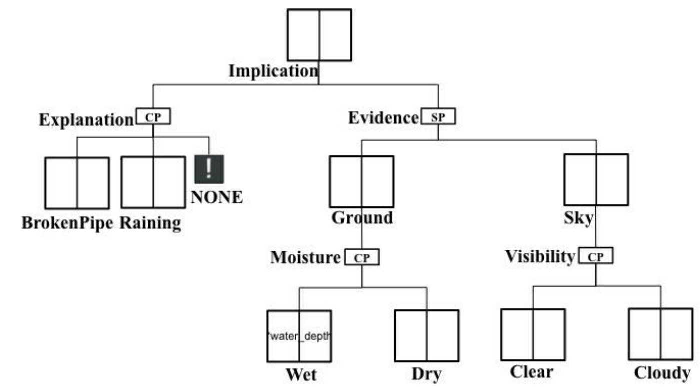
Spiking Neural Network for Asset Allocation Implemented Using The TrueNorth System

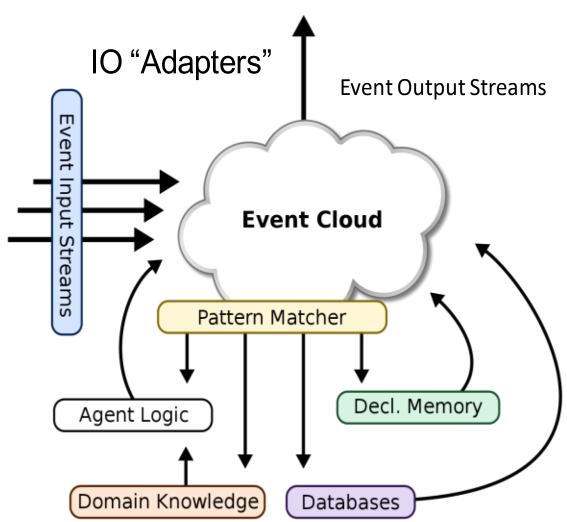
June 25, 2019

Chris Yakopcic (UD), Nayim Rahman (UD), Tanvir Atahary (UD), Md. Zahangir Alom (UD), Tarek Taha (UD), Alex Beigh (UDRI), Scott Douglass (711 HPW)

CECEP Architecture

- The CDO is the decision making engine within the CECEP architecture
- These very simple examples quickly become very complex in realistic systems
  - Billions of possible outcomes

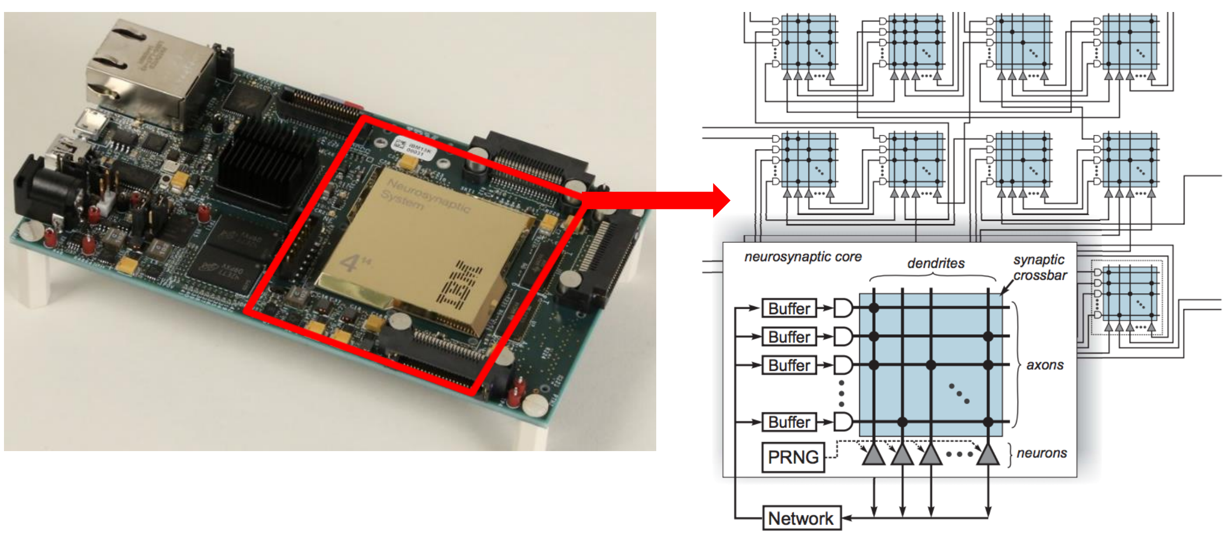




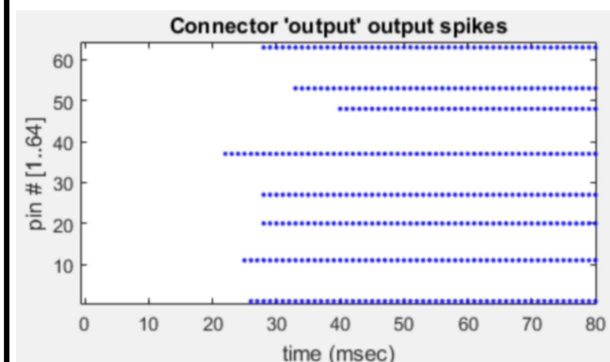
Name	Specification
Raining	IF Implication.explanation = Raining THEN Implication.evidence.Ground.moisture = Wet AND Implication.evidence.Sky.visibility = Cloudy
Broken Pipe	IF Implication.explanation = BrokenPipe THEN Implication.evidence.Ground.moisture = Wet OR (Implication.evidence.Sky.visibility = Clear AND NOT Implication.evidence.Ground.moisture = Dry)
Dry Ground	IFF NOT (Implication.explanation = Raining OR Implication.explanation = BrokenPipe) THEN Implication.evidence.Ground.moisture = Dry
Wet Ground	IFF Implication.evidence.Ground.moisture = Wet THEN Implication.explanation = Raining OR Implication.explanation = BrokenPipe

Objective

- Optimized resource allocation is extremely computationally expensive
- We need low SWaP alternatives, large problems are currently prohibitively expensive to solve.
- This is done using a series of spiking neurons that fire according to the most logical vehicle assignment options
- This work covers a MATLAB implementation of the spiking neuron based algorithm



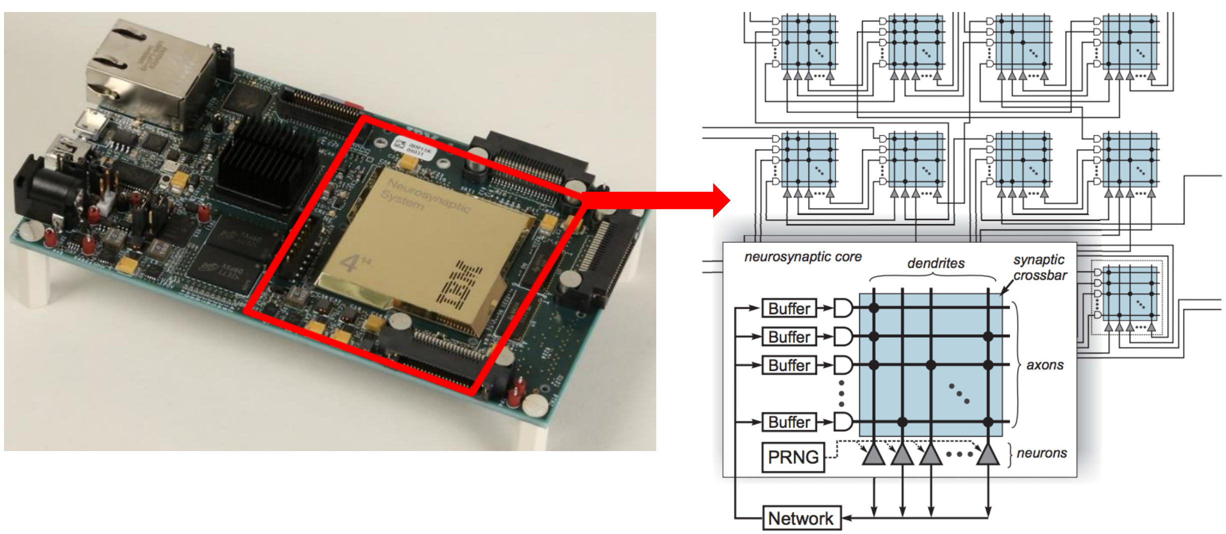
True North Spiking Output



Vehicle 1 Goes to Target 1
Vehicle 2 Goes to Target 3
Vehicle 3 Goes to Target 4
Vehicle 4 Goes to Target 3
Vehicle 5 Goes to Target 5
Vehicle 6 Goes to Target 8
Vehicle 7 Goes to Target 5
Vehicle 8 Goes to Target 7

Objective

- Optimized resource allocation is extremely computationally expensive
- We need low SWaP alternatives, large problems are currently prohibitively expensive to solve.
- This is done using a series of spiking neurons that fire according to the most logical vehicle assignment options
- This work covers a MATLAB implementation of the spiking neuron based algorithm



Allocation Problem Size	Number of Possible Solutions
2 × 2	9
4 × 4	625
6 × 6	117,649
8 × 8	43,046,721
10 × 10	25,937,424,601

Outline

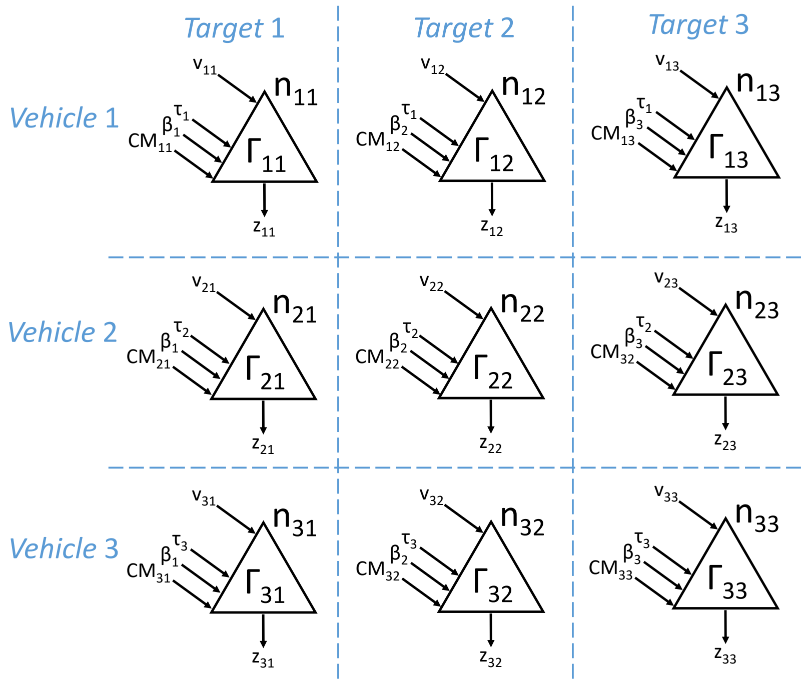
- CECEP Applications

  - M by N Asset Allocation in SNNs
    - Method
    - Algorithm
    - TrueNorth Implementation
    - Results in TrueNorth
    - Latest Implementation and Results on Loihi

Neuron Model

UNIVERSITY of DAYTON

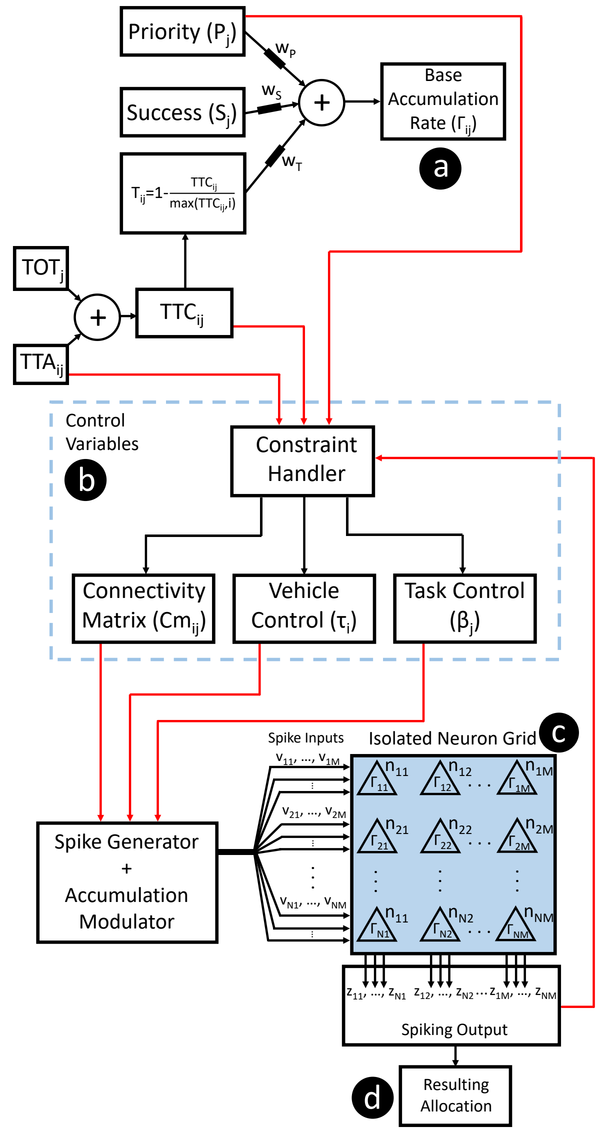
- Single neuron holds connection between one vehicle and one target
  - Capable of allocating N vehicles for M separate targets using N×M neurons
- Weight Parameters
  - TTA: Time to the target
  - Priority: Necessity of reaching target
  - TOT: Hold time for vehicle once target is reached
  - Probability of Success: Likelihood that a target will be completed by a certain vehicle
  - TTC = TOT + TTA
- Control Parameters
  - CM: Connectivity matrix hold vehicle-target compatibility
  - τ: A vehicle can only be assigned to one target
  - β: Penalize (but do not stop) multiple vehicles from reaching a single target



Algorithm Block Diagram

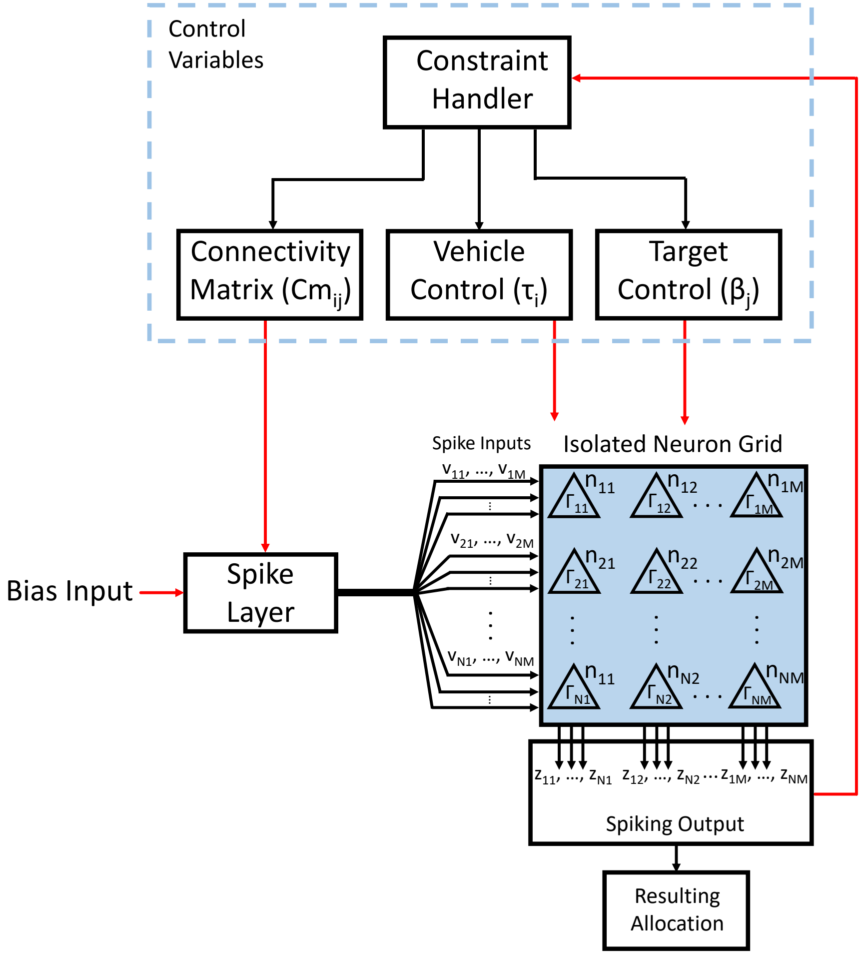
- Flow of inputs and outputs for the algorithm
  - Spike accumulation is proportional to a weighted sum of priority, success, and time
  - Spikes occur depending on compatibility, as well as vehicle and target status

  - a) Base Accumulation Rate
  - b) Control Variables
  - c) Neuron Grid
  - d) Allocation Result



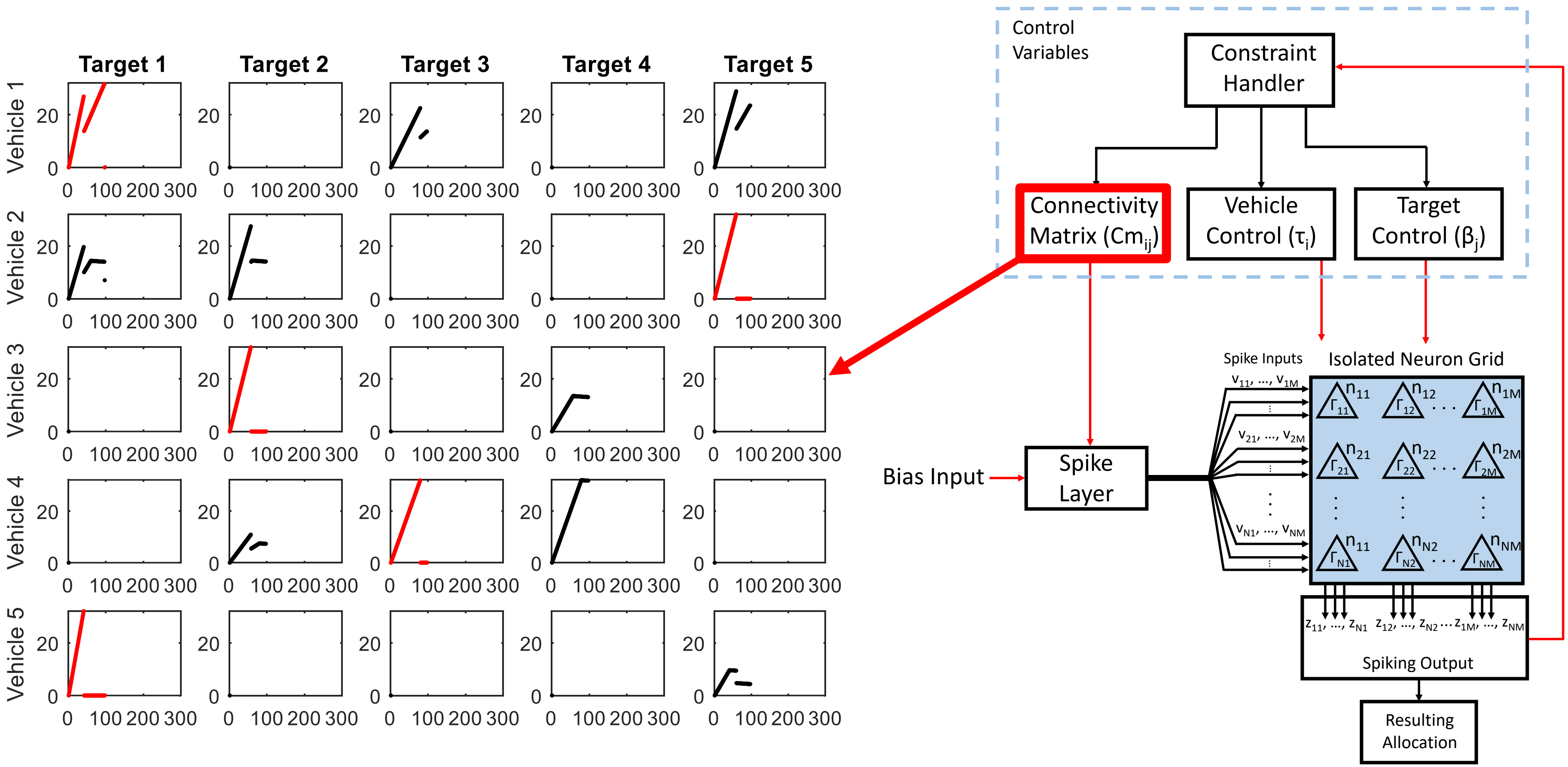
Simplified Algorithm Block Diagram

- Flow of inputs and outputs for the algorithm
  - Spike accumulation is proportional to a weighted sum of priority, success, and time
  - Spikes occur depending on compatibility, as well as vehicle and target status



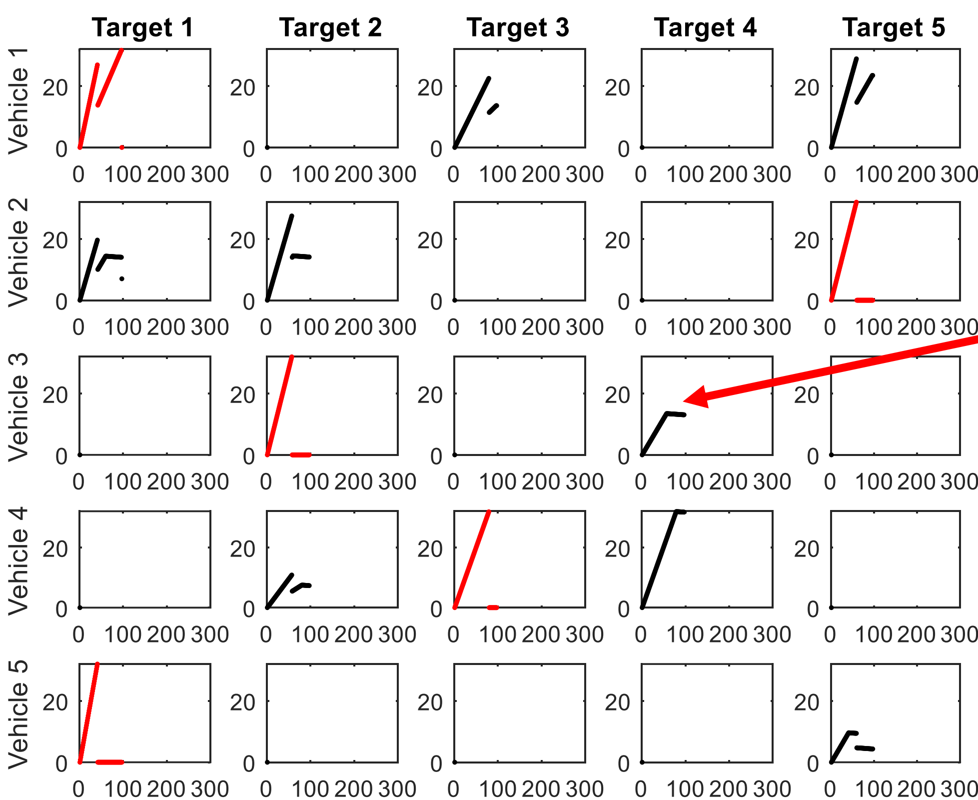
Example Allocation

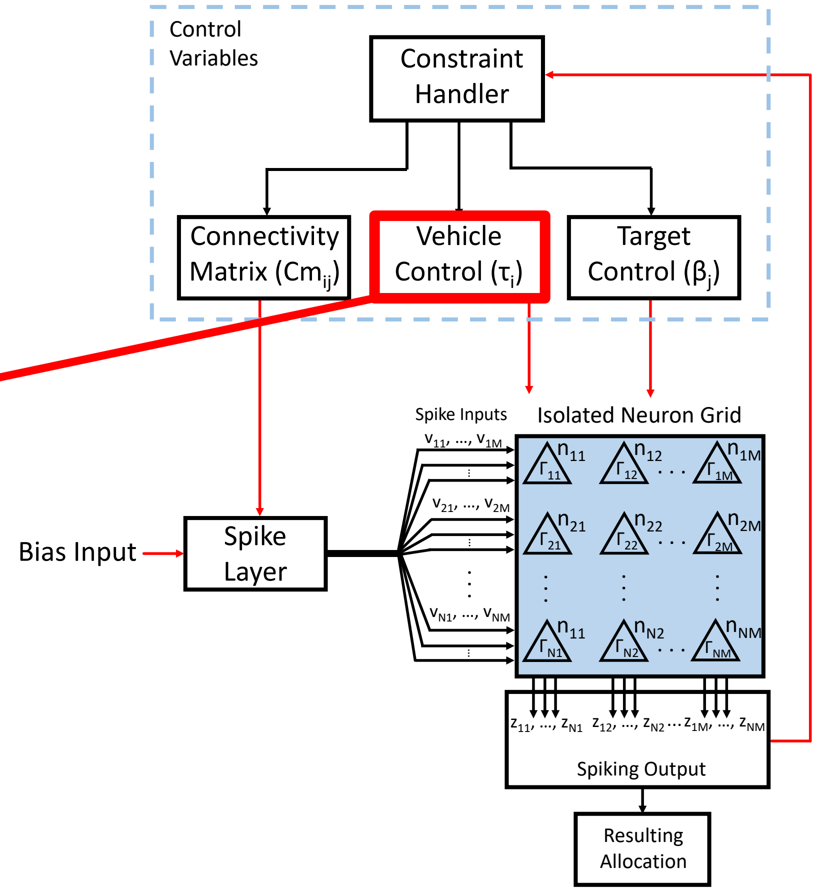
Connectivity Matrix Response



Example Allocation

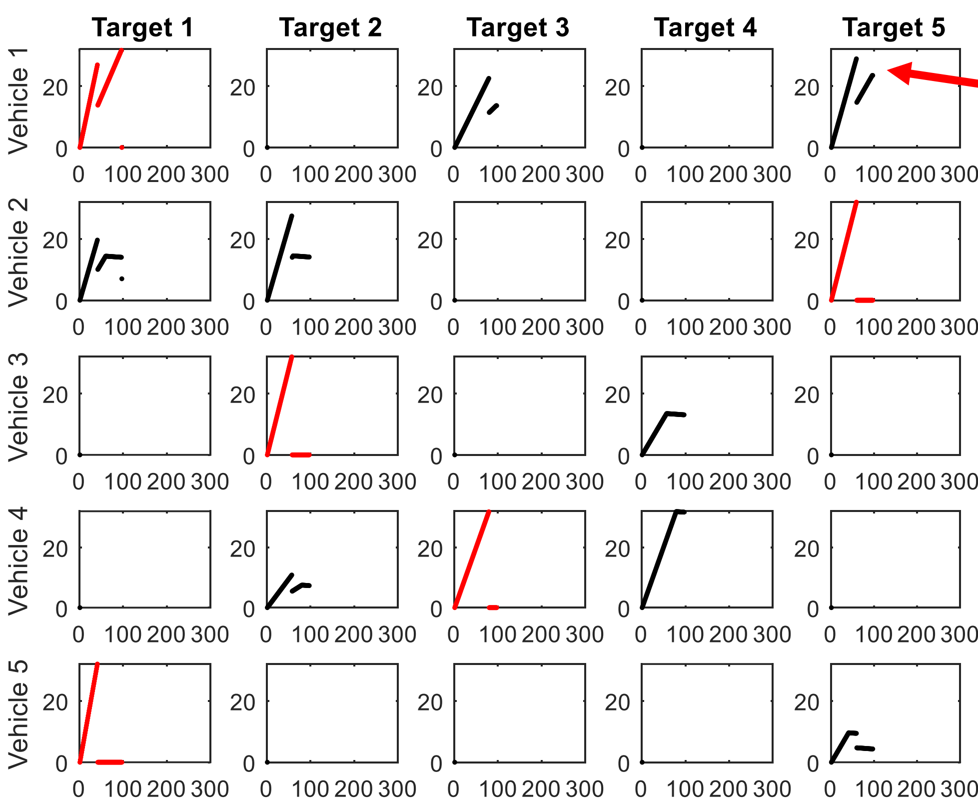
Vehicle Control Response

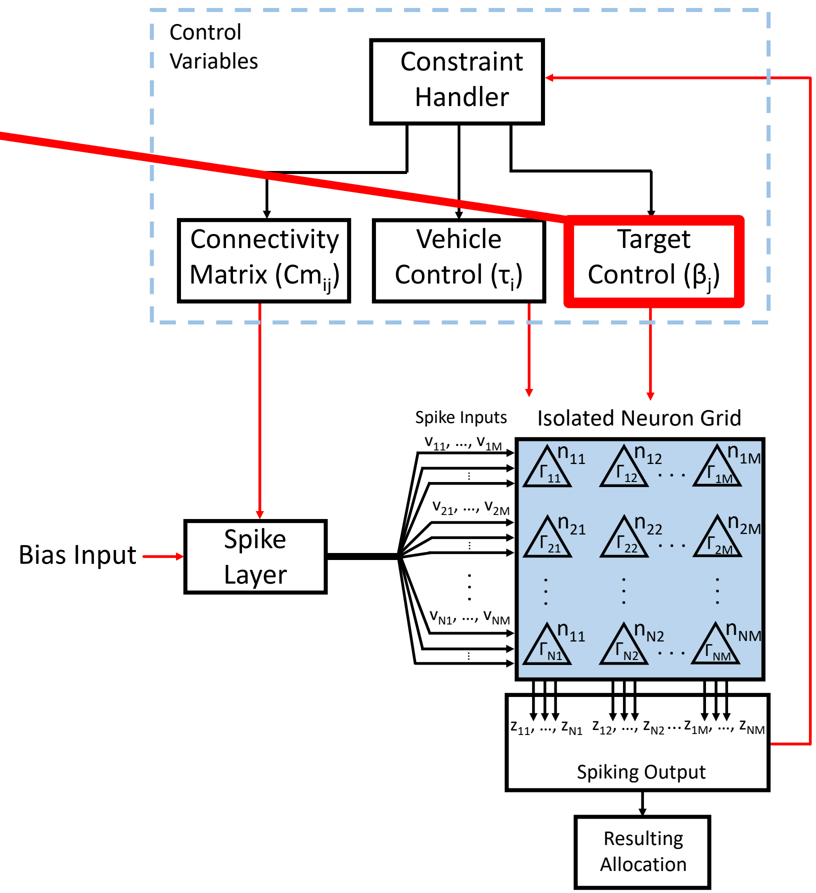




Example Allocation

Target Control Response

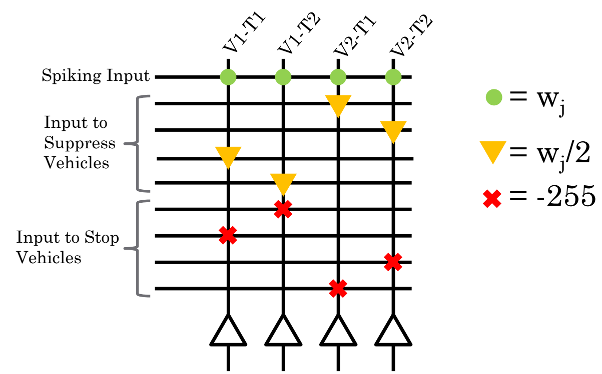




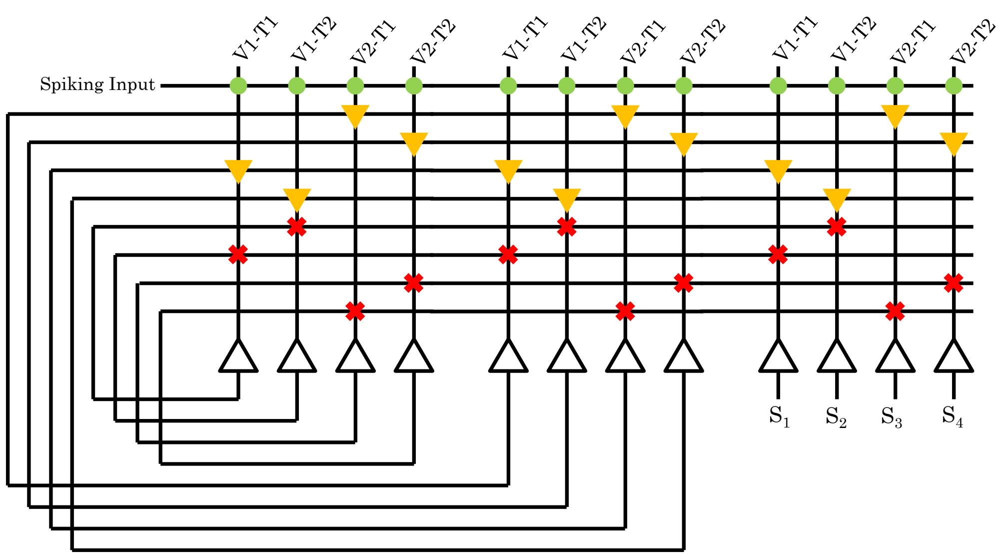
TrueNorth Implementation

- Circuits display the 2 by 2 scenario
  - Simple example mainly for demonstration
- 9 Inputs
  - 1 Uniform Spiking
  - 4 Vehicle Control
  - 4 Task Control
- 12 Outputs
  - 4 Vehicle Control Send
  - 4 Task Control Send
  - 4 Final Outputs
    - 3 Duplications of the same circuit

Neuron Accumulation

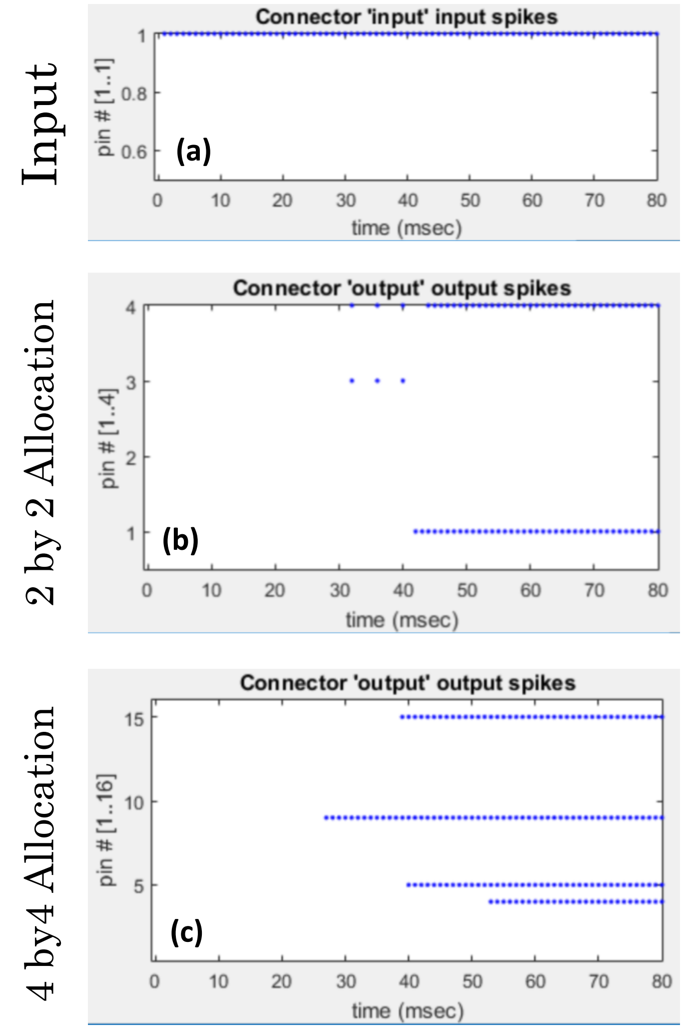


Neuron Accumulation with Control Variables



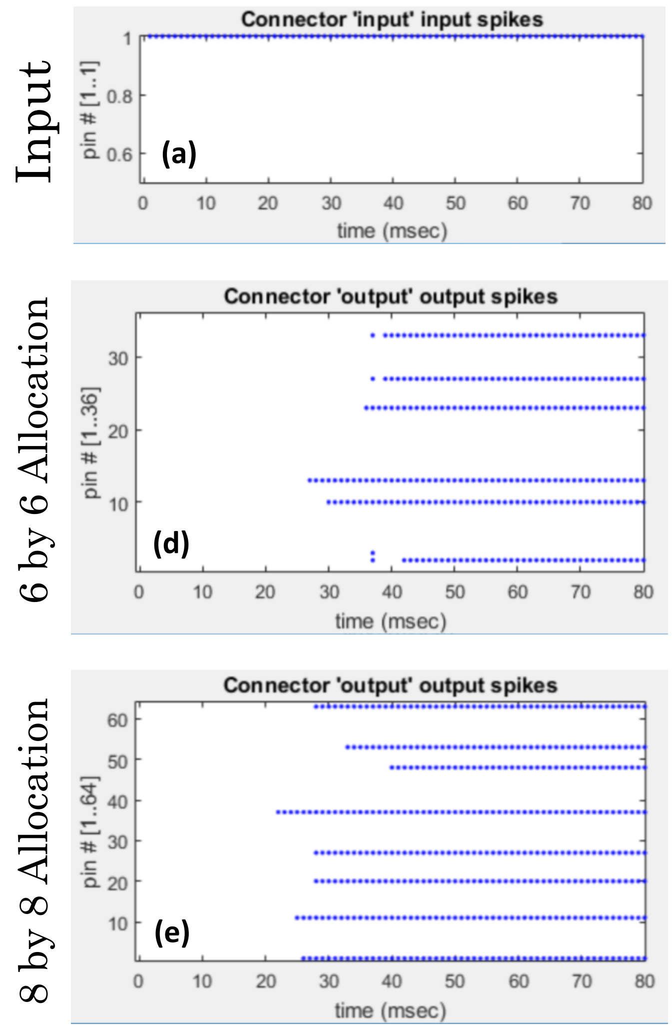
TrueNorth Allocation Results

- Allocation results show which neuron numbers are spiking at the end of an allocation execution
- 2 by 2
  - Neurons: 1 4
  - Result [1 2]
- 4 by 4
  - Neurons: 4 5 9 15
  - Result [4 1 1 3]



TrueNorth Allocation Results

- Allocation results show which neuron numbers are spiking at the end of an allocation execution
- 6 by 6
  - Result [2 4 1 5 3 3]
- 8 by 8
  - Result [1 3 4 3 5 8 5 7]



Algorithm Comparison

	Exhaustive Search Using K80 GPU		TrueNorth Spiking System			
Allocation Size	**Baseline CDO Reward**	**Baseline CDO Result**	**Effective Reward**	**Allocation Result**	**Answer Rank**	**Answer Percentile**
3×3	18.8703	[2 1 1]	18.4807	[2 1 2]	2 of 64	98.44%
4×4	11.377	[4 1 1 3]	11.377	[4 1 1 3]	1 of 625	100%
5×5	22.6219	[1 5 2 4 1]	22.6203	[1 5 2 3 1]	2 of 7776	99.99%
6×6	31.2628	[2 4 1 5 3 3]	31.2628	[2 4 1 5 3 3]	1 of 117649	100%
7×7	48.448	[4 2 2 6 5 6 7]	40.6019	[4 6 4 2 5 5 7]	6847 of 2.09M	99.67%
8×8	40.8782	[1 3 4 7 5 3 6 8]	39.1283	[1 3 4 3 5 8 5 7]	111 of 43.0M	~100%

- The best allocation for each case was determined using a GPU exhaustive search
- The table compares this result to the approximate result obtained from the Loihi spiking system

Timing Comparison

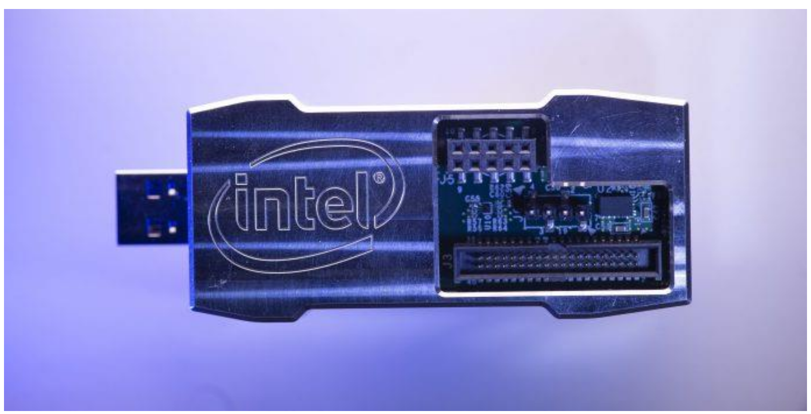
- Runtime comparison between the exhaustive search and spiking system

Allocation Size	CDO Search Time (GPU)	TrueNorth Execution Time	TrueNorth System Speedup
3×3	224 ms	47 ms	4.76×
4×4	231 ms	53 ms	4.36×
5×5	233 ms	52 ms	4.48×
6×6	234 ms	43 ms	5.44×
7×7	269 ms	56 ms	4.80×
8×8	955 ms	40 ms	23.88×

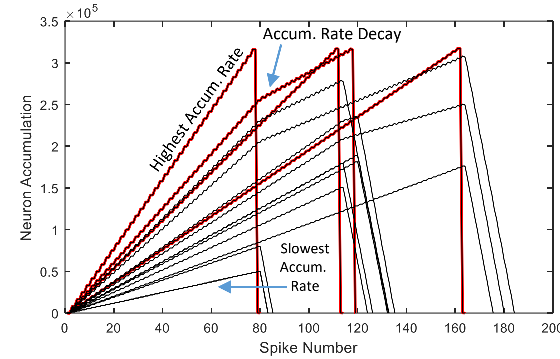
- Solution space on an asset allocation problem grows with problem size
  - Neuron utilization grows at a much smaller rate

Allocation Problem Size	Number of Possible Solutions	Number of Acc. Neurons
2 × 2	9	4
4 × 4	625	16
6 × 6	117,649	36
8 × 8	43,046,721	64
10 × 10	25,937,424,601	100

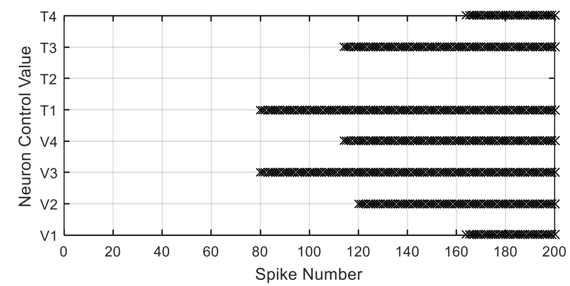
- Loihi implementation of the same algorithm
- Loihi photo shows portable USB stick
  - This work was performed via remote login



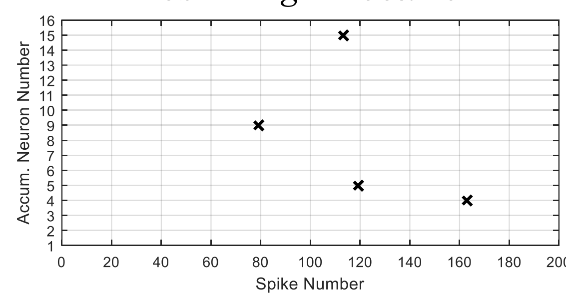
Neuron Grid Accumulations



Control Variable Spiking Layer



Resulting Allocation



Spiking System Comparison

- Loihi demonstrates significant speedup of TrueNorth
  - Mainly due to shorter cycle time and non uniform intervals

Allocation Size	CDO Search Time (GPU)	Loihi Execution Time	Loihi System Speedup	TrueNorth Execution Time	TrueNorth System Speedup
3×3	224 ms	0.312 ms	717×	47 ms	4.76×
4×4	231 ms	0.384 ms	601×	53 ms	4.36×
5×5	233 ms	0.319 ms	730×	52 ms	4.48×
6×6	234 ms	0.414 ms	565×	43 ms	5.44×
7×7	269 ms	0.428 ms	629×	56 ms	4.80×
8×8	955 ms	0.737 ms	1296×	40 ms	23.88×

SWaP Comparison

- General System Comparison
  - Exhaustive/Traditional vs. Embedded/Approximate
  - Approximate solution leads to dramatic increase in efficiency
  - In general, this result shows how the proposed algorithm enables portability of inference algorithms

	Trad. CPU / GPU System	Spiking Systems	Ratio
Size	2240 in^3	24 in^3	93×
Weight	20 lb	0.5 lb	40×
Power	500 W	< 70 mW	7142×
Accuracy	100%	99%	-

Next Steps

- Asset Allocation
  - More complete algorithm comparison
  - Study of maximum scalability
  - Comparison of methods for large scale allocation problems

- Loihi
  - Energy Benchmark
  - Maximum Scalability