}^T + \mathbf{RD}\widetilde{\mathbf{V}}^T. \tag{4}$$

Because the second term is a constant and the first one only depends on the first column of $\widetilde{\mathbf{U}}$, the "slope" of the equation, that is, the ratio of change between the second and first output (as the first column of weighting matrix $\widetilde{\mathbf{U}}$ is modified), can be written as

$$\text{slope}_{xy} = \frac{\begin{pmatrix} 13.04 & 0 & a & b \end{pmatrix} \tilde{\mathbf{v}}_2^T}{\begin{pmatrix} 13.04 & 0 & a & b \end{pmatrix} \tilde{\mathbf{v}}_1^T} = \frac{13.04 \cdot 0.46 + b}{13.04 \cdot 0.89 + a}. \tag{5}$$

It is clear that based on (5) the slope can be set to any value just by modifying the values of a and b. If the values of a and b are changed systematically between two extreme values, the slope of discovery will also oscillate along the principal component of the original matrix.

5. The Spiral Discovery Network Cell: A Neural Network-Based Formulation of SDM

The key observation of this paper is that SDM can be formulated in much simpler and at the same time more powerful terms using neural networks. The recurrent model shown in Figure 2 is capable of producing systematic, cyclic patterns similar to the original formulation, but at the same time it is adaptive based on a set of external feedback signals. The cell consists of the following modules:

(i) A *timer* that functions as a modulo counter for updating the state of the cell at discrete time steps

(ii) 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

(iii) 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 2. The updated activation at time t is

$$\mathbf{a}^{(t)} = \alpha^{(t)}\mathbf{x} + \beta\mathbf{p}^{(t)}, \tag{6}$$

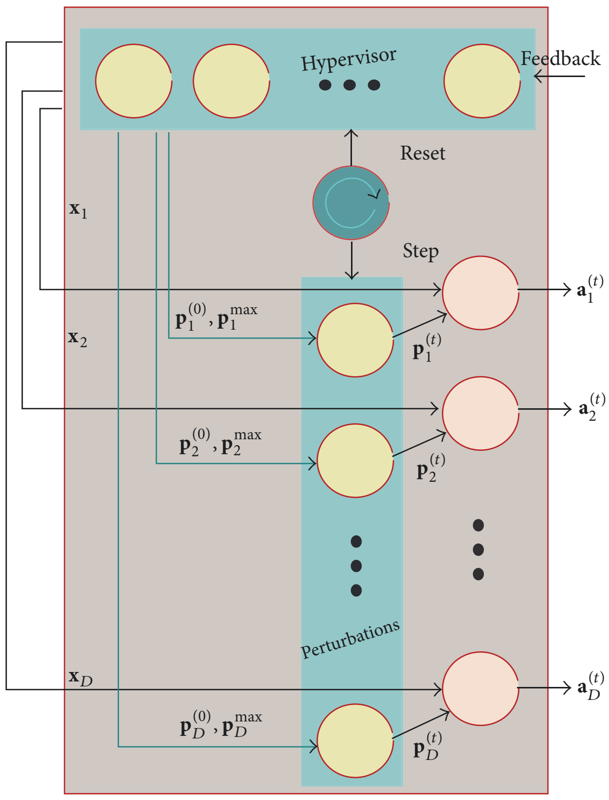

FIGURE 2: Neural network inspired formulation of an SDN cell. The cell includes a timer neuron, hypervisor neurons, perturbation neurons, and output neurons providing activation at each time step t.

where

$$\alpha^{(0)} = 1$$
$$\alpha^{(t)} = \alpha^{(t-1)} + \text{step\_sz}$$
$$\mathbf{p}^{(t)} = \text{sgn}\left(\text{cycle\_dir}^{(t)}\right) \cdot \frac{\mathbf{p}^{\max} - \mathbf{p}^{(0)}}{\text{cycle\_len}} \tag{7}$$
$$\text{cycle\_dir}^{(t)} = \begin{cases} 1, & \text{if } \dfrac{\text{cycle\_len}}{4} \le t < \dfrac{3 * \text{cycle\_len}}{4} \\ -1 & \text{otherwise} \end{cases}.$$

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 value of α 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 and can be calibrated based on the cycle length alone, owing to the fact that the principal component $\mathbf{x}$ is normalized to begin with). The direction and norm of $\mathbf{p}^{(t)}$, by contrast, which ultimately depends on the relationship between $\mathbf{p}^{(0)}$ and $\mathbf{p}^{(\max)}$, determine 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 initialized as follows:

$$\begin{aligned} p_{i,\text{unnormed}}^{(0)}[c] &= p_i^{(\arg\min_t h_i^t[c-1])}[c-1] \\ p_{i,\text{unnormed}}^{\max}[c] &\\ &= p_i^{(0)}[c] + \text{softmax}\left(\sigma_{h_i}[c-1]\right)\left(\sigma_{h_i}[c-1]+1\right) \\ &= p_i^{(0)}[c] + \frac{\exp\sigma_{h_i}[c-1]}{\sum_I \exp\sigma_{h_I}[c-1]}\left[\sigma_{h_i}[c-1]+1\right] \\ \mathbf{p}^{(0)}[c] &= \left\|\mathbf{p}_{\text{unnormed}}^{(0)}[c]\right\| \\ \mathbf{p}^{\max}[c] &= \left\|\mathbf{p}_{\text{unnormed}}^{\max}[c]\right\| \\ x_i[c] &= \left\|x_i[c-1] + \frac{p_i^{(0)}[c]}{\left\|\mathbf{p}^{(0)}[c]\right\|}\right\|. \end{aligned} \tag{8}$$

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 ith hypervisor cell at time t of cycle $c-1$. σ_{h_i} denotes the standard deviation of value of the ith hypervisor cell. Both $\mathbf{p}$ update equations ensure the following:

(i) 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 ith hypervisor cell).

(ii) 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.

(iii) The principal component, $\mathbf{x}$, is set to the initial principal component plus the normalized value of the perturbation.

It is worth noting that the way in which SDN cells encapsulate a complex set of functions with a specific functional logic is reminiscent of how long short-term memories reduce the complexity of backpropagation through time [19, 20]. In the case of SDN cells, the effects of a complete cycle are stored within the cell. Although these effects are deterministic, it would be worth investigating how the hyperparameters like step_sz might themselves be learned.

Another approach that can be mentioned in connection with SDN cells is Particle Swarm Optimization (PSO) [21, 22] and other metaheuristic approaches, such as genetic algorithms [23–25]. PSO and genetic algorithms are somewhat similar to SDN cells in the sense that exploration evolves towards more promising areas of the parametric space. However, the two categories of approaches are also different in the way that they make a compromise between exploration and exploitation: even when evolving towards more promising regions, SDN cells still represent alternative regions to an extent that depends on how varied the obtained feedback values were (exploration); it is the principal direction of the next cycle that in turn influences exploitation.

6. Simulation Example

As a simulation example, we consider a surface described by two parameters, x and y, that can take values of $(0, 10]$. The surface is expressed through the following relationship (see also Figure 3):

$$z = \begin{cases} 500 & \text{if } x, y \notin (0,10] \times (0,10] \\ 70 & \text{if } x, y \in [1, 1.5] \times [2.75, 4.5] \\ -10 & \text{if } x, y \in [3.25, 3.5] \times [3.5, 4.25] \\ (x-5)^2 + \cdots & \\ -2(y-2) + \cdots & \\ x + e^{1/(x+y)} & \text{otherwise.} \end{cases} \quad (9)$$

Figure 4 shows that the minimum location of the search (and parameters thereof) was found as early as in the 7th cycle, without recourse to any kind of gradient information. Although no location for the exact minimum (-10) was found, it can be argued that the obtained results come quite close to achieving this, for two reasons:

(i) The range of values of the loss function was between 70 and -10; hence the value of -8.44 falls within 2% of error.

(ii) The search itself was unconstrained (i.e., was not guided by the knowledge that only values between 0 and 10 were to be considered on the x- and y-axes): of course, as expected, the fact that locations outside of the specified range had a loss value of 500 helped to guide the search.

Although rudimentary, the example shows the potential value of SDM in dealing with optimization problems that are nonconvex and nondifferentiable.

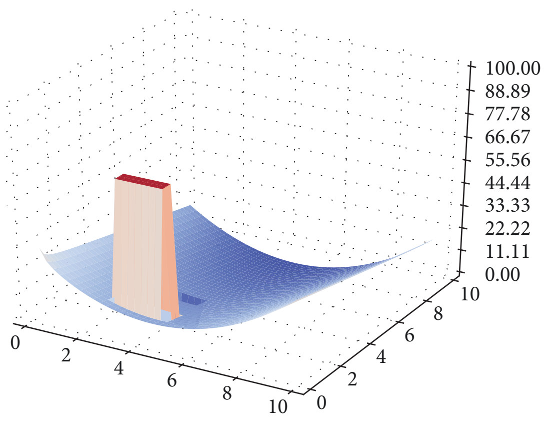
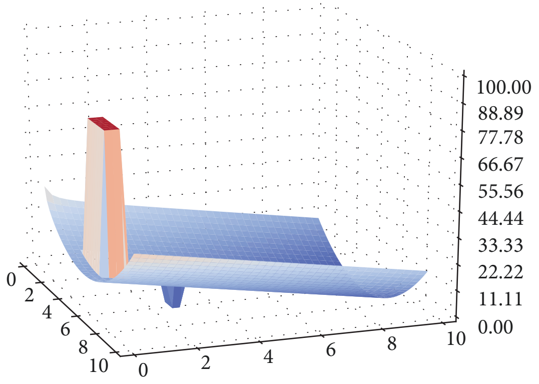

Figure 3: Two views of the complex surface to be minimized in simulation example.

7. Conclusions

In this paper, an extended, automated variant of the Spiral Discovery Method is proposed. The variant is formulated as a neural network, or rather as a component thereof, and is referred to as the Spiral Discovery Network (SDN) cell. The model of SDN cells incorporates several beneficial properties. First, it is capable of exploring large areas of parametric spaces through a parametric hyperspiral structure, such that the hyperspiral structure itself changes through adaptive cycles. Second, it can rely on any kind of quantitative (perhaps even qualitative) feedback, not only gradient information, to achieve its adaptivity. These properties combined make SDN cells a candidate solution for optimization problems in which the parametric space is nonconvex and potentially even nondifferentiable. A rudimentary simulation was described in the paper to demonstrate the capabilities of SDN cells. One possible avenue of investigation as part of future work would be to consider how SDN cells might be used as part of a network to further improve optimization performance.

Conflicts of Interest

The author declares that they have no conflicts of interest.

Acknowledgments

This work was supported by the FIEK program (Center for Cooperation between Higher Education and the Industries at Széchenyi István University, GINOP-2.3.4-15-2016-00003).

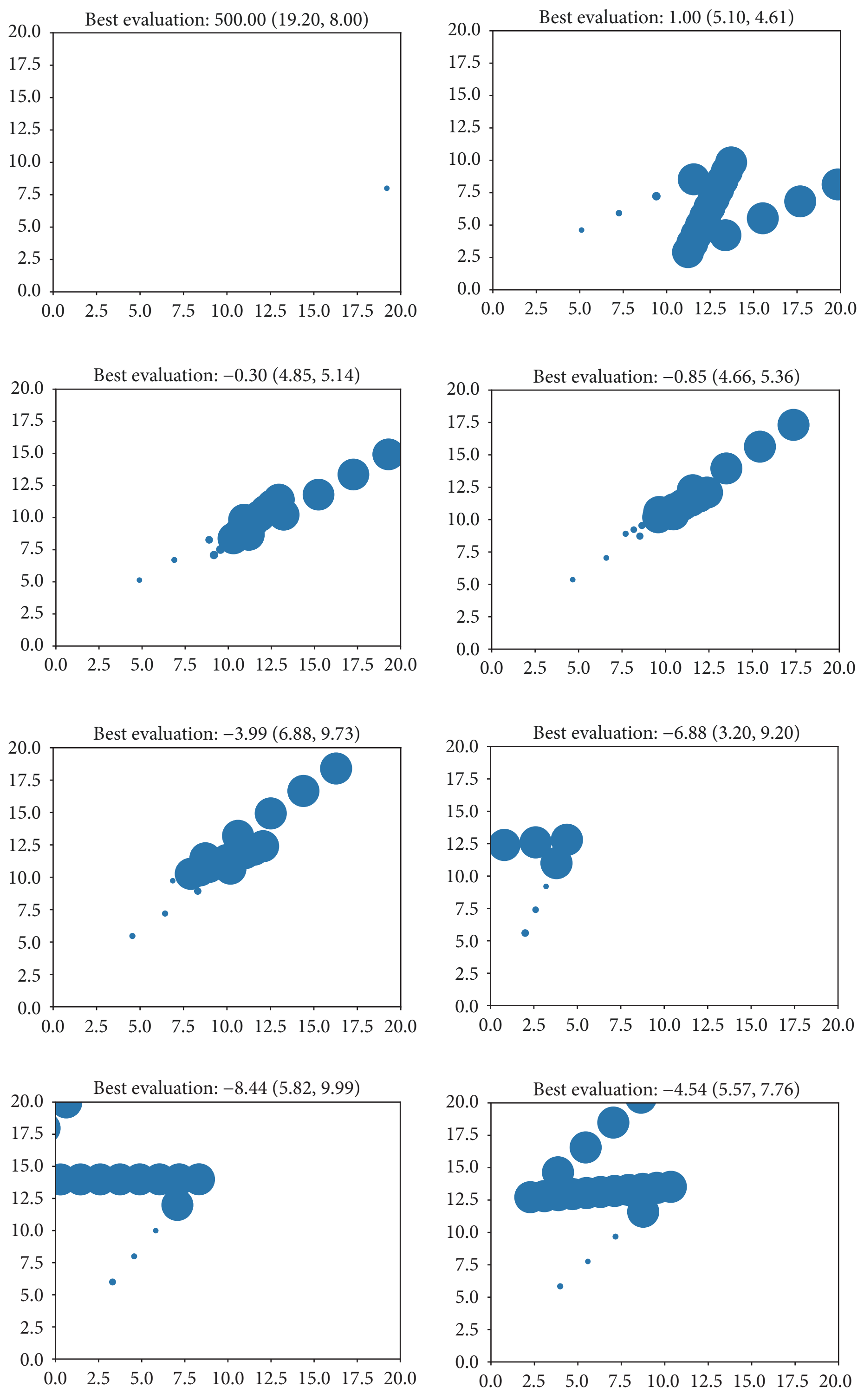

Figure 4: Cycles of SDN from left to right, top to bottom. Performance is indicated at the top of the plots, with the sizes of the evaluated locations inversely proportional to the value of the loss function (and scaled per plot). The figure shows that the smallest loss value, in a 2% vicinity of the minimum, was found as early as in the 7th cycle, without recourse to any kind of gradient value.

References

[1] P. Baranyi, A. Csapo, and G. Sallai, “Cognitive infocommunications (CogInfoCom),” *Cognitive Infocommunications (CogInfoCom)*, pp. 1–219, 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*, vol. 16, no. 2, pp. 358–367, 2012.

[3] H. Takagi, “Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation,” *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.

[4] H. Takagi and H. Iba, “Preface interactive evolutionary computation,” *New Generation Computing*, vol. 23, no. 2, pp. 113-114, 2005.

[5] S. Shalev-Shwartz, O. Shamir, and S. Shammah, *Failures of deep learning*, 2017, arXiv preprint arXiv:1703.07950.

[6] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87 of *Applied Optimization*, Springer, Amsterdam, The Netherlands, 2004.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[8] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[9] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, pp. 2176–2184, usa, June 2013.

[10] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2121–2159, 2011.

[11] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014, arXiv preprint arXiv:1412.6980.

[12] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML ’15)*, pp. 448–456, July 2015.

[13] A. Neelakantan, L. Vilnis, Q. V. Le et al., *Adding gradient noise improves learning for very deep networks*, 2015, arXiv preprint arXiv:1511.06807.

[14] P. Baranyi, “TP model transformation as a way to LMI-based controller design,” *IEEE Transactions on Industrial Electronics*, vol. 51, no. 2, pp. 387–400, 2004.

[15] P. Baranyi, Y. Yam, and P. Várlaki, *Tensor product model transformation in polytopic model-based control*, CRC Press, 2013.

[16] P. Baranyi, *TP-Model Transformation-Based-Control Design Frameworks*, Springer International Publishing, 2016.

[17] L. de Lathauwer, B. de Moor, and J. Vandewalle, “A multilinear singular value decomposition,” *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[18] M. Ishteva, L. De Lathauwer, P.-A. Absil, and S. Van Huffel, “Dimensionality reduction for higher-order tensors: algorithms and applications,” *International Journal of Pure and Applied Mathematics*, vol. 42, no. 3, pp. 337–343, 2008.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] N. Kalchbrenner, I. Danihelka, and A. Graves, *Grid long short-term memory*, 2015, arXiv preprint arXiv:1507.01526.

[21] J. Kennedy, “Particle swarm optimization,” in *Encyclopedia of Machine Learning*, pp. 760–766, Springer US, Boston, MA, USA, 2011.

[22] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[23] L. Davis, *Handbook of genetic algorithms*, 1991.

[24] M. Gen and R. Cheng, *Genetic algorithms and engineering optimization*, John Wiley & Sons, 2000.

[25] J. H. Holland, *Complexity: A Very Short Introduction*, Oxford University Press, 2014.