Constraint Satisfaction Problems solution through Spiking Neural Networks with improved reliability: the case of Sudoku puzzles

Constraint satisfaction problems solution through spiking neural networks with improved reliability: The case of Sudoku puzzles

Author
Riccardo Pignari, Vittorio Fra, Evelina Forno, Enrico Macii, Gianvito Urgese – Politecnico di Torino, Electronic Design Automation (EDA) Group, Torino, Italy

Citation
Pignari, R., Fra, V., Forno, E., Macii, E., Urgese, G. Constraint satisfaction problems solution through spiking neural networks with improved reliability: The case of Sudoku puzzles.

Introduction

Constraint satisfaction problems (CSPs) are a subset of NP-Complete problems: they belong to the NP (nondeterministic polynomial-time) class, a common example of CSP is the latin square problem in the variant of Sudoku puzzles. A possible method to find a solution is through a neuromorphic approach as shown in [1]. Specifically, a stochastic version of a Spiking Neural Network (SNN) made of Leaky Integrate-and-Fire (LIF) neurons and with a structure defined over the mathematical formulation of the CSP can be employed. Such a network describes a system that stochastically evolves towards the solution of the Sudoku puzzle, following the attractor dynamics, in the configuration space [2, 3]. Despite showing the capability of solving this and other CSP problems, [1] presents some limitations that should be addressed:

(i) each solution attempt requires a pre-set simulation time that cannot be modified once started, thus hindering the possibility of stopping the process once the solution has been found and introducing unnecessary energy consumption;

(ii) the validation method is carried out through the binning process, which consists in extracting the spikes from the neuromorphic platform and analyzing its state through an external platform which inherently implies additional energy consumption due to data preparation and transfer;

(iii) the original problem through a mapping process is encoded in the SNN network, given the complexity of some problem classes there may be limitations on the reliability of the network in modeling the initial addresses, observing a modification of these elements during the simulation.

Here we propose a fully spiking pipeline able to find a solution, validate it and stop the generation of spikes.

Methods

In our work [4], we adopted the same mapping of the Sudoku problem into an SNN as proposed in [1]. In this implementation, the variable associated with each of the 81 cells of the sudoku puzzle is represented by 9 populations of neurons, one for each possible value, in a winner-take-all configuration. The constraints are modelled through lateral-inhibition by connecting different populations across cells. The resulting structure defines a sparsely connected graph which encloses the formulation of the original problem in the number of neurons and their connections. In order to address the above mentioned issues of [1] we implemented two major changes. Hereinafter, we refer to the original SNN as in [1], being the one directly in charge of exploring the configuration space, as the *CSP-Solver*, and to our changes as the *Clue-Hold* and the *Spike-Val* subnetwork.

The *Spike-Val* subnetwork is introduced to deal with issues (i) and (ii), and it is composed of 3 blocks with different tasks:

- *Block-1*: it verifies that the constraints on the cell's values are met, and it consists of two levels:

1. It has a structure like that of the *CSP-Solver* with a reduced number of neurons per population, which is used to clear the SNN output of the background noise.

2. It allows us to verify that the 3 types of constraint, namely on the row, on the column and within the subgrid, are satisfied.

- *Block-2*: with a structure made of a single layer, it verifies the solution, interrupts the activity of the main network and activates *Block-3).*

- *Block-3*: it holds the activity state of the solution found by the main network.

The *Clue-Hold* tackles instead issue (iii) by implementing unidirectional lateral-inhibition from the cells corresponding to the clues to the empty ones. Such symmetry break ensures that the initial clues cannot be affected by the activity of the populations they must hinder from assuming specific values along a row, a column or within a subgrid.

Conclusion

To verify our implementation, we focused on the class *easy* of Sudoku puzzles, using the one reported in [1] and two additional ones of the same class [5] for 300 solution trials for each of them. All of the 900 trials have been performed with both the original implementation [1] and our proposed pipeline. In *Figure 1a,* the two approaches are compared in terms of solutions found, and validated, expressed as a percentage of the total set of experiments. The gain of more than 9% we achieved can be ascribed to the *Clue-Hold*, which ensures that the original formulation of the CSP is held during the overall simulation. A direct and quantitative evaluation of the impact of *Clue-Hold* is shown in *Figure 1b* where more than 11% of the experiments carried out with the original implementation were affected

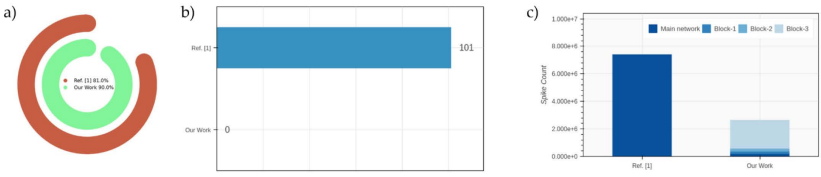

FIGURE 1
Results from 900 trials. **(a)** Percentage of solutions found and successfully validated. **(b)** Number of clues change occurring during the network activity. **(c)** Median value of the total spike count.

by such an issue, which was instead completely fixed with our SNN solver. *Figure 1c* reports the median of the spiking activity produced, where all the contributions from the different portions of the network are highlighted. Despite having additional neurons involved in the dynamic evolution of the system, our implementation reduces to about 3× the total number of spikes, such a result arises from the possibility, given by the *Spike-Val* subnetwork, of deactivating the main network once a solution is found. Starting from [1], we therefore have shown that the solution to a specific class of CSPs as that of Sudoku puzzles can be found and validated in an efficient, reliable and fully spiking way.

References

[1] G. A. Fonseca Guerra et al., Front. Neurosci., vol. 11, p. 714, Dec. 2017.

[2] J. J. Hopfield, Proc. Natl. Acad. Sci. U.S.A., vol. 79, n. 8, p. 2554-2558, Apr. 1982.

[3] G. Pedretti et al., IEEE Int. Symp. Circuits Syst., p. 1-5, Oct. 2020.

[4] R. Pignari et al., *under submission*.

[5] T. Mantere et al., IEEE CEC, p. 1382-1389, Sep. 2007.