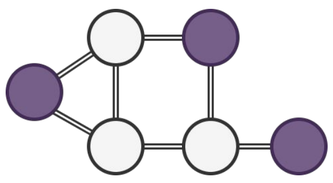
Implementation of a Distributed Minimum Dominating Set Approximation Algorithm in a Spiking Neural Network

Victoria Bosch, Arne Diehl, Daphne Smits, Akke Toeter, Johan Kwisthout

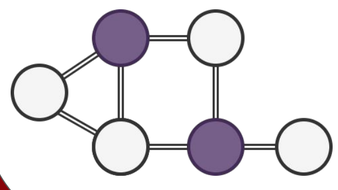
Spiking neural networks can be used to model **distributed algorithms**. **This work presents a translation of a distributed approximation algorithm for the minimum dominating set problem,** as described by Kuhn and Wattenhofer [1], to a spiking neural network. This translation shows that neuromorphic architectures can be used to implement distributed algorithms. Subcomponents of this implementation such as: "*the calculation of the minimum or maximum of two numbers*" and "*computation of the degrees of a vertex*", can be repurposed as foundational building blocks for other (graph) algorithms. **This work illustrates how leveraging neural properties for the translation of traditional algorithms contributes to a growing body of knowledge on neuromorphic applications for scientific computing using SNN simulation software and neuromorphic hardware.**

Minimum Dominating Set
Input: Undirected graph $G = (V, E)$.
Output: Subset D in which $D \subseteq V$ if $v \in V$ is in D or adjacent to D, and no subset of D is a dominating set of G.

Dominating Set



Minimum Dominating Set



Algorithm 1: Kuhn-Wattenhofer - $LP_{MDS} \rightarrow IP_{MDS}$

input : feasible solution $\underline{x}^{\alpha}$ for LP_{MDS}
output: IP_{MDS}-solution $\underline{x}_{DS}$ (dom. set)

calculate $\delta_i^{(2)}$ // Each step is computed for all vertices $v_i \in V$ simultaneously.
$p_i := min\{1, x_i^{\alpha} \cdot ln(\delta_i^{(2)} + 1)\}$
$x_{DS,i} := \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{otherwise} \end{cases}$
send $x_{DS,i}$ to all neighbours
if $x_{DS,j} = 0$ *for all* $j \in N_i$ **then**
  $x_{DS,i} := 1$
end

Algorithm 2: Kuhn-Wattenhofer - LP_{MDS} approximation

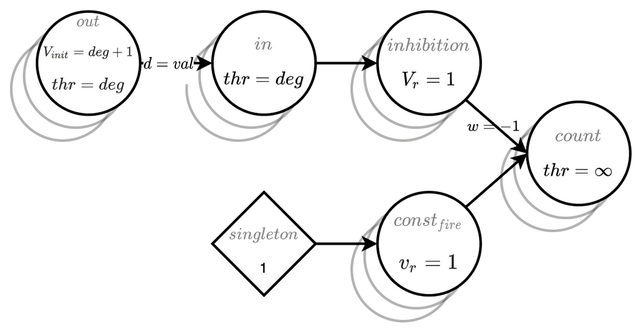
$x_i := 0$;
$\tilde{\delta}(v_i) := \delta_i + 1$
for $l := k\text{-}1$ *to* 0 *by* $\text{-}1$ **do**
  for $m := k\text{-}1$ *to* 0 *by* $\text{-}1$ **do**
    if $\tilde{\delta}(v_i) \geq (\Delta + 1)^{l/k}$ **then**
      $x_i := max\{x_i, \frac{1}{(\Delta+1)^{m/k}}\}$
    end
    Send $colour_i$ to all neighbours
    $\tilde{\delta}(v_i) := |\{j \in N_i \mid colour_j = \text{'white'}\}|$
    Send x_i to all neighbours
    if $\Sigma_{j \in N_i} x_j \geq 1$ **then**
      $colour_i :=$ 'gray'
    end
  end
end

Methods

- Radboud SNN Simulator (https://gitlab.socsci.ru.nl/snnsimulator/simsnn)
- Leaky-integrate-and-fire neurons
- Programming SNNs [2]:
  - View every neuron as a computational unit.
  - Every subnetwork implements a single function.
  - Implementation of algorithm is run in parallel for each vertex of the input graph.
  - CPU connects subnetworks together.
  - Sub-networks' connect together to compute final result.
  - Result is read off from LIF-neuron's state after computation.
- Manually programming and designing the network may be unconventional, yet this approach enables **increased control in tailoring of SNNs for various (graph) problems**. For example, Aimone et al. present a conversion method for the class of dynamic programs [3].

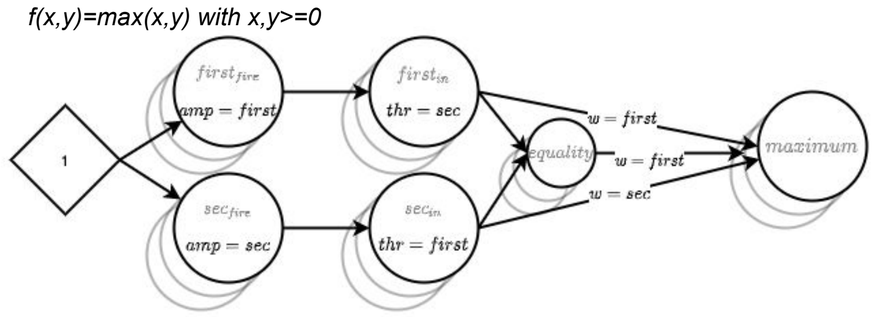
Examples of modular SNNs:

Maximum degree of graph

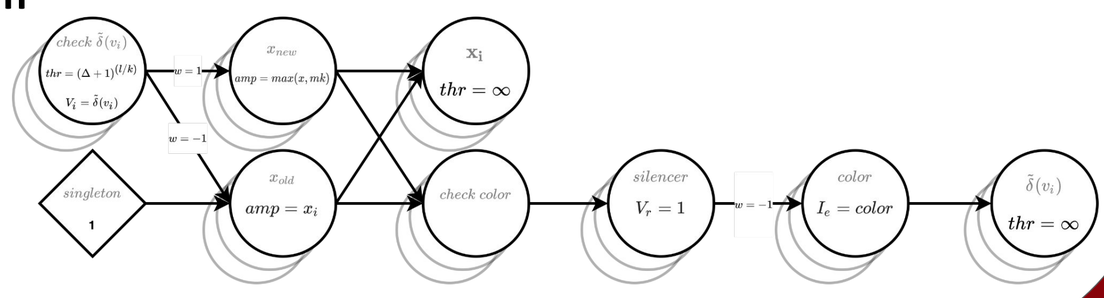


Maximum value of two values

f(x,y)=max(x,y) with x,y>=0



Update function in the approximation of LP_{MDS}



Results

- The implementation of the KW-algorithm in a spiking neural network has been successful as a proof of concept for use of neuromorphic applications for graph algorithms in scientific computing.
- Similar time, space, and energy complexity - but the lack of a **unified complexity theory** for SNNs and distributed algorithms hinders fair comparison.

Future work

- Merge modular subnetworks into a single SNN.
  - Potential energy & time complexity improvements.
- General tutorial on conversion of distributed graph algorithms to SNNs.

Literature

[1] Kuhn, F., Wattenhofer, R.: Constant-time distributed dominating set approximation. Distributed Computing 17(4), 303–310 (2005)
[2] Severa, W., Parekh, O., Carlson, K.D., James, C.D., Aimone, J.B.: Spiking network algorithms for scientific computing. IEEE ICRE (2016)
[3] Aimone, J. B., Parekh, O., Phillips, C. A., Pinar, A., Severa, W., & Xu, H.: Dynamic Programming with Spiking Neural Computing. ICONS (2019)