2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

On the use of associative memory in Hopfield networks designed to solve propositional satisfiability problems

Natalya Weber
Embodied Cognitive Science Unit
Okinawa Institute of Science and Technology Graduate University
Okinawa, Japan
ORCID: 0000-0002-1955-3612

Werner Koch
Independent Scholar
Dresden, Germany
ORCID:
0000-0001-7246-0434

Ozan Erdem
Independent Scholar
Toronto, Canada
ORCID:
0009-0003-7844-4832

Tom Froese
Embodied Cognitive Science Unit
Okinawa Institute of Science and Technology Graduate University
Okinawa, Japan
tom.froese@oist.jp

***Abstract*—Hopfield networks are an attractive choice for solving many types of computational problems because they provide a biologically plausible mechanism. The Self-Optimization (SO) model adds to the Hopfield network (HN) by using a biologically founded Hebbian learning rule, in combination with repeated network resets to arbitrary initial states, for optimizing its own behavior towards some desirable goal state encoded in the network. However, the solutions to the abstract problems used in the literature offer little insight into how HN arrive at solutions or partial solutions. In order to better understand that process, we demonstrate first that the SO model can solve *concrete* combinatorial satisfiability problems: The Liars problem and the map coloring problem. Based on these solutions, we discuss how under certain conditions critical information might get lost forever with the learned network producing seemingly optimal solutions that are in fact inappropriate for the problem it was tasked to solve. What appears to be an undesirable side-effect of the SO model, can provide insight into its process for solving intractable problems.**

***Index Terms*—self-optimization, Hopfield neural network, Hebbian learning, SAT problems, Liars problem, map coloring problem**

I. Introduction

The combination of domain knowledge and centralized control is an effective solution to a broad class of optimization problems. However, in the case of complex adaptive systems, the system's control tends to be distributed and it is often unclear what the most appropriate trajectory is and even the form of the optimal solution may simply be unknown. This is the case for many kinds of biological systems, but also social systems, that tend to be capable of giving rise to creative solutions even under novel circumstances. Such a complex adaptive system cannot necessarily rely on the availability of explicit extrinsic reward signals to improve its behavior, which raises the intriguing question of what other, more minimal mechanisms could be available.

A particularly interesting model of a distributed complex adaptive system is the Self-Optimization (SO) model. It is a simple model comprised of a Hopfield network (HN) [1] and Hebbian learning [2], two biologically founded [3] and well established mechanisms. Already in 1985 Hopfield and Tank showed that if an optimization problem is formulated in terms of desired optima subject to constraints (i.e. the connections of the network correspond to the constraints of the problem), the natural dynamics of the system is to converge to a stable state that will correspond to a locally optimal solution to that problem with the least violated constraints [4]. Intriguingly, in the SO model, under certain conditions, the combination of Hebbian learning with the dynamics of the HN allows the system to form an associative memory of its own behaviour and change its own dynamics to enhance its ability to find configurations that minimize the constraints between system's variables (nodes of the network), and "find solutions that are better than any solutions found before the application of such learning (i.e. *true optimisation*)" [5, emphasis added].

Closely related to the field of engineering are propositional satisfiability (SAT) problems. The goal in a SAT problem is to determine whether a given logical formula can be made true (i.e. satisfiable) by assigning appropriate truth values to its variables. Many important real-world problems in different scientific fields can be naturally expressed as Max-k-SAT (maximum satisfiability problems with a maximum of k literals per clause) [6]: routing and scheduling problems in industrial engineering, software and hardware debugging in computer science and computer engineering, different problems of bioinformatics in biological sciences, to name a few.

It was previously mentioned [7] that the initial weights of the HN in the SO model represent a weighted-Max-2-SAT problem, but it was never actually shown how one would start from a concrete SAT problem and use the SO model to solve it. Prior investigations into Hebbian learning in neural networks focused on abstract problems, providing a foundational understanding of the process and enabling a comprehensive analysis of its behavior. However, a knowledge gap exists regarding the application of Hebbian learning to concrete problems and its implications for constraint violations. This poses an obstacle for researchers from different fields of science interested in biological plausibility, as they may find it difficult to understand how to directly apply the model to their research.

Thus, our goal in this work is twofold. First, we want to make the SO model more accessible to the community and researchers who seek to apply it and understand its performance with respect to specific, potentially not fully-satisfiable problems. To this end, we promote a method developed in the 1990s to convert any SAT problem to the weights of the Hopfield network. Second, we want to show how, by using a concrete problem, we can further expand our knowledge about the SO model in general by being able to answer questions that we could not answer in the abstract case.

The rest of the paper is structured as follows: in Sec. II and Sec. III, we give a short background to and the description of the "Abdullah method," which is used in this work to translate a SAT problem to the HN weights, which can then be used in the SO simulation. We provide examples for two classic problems - the Liars problem and map coloring. Section IV is broken into two parts. We first show in Sec. IV-A that the SO model successfully solves both problems when a satisfiable solution exists. Then, in Sec. IV-B, we discuss what it means to break constraints in the context of an unsatisfiable problem. Finally, in Sec. V, we discuss the various paths for future research, and in Sec. VI, we draw the conclusions of this work.

II. Background: high-order Hopfield networks and logic for problem solving

Following Hopfield's seminal papers [1], [4], the HN energy function was generalized to include high-order terms. Here we adopt the following notation:

$$\begin{aligned}E^{(k)}(t) = &-\frac{1}{k}\sum_{\gamma_1}^{N}\sum_{\gamma_2}^{N}\cdots\sum_{\gamma_k}^{N} W^{(k)}_{\gamma_1\cdots\gamma_k}\prod_{i=1}^{k} s_{\gamma_i}(t)\\ &-\frac{1}{k-1}\sum_{\gamma_1}^{N}\cdots\sum_{\gamma_{k-1}}^{N} W^{(k-1)}_{\gamma_1\cdots\gamma_{k-1}}\prod_{i=1}^{k-1} s_{\gamma_i}(t) - \cdots\\ &-\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N} W^{(2)}_{ij} s_i(t)\, s_j(t) - \sum_{i}^{N} W^{(1)}_{i} s_i(t) - c,\end{aligned} \tag{1}$$

where N is the number of nodes, k is the order of the energy function, $W^{(k)}_{\gamma_1\cdots\gamma_k}$ is a k-th order tensor of size N^k, symmetric on all pairs of indices γ_i. The s_i are bipolar discrete elements of the system's state vector $\mathbf{S} = \{s_1(t), ..., s_N(t)\}$ of size N, and c is a constant. The standard HN energy function is then a special case of (1) with $k = 2$:

$$E(t) = -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N} W^{(2)}_{ij} s_i(t)\, s_j(t) - \sum_{i}^{N} W^{(1)}_{i} s_i(t) - c. \tag{2}$$

In parallel to researchers in the fields of physics and neuroscience expanding their knowledge of neural networks, researchers in the field of artificial intelligence started working on developing logic programming for problem solving [8]. Then in 1991, [9] expanded this work by showing an equivalence between the search problem of propositional logic satisfiability and the problem of minimizing the energy function (1). Building on that work, [10] presented a method to compute the synaptic weights of the network, which correspond to the propositional logic embedded in the system. The latter, termed later [11] as the "Abdullah method", is the method adopted in this work and is presented in Sec. III-C.

III. SAT Problem conversion to Hopfield network weights

The satisfiability problem in propositional logic (SAT) is a combinatorial problem of deciding for a given propositional formula Φ, whether there exists an assignment of truth values to the propositional variables appearing in Φ under which the formula Φ evaluates to "true". Satisfying assignments are called "models" of Φ and form the solutions of the respective instance of SAT.

In the following we provide two different classical examples of SAT problems - the Liars problem and the map coloring problem.

A. Liars problem

Imagine a room with $N = 4$ people: Alice, Bob, Cal, and Dan. Some (or all) of them can make statements, for example:

> Alice says "Dan is a liar.", Dan says "Bob is a truth-teller", Cal says "Bob is a liar".

The assumption is that each person can either be a liar (always lying) or a truth-teller (always telling the truth). The problem is to find whether a valid assignment of "Liar" or "Truth-teller" for each person exists given the statements above. These facts are represented by a vector of states $\mathbf{S} = \{s_1, \ldots, s_N\}$ with the fact that the i-th person is a liar or a truth teller represented by the state $s_i \in \{0, 1\}$, $i \in [1, N]$, where $s_i = 1$ denotes the person i being a Truth-teller, and $s_i = 0$ denotes the person i being a Liar. Using this notation we can translate the above statements into a *knowledge base,* which represents the constraints for the liars and truth-tellers problem[1]:

$$\{s_1 \Leftrightarrow \neg s_4, s_4 \Leftrightarrow s_2, s_3 \Leftrightarrow \neg s_2\}. \tag{3}$$

Most SAT algorithms operate on propositional formulae in Conjunctive Normal Form (CNF), which is as a *conjunction of disjunctions of literals.* Converting (3) to a CNF gives:

$$\begin{aligned}\Phi = &(\neg s_1 \vee \neg s_4) \wedge (s_4 \vee s_1) \wedge (\neg s_4 \vee s_2)\\ &\wedge (\neg s_2 \vee s_4) \wedge (\neg s_3 \vee \neg s_2) \wedge (s_2 \vee s_3).\end{aligned} \tag{4}$$

Next we consider another example and then in Sec. III-C we show how to convert (4) to the weights of a Hopfield network.

[1]For readers without a background in SAT we recommend the following two [12], [13] online resources for short, but to the point very clear description of SAT and its uses. For a detailed mathematical description see [6].

B. Map coloring

The map coloring problem is a special case of a graph coloring problem. Given the borders between all regions on a map, the goal is to color all regions by distinct colors such that no two bordering regions have the same color. In this case the state of the map is represented by a matrix $\mathbf{S} = \left\{s_1^1, s_1^2, \ldots, s_n^M\right\}$, $i \in [1,n]$, $j \in [1,M]$ with $s_i^j = 1$ denoting the region i being colored with the color j. Given $N = nM$ states for n regions, M colors, and the border adjacency matrix B with the elements $b_{ii'} = 1$ denoting region i and region i' sharing a border and $b_{ii'} = 0$ otherwise, the map coloring problem can be summarized into the three sets of constraints:

1) Each region has to be colored:

$$\varphi_1 = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{M} \left(s_i^j \right) \right). \tag{5}$$

2) A region cannot be two distinct colors at the same time:

$$\varphi_2 = \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{M} \bigwedge_{j' \neq j}^{M} \left(\neg \left(s_i^j \wedge s_i^{j'} \right) \right). \tag{6}$$

3) Regions that share a border should have a different color:

$$\varphi_3 = \bigwedge_{i=1}^{n} \bigwedge_{\{i' \mid b_{ii'}=1\}} \bigwedge_{j=1}^{M} \left(\neg \left(s_i^j \wedge s_{i'}^j \right) \right). \tag{7}$$

Taking the sets of clauses (5), (6), and (7) together gives the formula for the map coloring problem:

$$\Phi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3. \tag{8}$$

Already for $n = 4$ regions, and $M = 2$ colors the propositional formula Φ in (8) will have about 20 clauses so it is too lengthy to present in this paper, but we provide the full analytical derivation in [14]. In the next section we show how to convert (4) and (8) to the weights of a Hopfield network.

C. Using the Abdullah method to translate a logical formula to weights of the Hopfield network

According to [10], [15] determining the states $\mathbf{S}$ that will satisfy Φ is equivalent to a combinatorial minimization of the cost function $E_{\neg\Phi}$ of the *inconsistency* $\neg\Phi$. The value of $E_{\neg\Phi}$ depends on the number of clauses satisfied by the model, such that the more clauses are unsatisfied, the bigger the value of $E_{\neg\Phi}$, and $E_{\neg\Phi} = 0$ denoting that $\neg\Phi$ evaluates to “False”, which means that Φ evaluates to “True” and a state $\mathbf{S}$ that satisfies all the constraints was found. The cost function $E_{\neg\Phi}$ is given by a sum over one term each per negated clause in the logical formula Φ. For example, given formula (4), the inconsistency $\neg\Phi$ is

$$\begin{aligned}\neg\Phi = &(s_1 \wedge s_4) \vee (\neg s_4 \wedge \neg s_1) \vee (s_4 \wedge \neg s_2) \\ &\wedge (s_2 \wedge \neg s_4) \vee (s_3 \wedge s_2) \vee (\neg s_2 \wedge \neg s_3). \end{aligned} \tag{9}$$

The literals s_i and $\neg s_i$ in (9) are mapped to the terms $\frac{1}{2}(1+s_i)$ and $\frac{1}{2}(1-s_i)$, respectively. The entire disjunction of conjuctions is subsequently turned into a sum of products of such terms. In the example given above, the corresponding cost function $E_{\neg\Phi}$ takes the form

$$\begin{aligned} E_{\neg\Phi} = &\frac{1}{4}(1+s_1)(1+s_4) + \frac{1}{4}(1-s_4)(1-s_1) \\ &+ \frac{1}{4}(1+s_4)(1-s_2) + \frac{1}{4}(1+s_2)(1-s_4) \\ &+ \frac{1}{4}(1+s_3)(1+s_2) + \frac{1}{4}(1-s_2)(1-s_3) \\ = &\frac{1}{2}(-s_2 s_3 - s_1 s_4 + s_2 s_4) + \frac{3}{2}. \end{aligned} \tag{10}$$

We obtain the values of connections $W_{ij}^{(2)}$, the biases $W_i^{(1)}$, and the constant c by comparing the cost function $E_{\neg\Phi}$ (10) term by term with the energy function (2). Similarly, if the $E_{\neg\Phi}$ had higher order products, we would need to compare it to a higher order energy function (1).

Figure 1a and Figure 1d show the connections $W_{ij}^{(2)}$ for the Liars problem with $N = 50$ people and for a map coloring problem in a form of a checkerboard that has $n = 8 \times 8 = 64$ tiles that need to be colored by $M = 2$ distinct colors, thus having $N = 64 \times 2 = 128$ states.

Once we have the initial weight matrix $\mathbf{W}_0^{(2)}$, comprised of the elements $W_{ij}^{(2)}$, we can use the SO model to find the states $\mathbf{S}$ that will satisfy Φ. Since the mechanism [7], [16], and the implementation [17] of the SO model were previously covered in length, we do not expand on them here. The two examples serve to show how different the weight connections of the two problems are, which will affect the SO procedure as it will be shown in the next section.

In the next section we present the results of running the SO model for three different scenarios: in Sec. IV-A we show results for the the Liars problem, and the map problem of a checkerboard, and in Sec. IV-B we show the results for coloring the map of South America.

IV. Results

For all the results in this section we use the “on-the-fly” implementation developed in previous work [17], and the algorithm was implemented as a compiled FORTRAN module to be loaded from Python. The shapes comprising the map of South America in Sec. IV-B were obtained from the CShapes 2.0 Dataset [18]. The adjacency for the map coloring problem was determined geometrically from the country map shapes, and the statements for the Liars problem of the type described in Sec. III-A were chosen randomly. The clauses for each problem were generated by a dedicated Python code available in [14]. From the thus computed clauses, the initial weights $\mathbf{W}_0$ for each problem were obtained using the procedure described in Sec. III-C.

We first present in Sec. IV-A that the SO model successfully solves the two different instances of the Liars problem and coloring of a checkerboard. Then in Sec. IV-B we use the problem of coloring the map of South America to show what happens when certain constraints are broken and what are their implications for the problem.

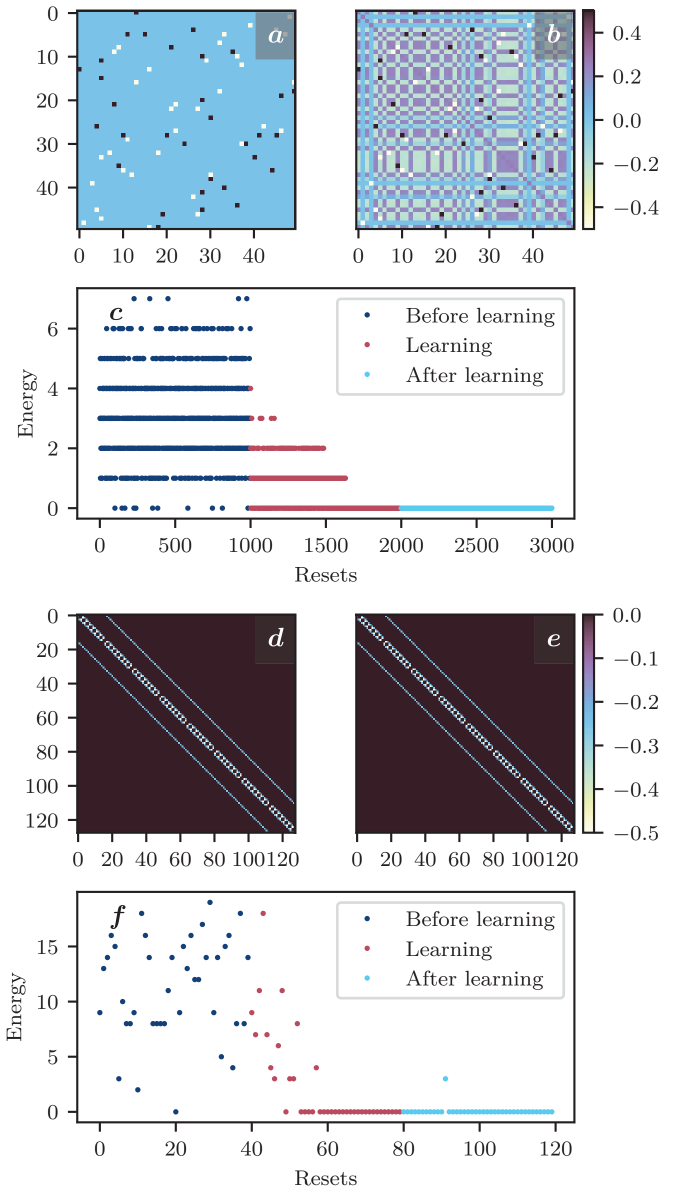


Fig. 1. Simulation results for (a-c) the Liars problem ($N = 50$ people, 34 statements, learning rate $\alpha = 2.5 \times 10^{-7}$, $20N$ steps), and (d-f) the Checkerboard map coloring problem ($N = 2 \cdot 64 = 128$ states, $\alpha = 8 \times 10^{-21}$, $10N$ steps). (a),(d) The initial weights $\mathbf{W}_0$ derived from the cost function $E_{\neg\Phi}$ comparison with (2), (b),(e) the weights $\mathbf{W}$ after learning, (c),(f) the energy at the end of convergence for a set without learning (resets 1–1000 and 1–40, blue), during learning (1001–2000 and 41–80, red), and after learning (2001-3000 and 81-120, light blue) for the two problems, respectively.

A. Solvable problems

The results for the two different instances of the Liars problem and the map coloring problem of a checkerboard are shown in Fig. 1a-c and Fig. 1d-f, respectively. We can see that for both of the problems the SO model converges to an energy $E = 0$, denoting that a state $\mathbf{S}$ that satisfies all the constraints was found for both cases. This is further exemplified by the correct coloring of the checkerboard in Fig. 2. We can see that the system in the checkerboard problem requires far fewer resets than in the Liars problem (about 20 compared to about 700) and a learning rate more than ten orders lower for convergence (resulting in a visually indistinguishable learned weight matrix, Fig. 1e), despite the fact that it has more nodes than in the Liars problem ($N = 128$ compared to $N = 50$). This is because in the case of the checkerboard, there is a band-diagonal structure in the constraints of the problem. It is a very easy problem and just requires the barest of nudges to settle into the correct configuration. In fact, with larger learning rates, the system converges early to an incorrect configuration without being able to escape it later on. Structure in the constraints is known to improve the chances of the SO model of finding a solution [16]. Such structure is missing for the Liars problem, making it difficult for the system to find the proper solution. We will come back to this point in Sec. V.

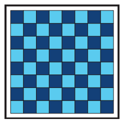


Fig. 2. Colored checkerboard from the learned state $\mathbf{S}$.

B. Coloring the map of South America with 2 colors

In this section we show that one can use the SO model for more difficult maps than just a checkerboard, specifically the map of South America, which requires four colors for a valid solution[2]. Here we use just two, so in this context it is an unsolvable problem. That said, given a situation when one is given just two colors when four are required, we can still ask what is the optimal coloring scheme to obtain a map as comprehensible as possible under those restrictions?

To answer this question we put additional weights ω_i on the different set of constraints φ_i in (8):

$$\Phi = \omega_1\varphi_1 \wedge \omega_2\varphi_2 \wedge \omega_3\varphi_3, \tag{11}$$

and tested 3 different scenarios:

1) $\omega = [1, 1, 1]$ - all constraints have the same weight.
2) $\omega = [5, 5, 1]$ - the color constraints φ_1, φ_2 have a higher weight compared to the set of border constraints φ_3. Meaning that each region first and foremost must be a proper color.
3) $\omega = \left[1, 1, b^L_{ii'}\right]$ - the border constraints φ_3 are weighted by a normalized border adjacency matrix B^L with the elements $b^L_{ii'}$ denoting the border length between countries i and i'. Meaning that adjacent regions are less forced to be colored differently if they have a short border.

The initial weights $\mathbf{W}_0$ and weight matrices $\mathbf{W}$ after the SO simulation for each of these weighed sets are shown in the first two columns of Fig. 3. We can see from Fig. 3 how the initial weights $\mathbf{W}_0$ change corresponding to the different additional weights ω on each of the set of constraints. The second column shows the weight matrices $\mathbf{W}$ after learning (after the learning period shown in red in the third column of Fig. 3). From Fig. 3 we can see several things. First, for the same amount of time, each of the scenarios end in a different result. The lowest energy (corresponding to

[2]While the checkerboard example only required two colors, a general map may require up to four colors. The proof for that statement was a significant achievement [19].

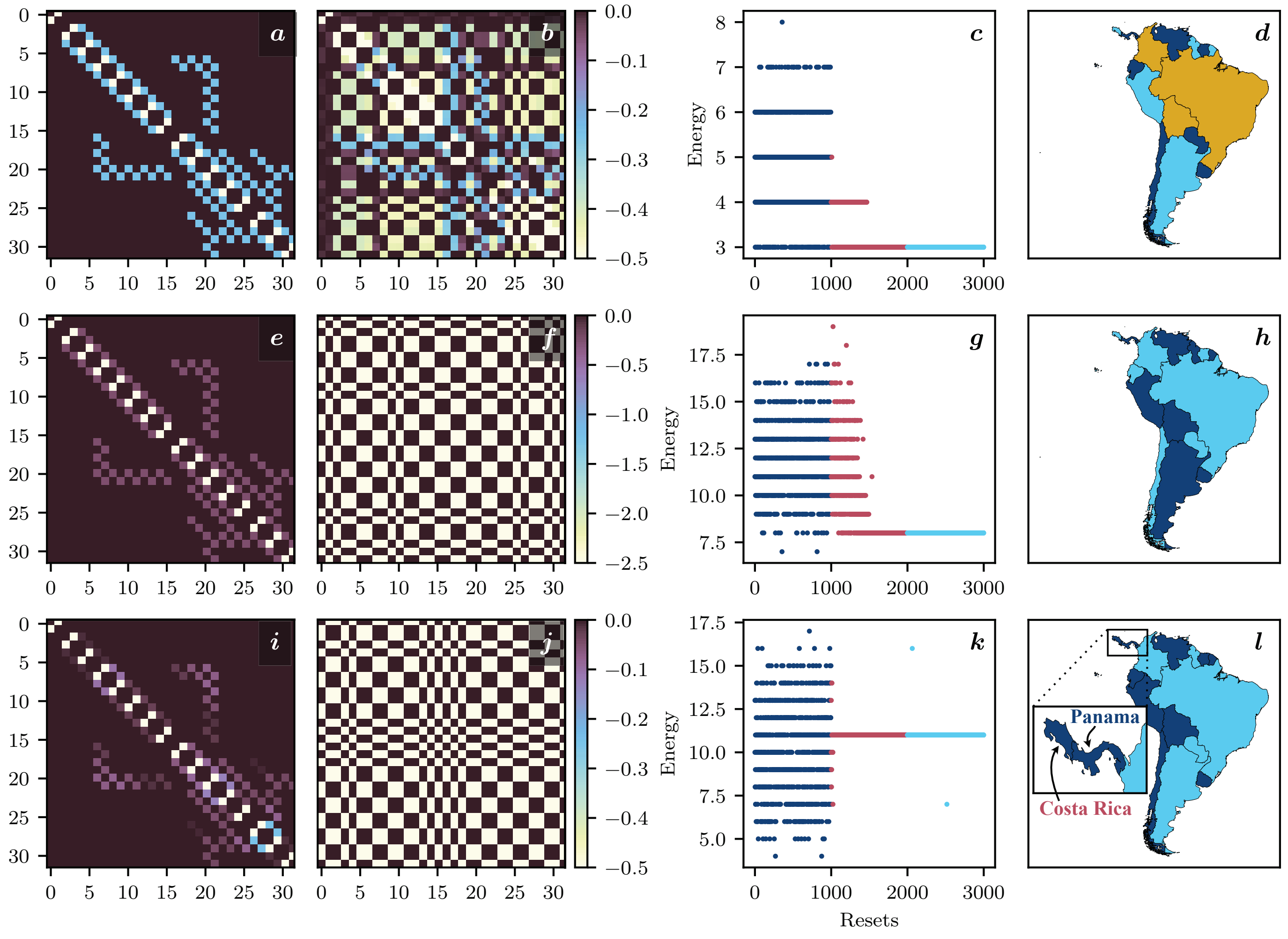


Fig. 3. Self-optimization simulation results for coloring the map of South America ($N = 2 \cdot 16 = 32$ states, $20N$ steps) for (a-d) $\omega = [1, 1, 1]$ (learning rate $\alpha = 8 \times 10^{-7}$), (e-h) $\omega = [5, 5, 1]$ ($\alpha = 2.1 \times 10^{-5}$) and (i-l) $\omega = \left[1, 1, b^L_{ii'}\right]$ ($\alpha = 2.1 \times 10^{-5}$). First and second columns show the weight matrices before and after learning, respectively. Third column shows energy at the end of convergence for a set without learning (resets 1-1000, blue), during learning (1001-2000, red), and after learning (2001-3000, light blue). The right column shows the colored map resulting from the learned state $\mathbf{S}$. Note that while the Hopfield dynamics and learning are computed using modified weights for the lower two rows, the energies in sub-plots g and k are computed from the non-weighted constraints identical to sub-plot c for ease of comparison.

the lowest number of broken clauses) is achieved for the $\omega = [1, 1, 1]$ scenario (Fig. 3, c and d), which corresponds to the regular unweighted formula Φ. For comparison we checked the CNF instance of the problem with the RC2 solver in the PySAT package [20] and it produced the same result. This result, however, shows that three countries do not have a proper color (indicated by yellow color in Fig. 3d). In comparison, the additional weights on the constraints $\omega = [5, 5, 1]$ (Fig. 3, second row) and $\omega = \left[1, 1, b^L_{ii'}\right]$ (Fig. 3, third row) produced proper colors for all countries, which however results in a higher energy (higher number of border constraints broken). Although the two weights on the constraints $\omega = [5, 5, 1]$ and $\omega = \left[1, 1, b^L_{ii'}\right]$ serve the same function of making sure that all countries have proper color, everything else taken equal, they produce different results. An interesting result is obtained for $\omega = \left[1, 1, b^L_{ii'}\right]$ (Fig. 3, k and l). We can see that Costa Rica has the same color as Panama, despite the fact that it does not cost anything to make it a different color. In fact, making it a different color would decrease the energy. But what happens here is that learning is too fast, so it changes the energy landscape to the extent that this constraint is broken “forever”. The Costa Rica state is no longer a local minimum for the original weight matrix. This is easy to fix (see Appendix A) but provides valuable insight into the interplay between learning and constraints as discussed in the next section.

V. Discussion

A. Related work

Solving a SAT problem with Hopfield networks is not a new endeavor, albeit not much research has been done on it. Lima and colleagues introduced the SATyrus [21] that uses the method described in [9] to translate a list of clauses of a SAT problem into the energy function of a HN. The architecture uses simulated annealing [22] to reach global minima. The authors show that SATyrus successfully solves the graph coloring problem, the traveling salesperson problem, and a chemistry problem. On the other side of the globe, using the Abdullah method [10], Kasihmuddin et al. [23], showed that a HN with a relaxation rate augmented update rule [24] is capable of reaching global minima, successfully solving randomly generated instances of Max-2-SAT and Max-3-SAT. The main difference between the SO model and these two research lines is that in the SO model, Hebbian learning is used for optimization. Through Hebbian learning, the weights of the system are constantly changing. This is an advantage of the SO model, where an analysis of the weights allows to study the relationship between biologically plausible unsupervised learning and the constraints of the initial problem, something that is challenging to do with methods such as simulated annealing or relaxation rate update rule.

Hebbian learning in the context of logic of neural networks was investigated before in [15], but with a different aim - to perform a reverse analysis and derive the logical clauses acquired by the system's weights through learning. In comparison, in the current work Hebbian learning is used along with periodic resets of the system in order to enhance the ability of the system to find configurations that minimize the violations of constraints to find optimal solutions.

Similar dynamics to that of the SO model were shown by Power [25] to be present in the coevolution of ecological networks. The author demonstrates how the constraint satisfaction behavior of evolving ecological network enables it to solve Sudoku puzzles. That work, however, does not discuss the implications of learning and breaking constraints on the end result of the solution. In addition, here we further generalize the SO model by re-introducing an underexposed relatively old method, for solving any SAT problem on the SO model. This enables researchers from diverse disciplines to apply the SO model for their specific needs within the community.

B. Limitations of the SO model

In all the example problems used, there is a maximum $k = 2$ literals per clause. The examples of adding additional weights on the constraints in Sec. IV-B can be viewed as weighted-Max-2-SAT problems. The current implementation of the SO model cannot handle SAT instances with $k > 2$ literals per clause[3]. That said it is known that any k-SAT can be reduced to 3-SAT problem, and any 3-SAT problem can be reduced to Max-2-SAT [26] (in both cases at the expense of a linear number of new variables). This means that, in principle, any k-SAT problem can be reduced to Max-2-SAT and then the SO model can be used to solve it.

Another limitation of the model is its poor handling of problems without an underlying structure. As we have seen in Sec. IV-A, although the model does find the solution for the Liars problem, it takes a considerable amount of time even though it was a small problem. This becomes even worse for problems of bigger size. This is related to a known limitation of selective associations [16]. For such problems, the stochastic nature of repeated resets also can be problematic. The Liars Problem discussed in Sec IV-A, for instance, requires fine tuning of the learning rate depending on the seed of the random number generator to converge to the zero energy configuration. Thus one should consider the structure of their problem before using the SO model to solve it.

Despite these limitations, there are many benefits for using the SO model. First, in comparison to existing SAT solvers, although the SO model is not comparable speed-wise, different than SAT solvers the SO model a) finds a solution in a biologically realistic way, and b) it has potential for parallelization even though our implementation is sequential. We note also that in the current work, at each reset, the entire system was reset instantaneously. However, recently we showed [27] that resetting just part of the system allows the system to reach the lowest converged energy in far fewer resets. This adds both more biological plausibility and faster processing.

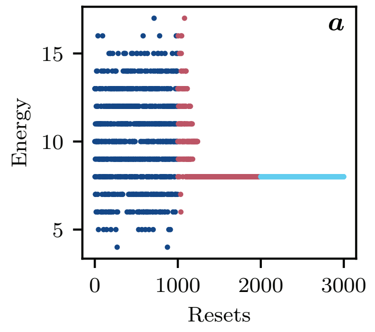
Moreover, even if for some combinatorial optimization problems the SO model might not produce solutions that will satisfy all the constraints, our experiments indicate that it can still yield high quality solutions. These solutions could hypothetically prove beneficial in various contexts in optimization solvers. For instance, such a high quality, but imperfect, solution to a combinatorial optimization problem provides valuable insight into the local consistencies of variable assignments. These assignments could potentially be used to guide the search in a SAT solver, serving as variable selection or phase selection heuristics [6]. A concrete application of such solutions is described in [28], which can use a partial, hence imperfect, assignment to drive a correct solution. We posit that SO can be successfully utilized in similar situations, and we plan to explore these possibilities in future research.

VI. Conclusion

This work demonstrates the practical application of the Self-Optimization algorithm to concrete combinatorial problems in SAT form. Through two different examples, the Liars problem and the map coloring problem of a checkerboard, we showcase the model's capability to find optimal configurations that represent the solutions to these problems. We further expanded the approach by analyzing a more complex problem: coloring the map of South America with just two colors. This problem has no solution. In principle, four colors are needed to avoid color clashes at borders. The goal of that example was to provide insight into what happens to the end result when individual constraints of the problem are broken.

Specifically, the last example of Sec. IV-B, shows how Hebbian learning has dramatically modified the weight matrix that encodes the problem constraints such that some of these constraints are no longer contained within. In the example, in the unmodified weight matrix the color of Costa Rica can be trivially changed to satisfy the constraint on having opposite colors across borders, however a state with that color flipped is no longer an energetic minimum of the learned weight matrix.

[3]For example, in the Liars problem, $k = 3$ literals would be in the case if a person made a statement about two people (e.g. "Cal says Alice and Bob are both liars"). In the map coloring problem, that would be the case if there would be three colors available.



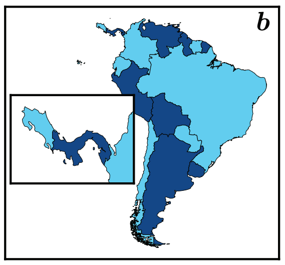


Fig. 4. Self-optimization simulation results for coloring the map of South America ($N = 2 \cdot 16$ states, $20N$ steps) $\omega = \left[1, 1, b^L_{ii'}\right]$ ($\alpha = 6 \times 10^{-6}$). Sub-plot a shows energy at the end of convergence for a set without learning (resets 1-1000, blue), during learning (1001-2000, red), and after learning (2001-3000, light blue). Subplot b shows the colored map resulting from the learned state $\mathbf{S}$. For this learning rate, the weights on the constraints $\omega = \left[1, 1, b^L_{ii'}\right]$ result in the system converging to the same energy state as for $\omega = [5, 5, 1]$ (compare to Fig. 3g-h).

An observation of this form cannot be derived from changes to the weight matrix in an abstract problem. At this point we do not know how to exploit that observation to improve the model or the approach, but this should definitely provide an avenue for further research. For instance it is interesting to investigate how breaking some constraints helps in solving the rest of the problem. Another fruitful direction of inquiry might be to check whether broken constraints can later be mended. A concrete problem can be utilized in these cases to understand the effect of changes to the model or the solution strategy.

VII. Data availability statement

The model from the main text as well as the code used for the simulation are available at [14]. The shapes comprising the map of South America in Sec. IV-B were obtained from the CShapes 2.0 Dataset [18].

VIII. Acknowledgment

We are grateful for the help and support provided by the Scientific Computing and Data Analysis section of Research Support Division at OIST.

Appendix

The broken border color constraint resulting from constraint weighting by border length as depicted in Fig. 3 sub-plot l can be fixed by employing a less aggressive learning rate as shown in Fig. 4.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[2] D. O. Hebb, *The organization of behavior, a neuropsychological theory*, ser. The organization of behavior, a neuropsychological theory. Oxford, England: Wiley, 1949.

[3] F. Pulvermüller, R. Tomasello, M. R. Henningsen-Schomers, and T. Wennekers, "Biological constraints on neural network models of cognitive function," *Nat Rev Neurosci*, vol. 22, no. 8, pp. 488–502, Aug. 2021.

[4] J. J. Hopfield and D. W. Tank, ""Neural" computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152, Jul. 1985.

[5] R. Watson and M. Levin, "The collective intelligence of evolution and development," *Collective Intelligence*, vol. 2, no. 2, p. 26339137231168355, Apr. 2023.

[6] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. NLD: IOS Press, Jan. 2009.

[7] R. A. Watson, C. L. Buckley, and R. Mills, "Optimization in "self-modeling" complex adaptive systems," *Complexity*, vol. 16, no. 5, pp. 17–26, 2011.

[8] R. Kowalski, *Logic for problem solving*, ser. Artificial intelligence series. New York: Elsevier North Holland, 1979.

[9] G. Pinkas, "Symmetric Neural Networks and Propositional Logic Satisfiability," *Neural Computation*, vol. 3, no. 2, pp. 282–291, Jun. 1991.

[10] W. A. T. W. Abdullah, "Logic programming on a neural network," *International Journal of Intelligent Systems*, vol. 7, no. 6, pp. 513–519, 1992.

[11] S. Sathasivam and W. A. T. W. Abdullah, "Logic Learning in Hopfield Networks," *Modern Applied Science*, vol. 2, no. 3, p. p57, May 2008.

[12] T. W. Neller, Z. Markov, I. Russell, and D. Musicant, "Clue Deduction: an introduction to satisfiability reasoning," May 2008.

[13] O. Erdem, "Encoding Problems in Boolean Satisfiability," https://ozanerdem.github.io/jekyll/update/2019/11/17/representation-in-sat.html, Nov. 2019.

[14] N. Weber, "SO for SAT problems," *(available at: https://github.com/nata-web/SO_for_SAT)*, Jul. 2023.

[15] W. Ahmad Tajuddin Wan Abdullah, "The logic of neural networks," *Physics Letters A*, vol. 176, no. 3, pp. 202–206, May 1993.

[16] R. A. Watson, R. Mills, and C. Buckley, "Transformations in the scale of behavior and the global optimization of constraints in adaptive networks," *Adaptive Behavior*, vol. 19, no. 4, pp. 227–249, Aug. 2011.

[17] N. Weber, W. Koch, and T. Froese, "Scaling up the self-optimization model by means of on-the-fly computation of weights," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2022, pp. 1276–1282.

[18] G. Schvitz, L. Girardin, S. Rüegger, N. B. Weidmann, L.-E. Cederman, and K. S. Gleditsch, "Mapping the International System, 1886-2019: The CShapes 2.0 Dataset," *Journal of Conflict Resolution*, vol. 66, no. 1, pp. 144–161, Jan. 2022.

[19] D. MacKenzie, "Slaying the Kraken:: The Sociohistory of a Mathematical Proof," *Soc Stud Sci*, vol. 29, no. 1, pp. 7–60, Feb. 1999.

[20] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: A Python Toolkit for Prototyping with SAT Oracles," in *Theory and Applications of Satisfiability Testing – SAT 2018*, ser. Lecture Notes in Computer Science, O. Beyersdorff and C. M. Wintersteiger, Eds. Cham: Springer International Publishing, 2018, pp. 428–437.

[21] P. Lima, M. Morveli-Espinoza, G. Pereira, and F. Franga, "SATyrus: a SAT-based neuro-symbolic architecture for constraint processing," in *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, Nov. 2005, p. 6.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[23] M. S. M. Kasihmuddin, M. A. Mansor, and S. Sathasivam, "Discrete hopfield neural network in restricted maximum k-satisfiability logic programming," *Sains Malaysiana*, vol. 47, no. 6, pp. 1327–1335, Jun. 2018.

[24] X. Zeng and T. R. Martinez, "Improving the Performance of the Hopfield Network By Using A Relaxation Rate," in *Artificial Neural Nets and Genetic Algorithms*, A. Dobnikar, N. C. Steele, D. W. Pearson, and R. F. Albrecht, Eds. Vienna: Springer, 1999, pp. 67–72.

[25] D. Power, "Distributed associative learning in ecological community networks," Ph.D. dissertation, University of Southampton, Mar. 2019.

[26] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[27] T. Froese, N. Weber, I. Shpurov, and T. Ikegami, "From autopoiesis to self-optimization: Toward an enactive model of biological regulation," *Biosystems*, vol. 230, p. 104959, Aug. 2023.

[28] S. Cai, X. Zhang, M. Fleury, and A. Biere, "Better Decision Heuristics in CDCL through Local Search and Target Phases," *Journal of Artificial Intelligence Research*, vol. 74, pp. 1515–1563, Aug. 2022.