Increasing Simulated Radiation Robustness of Graph Spiking Neural Networks

Leveraging brain-adaptation mechanisms in a Minimum Dominating Set Approximation Algorithm

Akke Toeter

Supervisors
TU Delft:
Dr. D.M. Stam
Dr. A. Menicucci
Radboud University:
Dr. J.H.P. Kwisthout

TU Delft

Acknowledgements

I would like to express my gratitude to Dr. J.H.P. Kwisthout for consistently and kindly providing me with feedback and guidance. Dr. D.M. Stam for enabling me to pursue this research from the earliest conception of the idea, and Dr. A. Menicucci for providing me with technical and strategic advice throughout this project. Elia Montanari and the SpaceBrains foundation for supporting this work. Nathal Dorval from ESA for providing me with feedback on my progress during this work. The NEUROTECH consortium for supporting in this work.

I thank Wies, for her support and the joy she brings to my life, Erik, Jelmer and Diederik, for their trust, and for leading by example. Victoria, Jochem, David, Arne and Antonie for providing feedback on my work. My fellow students for joining the exploration into new ventures.

Most of all, I would like to express my deepest gratitude to my parents and sister. Throughout this incredible journey of pursuing my master's degrees, their unwavering support, love, and encouragement has been instrumental in my success.

List of Figures

Acronyms

LIF Leaky-Integrate-and-Fire. 4, 5, 25, 31

MDS Minimum Dominating Set. ix, 31

MDSA Minimum Dominating Set Approximation. ix, 2, 5, 6, 9–11, 16–21, 25, 27, 29–31

RRAM Resistive Random Access Memory. 1, 31

SEE Single-event Effects. 11, 17–19, 31

SNN Spiking Neural Network. ix, 1, 3–11, 13, 17–21, 23–25, 27–31

STDP Spike-Timing-Dependent Plasticity. 1, 8, 31

Contents

Abstract

Neuromorphic architectures are of interest in space application due to their energy efficiency. However, space radiation has been shown to damage computational hardware. Research has been performed on the radiation robustness of (pre-trained) spiking neural networks, and the brain has shown to be able to recover from certain types of lesions. Other work has shown Spiking Neural Networks (SNNs) can obtain advantages for graph algorithms. Such (distributed) SNN graph algorithms may become relevant for space applications. For example, SNNs may form a neuromorphic implementation of the Minimum Dominating Set (MDS) algorithm that others have proposed for distributed satellite swarm coordination. This research combines the trend of SNN implementations of distributed graph algorithms with neuromorphic space applications and brain-adaptation induced robustness as inspiration. It shows that brain inspired adaptation mechanisms can increase the simulated radiation robustness of an SNN implementation of an unweighted, distributed Minimum Dominating Set Approximation (MDSA) algorithm. An SNN implementation of the MDSA algorithm by Alipour et al. is created and used for this experiment. That SNN is adapted with a population coding approach, and with a sparse redundancy approach. These three SNNs are exposed to simulated radiation effects in the form of synaptic weight increases which occur with a probability of 0.001% to 20% per synapse per timestep. Separate simulations are performed with a simulated radiation induced permanent neuron death with a probability of 0.01 % to 25 % per neuron per timestep. The SNN performance is measured by comparing its output to the unradiated algorithm output. The sparse redundancy increases the robustness against simulated radiation induced neuron death for radiation probabilities from 0.5 % to 25% per neuron per time step, for the MDSA SNN. Below these probabilities, the adaptation mechanism is contra productive. Similarly, population coding is contra productive below 0.1% and increases radiation robustness for simulated synaptic weight increase probabilities of up to 5%.

Chapter 1

Introduction

Space exploration poses several challenges. Launching material into orbit is still expensive, spacecrafts are subject to strict energy budgets, and once in space, the environment can be harsh due to extreme temperatures, a vacuum and radiation. On top of that, deep space missions have to deal with increasing communication delays with greater distances from Earth. Neuromorphic engineering is a field that can yield applications and use-cases in the domain of space exploration to overcome some of these challenges.

Neuromorphic architectures are based on the structure of the brain, in the sense that they perform computations using neurons and synapses, with event-driven communication [34]. This allows for large parallel computations with, low power consumption in various applications, such as event-based vision cameras, and graph algorithms [3, 5, 38]. Neuromorphic hardware can accelerate data processing tasks by leveraging its energy efficiency and parallel processing capabilities [45]. Its ability to perform pattern recognition and data analysis on board can reduce the need for transmitting raw data back to Earth. This can save valuable bandwidth and reduce communication latencies [6].

Additionally, neuromorphic hardware is capable of performing decision-making and real-time processing tasks [13, 25, 35]. This is valuable in space missions where immediate responses to its environment, and adaptive behaviour enables autonomous spacecraft to process sensory inputs and make decisions without relying heavily on Earth-based communication and control systems. This can increase the rate of exploration of autonomous space robots [39]. Besides strict energy constraints of space applications, space radiation forms another challenge for space hardware [20]. Von Neumann hardware can be vulnerable to such environmental factors and typically requires protective measures such as radiation hardening and majority voting systems [11, 12, 20]. Neuromorphic hardware, with its parallel and fault-tolerant architecture, may be more robust- and/or resilience against space radiation than traditional Von Neumann architectures. Research is performed on the radiation effects on neuromorphic space hardware and its radiation robustness. For example, Zhilu Ye et al. evaluated the radiation effects in a Resistive Random Access Memory (RRAM)-based neuromorphic computing system for inference [49], and Dahl et al. simulated radiation effects on a pre-trained SNN [21]. The latter demonstrated that the Spike-Timing-Dependent Plasticity (STDP) learning implemented in the SNN enabled it to recover from radiation damage. The brain has also been shown to adapt to regain lost functionality after certain lesions through plasticity [43]. However, to this date, no radiation robustness studies have been found that were performed on neuromorphic graph algorithms. This research sets out to investigate whether brain-inspired adaptation mechanisms can be used to increase the radiation robustness of graph SNNs in neuromorphic space hardware.

Work by Aimone et al. demonstrated that SNNs running on neuromorphic hardware can also be advantageous for graph algorithms, and mentioned that inspiration for efficient SNNs may be sought in distributed graph algorithms [3]. Distributed graph algorithms are frequently applied in robotic swarms [9, 22, 23]. In particular, a weight based dominating set clustering algorithm has been presented for small satellite networks [42]. This algorithm by Qin et al. starts by computing a minimum dominating set, and uses that set as cluster heads from which the entire swarm will be clustered [42]. Alipour et al. note in [4] that finding a minimum dominating set is NP-complete [32], even if planar graphs are permitted with at most 3 neighbours per vertice (degree 3) [7].

Combining these trends for energy efficient, robust neuromorphic graph algorithms, such as finding a minimum dominating set (see chapter 2), for distributed space applications, this research aims to determine whether brain-inspired adaptations can be used to increase the simulated radiation robustness of a SNN

implementation of a distributed Minimum Dominating Set Approximation (MDSA) algorithm. Chapter 2 describes the experiment setup that is used to investigate this idea. The software used to perform the experiment is described in chapter 3. The experiment results are presented in chapter 4, along with the adaptation costs, in terms of neuronal-, synaptic- and energy complexity. These results are discussed in chapter 5 and the research conclusion is presented in chapter 6. Several recommendations for future research are included in chapter 7.

Chapter 2

Methodology

This section describes the experiment setup that is used to generate the results in chapter 4. First, the algorithm that is selected for this experiment is given in section 2.1. Section 2.2 describes how that algorithm is implemented in an SNN. Section 2.3 describes how the radiation effects that may occur on neuromorphic space hardware, are simulated. The adaptation mechanisms, tailored to these effect types, are then described in section 2.4. Section 2.5 contains the specification of the experiment parameters that are used to generate the results. The SNN performance measurement procedure is detailed in section 2.6, and the method to quantify the adaptation costs is given in section 2.7.

2.1. Algorithm Selection

As motivated in the introduction, this experiment uses the minimum dominating set approximation (MDSA) algorithm, as presented by Alipour et al. [4]. The decision variant of the dominating set problem can be specified as [47]:

> **Dominating Set**
> **Input:** A graph $G = (V, E)$ and a positive integer k.
> **Question:** Does there exist a dominating set $V' \subseteq V$ such that $|V'| \leq k$? (Here a vertex set V' is called a dominating set if for every vertex $v \in V$ we have either $v \in V'$ or $u \in V'$ for some u such that $(u, v) \in E$.)

A minimum dominating set $V^* \subseteq V$ for graph G consists of the smallest number (k^*) of vertices $V^* \subseteq V$ that form a dominating set in graph G. Furthermore, a minimum total dominating set $V^{\cdot} \subseteq V$ for graph G is a minimum dominating set for which all vertices $V^{\cdot}$ are neighbours, meaning the minimum total dominating set forms a connected subgraph $G^{\cdot} \subseteq G$. An approximation of the minimum dominating set typically contains more vertices than the number of vertices in the minimum dominating set. The algorithm used by Alipour et al. has an approximation factor of at most 32 for the minimum dominating set. That means the dominating set that is returned with the approximation contains at most 32 times as many vertices as the actual minimum dominating set. This minimum dominating set approximation algorithm is specified in algorithm 1.

Algorithm 1: Distributed Algorithm for computing a total dominating set in a graph with given integer $m \geq 0$.

Input: Connected, planar, triangle-free graph of size n.
Output: Set of vertices that form a total dominating set (TDS).

1 In the first round, each vertice v_i chooses a random number $0 < r_i < 1$ and computes its weight $w_i = d_i + r_i$ and sends w_i to its adjacent neighbours.;
2 In the second round, each vertice v marks a neighbour vertex v_i whose weight w_i is maximum among all the other neighbours of v.;
3 **for** m rounds **do**
4 Let x_i be the number of times that a vertex is marked by its neighbour vertices, let $w_i = x_i + r_i$;
5 Unmark the marked vertices.;
6 Each vertex marks the vertex with maximum w_i among its neighbour vertices.;
7 **end**
8 8: The marked vertices are considered as the vertices in our total dominating set for G.;

The implementation takes in connected, triangle-free, planar graphs (e.g., fig. 2.1). The following aspects of this algorithm are highlighted, as they are relevant in section 2.2:

1. The value of the mark is explicitly defined as: a single mark has a value of $mark_{val} = 1$ in this algorithm.
2. It is noted that the function of the random numbers r_i in algorithm 1 is to generate a random ordering with random values that do not exceed the value of the mark $mark_{val}$. So if the single mark value $mark_{val}$ increases with a factor n, one can increase the maximum value of r_i with a factor n whilst preserving which vertices are marked in step 8 of algorithm 1.

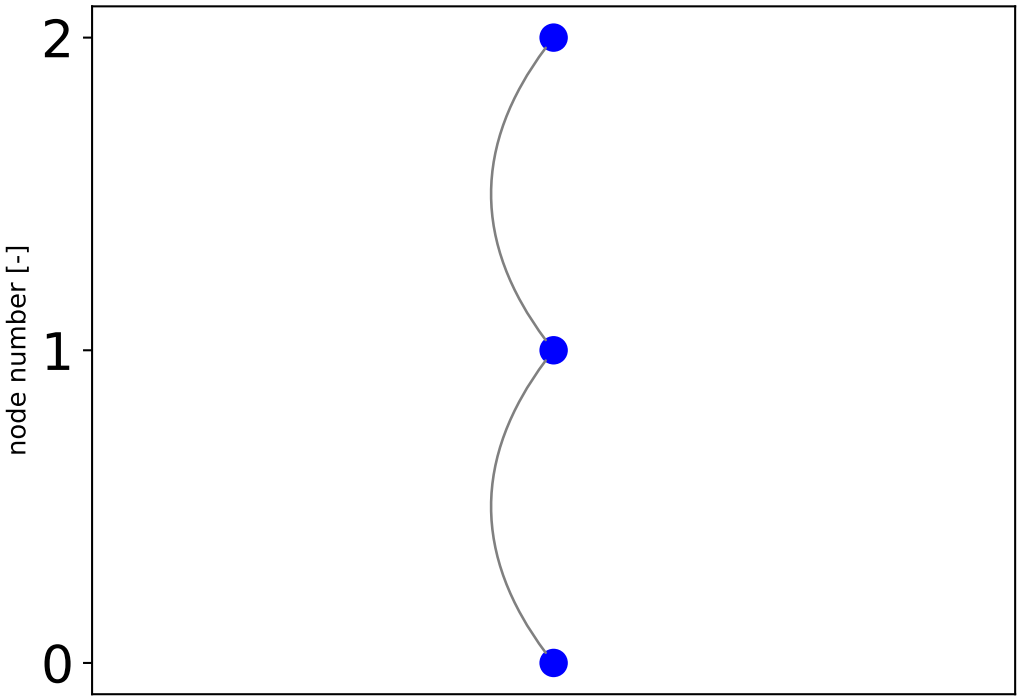


Figure 2.1: An undirected, connected, planar, triangle-free input graph of size $n = 3$ vertices.

2.2. SNN Algorithm Implementation

Algorithm 1 is implemented in an SNN using Leaky-Integrate-and-Fire (LIF) neurons as documented in the source code of the LIF object of the Lava framework for neuromorphic computing release V0.7.0 [27]. These neurons can be described with the following set of equations:

1. $u(t) = u(t-1) \cdot (1 - du) + a_{in}(t)$
2. $v(t) = \begin{cases} 0 & \text{if } v(t) > vth \\ v(t-1) \cdot (1 - dv) + u(t) + bias & \text{otherwise} \end{cases}$
3. $s_{out}(t) = v(t) > vth$

Where:

- $u(t)$ can be described as the current of the neuron at time step t.
- du can be described as a current leakage factor. It is constant over time.

- $a_{in}(t)$ is the cumulative sum of the input signals that the neuron receives at time step t.
- vth is the spike threshold. If the neuron voltage exceeds this threshold, the neuron spikes.
- v is the voltage/potential of the neuron, at time step t. After a spike, the neuron potential resets to $0[V]$, regardless of the value of the (non-recurrent) output signals of that neuron.
- dv can be described as the potential leakage factor. It is constant over time.
- $bias$ can be described as the potential leakage term. It is constant over time.
- $s_{out}(t)$ is a boolean indicating whether or not the neuron will spike at that time step or not.

For each input graph, the LIF neurons are composed to replicate the functionalities specified in algorithm 1. To give a high-level explanation of the SNN implementation of the MDSA algorithm, the following terminology is used:

- n is the number of vertices in the input graph G.
- i represents an index of a vertice in the input graph G, with $0 \leq i < n$ and $i \in \mathbb{N}$.
- d_i is the degree (number of neighbours) of vertice i in the input graph G.
- j_i represents a neighbour vertice of vertice i in the input graph G, with $0 \leq j < n-1$, $i \neq j$ and $j \in \mathbb{N}$.
- r_i represents the random number of vertice i in the input graph G, with $0 \leq r_i < n$ and $r_i \in \mathbb{N}$. Note, the function of the random numbers r_i, is to generate an order among the vertices i in graph G (in case of identical degrees for different vertices). To full-fill this task without constraints, it is irrelevant whether they are floating numbers or natural numbers, nor is it relevant what their range is, as long as all the numbers r_i are unique and smaller than the value of a single mark.
- m_{max} is the maximum number of iterations for which the MDSA algorithm is ran. It is noted by Alipour et al. that the algorithm converges for low m values, their benchmark range is: $m \in [1,2,5]$ accordingly, [4].
- m represents the approximation round of the algorithm, with $0 \leq m \leq m_{max}$ and $m \in \mathbb{N}$.
- $ceiling\_inhibition$ is a single value per input graph, that is used to ensure the current $u(t)$ of each degree_receiver neuron initialises negative, after it has received its weight w as input signals.

Looking at fig. 2.1, one can see that $n = 3$, with indices $i \in [0,1,2]$, and a maximum degree of 2 is found for vertice 1 (because it has two neighbours). Furthermore, fig. 2.2 shows the basic, unadapted SNN for the input graph displayed in fig. 2.1. Section 2.2.1 describes how this SNN is constructed.

2.2.1. Network Construction

The SNN consists of 3 main segments, which each consist of groups of neurons: the initialisation segment, the round computation segment, and the output storage segment. The initialisation- and storage segments scale linearly with the number of vertices of the input graph, but are constant otherwise. The middle/round computation segment, scales linearly with the edges in the input graph, and grows with one layer for every approximation round m. This network construction explanation starts with that middle segment, and proceeds with the storage- and initialisation segment that are connected to this middle/round computation segment. To see an example of the network with a middle/round computation segment of 2 layers ($m = 1$), one can look at Figure 2.2.

For each round m of the algorithm, for each vertice i in the input graph G, a pair is created. This pair consists of one group of $degree\_receiver_{i,j_i,m}$ neurons, and one $selector_{i,m}$ neuron. Those pairs are used to determine which vertices should receive marks. These marks are then passed to the relevant $degree\_receiver_{i,j_i,m+1}$ neurons of the next round $m+1$ for $m < m_{max}$.

For the last round of the algorithm ($m = m_{max}$), these marks are passed to the $counter_{j_i}$ neurons. The $counter_{j_i}$ neurons store the output of the SNN as the current $u(t)$.

The degrees d_i of vertices i of graph G, are encoded as synaptic inputs from the $spike\_once_i$ neurons into the $degree\_receiver_{i,j_i,m}$ neurons of the first round ($m = 0$). The random number r_i belonging to a

vertice i, is encoded as synaptic input from the $rand_i$ neuron into the $degree\_receiver_{i,j_i,m}$ neurons for every round m. To ensure those $degree\_receiver_{i,j_i,m}$ neurons are initialised with a negative current $u(t)$, they are inhibited by the $rand_i$ incoming neuron synapses with the $ceiling\_inhibition$ value that is fixed per input graph G.

Per round m, the group of $degree\_receiver_{i,j_i,m}$ neurons belonging to vertice i, are charged by the $selector_{i,m}$ neurons, until the first $degree\_receiver_{i,j_i,m}$ neuron in that group belonging to vertice i, spikes. Then, the winning $degree\_receiver_{i,j_i,m}$ neuron inhibits the other $degree\_receiver_{i,k_i,m}$ neurons in that group, and inhibits the $selector_{i,m}$ neuron. The random number is hence used to brake any tie that may occur between $degree\_receiver$ neurons. Once all $selector_m$ neurons of round m have been inhibited, a $next\_round$ neuron spikes to ensure the $selector_{m+1}$ neurons start spiking, granted the SNN is not at the final round ($m < m_{max}$).

Once the network has reached its final round m_{max}, and all $selector_{m_{max}}$ neurons have been inhibited, the $terminator\_node$ spikes to indicate the network is done with its computations. The $counter$ neurons with currents $u(t) > 0$ form the total dominating set of input graph G, and approximate the minimum dominating set with at most a factor 32.

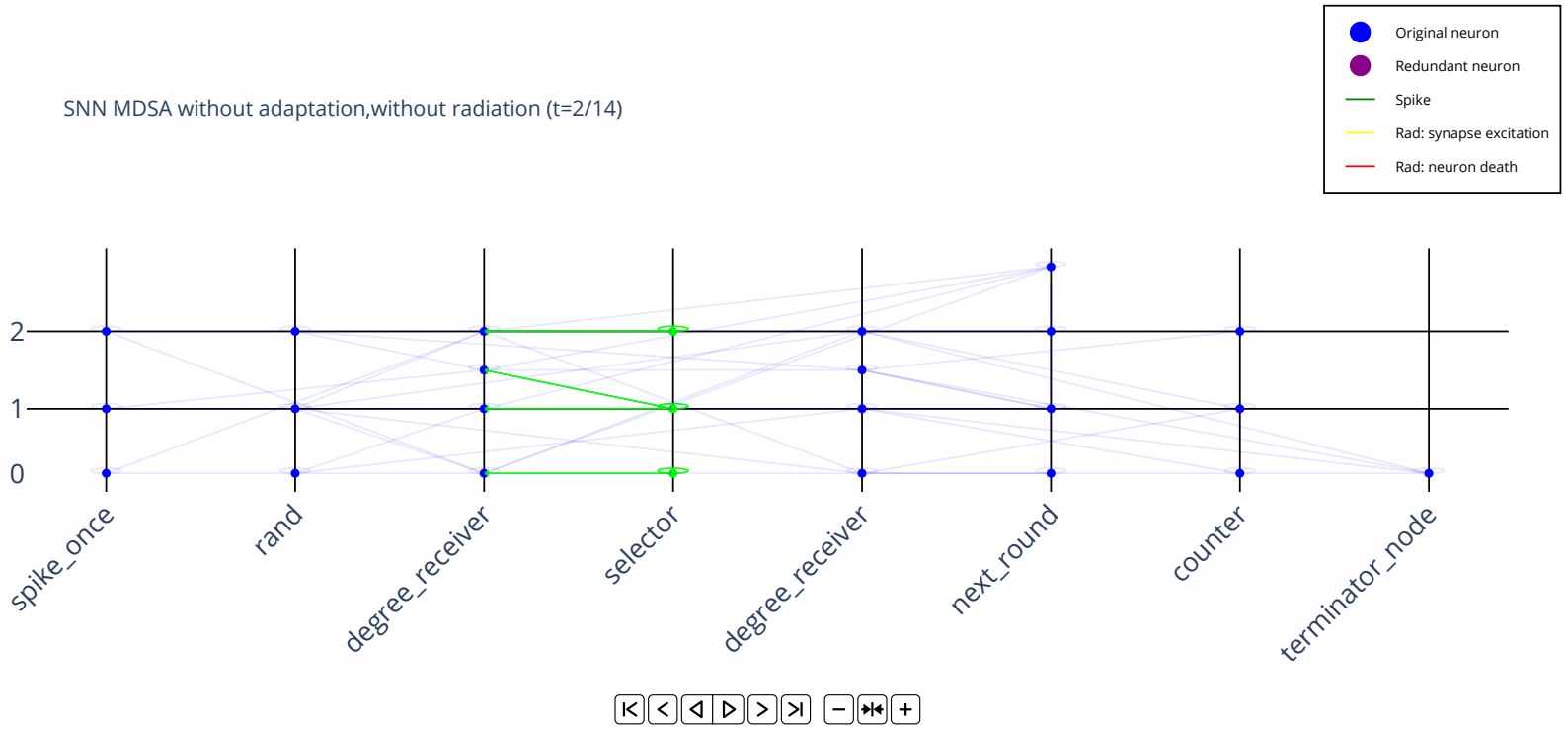


Figure 2.2: SNN implementation of the MDSA approximation for the input graph shown in fig. 2.1, with 2 approximation rounds ($m = 1$). The circles above the vertices are recurrent synapses.

2.3. Radiation Effect Simulation

In this research, the radiation effects that are observed and used in other studies, are applied by simulating those radiation effects on the neurons and synapses in the implemented SNN. Section 2.3.1 discusses how the simulated radiation effect settings for synaptic weight increases are derived. Section 2.3.2 discusses how the simulated radiation effect settings for neuron death are derived.

2.3.1. Synaptic Weight Increases

Neuromorphic hardware typically simulates synapses using memristors because they are able to change their resistance based on the current that passes through them [16, 26, 28, 48]. This makes them suitable to simulate synapses, which typically also change in weight based on how many signals pass through them [1]. Multiple studies have modelled increases in synaptic weight due to radiation [14, 15] , and others have experimentally observed decreases in memristor conductance due to radiation [18, 36]. These studies were performed

using floating point synapse weights in the range [0,1]. In this study, algorithm 1 is implemented using integer values to simplify debugging. This however is not considered an issue as the synapse values could also be mapped to the range [0,1] whilst keeping the functionality and output identical. Based on [8, 14, 19], the radiation effect timing is assumed to follow a Poisson distribution, and the radiation effect amplitude is expected to follow a Gaussian distribution, similar to [14]. In [15], an average synaptic weight increase is observed in the range of 56% to 100% as a consequence of simulated radiation effects on a memristor based neuromorphic circuit with 25 pre-synaptic neurons and 1 post-synaptic neuron. In [14], a synaptic weight increase in the range of 8 % to 36% is observed for a modified ideal memristor model [44] made using Verilog-A. These simulations used a Poisson distribution for the radiation effect interval periods, and a Gaussian distribution for the radiation effect magnitude. Additionally, a memristor resistance reduction has been found of -77% by [18] in an experiment where TiO_2 memristors were exposed to an alpha-particle radiation of $10^{14}\,\frac{alphas}{cm^2}$, where the particles had an energy level of $1[MeV]$.

In [15], the authors use a range of 10 to 30 radiation effect pulses to realise average synaptic weight increases. It is noted that their simulation has a fixed duration, whereas the SNN implementation of algorithm 1 has a longer radiation exposure period for larger values of m, and for larger input graphs.

To summarise, to simulate radiation induced synaptic excitation, typically, a Poisson distribution is used to determine when the radiation effects occur, and a Gaussian distribution is used to determine the magnitude of the synaptic weight excitation. This configuration is used for this experiment, and to determine which synapses are subjected to the synaptic weight excitation, a uniform distribution is used. Since the simulation duration of the (adapted) SNNs are a function of the number of iteration rounds m, and the input graph size, these parameters influence the simulated radiation exposure. To preserve this relation, the number of simulated radiation events is expressed as the probability of occurrence per synapse per timestep. This value is then used to compute the number of simulated radiation induced synaptic weight increases ($nswi_{rad}$) for each input graph. This $nswi_{rad}$ is then fed into the Poisson distribution to get the timesteps at which the synaptic weight increases occur. Hence, the following parameters are used to generate the synaptic weight increases:

- $seed$ [−], with $seed \in \mathbb{N}$, as the pseudo-random number that is used to get different, yet predictable radiation effects.
- $sim\_duration$ [$timesteps$] as the simulation duration of the unradiated SNN.
- $n_{synapses}$ [$synapses$] as the number of synapses in the SNN.
- p_{swi} [%] is the probability that a synapse experiences a synaptic weight increase at a timestep t. This probability is an explicitly defined experiment parameter.
- The frequency of radiation effect occurrence λ is derived from the probability of a synaptic weight increase occurring per synapse per timestep p_{swi} [%], and the number of synapses $n_{synapses}$ as:

$$\lambda = p_{swi} \cdot n_{synapses} \tag{2.1}$$

- $nswi_{rad}$ is the number of simulated radiation induced synaptic weight increases per simulation. This is computed using eq. (2.2).

$$nswi_{rad} = \lambda \cdot sim\_duration \tag{2.2}$$

- $sample\_size = n_{synapses} \cdot sim\_duration$ as the number of binary samples that are taken from this Poisson distribution.

These settings yield a varying number of synaptic weight increases per simulation, depending on the random seed, input graph size, and the number of approximation iterations m. The magnitude of these effects are sampled from a Gaussian distribution defined by mean μ and standard deviation σ. There are infinitely many combinations of λ, μ and σ that yield each value of average synaptic weight increase. Therefore, analogue to [14], the ratio of μ and σ is kept to:

$$\frac{\mu}{\sigma} = \frac{2}{1} \tag{2.3}$$

To limit the computational costs, the mean and standard deviation for the Gaussian distribution are set to: $\mu = 1$ and $\sigma = 0.5$. Each unique combination of $seed, \lambda, \mu$ and σ that is defined in this experiment, is used to generate a unique synaptic excitation pattern that is used in the simulations.

2.3.2. Neuron Death

The work by Cantley et. al [10] states that radiation that permeates beyond the hardening may lead to neuron death due to circuit failure in the SNN. That same study performs a simulation of neuron death percentages up to 25% for an SNN that implements an STDP learning rule. This research measures the SNN adaptation mechanisms robustness with simulated radiation effects in the form of neuron death probabilities of up to 25% per neuron per timestep. The neuron deaths are modelled as permanent, and their occurrence (at which time steps) is sampled from a Poisson distribution with the same variables as for the Poisson distribution sampling procedure outlined in section 2.3.1, except that the number of synapses $n_{synapses}$ is replaced with the number of neurons $n_{neurons}$. The neurons that are subject to neuron death are sampled from a uniform distribution for each of these time steps.

2.4. SNN Adaptation Mechanisms

After considering the following four methods of brain adaptation: redundancy, reorganisation, niche construction and timing of developmental trajectories, the redundancy is selected for application in the adaptation mechanisms. It is noted that higher vertebrate brains often have multiple neural pathways that can support certain behaviour [30]. Using multiple pathways is considered a viable strategy of implementing a brain inspired adaption mechanism in the generated SNN algorithm. The re-organisation is not used because there is no hierarchy within the selected SNN implementation of the minimum dominating set approximation. The niche construction is unused because the SNN implementation does not have the liberty to select its own input, nor influence its own environment. Timing of developmental trajectories are ignored because the algorithm does not learn. Since there are two different types of simulated radiation expected in the SNNs, two adaptation techniques are tested. Accordingly, the redundancy is implemented using the following approaches:

- Population coding - Represents stimuli using combined activities of multiple neurons [40].

- Sparse redundancy - Generates custom redundant neurons for each neuron (type) in the original SNN.

The population coding is implemented with a population size of p, which yields a redundancy of $p-1$. To preserve the original functionality of the algorithm, the thresholds of the populations are modified per neuron type in the original SNN. It is a general adaptation strategy that is expected to show robustness against simulated radiation in the form of synaptic excitation as well as against neuron death.

The sparse redundancy is created by adding r redundant neurons for each neuron. If the original neuron does not spike, the first redundant neuron takes over, if that neuron also died, the second redundant neuron takes over etc. This induces a time delay which increases the overall simulation duration with roughly a factor r for a redundancy level of r. Furthermore, there is a limitation on the level of redundancy that was found effectively for this SNN implementation as there were no neuron properties found that enabled a neuron to spike d steps after receiving an input signal at an arbitrary time t with $1 \leq d < r+1$ with $d \in \mathbb{N}$ and $r \in \mathbb{N}$. This form of adaptation is tailored to the simulated radiation effect of neuron death.

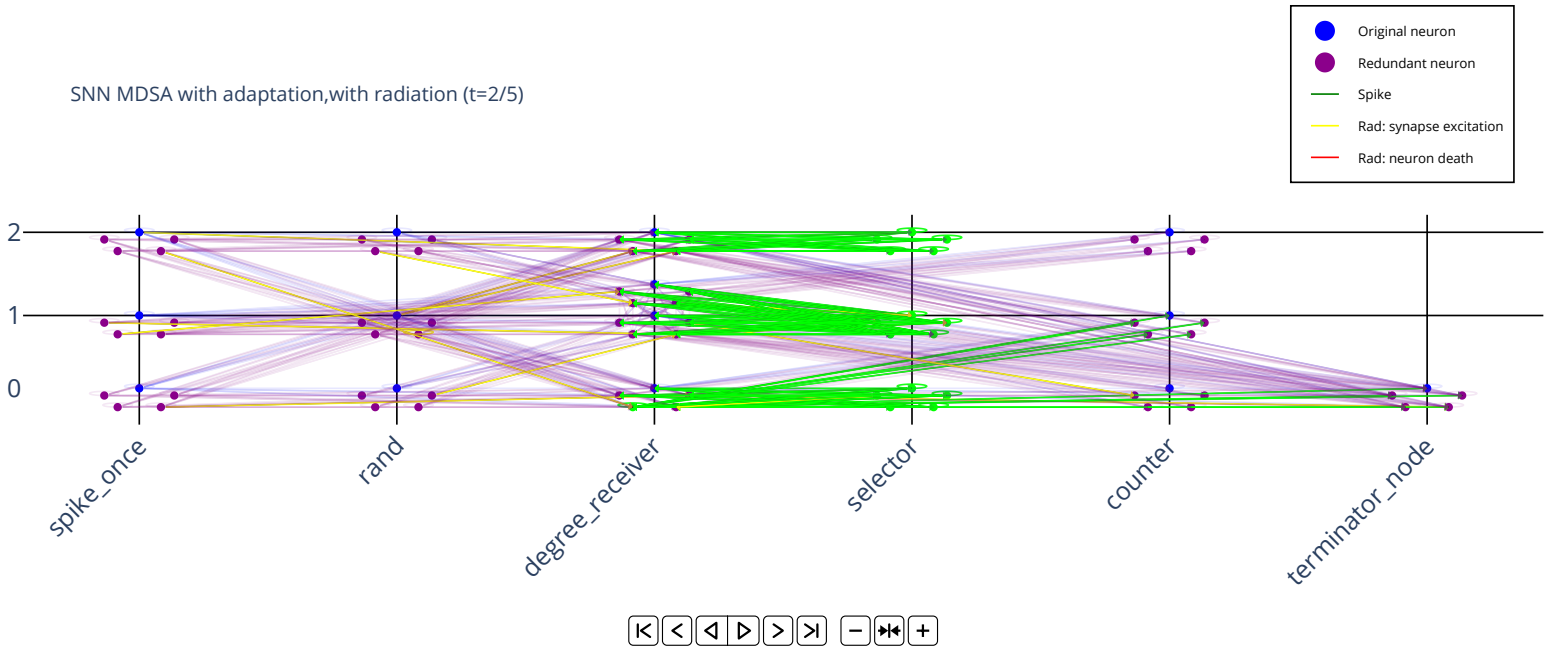


Figure 2.3: Population coding adaptation of the SNN MDSA implementation for the input graph shown in fig. 2.1, with simulated radiation in the form of synaptic excitations (in yellow) and a population size of $p = 5$. The synapse excitations are only shown when they first occur, and then ignored in the visualisation in the consecutive steps. However they are computed as non-transient, and hence are cumulative.

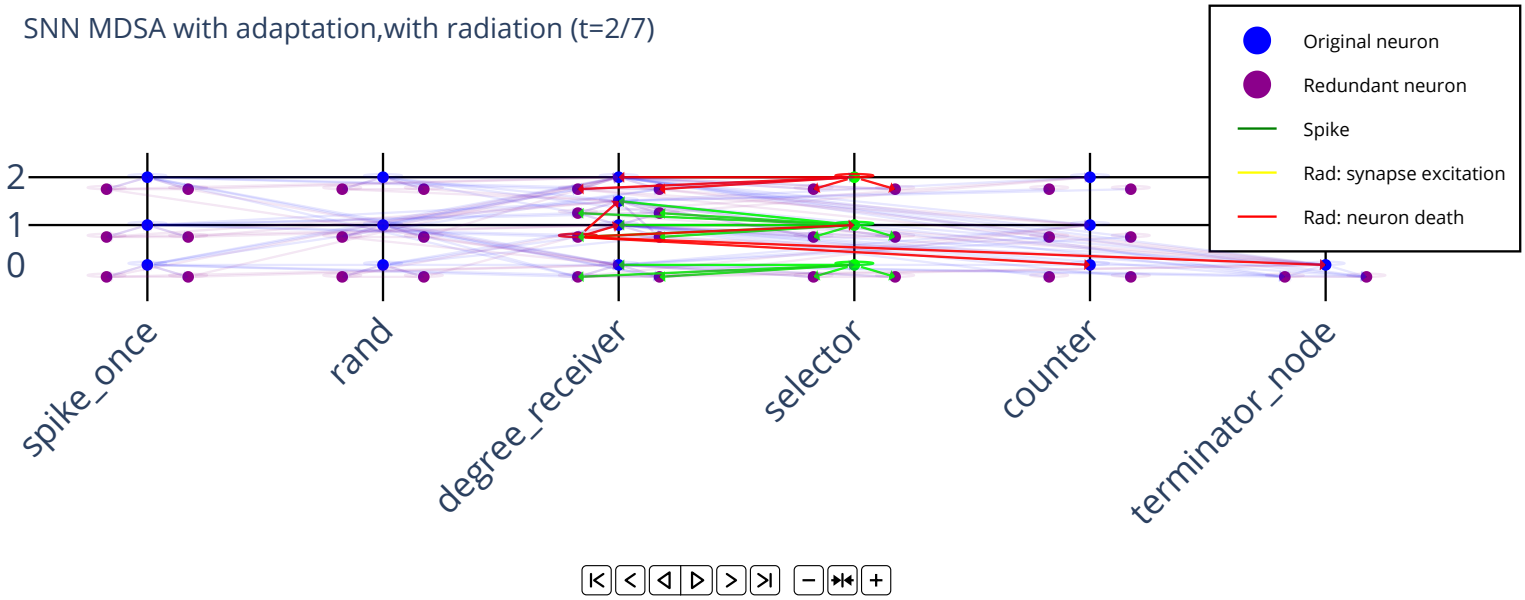


Figure 2.4: Sparse redundancy adaptation (consisting of $r = 2$ redundant neurons per original neuron) of the SNN MDSA implementation for the input graph shown in fig. 2.1 with simulated radiation in the form of neuron death. The neuron deaths, visualised as by the red synapses, are only visualised when they first occur, and then ignored in the visualisation in the consecutive steps. However they are computed as non-transient, and hence are cumulative.

2.5. Experiment Parameters

To test whether the adaptation increases the robustness of the SNN against the simulated radiation effects, their performance is measured against the unadapted SNN. To limit the computational costs, the m values

are set to $m = 1$. This value (along with $m = 2$ and $m = 5$), is also used in the benchmark by Alipour et al. [4]. Since algorithm 1 takes in planar, triangle free undirected graphs, input graphs of different sizes are generated that each have these all of these three graph properties. Per graph size, different graph shapes (non-isomorphic graphs) are used. To further reduce the computational costs of the experiment, the input graph sizes range from 5 to 20 [vertices]. Since algorithm 1 uses random weights to initialise the algorithm, a list of pseudo-random numbers in range 1 to 100 is used to generate the random weights per input graph, and to generate the radiation effects. The Radboud SNNSimulator is modified to store the SNN behaviour, and used to simulate the SNNs [46]. For the synaptic weight increases, the probability of occurrence per timestep are set to: $1e-05$, $5e-05$, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.2 and for the neuron deaths, the probabilities of occurrence per timestep are set to: 0.0001, 0.00025, 0.0005, 0.00075, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.25 respectively.

2.6. SNN Robustness Measurement Procedure

Whether an MDSA SNN output is considered correct or not, in terms of radiation robustness, is determined by comparing the output of the SNN algorithm (the current $u(t)$ of the counter neurons) with the count per vertice as computed by the Neumann implementation of algorithm 1. The robustness of the SNNs is accordingly measured as the fraction of correct SNN outputs out of the total number of outputs. Since this is a binary output, 1 for correct, 0 for incorrect, an Anova test is performed that determines whether the adaptation mechanisms are effective in increasing the number of correct outputs compared to no adaptation.

To visualise the performance of the adaptations, the correctness (1 or 0) of the SNNs are clustered per random seed into a score using:

$$score(pseudo\_random\_seed) = \frac{n_{correct}}{n_{instances}} \tag{2.4}$$

With:

- $pseudo\_random\_seed$ as the pseudo random seed for which the score is computed.
- $n_{instances}$ as the number of input graphs for which the score is computed.
- $n_{correct}$ as the number of input graphs for which a correct output was given by the SNN.

Each score hence encompasses a range in input graphs with different sizes, shapes, and contains, for each of those input graphs, one random number allocation, and one simulated radiation effect pattern. One score is computed per simulated radiation type, per simulated radiation intensity, per adaptation type, per pseudo-random seed. These scores are then used to create boxplots to indicate the robustness of the SNN adaptation mechanism, under the simulated radiation types and probabilities specified in section 2.5.

To indicate the statistical significance of the results, the p-values per adaptation mechanism setting, as computed using the Anova test, are plotted on the y-axis, with one line per radiation probability value used in this experiment.

2.7. Adaptation Cost Measurement Method

The adaptation costs in the SNN serve an essential role in understanding the efficiency and performance of these adaptation mechanisms. These costs can be divided into the three categories discussed in section 2.7.1 to section 2.7.3.

2.7.1. Neuronal Cost

The neuronal costs in the context of this experiment refer to the average number of neurons that are used in the SNNs. It provides an indication on the how much the adaptation mechanisms increase the number of neurons in the SNNs. This information can be used to select neuromorphic hardware platforms, or vice versa, select adaptation mechanisms that are suitable for a given neuromorphic hardware platform.

2.7.2. Synaptic Cost

Analogue to the neuronal costs, the synaptic costs refer to the average number of synapses that are used in the SNNs. It provides an indication on the how much the adaptation mechanisms increase the number of

synapses in the SNN. The synaptic costs can serve as a first indicator to assess whether the synaptic fan-in and fan-out of a neuromorphic hardware platform may not be large enough to implement the adaptation mechanisms. Furthermore, it can be used to determine whether the neuromorphic hardware platform is able to support the desired levels of adaptation, or alternatively, whether the adaptation mechanisms are suitable for the neuromorphic hardware platform.

2.7.3. Energy Cost

The energy costs are measured in terms of the average number of spikes that are generated by the SNNs whilst running the MDSA algorithm. This is an indication on the how much the adaptation mechanisms increase the number of spikes in the SNNs. The number of spikes provides an insight in the energy consumption of the neuromorphic hardware on which the SNNs are ran. Since energy efficiency is a critical concern in many real-world applications of SNNs, especially in space applications, measuring energy costs helps understanding the power requirements of SNNs with adaptation mechanisms. By comparing energy consumption between adapted and non-adapted SNNs running on the same input data, a deeper understanding is gained into how the adaptation effects the power efficiency of the SNNs.

Furthermore, this study focuses on the impact of adaptation on SNNs when subjected to the same input data or input graphs. This controlled experiment allows isolation of the effects of adaptation mechanisms and facilitates observing how they influence the network's structure and performance.

It is important to note that this analysis ignores certain factors that might influence network complexity. For instance, the potential changes in complexity resulting from excitatory or inhibitory simulated synaptic plasticity effects (SEE) are not taken into consideration. These effects, while important in biological neural networks, are excluded from this current investigation to isolate the specific contributions of adaptation mechanisms on the metrics mentioned above.

In summary, this research provides a comprehensive assessment of the adaptation costs in the MDSA SNNs, taking into account neuronal-, synaptic-, and energy costs. By conducting comparative analyses, light is shed on how adaptation mechanisms influence network performance and resource utilisation, which contributes to the broader understanding of spiking neural network dynamics.

Chapter 3

Software

This section discusses the software that is used in section 3.1. The dependencies are descripted in section 3.2, and the the verification of the software is discussed in section 3.3. A proposal to validate the the software is given in section 3.4.

3.1. Architecture

The software used in this research project consists of 6 modular components. Each of these is released as a Python pip package. These pip packages are listed below:

- **snncompare** - The main module that orchestrates running the experiment and generating the results.
- **simsnn** - A modified Radboud SNN simulator.
- **snnalgorithms** - A module that implements the SNN algorithm(s).
- **snnadaptation** - A module that is able to adapt the snn.
- **snnbackends** - A module that allows the simulation to be performed on both Von Neumann hardware and neurmorphic backends.
- **snnradiation** - A module that applies simulated radiation effects to the SNNs that are ran.

The architecture of the software is shown in fig. 3.1.

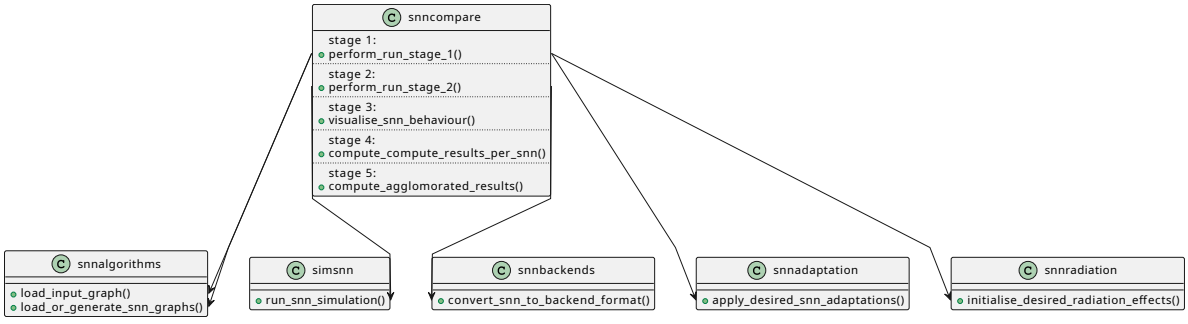


Figure 3.1: A high-level schematic overview of the experiment software.

In fig. 3.1, one can see that the experiment is compsed of 5 different stages. The first stage loads the SNN graphs that are to be simulated. The second stage then runs the simulation. The third stage allows one to visually the behaviour of the SNN, and the fourth stage computes the results of the simulation. The fifth stage then agglomorates the results of the individual simulations to provide an insight in the overall performance of the adaptation mechanisms against the simulated radiation.

3.2. Dependencies

The dependencies are included in the below Conda environment.yml:

```yaml
# This file is to automatically configure your environment. It allows you to
# run the code with a single command without having to install anything
# (extra).

# First run: conda env create --file environment.yml
# If you change this file, run: conda env update --file environment.yml

# Remove with:
# conda remove --name snncompare --all

# Instructions for this networkx-to-lava-nc repository only. First time usage
# On Ubuntu (this is needed for lava-nc):
# sudo apt upgrade
# sudo apt full-upgrade
# yes | sudo apt install gcc

# For code structure visualisation:
# yes | sudo apt install graphviz

# Conda configuration settings. (Specify which modules/packages are installed.)
name: snncompare
channels:
  - conda-forge
dependencies:
# Specify specific python version.
  - python=3.10

  # Install git without sudo if it is not installed.
  - git
  # Generate plots.
  - matplotlib
  # Pre-commit packages for code quality:
  # Run python tests.
  - pytest-cov
# Turns relative import paths into absolute import paths.
  - absolufy-imports
# Auto format Python code to make it flake8 compliant.
  - autoflake
# Scan Python code for security issues.
  - bandit
# Code formatting compliance.
  - black
# Correct code misspellings.
  - codespell
# Verify percentage of code that has at least 1 test.
  - coverage
# Show graph plots live in browser.
  - dash
# Auto formats the Python documentation written in the code.
  - docformatter
# Auto generate docstrings.
  - flake8
# Create gif of SNN behaviour.
```

```
  - imageio
# Auto sort the import statements.
  - isort
# Auto format Markdown files.
  - mdformat
# Auto check static typing.
  - mypy
  # Run graph software quickly.
  # Lava depends on networkx 2.8.7
  - networkx>=2.8.7
# used to visualise dash plots.
  - pandas
# Auto generate documentation.
  - pdoc3
# Another static type checker for python like mypy.
  - pyright
# Include GitHub pre-commit hook.
  - pre-commit
# Visualise the structure of the code based on a command.
  - pycallgraph2
# Automatically upgrades Python syntax to the new Python version syntax.
  - pyupgrade
# Auto generate docstrings.
  - pyment
# Auto check programming style aspects.
  - pylint
# Used to visualise dash plots
  - seaborn
# Enable publishing the pip package.
  - twine
# Ensure the python function arguments are verified at runtime.
  - typeguard
# Enable creating the pip package.
  - setuptools
  - wheel
# pip packages:
  - pip
  - pip:
    # Run pip install on .tar.gz file in GitHub repository (For lava-nc only).
    - https://github.com/lava-nc/lava/releases/download/v0.5.0/lava-nc-0.5.0.tar.gz
    # Show table with radiated neuron failure modes.
    - dash-bootstrap-components
    # Install visualisation tool.
    - dash-daq
    # Support parsing Json files.
    - jsons
    # Visualise dash plots.
    - kaleido
    # Seems to be an autoformatter like black, but installed using npm instead of pip.
    - prettier
    # Tag files using Python.
    - pytaggit
    # Allow for auto generation of type-hints during runtime.
    - pyannotate
    # Time function durations.
    - customshowme
```

```
110     # Create boxplots.
111     - simplt
```

3.3. Verification

The different pip packages are verified using unit tests and integration tests. The unit- and integration tests are written using the pytest framework. The SNN implementations of the MDSA algorithm are verified using the Von Neumann algorithm implementation. The output of the SNN is read from the neurons and this is compared to the output of the Von Neumann algorithm. This output is given in the form of selected neurons that approximate a Minimum Dominating Set on the input graph. This verification is performed for all SNN networks whose results are computed, as a default check. This verification method is applied to both the unadapted and adapted SNNs. Separate tests are written that verify that the neurons that are selected to undergo a simulated radiation effect in the form of neuron death, do not convey spike signals to their downstream neurons after the effect takes place. Similarly, for the simulated radiation effect in the form of synaptic excitation, the input received at the downstream neurons is tested to ensure they receive the simulated radiation effect when it takes place.

3.4. Validation

This research could be validated by selecting relevant neuromorphic hardware and submitting it to the appropriate radiation levels. The following process is proposed to determine which neuromorphic hardware is relevant, and what level of radiation exposure is appropriate. It is recommended to select a specific space mission objective. From this objective, any neuromorphic computational requirements could be derived, along with a flight trajectory. This flight trajectory can then be used to derive the expected radiation margins. The space hardware composition and/or expected spacecraft orientations can then be used to derive the expected radiation exposure to the neuromorphic hardware, and/or co-processors. Based on this radiation pattern, one can determine whether or not radiation shielding would be necessary. The simulated radiation effects from this study can then be used to determine obtain a first insight into whether the softwarematic adaptations are considered a viable option to (partially) mitigate the radiation effects and/or whether physical radiation hardening techniques are required. At this stage, it is essential that the bottleneck components, in terms of radiation sensitivity, are identified within the considered neuromorphic hardware. Iteratively, mitigation strategies for these bottlenecks may be designed until neuron death and/or synaptic excitation become a radiation failure mode bottleneck. In this case, the presented adaptation mechanisms can be validated through a radiation test campaign.

3.4.1. Radiation Testing

To minimise the costs and complexity of the radiation tests, a test-sequence is proposed that starts at small components, and works its way up to high-level SNN behaviour under radiation exposure. For the selected hardware, the components of the hardware that are most sensitive to radiation effects are identified. These components are then tested in isolation, to determine the radiation effects on those components. These components are then tested in combination, to determine the radiation effects on the components when they are used in combination to run the SNN. These tests can then be used to determine the required level of redundancy. The presented adaptation mechanisms can then be implemented with the required level of redundancy, and two final radiation tests can be conducted in which the respective adaptation mechanisms are tested.

In these radiation tests, the performance of multiple runs of the MDSA algorithm on pseudo-random input graph sizes varying from 10 to 50 vertices, are to be compared with- and without adaptation. Yielding 4 runs in total. Two runs are to be performed with radiation types that yield synaptic excitation, where the first run is performed without adaptation mechanism, and another run is performed with a population coding adaptation mechanism. If the run with the adaptation mechanism yields a higher score with p-values below 0.05 [-] in the accompanying Anova test, the presented adaptation mechanism is considered validated. Similarly, another two runs are proposed with a radiation type that yields neuron death in the neuromorphic hardware. One run is to be performed without adaptation and one with sparse redundancy. If the runs with adaptation yield a higher score with p-values below 0.05 [-] in the accompanying Anova test, the sparse redundancy is considered validated.

Chapter 4

Results

The experiment results of the SNN simulations on the MDSA algorithm are presented in this section. The experiment results are described in terms of the simulated radiation robustness and characterised with the adaptation costs for the respective SNN adaptation methods. The simulated radiation robustness results against synaptic excitation are presented in section 4.1.1, and the robustness results against the simulated radiation effects in the form of neuron death, are presented in section 4.1.2. For each of these results, the accompanying P-value of the Anova test is included to convey the statistical significance.
The adaptation costs are described in terms of space complexity by visualising how many neurons and synapses are required to implement the SNNs with- and without the respective adaptations. The energy complexity is visualised by showing the number of spikes required for the MDSA SNNs. The simulated radiation robustness of the SNNs with- and without adaptation, is given in section 4.1. The adaptation cost measurements in terms of space- and energy complexity, are given in section 4.2.

4.1. Simulated Radiation Robustness

The measured SNN robustness against simulated SEEs in the form of synapse excitation, is presented in fig. 4.1. The robustness scores of the SNN adaptation mechanisms are displayed on the vertical axis in that figure. This robustness score is computed by comparing the output of the respective SNN to the output of a traditional Von Neumann implementation of the MDSA algorithm by Alipour et al. [4]. If the SNN selects the same vertices at as minimum dominating set approximation as the unradiated, unadapted Von Neuman implementation, it receives a score of 1 [-]. If the SNN selects a different set of vertices, it receives a score of 0 [-]. To ensure the experiment is statistically significant, it is repeated on different graphs, each with 100 different random number initialisations. Each of these random number seeds yields a single dot in the boxplots of fig. 4.1 and fig. 4.2. Each dot is composed of multiple unique input graphs per input graph size, and the input graph sizes are in range 5 to 20 [vertices]. This results in a single list of zeros and ones per set of random numbers. These ones and zeros are then averaged into a score in range 0 to 1. This averaged score is then represented as the y-coordinate of the dot in the boxplots of section 4.1.1. These boxplot scores of the SNNs are computed for each of the redundancy levels of each of the adaptation types. These adaptation types and their respective redundancy levels are positioned on the horizontal axis. The redundancy level is varied from 0 to a maximum of 6 [redundant neurons].

4.1.1. Synaptic Excitation

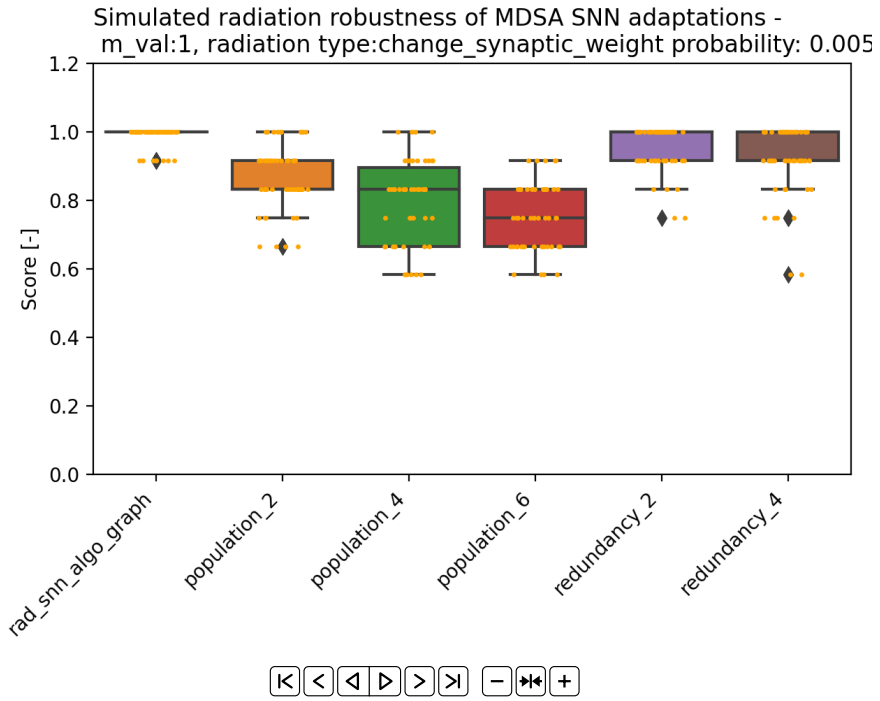
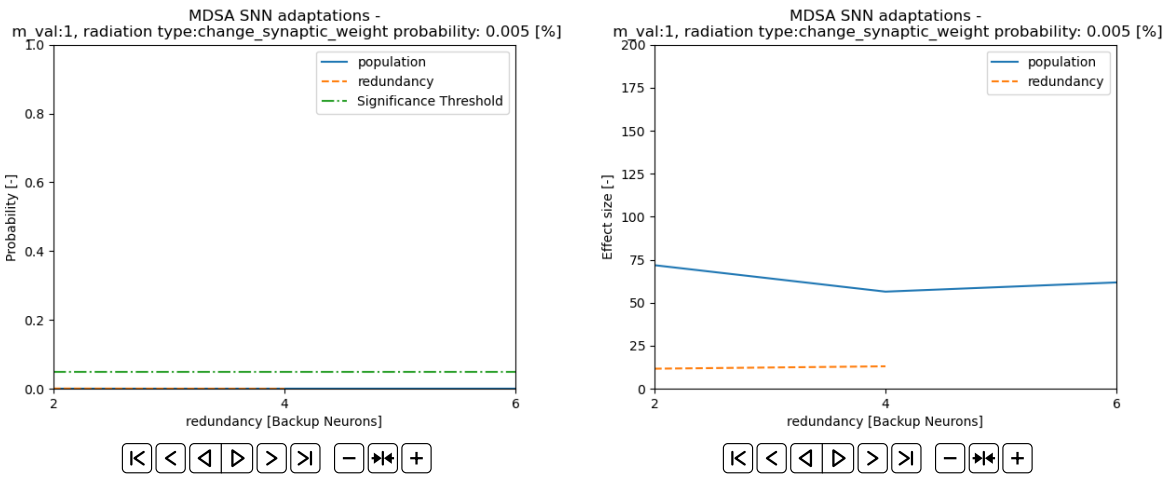


Figure 4.1: Simulated radiation robustness against simulated SEE in the form of synapse excitation, for SNNs with- and without adaptation.



(a) The p-value of the Anova test for the redundancy levels from 2 to 6 for the population coding and sparse redundancy adaptation mechanisms, as observed against simulated radiation induced synaptic weight increases. Values below 0.05 are considered statistically significant.

(b) The f-statistic of the Anova test for the redundancy levels from 2 to 6 for the population coding and sparse redundancy adaptation mechanisms, as observed against simulated radiation induced synaptic weight increases.

Figure 4.1 shows that for synaptic weight increase probabilities below 1 % chance per synapse per timestep, both population coding and sparse redundancy are contra productive as it decreases the SNN robustness. For probabilities of 1% or larger, the both population coding and sparse redundancy are an effective method to increase the simulated radiation robustness of the SNN implementation of the MDSA algorithm. In both adaptation mechanisms more redundancy yields a better performance. No significant performance difference is found between the two adaptation mechanisms, when comparing the highest levels of redundancy. For lower levels of redundancy the sparse redundancy shows the highest simulated radiation robustness.

4.1.2. Neuron Death

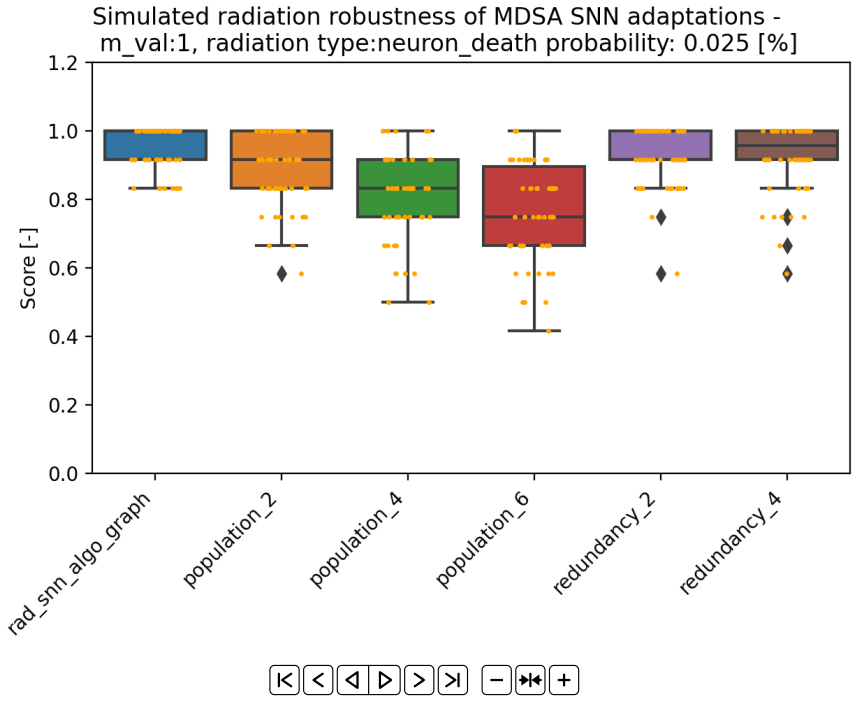
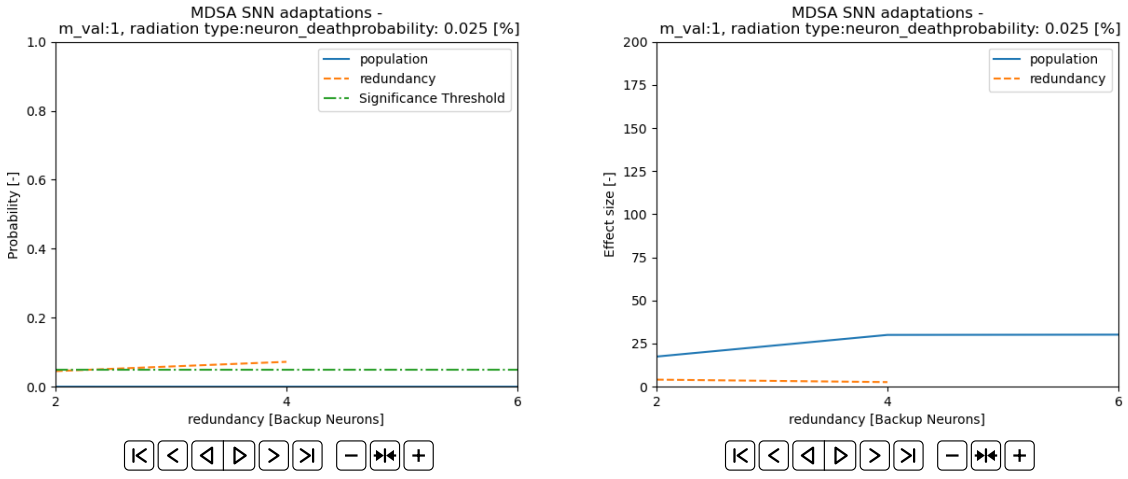


Figure 4.2: Simulated radiation robustness against simulated SEE in the form of neuron death, for SNNs the population coding and sparse redundancy adaptations. Each box contains 100 dots, corresponding to 100 unique random seeds in range 1 to 100, which are used to initialise the random weights in the SNNs, as well as the radiation patterns. Each dot represents the fraction of outputs out of 12 input graphs into adapted SNN that are identical to that of the (unradiated) algorithm specified in algorithm 1.



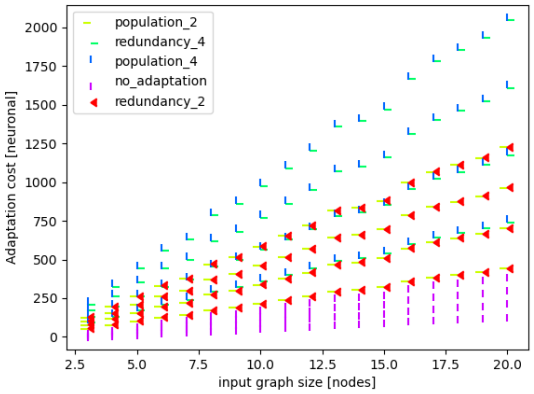
(a) The p-value of the Anova test for the redundancy levels from 2 to 6 for the population coding and sparse redundancy adaptation mechanisms, as observed against simulated radiation induced neuron death. Values below 0.05 are considered statistically significant.

(b) The f-statistic of the Anova test for the redundancy levels from 2 to 6 for the population coding and sparse redundancy adaptation mechanisms, as observed against simulated radiation induced neuron death.

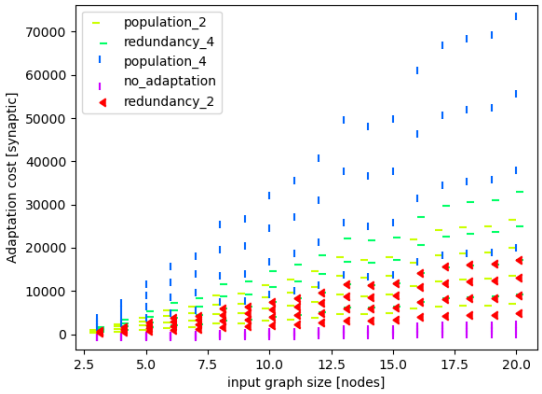
Figure 4.2 shows that the population coding is not effective against the simulated radiation induced neuron death. For neuron death probabilities below 0.5 % chance per neuron per timestep, sparse redundancy is contra productive as it decreases the SNN performance. For probabilities of 0.5% or larger, sparse redundancy is an effective method to increase the simulated radiation robustness of the SNN implementation of the MDSA algorithm.

4.2. Adaptation Costs

To characterise the adaptation mechanisms, the cost of implementing the redundancy of the adaptations is presented in this subsection. This information can assist trade-offs in selecting the SNN adaptation mechanisms for specific space mission applications. The energy costs of the SNNs are measured in terms of the number of spikes required to complete the computation of MDSA for the selected input graphs. It can function as an indication of the energy consumption of running the adapted MDSA SNN on actual neuromorphic hardware. The space complexity costs of the SNNs are measured in terms of the number of neurons and synapses required to implement the redundancy. This can be relevant in selecting the neuromorphic architecture, or maximum input graph size on which the MDSA algorithm is ran. The adaptation costs are measured for the population coding and sparse redundancy adaptation mechanisms. The adaptation costs are measured for the SNN with- and without adaptation. The adaptation costs are measured for the MDSA algorithm instances with $m \in [0,1]$. The neuronal-, synaptic- and energy complexity of the SNNs with- and without adaptation are visualised in fig. 4.3a, fig. 4.3b and fig. 4.4 respectively. These figures are used to show the adaptation costs in the adapted SNNs.



(a) The number of neurons for the SNNs for the algorithm instances with $m \in [0,1]$.



(b) The number of synapses for the SNNs for the algorithm instances with $m \in [0,1]$.

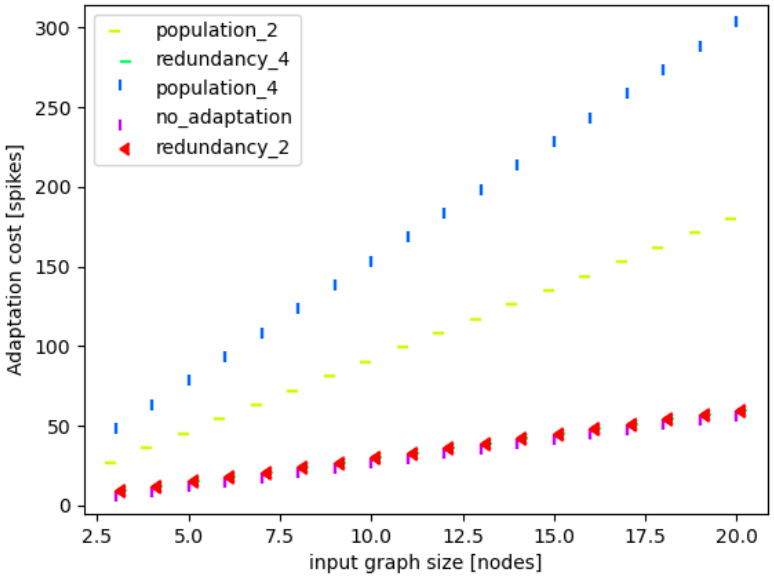


Figure 4.4: The number of spikes measured in the SNNs for the algorithm instances with $m \in [0,1]$. Note that the adaptation costs are zero for a sparse redundancy of 2 and 4 compared to the unadapted SNN implementation of the MDSA algorithm.

The adaptation costs of both the population coding and sparse redundancy, scale linearly with the redundancy level in terms of the number of neurons and quadratic in the number synapses. The energy costs, in terms of spikes, scale linearly with the redundancy level for the population coding. There are no adaptation costs in terms of spikes for the sparse-redundancy adaptation.

4.3. Qualitative Analysis

The SNN adaptation mechanisms on the MDSA algorithm can become effective for larger levels of radiation exposure. Not every adaptation mechanisms is effective against each type of simulated radiation effect, and if the adaptation mechanism is effective, a trend is observable where higher levels of redundancy yield increased simulated radiation robustness. The sparse redundancy is effective against both of the simulated radiation effect types, and it starts being effective at a lower levels of redundancy than the population coding adaptation. Furthermore, it has a lower energy cost and a lower neuronal cost. A drawback of the sparse redundancy implementation is that it has a limited supported maximum level of redundancy of 4 [redundant neurons], whereas the population coding adaptation does not have such an inherent redundancy level limitation. The population coding adaptation is only effective against the simulated radiation effect in the form of synaptic excitation, and more redundancy yields a higher degree of simulated radiation robustness for this adaptation mechanism.

Chapter 5

Discussion

In section 5.1 the results are discussed, and section 5.2 focuses on the methodology and SNN design.

5.1. Observed Simulated Radiation Robustness

The analysis of the SNN performance displayed in fig. 4.1 and fig. 4.2, revealed that both the population coding and sparse coding adaptation mechanisms perform worse than the unadapted SNN for low radiation levels. Investigating the underlying causes for this behaviour led to the identification of the following possible explanations of this phenomenon:

- The increased simulation duration of the adapted networks leads to higher radiation impact which yields more failures. This is considered unlikely, because that would also lead to poorer performance at higher radiation levels.
- The lack of redundancy in the next-round neurons lead to failure in these networks. This however, would not explain for the performance decrease in the population coding adaptation mechanism at low radiation levels, because the next-round neurons are implemented as a population as well for this adaptation mechanism, and do not lack redundancy for that adaptation mechanism.
- Unexpected interactions between the adapted neurons and original neurons yield network failures due to radiation.

A SNN failure mode visualisation tool has been built that shows the radiation induced deviations from the unradiated adapted networks. This tool lists the neurons whose spike patterns differ from those in the unradiated SNN and at which timesteps they differ. A neuron that spikes when its unradiated counterpart does not, is displayed with green text, and a neuron that does not spike when its unradiated counterpart does, is displayed with red text, along with the timestep of this deviation in spike behaviour. A complicating factor in the root cause analysis of the performance reduction at increased redundancy levels at low radiation levels is that it is challenging within this tool to determine whether any additional deviations downstream of the initial deviation is caused due to the initial radiation induced deviation, or induced by additional simulated radiation effects. No particular failure mode(s) have been identified that explain the reduced performance at increased redundancy levels for low radiation.

An additional tool, henceforth referred to as isolated failure mode inspector, could be built to register and separate each individual simulated radiation effect. This could then be combined with a tool, henceforth referred to as isolated performance measurement tool, that recomputes the performance measurements for the isolated single radiation effects. If the performance on the isolated radiation effects displays the same trend of reduced performance at increasing redundancy levels for low levels of radiation, the isolated failure mode inspector could be used to determine whether any SNN failure mode pertaining to a single neuron type can be identified that contributes to the reduced performance at increased redundancy levels for low radiation levels. If the isolated performance measurement tool does not display a reduced performance at increased redundancy levels for low radiation levels, the reduced performance at increased redundancy levels for low radiation levels is caused by the interaction between the initial radiation effect and the additional radiation effects. In that case, the isolated failure mode inspector tool does not add value to the root cause analysis of the performance decrease at low radiation levels. Instead the SNN failure mode visualisation tool

could be used to cluster the different neuron combinations for the various failure modes to see whether any particular failure-mode pattern is observable in the adapted network that is not present in the unadapted network.

Constructing the two additional tools for isolated performance measurements and isolated failure mode inspection is considered beyond the scope for this research.

In fig. 4.2, the performance of the SNN with a sparse redundancy of 4 [redundant neurons] seems to be increasing from a radiation exposure level that goes from 10 to 20 % probability of occurrence per neuron per timestep. This is shown by the horizontal centerline in the box shifting upwards. It is expected that as radiation levels increase, performance goes down, with and without adaptation. The increase in performance is attributed to an observed larger spread in performance for the highest radiation exposure, as indicated by the upper and lower T-bars on the top and bottom of the box. No particular outlier is detected that contributes towards this performance increase. It is assumed that this randomness induced variation allows for the unexpected increase in performance.

5.2. Experiment Method Limitations

Several limitations are defined within the research methodology outlined in chapter 2. These limitations are summarised as follows:

1. The network initialisation costs are not included in this research. They could be used to determine whether the SNN graph algorithm is more computationally efficient than running the algorithm 1 on Von Neumann hardware.

2. The simultaneous implementation of both adaptations is not investigated. Such an implementation may consist of parallel or hybrid forms. In a parallel composition, a solution has to be found to synchronise the time-delay between the population coding approach and the sparse redundancy implementation. Furthermore, the synaptic output weights of both the population neurons and sparse neurons need to be adjusted along with the post-synaptic neuron threshold to preserve the same overall SNN behaviour. Alternatively, in a hybrid composition, some elements of the sparse coding implementation may be transformed into populations themselves. Such implementations are considered outside the scope of this research.

3. Neuromorphic architectures may have different radiation failure bottlenecks that result in system failure before the simulated radiation effects occur [49]. For example, the memories of the Intel Loihi chip store the connectivity of the neurons that are mapped to its neuromorphic core [17]. If radiation corrupts these memories, the neuronal signals may not arrive, or arrive at the wrong location. The presented adaptation mechanisms do not provide protection against such high-level failure modes. The presented adaptation mechanisms are only able to protect against radiation induced failures that occur at the level of the individual neurons and synapses.

4. No source has been presented nor found that demonstrates an explicit causal chain on how radiation effects in a neuromorphic chip may lead to neuron death. While it seems intuitive to investigate the possibility of radiation induced neuron deaths in neuromorphic hardware, based on studies demonstrating radiation induced neuron death in organic tissue [29, 31, 37, 41], such failure modes may not occur in neuromorphic hardware if its architectures and/or components exhibit different radiation induced failure modes in space applications. All hardware/non-biological studies that were found on the concept of radiation induced neuron death rely on the explanation of that concept, given in [10]. However, no explicit radiation interaction mechanism have been found in that work, with which radiation on neuromorphic hardware components may prevent neurons from spiking anymore.

5. A significant increase in radiation robustness may be found if the redundancy in the SNN can be implemented in the form of a majority vote, that allows a small subset of the total population to still convey the relevant spike information. However, no such implementation has been found, and it is expected that it would require a redesign of the SNN and possibly a different information encoding strategy.

6. The implementation of the sparse redundancy does not permit arbitrary levels of redundancy, and induces a network simulation duration increase factor that is at most equal to the level of the redundancy. This limitation occurs due to the network design, which requires the redundant next-round neuron to spike once, r timesteps after receiving an input spike, at an arbitrary time t, for a redundancy level

of r. A tool was developed that performs grid searches to find neuron constellations that satisfy the behavioural properties needed in the MDSA SNN for greater levels of redundancy r. However, no LIF neurons wre found that satisfy those requirements for redundancy levels greater than $r = 4$.

7. The sparse redundancy still has critical bottleneck neuron types that are not as redundant as the rest of the SNN (e.g., the next round neuron). These bottlenecks are a consequence of the chosen redundancy paradigm and can act as a limiting factor in the simulated radiation robustness of the sparse redundancy adaptation. However, inspection with the SNN failure mode visualisation tool did not reveal any patterns that can substantiate the status of the next-round neurons as bottleneck in terms of simulated radiation robustness for the sparse redundancy adaptation of the MDSA SNN.

8. Since algorithm 1 is an approximation algorithm, with an approximation bound of 32 times larger than the minimum dominating set, the probabilistic nature of the radiation effects may be used to present a probabilistic approximation bound. One difficulty in specifying such a probabilistic approximation bound, is in the sensibility of the SNN output when exposed to the selected simulated radiation effects. In the chosen SNN implementation, the network is able to output non-dominating sets under the influence of simulated radiation effects. To establish a reliable probabilistic bound, the SNN could be re-designed to yield non-minimal dominating sets, instead of no dominating sets, under the simulated radiation effects. Alternatively, an additional probabilistic bound may be established that is based on the probability of the SNN outputting a non-dominating set, instead of a dominating set, under the simulated radiation effects.

Chapter 6

Conclusion

An experiment is performed that simulated radiation effects on spiking neural networks by randomly terminating neuron activity and through random synapse excitation. The SNN in this experiment consists of an implementation of the MDSA algorithm by Alipour et al. [4]. The MDSA algorithm is a graph algorithm that can be used to approximate the minimum dominating set of a graph. Running this MDSA algorithm on neuromorphic space hardware in a radiation prone environment may lead to invalid SNN outputs. To increase the SNN robustness against radiation effects, the inspiration for two adaptation mechanisms was found in the brain. In particular, a population coding and sparse redundancy adaptation mechanism are implemented, each in a separate copy of an SNN implementation of the MDSA algorithm by Alipour et al. [4]. The population coding adaptation mechanism is implemented by increasing the number of neurons in the SNN and by increasing the number of synapses per neuron. The sparse redundancy adaptation mechanism was implemented by adding backup neurons and synapses to the SNN. The backup neurons and synapses were tailored to take over/spike if both their original neuron and their preceding backup neurons fail to spike. The experiment is ran on the Radboud Spiking Neural Network Simulator. The results of the experiment are analysed and discussed.

Brain inspired adaptation mechanisms can be used to increase the simulated radiation robustness of the manually designed spiking neural network implementation of the minimum dominating set approximation algorithm by [4] if the simulated radiation effect probabilities are high enough. For low radiation probabilities, the adaptation strategies lower the simulated radiation robustness.

For simulated radiation induced synaptic excitation probabilities of 1% or larger, both population coding and sparse redundancy are an effective method to increase the simulated radiation robustness of the SNN implementation of the MDSA algorithm. In both adaptation mechanisms more redundancy yields a better performance at synaptic excitation probabilities of 1% or larger. No significant performance difference between the two adaptation mechanisms is found, when comparing the highest levels of redundancy. For lower levels of redundancy the sparse redundancy shows the highest simulated radiation robustness.

For simulated radiation induced neuron death, a difference between the two adaptation strategies is found. The population coding is not effective, whereas the sparse redundancy is effective at sufficiently high levels of simulated radiation exposure. For neuron death probabilities below 0.5 % chance per neuron per timestep, sparse redundancy is contra productive as it decreases the SNN performance. For probabilities of 0.5% or larger, sparse redundancy is an effective method to increase the simulated radiation robustness of the SNN implementation of the MDSA algorithm.

The adaptation costs of the population coding and sparse redundancy adaptation mechanisms are characterised in terms of additional neurons, synapses and spikes that the adaptations induce compared to the unadapted SNN implementation of the MDSA. The adaptation costs in terms of the number of additional neurons are linear in the redundancy levels, and quadratic with the redundancy levels in terms of the number of additional synapses. For the population coding, the energy costs are linear with the population size. Manually tailoring the sparse redundancy to the unadapted SNN facilitates an increased radiation robustness against simulated radiation induced neuron death at no energy costs.

This is significant because it shows that it is possible to improve the radiation robustness of spiking neural networks (SNNs) without increasing their energy consumption. This is important because SNNs are a promising, energy efficient alternative to traditional artificial neural networks (ANNs) for a variety of space applications, including neuromorphic computing and brain-inspired computing. This result can stimulate

the development of more robust SNNs for a variety of neuromorphic space applications.

The presented results highlight the significance of considering specific adaptation mechanisms in SNN implementations of graph algorithms to improve the radiation robustness of neuromorphic space hardware. By incorporating brain-inspired principles, researchers can improve the robustness of neuromorphic systems, facilitating their effective operation in radiation-prone environments. Further exploration of different adaptation mechanisms and their impact on SNN performance under varying radiation scenarios holds promising potential for advancing space-related applications.

Chapter 7

Recommendations

A more detailed space mission plan, that includes a selection of the flight-plan, on-board Von Neumann hardware, and/or neuromorphic hardware, as well as an exact requirement on the computational needs of the space hardware, allows for deciding for which computations the neuromorphic hardware will be used, and which adaptation strategies may be applied. The flight-plan can be used to determine the radiation profile that is experienced throughout the mission, the neuromorphic hardware selection can then be used to determine the expected radiation effects, and the mission objective can be used to derive the computational needs of the space hardware. Based on the radiation effects and computational needs, an assessment can be made whether a SNN implementation is used to perform computations or whether Von Neumann hardware is used. To navigate the design space between energy efficient and radiation robust SNN implementations, the following recommendations are made for rule-based SNN algorithms, based on the notion that manually designing a radiation robust, energy efficient SNN is highly-labour intensive:

1. Where possible, one would like to select high-level implementation approaches that are inherently robust. For example, in the case of the minimum dominating set approximation, one would ideally design an SNN such that the expected radiation failure modes always lead to dominating sets, albeit larger, instead of non-sensical output. Another field where such design options are prevalent are routing strategies.

2. When selecting a neural coding paradigm, one might also want to consider a shift to alternatives of sparse- and population coding. For example, rate-coding approaches where one could modulate the neuron firing frequency to lower the impact of radiation induced noise, may be effective. Work performed on robust neural coding schemes for SNNs [24, 33] may serve as inspiration for adaptive robust graph SNN approximation algorithms.

3. One of the main challenges experienced in the design of the adaptation, is that adding more redundant/backup neurons does not necessarily increase the radiation robustness, as those additional neurons are subject to the same probability of failure as the original neuron they back up. Hence, one can try finding adaptation mechanisms that fail only if a majority/all of the neurons fail. An example of this is a majority voting implementation, which is what is traditionally used has a radiation hardening technique of Von Neumann space hardware. Since the output of such a majority voting cluster also needs redundancy, this challenge can propagate throughout the entire SNN. Since this is difficult to oversee manually, an evolutionary approach is expected to be an effective SNN design exploration approach for explicit/rule-based algorithms. However, one can also follow a more guided SNN design strategies aimed at increasing the radiation robustness of SNN implementations. For example, one could first explore whether a high-level requirement specification of the network sub-components is fruitful in combination with a constraint satisfaction approach, as this may help navigating the inter-dependencies between (redundant) neurons. This approach can lead to large search spaces of various neuron constellations which do not necessarily have a satisfactory solution. To overcome this, one could combine the constraint-satisfaction approach with proof-assistant languages such as Coq. This combination may be used to prove that certain neuron constellations, which may follow from the requirements, are non-existent. Such proofs could then be used to reduce the search space.

4. Instead of manually constructing a SNN for each MDSA input put graph, one could alternatively build

an MDSA SNN that takes the encoded input graph as a stream of bits or synaptic inputs instead. This would allow for a more general SNN implementation, that can be used for any MDSA input graph. However, this approach may be more difficult to implement, and may require more resources, as the SNN would need to be able to handle any input graph, instead of a specific one. This would also stimulate the possibility of using alternative adaptation strategies, such as frequency modulation in rate-coded or temporal-coded SNNs.

5. Due to the complexity of embedding the adaptation mechanisms into the tailored SNN constellations, it may be more effective to first embed the adaptation mechanisms in small, modular building blocks composed of SNN networks that can be compounded to form larger SNN networks. This would allow for a more modular approach to the adaptation mechanisms, and may allow for a more effective exploration of the design space. This work by Aimone et al. could be a good starting point as it presents the Fugu framework for modular building blocks to create neural algorithms [2].

6. For space applications of neuromorphic graph algorithms, it is recommended to take robustness into account in the original algorithm selection and to consider its robustness in combination with the accompanying SNN implementation.

Bibliography

[1] Shyam Prasad Adhikari, Changju Yang, Hyongsuk Kim, and Leon O Chua. Memristor bridge synapse-based neural network and its learning. IEEE Transactions on neural networks and learning systems, 23 (9):1426–1435, 2012.

[2] James B. Aimone, William Severa, and Craig M. Vineyard. Composing neural algorithms with fugu. In Proceedings of the International Conference on Neuromorphic Systems, ICONS '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376808. doi: 10.1145/3354265.3354268. URL https://doi.org/10.1145/3354265.3354268.

[3] James B Aimone, Yang Ho, Ojas Parekh, Cynthia A Phillips, Ali Pinar, William Severa, and Yipu Wang. Provable advantages for graph algorithms in spiking neural networks. In Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures, pages 35–47, 2021.

[4] Sharareh Alipour, Ehsan Futuhi, and Shayan Karimi. On distributed algorithms for minimum dominating set problem, from theory to application. arXiv preprint arXiv:2012.04883, 2020.

[5] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7243–7252, 2017.

[6] Gennadi Bersuker, Maribeth Mason, and Karen L Jones. Neuromorphic computing: The potential for high-performance processing in space. pages 1–12.

[7] Ronald V Book. Michael R. Garey and David S. Johnson, computers and intractability: a guide to the theory of np-completeness. 1980.

[8] Brendan Bridgford, Carl Carmichael, and Chen Wei Tseng. Single-event upset mitigation selection guide. Xilinx Application Note, XAPP987 (v1. 0), page 69, 2008.

[9] Cindy Calderón-Arce and Rebeca Solis-Ortega. Swarm robotics and rapidly exploring random graph algorithms applied to environment exploration and path planning. International Journal of Advanced Computer Science and Applications, 10(5), 2019.

[10] Kurtis D Cantley. Impact of radiation on pattern recognition in memristor-based neuromorphic circuits. Technical report, Boise State University, 2021.

[11] Chun-Wei Jacky Chang, Hsuan-Ming Ryan Huang, Yuwen Lin, and Charles H.-P. Wen. SERL: Soft error resilient latch design. In 2016 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), pages 1–4. IEEE. ISBN 978-1-4673-9498-7. doi: 10.1109/VLSI-DAT.2016.7482555. URL http://ieeexplore.ieee.org/document/7482555/.

[12] Herming Chiueh, Chia-Hsiang Yang, Charles H-P Wen, Chao-Guang Yang, Po-Hao Chien, Ching-Yang Hung, Yu-Jui Chen, Yao-Pin Wang, Chin-Fong Chiu, and Jer Lin. Radiation-harden RISC processor for micro-satellites in standard CMOS. In 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), pages 1–2. IEEE. doi: 10.1109/VLSI-DAT49148.2020.9196348.

[13] Federico Corradi, Hongzhi You, Massimiliano Giulioni, and Giacomo Indiveri. Decision making and perceptual bistability in spike-based neuromorphic vlsi systems. In 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pages 2708–2711. IEEE, 2015.

[14] Sumedha Gandharava Dahl, Robert Ivans, and Kurtis D. Cantley. Modeling memristor radiation interaction events and the effect on neuromorphic learning circuits. In Proceedings of the International Conference on Neuromorphic Systems. ACM, July 2018. doi: 10.1145/3229884.3229885. URL https://doi.org/10.1145/3229884.3229885.

[15] Sumedha Gandharava Dahl, Rober C. Ivans, and Kurtis D. Cantley. Radiation effect on learning behavior in memristor-based neuromorphic circuit. In 2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, August 2019. doi: 10.1109/mwscas.2019.8885288. URL https://doi.org/10.1109/mwscas.2019.8885288.

[16] Sumedha Gandharava Dahl, Rober C Ivans, and Kurtis D Cantley. Radiation effect on learning behavior in memristor-based neuromorphic circuit. In 2019 IEEE 62nd international midwest symposium on circuits and systems (MWSCAS), pages 53–56. IEEE, 2019.

[17] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. Ieee Micro, 38(1):82–99, 2018.

[18] Erica Deionno, Mark D. Looper, Jon V. Osborn, Hugh J. Barnaby, and William M. Tong. Radiation effects studies on thin film tio 2 memristor devices. In 2013 IEEE Aerospace Conference, pages 1–8. IEEE, 2013.

[19] Paul E Dodd, Marty R Shaneyfelt, Richard S Flores, James R Schwank, Thomas A Hill, Dale McMorrow, Gyorgy Vizkelethy, Scot E Swanson, and Scott M Dalton. Single-event upsets and distributions in radiation-hardened cmos flip-flop logic chains. IEEE Transactions on Nuclear Science, 58(6):2695–2701, 2011.

[20] Gianluca Furano, Gabriele Meoni, Aubrey Dunne, David Moloney, Veronique Ferlet-Cavrois, Antonis Tavoularis, Jonathan Byrne, Léonie Buckley, Mihalis Psarakis, Kay-Obbe Voss, et al. Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities. IEEE Aerospace and Electronic Systems Magazine, 35(12):44–56, 2020.

[21] Sumedha Gandharava Dahl, Robert C Ivans, and Kurtis D Cantley. Effects of memristive synapse radiation interactions on learning in spiking neural networks. SN Applied Sciences, 3:1–16, 2021.

[22] Li Gao, Duanfeng Chu, Yongxing Cao, Liping Lu, and Chaozhong Wu. Multi-lane convoy control for autonomous vehicles based on distributed graph and potential field. In 2019 ieee intelligent transportation systems conference (itsc), pages 2463–2469. IEEE, 2019.

[23] Juan Sebastián Gonzalez-Rojas, Juan Martinez-Piazuelo, and Nicanor Quijano. Distributed assignment with energy constraints of a uav fleet for resource distribution. In 2021 IEEE 5th Colombian Conference on Automatic Control (CCAC), pages 68–73. IEEE, 2021.

[24] Wenzhe Guo, Mohammed E Fouda, Ahmed M Eltawil, and Khaled Nabil Salama. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. Frontiers in Neuroscience, 15:638474, 2021.

[25] Daniel Gutierrez-Galan, Juan P Dominguez-Morales, Fernando Perez-Peña, Angel Jimenez-Fernandez, and Alejandro Linares-Barranco. NeuroPod: a real-time neuromorphic spiking CPG applied to robotics. 381:10–19. Publisher: Elsevier.

[26] Qinghui Hong, Liang Zhao, and Xiaoping Wang. Novel circuit designs of memristor synapse and neuron. Neurocomputing, 330:11–16, 2019.

[27] Intel. Lava framework for neuromorphic computing.

[28] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale memristor device as synapse in neuromorphic systems. Nano letters, 10(4):1297–1301, 2010.

[29] Mark D Johnson, Hong Xiang, Susan London, Yoshito Kinoshita, Michael Knudson, Marc Mayberg, Stanley J Korsmeyer, and Richard S Morrison. Evidence for involvement of bax and p53, but not caspases, in radiation-induced cell death of cultured postnatal hippocampal neurons. Journal of neuroscience research, 54(6):721–733, 1998.

[30] Mark H Johnson, Emily JH Jones, and Teodora Gliga. Brain adaptation and alternative developmental trajectories. Development and psychopathology, 27(2):425–442, 2015.

[31] Joaquın Jordan, Marıa F Galindo, Jochen HM Prehn, Ralph R Weichselbaum, Michael Beckett, Ghanashyam D Ghadge, Raymond P Roos, Jeffrey M Leiden, and Richard J Miller. p53 expression induces apoptosis in hippocampal pyramidal neuron cultures. Journal of Neuroscience, 17(4):1397–1405, 1997.

[32] Richard M Karp. Reducibility among combinatorial problems. Springer, 2010.

[33] Youngeun Kim, Hyoungseob Park, Abhishek Moitra, Abhiroop Bhattacharjee, Yeshwanth Venkatesha, and Priyadarshini Panda. Rate coding or direct coding: Which one is better for accurate, robust, and energy-efficient spiking neural networks? In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 71–75. IEEE, 2022.

[34] Johan Kwisthout and Nils Donselaar. On the computational power and complexity of spiking neural networks. In Proceedings of the 2020 Annual Neuro-Inspired Computational Elements Workshop, pages 1–7, 2020.

[35] Dongchen Liang and Giacomo Indiveri. A neuromorphic computational primitive for robust context-dependent decision making and context-dependent stochastic computation. IEEE Transactions on Circuits and Systems II: Express Briefs, 66(5):843–847, 2019.

[36] Matthew J. Marinella, Scott M. Dalton, Patrick R. Mickel, Paul. E. Dodd Dodd, Marty R. Shaneyfelt, Edward Bielejec, Gyorgy Vizkelethy, and Paul G. Kotula. Initial assessment of the effects of radiation on the electrical characteristics of tao_x memristive memories. IEEE Transactions on Nuclear Science, 59(6): 2987–2994, 2012. doi: 10.1109/TNS.2012.2224377.

[37] FD Miller, CD Pozniak, and GS Walsh. Neuronal life and death: an essential role for the p53 family. Cell Death & Differentiation, 7(10):880–888, 2000.

[38] Hesham Mostafa, Lorenz K Müller, and Giacomo Indiveri. An event-based architecture for solving constraint satisfaction problems. Nature communications, 6(1):8941, 2015.

[39] Issa AD Nesnas, Lorraine M Fesq, and Richard A Volpe. Autonomy for space robots: Past, present, and future. Current Robotics Reports, 2(3):251–263, 2021.

[40] Stefano Panzeri, Jakob H Macke, Joachim Gross, and Christoph Kayser. Neural population coding: combining insights from microscopic and mass signals. Trends in cognitive sciences, 19(3):162–172, 2015.

[41] David S Park, Erick J Morris, Rod Bremner, Elizabeth Keramaris, Jaya Padmanabhan, Michele Rosenbaum, Michael L Shelanski, Herbert M Geller, and Lloyd A Greene. Involvement of retinoblastoma family members and e2f/dp complexes in the death of neurons evoked by dna damage. Journal of Neuroscience, 20(9):3104–3114, 2000.

[42] Jing Qin, Xiang Mao, and Janise McNair. Weight based dominating set clustering algorithm for small satellite networks. In 2012 IEEE International Conference on Communications (ICC), pages 3195–3199. IEEE, 2012.

[43] Ian H Robertson and Jaap MJ Murre. Rehabilitation of brain damage: brain plasticity and principles of guided recovery. Psychological bulletin, 125(5):544, 1999.

[44] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The missing memristor found. nature, 453(7191):80–83, 2008.

[45] J Tani, G Ruvkun, MT Zuber, and CE Carr. On neuromorphic architectures for efficient, robust, and adaptable autonomy in life detection and other deep space missions. 1989:8080.

[46] Radboud University. Radboud SNN simulator. `https://gitlab.socsci.ru.nl/Akke.Toeter/simsnn`. URL `https://gitlab.socsci.ru.nl/Akke.Toeter/simsnn`.

[47] Iris Van Rooij, Mark Blokpoel, Johan Kwisthout, and Todd Wareham. Cognition and intractability: A guide to classical and parameterized complexity analysis. Cambridge University Press, 2019.

[48] Jingjuan Wang, Deliang Ren, Zichang Zhang, Hongwen Xiang, Jianhui Zhao, Zhenyu Zhou, Xiaoyan Li, Hong Wang, Lei Zhang, Mengliu Zhao, Yuxiao Fang, Chao Lu, Chun Zhao, Ce Zhou Zhao, and Xiaobing Yan. A radiation-hardening ta/ta $_2$ o $_{5-\ x}$ /al $_2$ o $_3$ /InGaZnO $_4$ memristor for harsh electronics. 113(12): 122907. ISSN 0003-6951, 1077-3118. doi: 10.1063/1.5045649. URL http://aip.scitation.org/doi/10.1063/1.5045649. Number: 12.

[49] Zhilu Ye, Rui Liu, Jennifer L Taggart, Hugh J Barnaby, and Shimeng Yu. Evaluation of radiation effects in rram-based neuromorphic computing system for inference. IEEE Transactions on Nuclear Science, 66 (1):97–103, 2018.

Glossary

clustering algorithm Groups data points based on similarities, facilitating data analysis, pattern recognition, and organization. 1, 31

distributed computation Computation performed by indepented processing units, typically connected through a network. 31

graph algorithm Computational procedures designed to analyze and/or manipulate data represented as interconnected vertices and edges. ix, 1, 24, 27, 28, 31

neuromorphic architecture Computational platform whose components mimic the brain, typically using neurons and synapses. ix, 31

parallel computation Computational performed simultaneously, typically on the same device by multiple processing units. 1, 31

population coding A neural information processing scheme where the activity of a group of neurons encodes specific information. ix, 8, 16, 18–21, 27, 31

resilience Speed with which a system can recover from a failure. 31

robustness The amount of damage a system can take before failure occurs. ix, 1, 8–10, 17–19, 24, 27, 28, 30, 31

sparse redundancy Defined within the scope of this research as a SNN robustness strategy where each backup neuron is tailored to the original neuron it backs. ix, 8, 16, 18–20, 24, 25, 27, 31

spiking neural network A type of artificial neural network that models the behavior of biological neurons by using discrete, asynchronous spikes. ix, 31

synaptic weight The strength of a connection between two neurons in a neural network, influencing signal transmission. 31

Von Neumann hardware Traditional computer architecture, that typically uses sequential instructions to perform computations on memory that is stored separately from the processing unit. 1, 13, 24, 29, 31