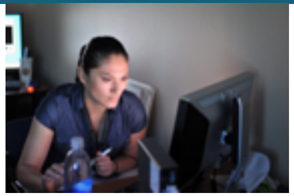
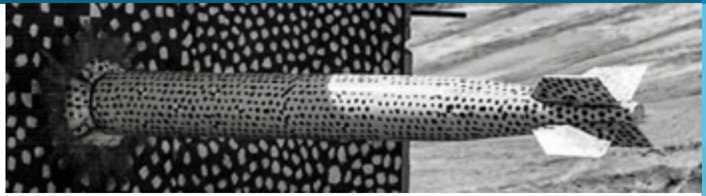
Dynamic Programming with Spiking Neural Computing





PRESENTED BY

Ojas Parekh

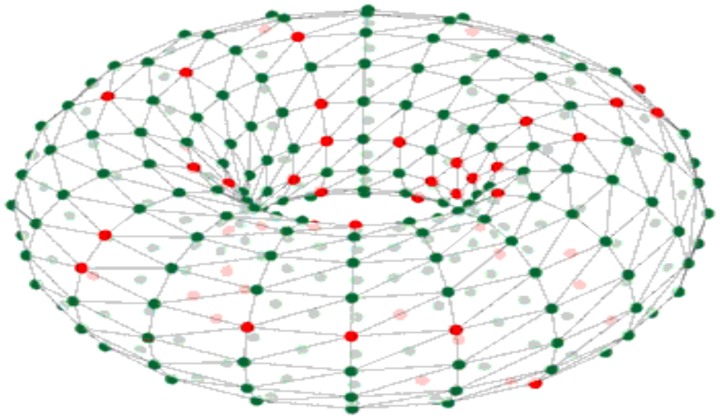
with Brad Aimone, Cindy Phillips, Ali Pinar, William Severa, Helen Xu (MIT), and Yipu Wang (UIUC)

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Neuromorphic Computing Devices

Neuromorphic computing devices will offer billions of energy-efficient neurons
Yet concrete non-learning applications realizing this potential remain elusive

- E.g., SpiNNaker, IBM True North, and Intel Loihi
- Simple artificial neurons offer massive localized parallelism
- Offer millions of energy-efficient neurons in compact footprints, with billions likely in the near future

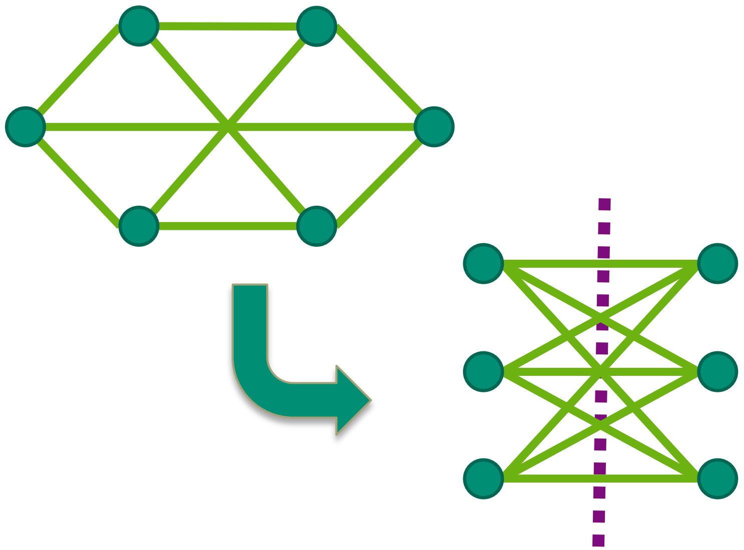


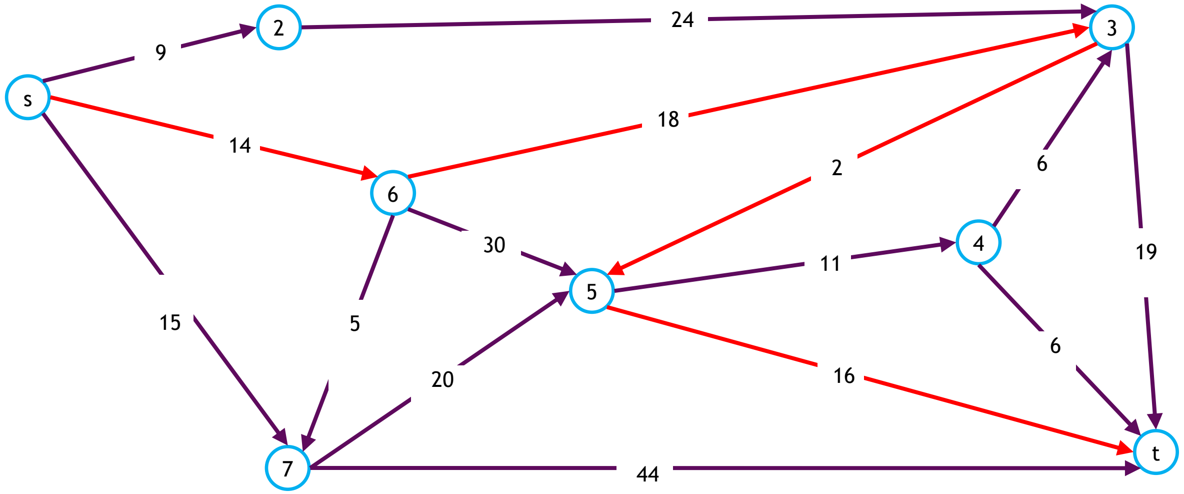
- Motivated by learning-oriented applications
- Limited benchmarks demonstrating fair and rigorous advantage
- **Neuromorphic architectures are massive graphs!**

[http://apt.cs.manchester.ac.uk/projects/SpiNNaker]

Graph Algorithms

Graph algorithms are at the heart of myriad applications:
navigation, social network analysis, cybersecurity, logistics, DNA analysis, …

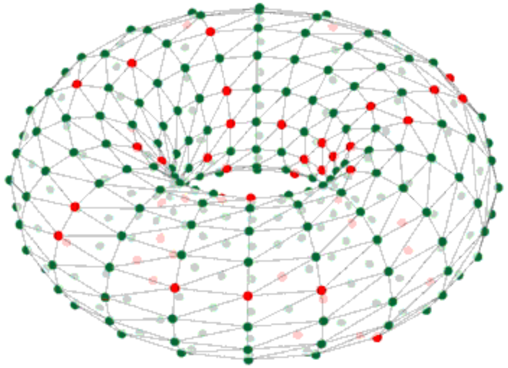


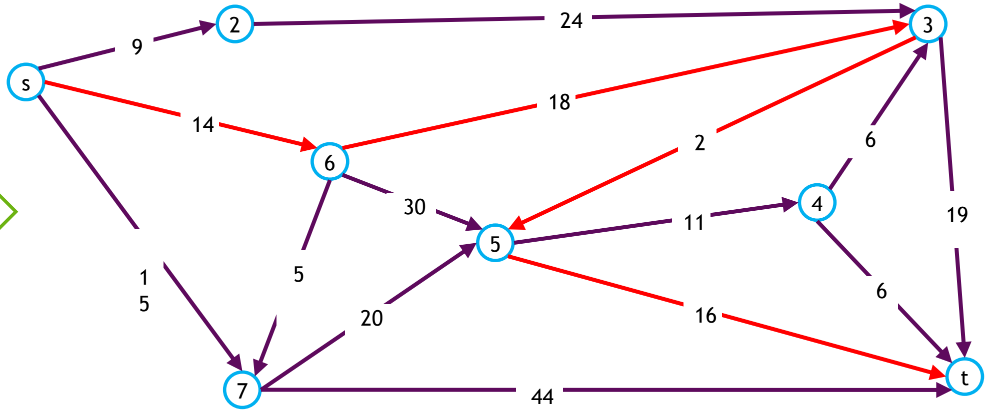


- Based on primitive algorithms for: paths, flows, cuts, clustering, …
- Overall performance often dominated by performance on primitives
- Graph500 benchmarks primitives at massive scale
- Current approaches hampered by demise of Moore’s law

Neuromorphic Graph Algorithms

Neuromorphic graph algorithms naturally leverage massive-scale neuromorphic devices



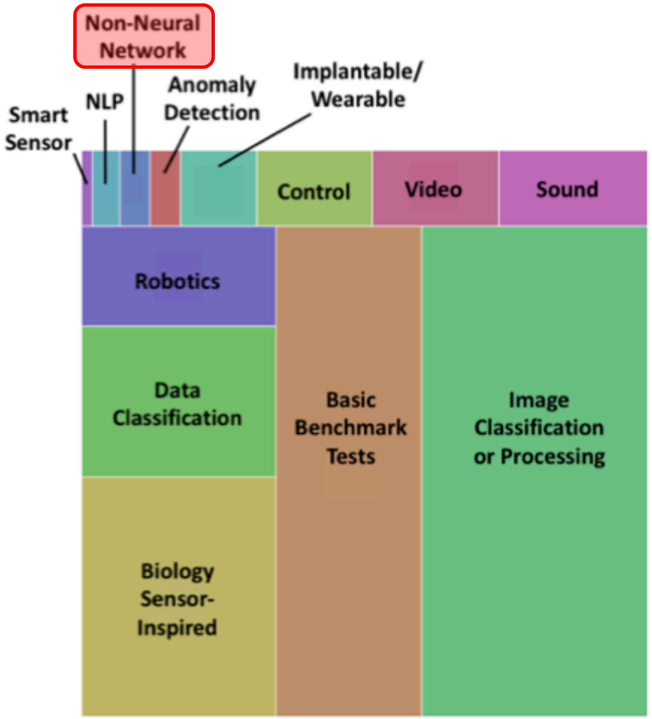


1. Scalable graph analysis as data needs grow well into the future
2. Rigorous assessment and validation of neuromorphic computing
3. **Bonus:** extending the scope of neuromorphic beyond learning

Current Neuromorphic Graph Algorithms

Study of Neuromorphic Graph Algorithms (NGAs) is limited

- Recent survey by Schuman et al. of neuromorphic computing covering 2500+ references had only 8 citations of graph applications (see figure)
- Most of above graph applications have a learning-oriented component (Hopfield networks or Boltzmann machines)
- A few spike-based graph primitives papers have emerged recently (e.g., [Hamilton, Mintz, Schuman, https://arxiv.org/abs/1903.10574, 2019])
- Timely opportunity for NGAs!



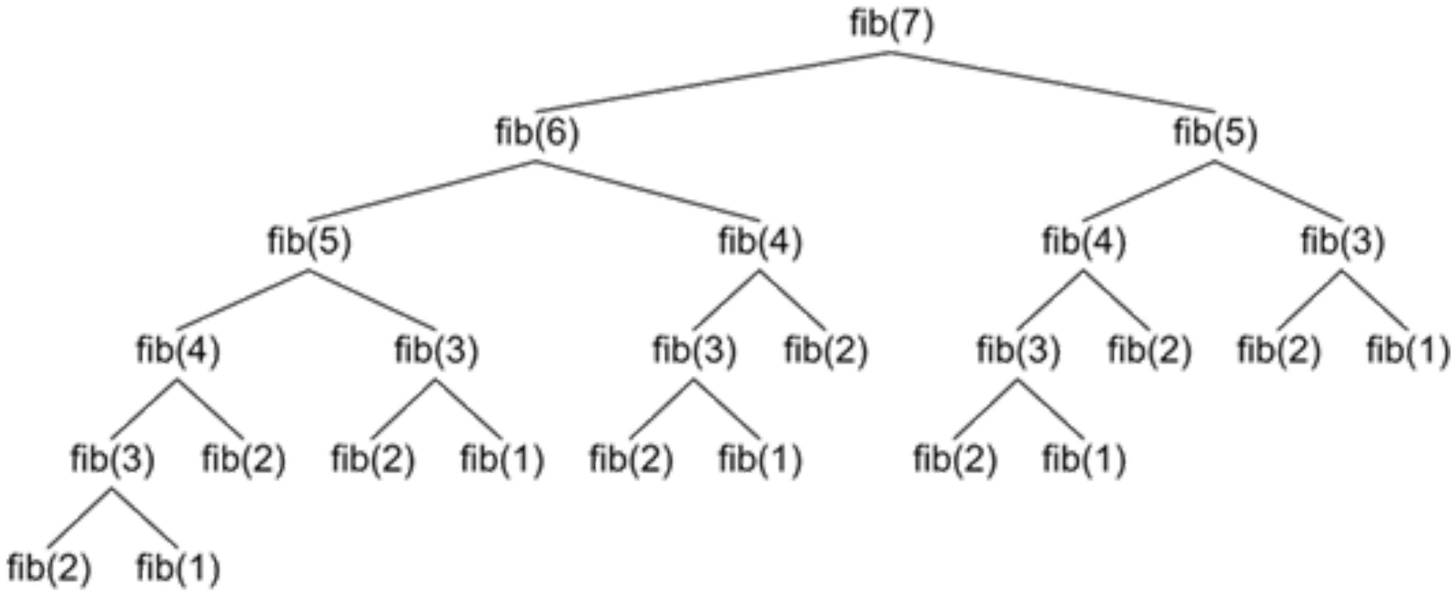
Landscape of current neuromorphic applications based on 2500+ references
[Schuman et al., https://arxiv.org/abs/1705.06963, 2017]

Dynamic Programming

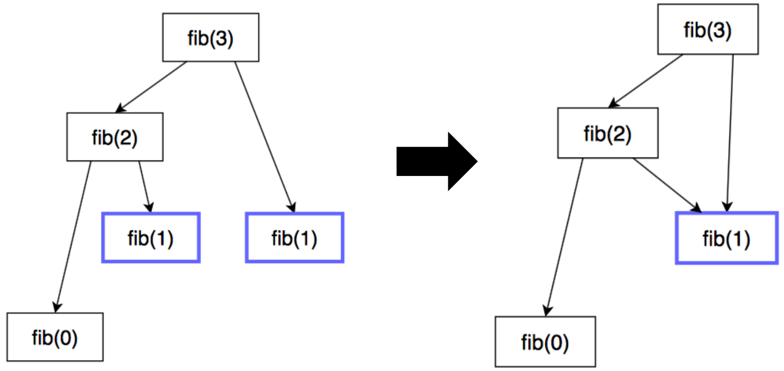
Dynamic programming is a *general technique* for solving certain kinds of discrete optimization problems

Dynamic programming consolidates redundant computation

$$fib(n) = fib(n-1) + fib(n-2); fib(1) = 1, fib(2) = 1$$



> Those who cannot remember the past are condemned to repeat it.
>
> -Dynamic Programming



[https://blog.usejournal.com/top-50-dynamic-programming-practice-problems-4208fed71aa3]
[https://programming.guide/dynamic-programming-vs-memoization-vs-tabulation.html]
[https://medium.com/@shmuel.lotman/the-2-00-am-javascript-blog-about-memoization-41347e8fa603]

Broad Applications of Dynamic Programming

Dynamic programming is a *general technique* for solving certain kinds of discrete optimization problems

- Recurrent solutions to lattice models for protein-DNA binding
- Backward induction as a solution method for finite-horizon discrete-time dynamic optimization problems
- Method of undetermined coefficients can be used to solve the Bellman equation in infinite-horizon, discrete-time, discounted, time-invariant dynamic optimization problems
- Many string algorithms including longest common subsequence, longest increasing subsequence, longest common substring, Levenshtein distance (edit distance)
- Many algorithmic problems on graphs can be solved efficiently for graphs of bounded treewidth or bounded clique-width by using dynamic programming on a tree decomposition of the graph.
- The Cocke–Younger–Kasami (CYK) algorithm which determines whether and how a given string can be generated by a given context-free grammar
- Knuth's word wrapping algorithm that minimizes raggedness when word wrapping text
- The use of transposition tables and refutation tables in computer chess
- The Viterbi algorithm (used for hidden Markov models, and particularly in part of speech tagging)
- The Earley algorithm (a type of chart parser)
- The Needleman–Wunsch algorithm and other algorithms used in bioinformatics, including sequence alignment, structural alignment, RNA structure prediction
- Floyd's all-pairs shortest path algorithm
- Optimizing the order for chain matrix multiplication
- Pseudo-polynomial time algorithms for the subset sum, knapsack and partition problems
- The dynamic time warping algorithm for computing the global distance between two time series
- The Selinger (a.k.a. System R) algorithm for relational database query optimization
- De Boor algorithm for evaluating B-spline curves
- Duckworth–Lewis method for resolving the problem when games of cricket are interrupted
- The value iteration method for solving Markov decision processes
- Some graphic image edge following selection methods such as the "magnet" selection tool in Photoshop
- Some methods for solving interval scheduling problems
- Some methods for solving the travelling salesman problem, either exactly (in exponential time) or approximately (e.g. via the bitonic tour)
- Recursive least squares method
- Beat tracking in music information retrieval
- Adaptive-critic training strategy for artificial neural networks
- Stereo algorithms for solving the correspondence problem used in stereo vision
- Seam carving (content-aware image resizing)
- The Bellman–Ford algorithm for finding the shortest distance in a graph
- Some approximate solution methods for the linear search problem
- Kadane's algorithm for the maximum subarray problem
- Optimization of electric generation expansion plans in the Wein Automatic System Planning (WASP) package

Wikipedia: 30 applications across diverse domains
[https://en.wikipedia.org/wiki/Dynamic_programming]

Another list with 50 applications
[https://blog.usejournal.com/top-50-dynamic-programming-practice-problems-4208fed71aa3]

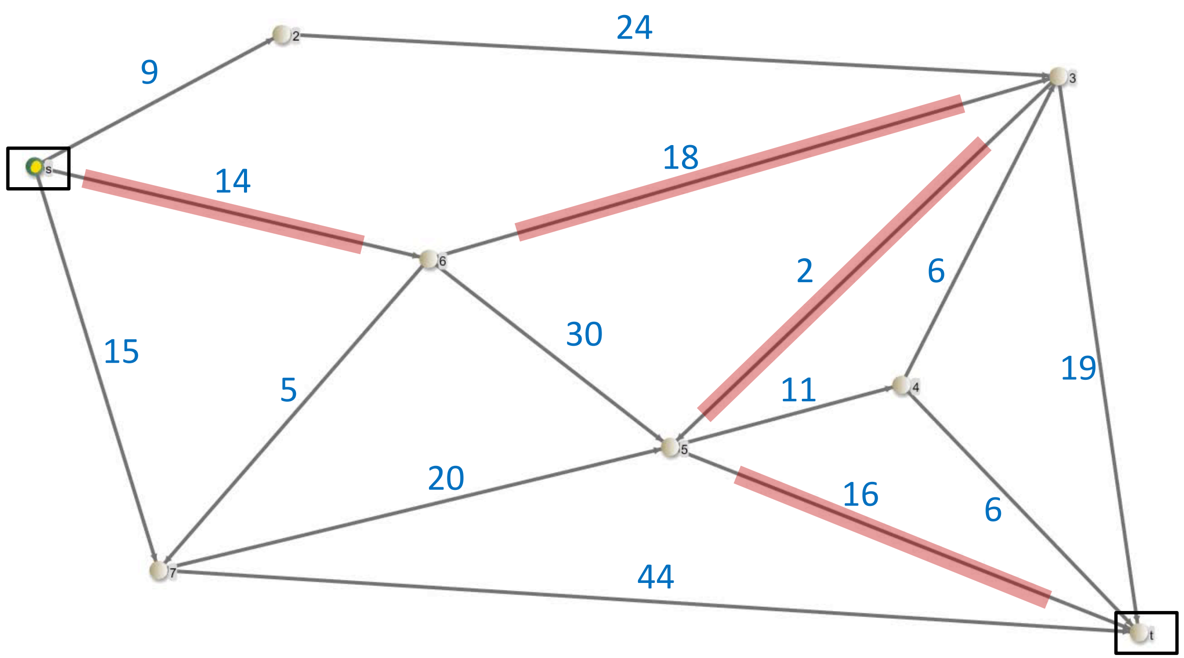
Spiking Dynamic Programming Approach

New neuromorphic algorithms for dynamic programming

Generically solves a broad class of dynamic programs

Spiking shortest paths algorithm

[Aibara et al., IEEE Int. Symp. on Circuits and Systems, 1991]



- Our dynamic programming algorithm leverages shortest path NGA
- Single neuron per dynamic program table entry
- Employs delays on links (simulable using recurrent neurons)
- **Novel temporal encoding:** time when neuron first fires represents value of dynamic program table entry

Spiking Dynamic Programming Example

New neuromorphic algorithms for dynamic programming

Spike times encode dynamic programming table values

Dynamic Program for Knapsack Problem

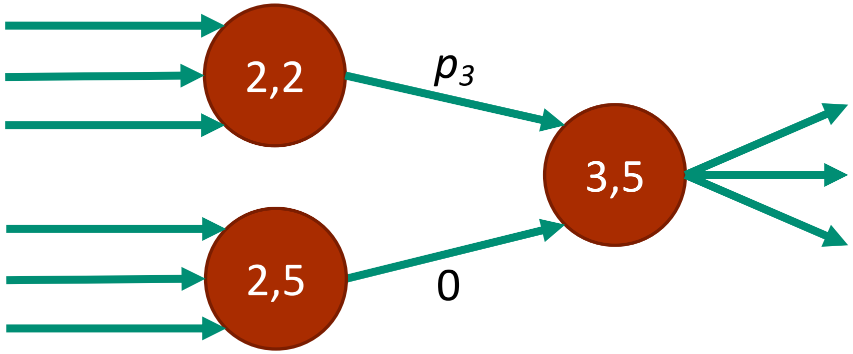
k \ W	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1
2	0	1	2	3	3	3	3	3	3	3	3
3	0	1	2	4	5	6	7	7	7	7	7
4	0	1	2	4	5	6	7	8	9	9	9
5	0	1	2	4	5	6	7	8	9	10	11

Each table entry is value of best knapsack solution of weight at most W using items {1,…,k}

Knapsack Problem:
N items, each with weight w_i and value v_i

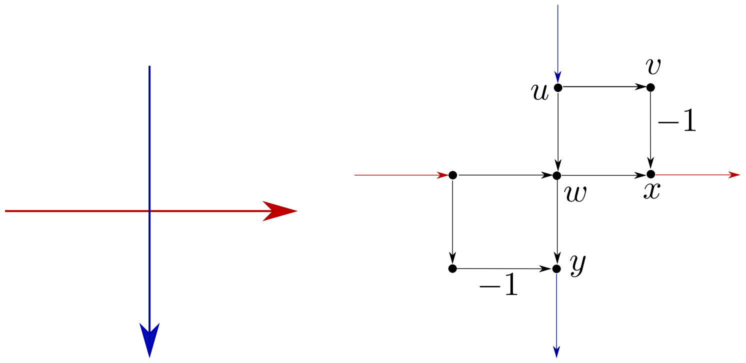
Goal: pick subset of items of weight at most W, maximizing total value.

$$T[3,5] = max\{\overbrace{T[2,5-w_3]+p_3}^{=6}, \overset{=3}{T[2,5]}\}$$



Spiking approach: T[i,j] encoded as time neuron (i,j) receives incoming spike on last of its incoming links

Practical Considerations and Extensions



- Dynamic program graph must be simulated on neuromorphic hardware graph
  New graph embedding problems and techniques

- Neuromorphic hardware has a fixed minimum delay
  Problem-specified delays must be scaled, introducing multiplicative factor to running time

- Dynamic programming graph loading and readout (I/O) costs may present bottlenecks
  Optimized problem-specific algorithms possible (we do so for longest increasing subsequence)

- Spiking approach as presented only gives *value* solution
  Can use O(log n) extra neurons per graph node as memory to store solution
  Novel Hebbian learning approach on edges also works!