Big Data Systems and Analytics: Lecture 1

Ling Liu
Professor
Distributed Data Intensive Systems Lab
School of Computer Science
Georgia Institute of Technology

Themes of the course

Big Data Systems +

- optimizations for big data applications,
- scaling techniques/methods for big data driven ML/AI algorithms

Big Data Analytics

- innovation in big data driven learning algorithms,
- innovation in optimizing ML/AI algorithms/Models for Big Data

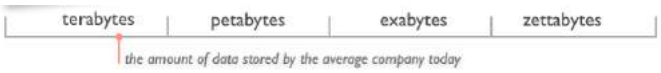
What is Big Data: A CS Perspective

- Big data refers to datasets
  - that are *beyond the ability of legacy approaches* to manage at an acceptable level of quality and / or
  - that *exceed the capacity of conventional systems* (hardware and/or software) to process within an acceptable elapsed time.
- Definition of big data: Subjective & evolving
  - As technology advances over time, the size of datasets that qualify as big data will also increase.
  - The definition is varying by sector, depending on
    - what kinds of software tools are commonly available and
    - what sizes of datasets are common in a particular industry or science domain.

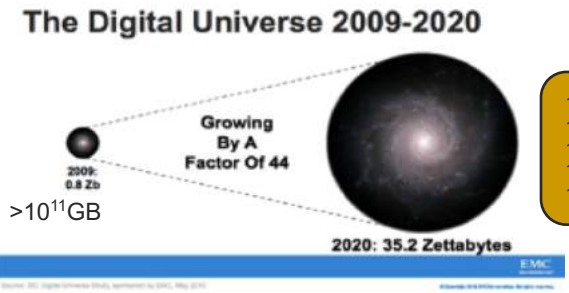
Characteristics of "Big" Datasets

- Huge in Volume
- Distributed
  - Dispersed over many servers
- Dynamic (Velocity)
  - Items add/deleted/modified continuously
- Heterogeneous (Variety)
  - Many agents access/update data
- Noisy
  - Inherent
  - Unintentional
  - Malicious
- Unstructured / semi-structured
  - No database schema

Characteristics of Big Data (1): Scale (Volume)



- **Data Volume**
  - 44x increase from 2009 - 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing exponentially



$>10^{11}$GB

1 terabyte (TB) =1,024GB ~10^3GB
1 petabyte (PB) =1,024TB ~ 10^6GB
1 exabyte (EB) =1,024PB~10^9GB
1 zettabyte (ZB) =1,024EB~10^{12}GB.
1 yottabyte (YB) =1,024ZB~10^{15}GB.

Georgia Institute of Technology | Georgia Tech College of Computing

Big Data: The Volume Challenge

- Challenges for sensing, collection, generation of more data, more meaningful data, more useful data
- Challenges for computation on data bigger in volume
- Challenges for communication on data bigger in volume
- Challenges for analysis (algorithms) on data bigger in volume
- How would we design **volume adaptive** big data processing **framework, systems, algorithms** and **services**?

Georgia Institute of Technology | Georgia Tech College of Computing

Characteristics of Big Data (2): Speed (Velocity)

- Data are generated fast and need to be processed fast
- Online Data Analytics + Real time Analytics
- Delayed decisions ➔ missing opportunities
- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body responses ➔ any abnormal measurements require immediate reaction

Georgia Institute of Technology | Georgia Tech College of Computing

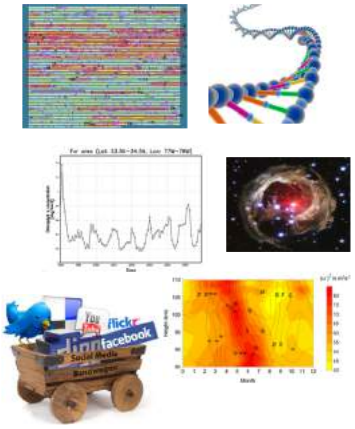
Big Data: The Velocity Challenge

- Challenges for collecting sensor data in real time
  - fast rates of data generations and consumption, regarding data input sources, data output rate, varying rates to sense and collect data, the varying rates of data output
- Challenges for analyzing sensor data in real time:
  - how fast to analyze, and its implication/impact.
- What would be **velocity adaptive** big data processing **framework, systems, algorithms** and **services**?

Georgia Institute of Technology | Georgia Tech College of Computing

Characteristics of Big Data (3): Complexity (Varity)

- Input data related complexity
  - Various formats, types, and structures
  - Text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc…
  - Static data vs. streaming data
  - A single application may be generating/collecting many types of data



To extract knowledge➔ all these types of data need to linked together

Characteristics of Big Data (3): Complexity (Varity)

- Algorithms related complexity
  - Different algorithms may derive different insights/outputs from the same collection of datasets
  - Which analysis does make sense?
  - How do we compare / combine different algorithmic recommendations?
- Metrics (Quality and Performance measurement)
  - Different measures can be used to measure performance, effectiveness, and utility of big data
  - **What metrics do make sense?**

To leverage different algorithms and metrics ➔ derive high quality value and data utility

Many Sources Generate Big Data

Social media and networks
(all of us are generating data)

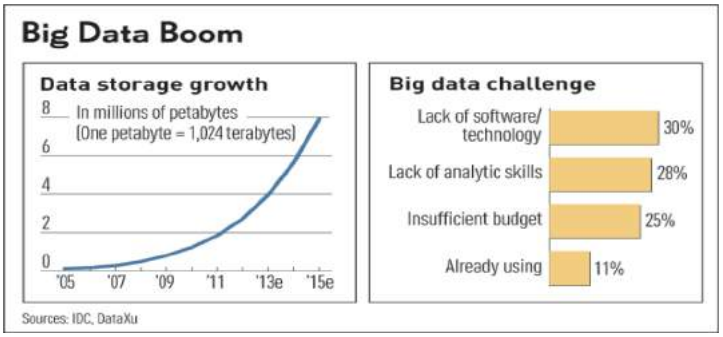
Scientific instruments
(collecting all sorts of data)

Mobile devices
tracking all objects all the time)

Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation are no longer hindered by the ability to collect data; but by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

Challenges in Handling Big Data



- **The Bottleneck is in technology**
  - New architecture, algorithms, techniques are needed in data collection, storage, processing, data center management, data security and privacy …
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data

New challenges
Restricted access

- Large data sets are kept on magnetic devices



- Access to data is sequential
- Random access is costly

New challenges
Search the data

- Traditionally, input data is:
  - Either small and thus easily searchable
  - Moderately large, but organized in database tables.
- In large data sets, input data is:
  - Immense
  - Disorganized, unstructured, non-standardized

Hard to find what you want

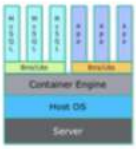
New challenges
The Scalability Dilemma

- State-of-the Art Machine Learning techniques do not scale to large data sets.
  - Memory constraints
  - Network bandwidth constraints
- Data Analytics frameworks can't handle lots of incomplete, heterogeneous, dirty data.
- Processing architectures struggle with increasing diversity of programming models and job types.
- Goal driven feature extractions demand a lot more innovations on learning and modeling techniques
- …

Google Cluster Trace + Measurement Study

Google Cluster 2011 trace at https://github.com/google/cluster-data.

186 GB of data



Detailed workload

- ~700000 jobs
- Each job, multiple tasks on LXCs (Linux Containers)

12583 heterogeneous machines

- Each task assigned to a single physical machine

Exhaustive information

- **Actual** CPU and memory usage per task, execution time, job priorities...
- Collected during **29 days** on 5-minute monitoring intervals

Priority groups

- Low-priority tasks can be evicted/migrated in favor of higher-priority ones

Google Cluster Trace

A motivating example

- Trace data for 29 days in May 2011
  - ~12,500 machines
  - ~672,000 jobs
  - ~144,000,000 task events
  - 12 priorities
  - 4 scheduling classes

- Task Events:

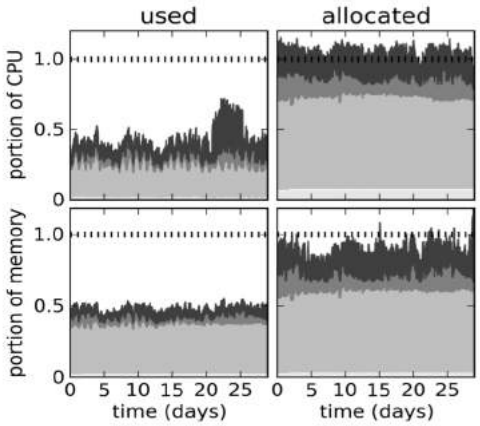
job_id,
task_index
machine_id
event_type
Scheduled, evicted, finished
priority, scheduling_class
memory_request,
cpu_request
Disk_space_request

"Preemptive Scheduling with Application-Transparent Checkpointing in Shared Clusters", ACM Compware 2015
Jack Li, Calton Pu, Yuan Chen, Vanish Talwar, Dejan Milojicic

18

Memory is not efficiently used

- Google datacenter usage analysis (1)



Mapping of original times to times emitted in the trace.

Source: *Towards understanding heterogeneous clouds at scale: Google trace analysis* Charles Reiss (UC Berkeley), Alexey Tumanov (CMU), Gregory R. Ganger (CMU), Randy H. Katz (UC Berkeley), Michael A. Kozuch (Intel Labs)
Intel Science & Technology Center for Cloud Computing, Carnegie Mellon University
http://sbac.lip6.fr/2014/session%209/3-JALorenzo.pdf

Sample Task Trace

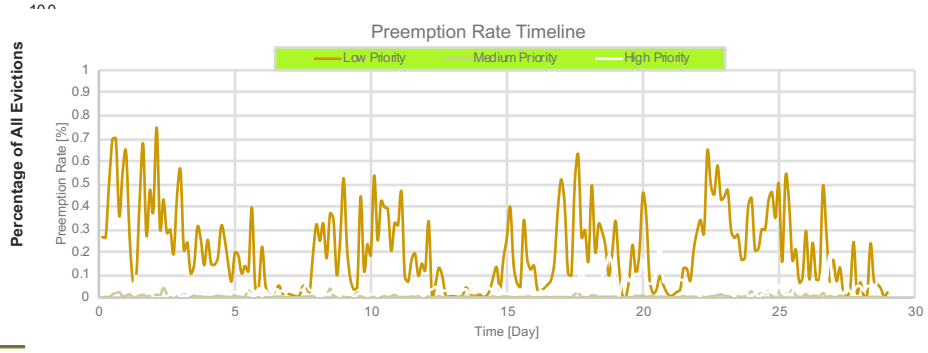
timestamp	job_id	Task index	machine_id	event_type	scheduling_class	priority	cpu_request	memory_request	disk_space_request
0	6250345153	822	0	Submitted	0	0	0.06873	0.01193	0.000115
0	6250345153	822	351650065	Scheduled	0	0	0.06873	0.01193	0.000115
13.7093	6250345153	822	351650065	Evicted	0	0	0.06873	0.01193	0.000115
13.7093	6250345153	822	0	Submitted	0	0	0.06873	0.01193	0.000115
15.2124	6250345153	822	4479483611	Scheduled	0	0	0.06873	0.01193	0.000115
1634.089	6250345153	822	4479483611	Evicted	0	0	0.06873	0.01193	0.000115
1634.089	6250345153	822	0	Submitted	0	0	0.06873	0.01193	0.000115
1636.392	6250345153	822	1335575	Scheduled	0	0	0.06873	0.01193	0.000115
2169.126	6250345153	822	1335575	Finished	0	0	0.06873	0.01193	0.000115

"Improving Preemptive Scheduling with Application-Transparent Checkpointing in Shared Clusters", ACM Middleware 2015
Jack Li, Calton Pu, Yuan Chen, Vanish Talwar, Dejan Milojicic

Google Trace Analysis (2)

- 12.4% of scheduled tasks evicted
- Up to 30k CPU-hours (35% of total capacity) wasted!
- Even latency-sensitive tasks are evicted

Task Priority	Num. of Tasks	Percent Evicted
Free (0-1)	28.4M	20.26%
Middle (2-8)	17.3M	0.55%
Production (9-11)	1.70M	1.02%



"Improving Preemptive Scheduling with Application-Transparent Checkpointing in Shared Clusters", ACM Middleware 2015
Jack Li, Calton Pu, Yuan Chen, Vanish Talwar, Dejan Milojicic

21

Problems Exposed

- Jobs got evicted even when there are sufficient CPU and memory available in the cluster! Why?
  - Some VMs experienced memory / CPU resource contention while other VMs have lot of idle CPU/Memory Resources
  - Preemption in Cluster Scheduling is not optimized
    - Checkpoint cost-aware eviction
    - Choose proper policies (e.g., kill vs.. checkpoint, local vs. remote) based on application progress, overhead and resource availability

"Improving Preemptive Scheduling with Application-Transparent Checkpointing in Shared Clusters", ACM Middleware 2015, Jack Li, Calton Pu, Yuan Chen, Vanish Talwar, Dejan Milojicic

A root cause of the problem:
Memory Utilization of different VMs are not well balanced against their workloads.

One Goal of this Short Course

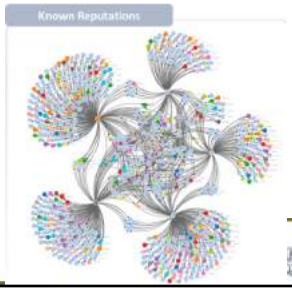
- **Rethinking of how the software defined intelligence should be learned, implemented and delivered.**
- **Three Lectures**
  - Lecture 1: A Strawman's perspective + case studies
  - Lecture 2: Wisdom of Crowd + Power of Ensemble
  - Lecture 3: Robustness of ML/AI against Deception

Lecture 1

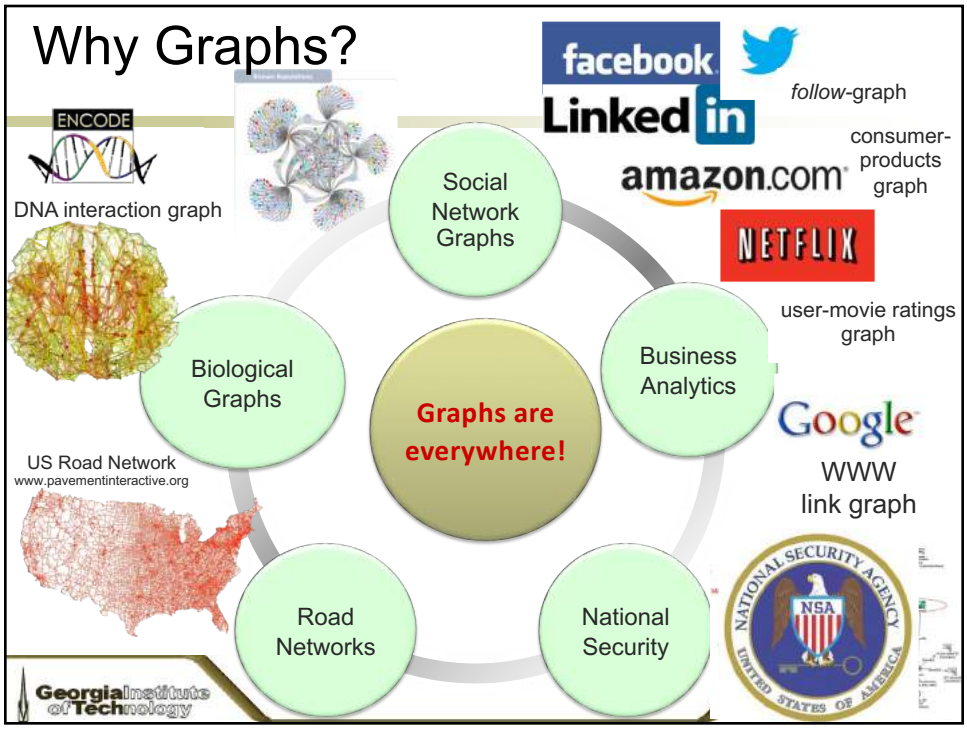
- A Strawman Big Data System: Big Graphs
  - Walk through the process of big graph data processing
  - Discussion: many problems with simple implementation
- A Strawman Data Analytic Algorithms: Big Graph
  - Walk through the process of data analytic algorithms on sensor networks
  - Discussion: many problems with simple implementation
  - Third case Study: Deep Learning needs to be combined with other algorithms, such as search, index, vision
    - Amazon Go + Google Alpha Go

BigGraph: BigData with *Structure*

- A graph is
  - a collection of binary relationships
  - viewed as networks of pairwise interactions between (physical or conceptual) entities
- Real information networks are represented as graphs and analyzed using graph theory
  - Internet
  - Road networks
  - Utility Grids
  - Protein Interactions



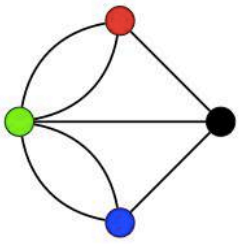
Why Graphs?



Scale of the first graph

Nearly 300 years ago the first graph problem consisted of 4 vertices and 7 edges—*Seven Bridges of Königsberg* problem.

[Paul Burkhardt, Chris Waring 2013]



Crossing the River Pregel

Is it possible to cross each of the Seven Bridges of Königsberg exactly once?

The data set and the algorithm can easily fit in the DRAM "memory"

Georgia Institute of Technology

Georgia Tech College of Computing

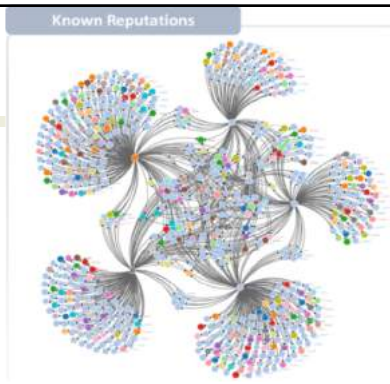
Road Scale

>24 million vertices
>58 million edges
*Route Planning in Road Networks - 2008

Georgia Institute of Technology

Georgia Tech College of Computing

Social Scale



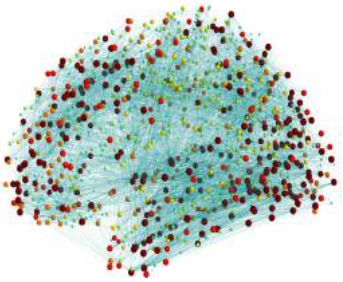
>1 billion vertices
~ 1 trillion edges
*Facebook Engineering Blog

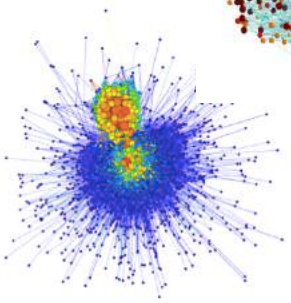
~41 million vertices
>1.4 billion edges
*Twitter Graph- 2010

Georgia Institute of Technology

Georgia Tech College of Computing

Web Scale

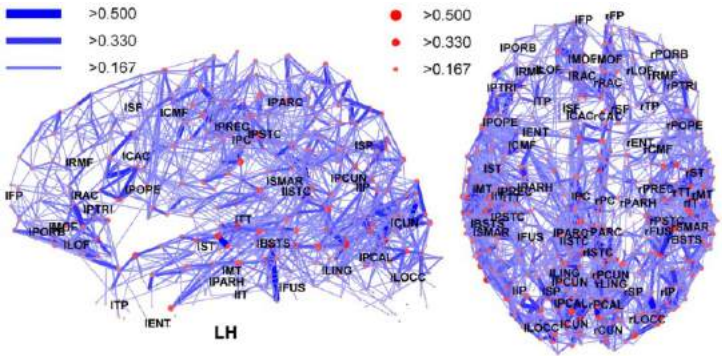




>50 billion vertices
>> 1 trillion edges
*NSA Big Graph Experiment- 2013

Georgia Institute of Technology | Georgia Tech College of Computing

Brain Scale



>100 billion vertices
>100 trillion edges
*NSA Big Graph Experiment- 2013

Georgia Institute of Technology | Georgia Tech College of Computing

Scale of Today's Graphs

in most of the Literature

- **Scale of Graphs studied in current CS literature**: on the order of billions of edges, tens of

[Paul Burkhardt, Chris Waring 2013]

Popular graph datasets in current literature

	n (vertices in millions)	m (edges in millions)	size
AS-Skitter	1.7	11	142 MB
LJ	4.8	69	337.2 MB
USRD	24	58	586.7 MB
BTC	165	773	5.3 GB
WebUK	106	1877	8.6 GB
Twitter	42	1470	24 GB
YahooWeb	1413	6636	120 GB

AS by Skitter (AS-Skitter) — internet topology in 2005 (n = router, m = traceroute)
LiveJournal (LJ) — social network (n = members, m = friendship)
U.S. Road Network (USRD) — road network (n = intersections, m = roads)
Billion Triple Challenge (BTC) — RDF dataset 2009 (n = object, m = relationship)
WWW of UK (WebUK) — Yahoo Web spam dataset (n = pages, m = hyperlinks)
Twitter graph (Twitter) — Twitter network[1] (n = users, m = tweets)
Yahoo! Web Graph (YahooWeb) — WWW pages in 2002 (n = pages, m = hyperlinks)

Big Graph: Storage and Preprocessing

- Basic Graph Storage Structure
  - **Raw Graph Dataset**
    - Vertices, Edges, Vertex State, Edge Weight
  - **Adjacency Matrix**
    - A N × N Matrix representing connectivity of a graph of size N.
  - **Adjacency List**
    - Vertex degree and vertex neighborhood (in-edges, out-edges)
  - **Index for fast access to vertex state**
  - **Index for fast access to edge state**

Iterative Graphomputation

Graph Queries, Subgraph Matching

Real World Graphs: How Big?

1 terabyte (TB) =1,024GB ~10^3GB
1 petabyte (PB) =1,024TB ~ 10^6GB
1 exabyte (EB) =1,024PB~10^9GB
1 zettabyte (ZB) =1,024EB~10^{12}GB.
1 yottabyte (YB) =1,024ZB~10^{15}GB.

Social Scale

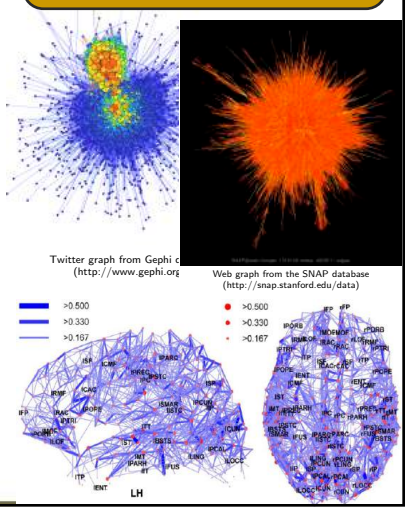
- 1 billion vertices, 100 billion edges
  - 111 PB adjacency matrix
  - 2.92 TB adjacency list

Web Scale

- 50 billion vertices, 1 trillion edges
  - 271 EB adjacency matrix
  - 29.5 TB adjacency list

Brain Scale

- 100 billion vertices, 100 trillion edges
  - 1.1 ZB adjacency matrix
  - 2.83 PB adjacency list



Top Super Computer Installations

Largest supercomputer installations do not have enough memory to process the Brain Graph (3 PB)!

- Titan Cray XK7 at ORNL — #1 Top500 in 2012
  - 0.5 million cores
  - 710 TB memory
  - 8.2 Megawatts[3]
  - 4300 sq.ft. (NBA basketball court is 4700 sq.ft.)
- Sequoia IBM Blue Gene/Q at LLNL — #1 Graph500 in 2012
  - 1.5 million cores
  - 1 PB memory
  - 7.9 Megawatts[3]
  - 3000 sq.ft.

1 terabyte (TB) =1,024GB ~10^3GB
1 petabyte (PB) =1,024TB ~ 10^6GB
1 exabyte (EB) =1,024PB~10^9GB
1 zettabyte (ZB) =1,024EB~10^{12}GB. .

Brain Scale

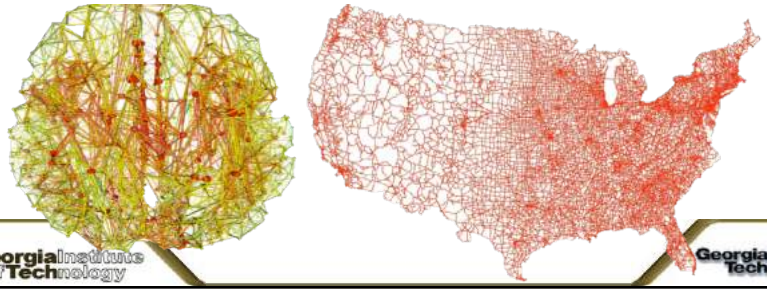
- 100 billion vertices, 100 trillion edges
  - 1.1 ZB adjacency matrix
  - 2.83 PB adjacency list

[Paul Burkhardt, Chris Waring 2013]

Big Graph Challenge:

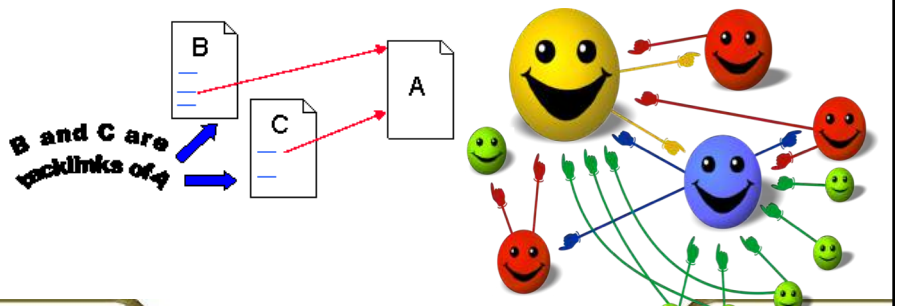
For an Algorithm working for small graph, Can it work for big graph?

BigGraph = BigData with *Structure*



Google PageRank

- Quality of a page is related to its in-degree
- Ranking pages based on links to them
  - Links from many pages → high rank
  - Link from a high-rank page → high rank

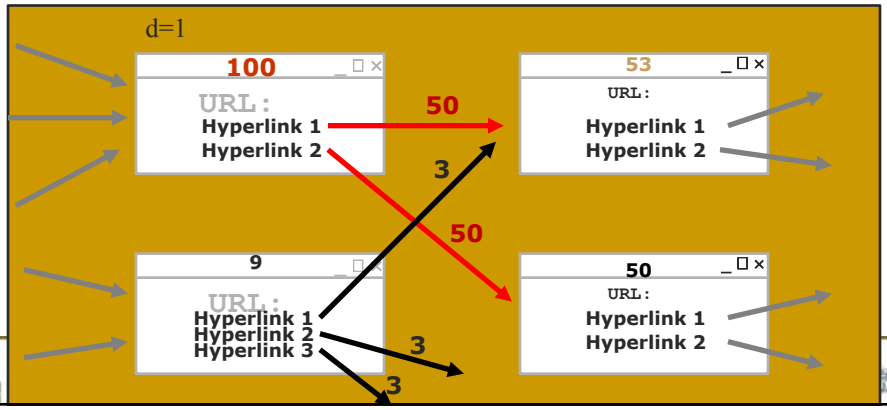


PageRank: An Example

- Simplified Definition of PageRank

$$R(u) = d \sum_{v \in B_u} \frac{R(v)}{N_v}$$

F_u : the set of pages u points to
B_u : the set of pages that point to u
N_u = | F_u |
d: damping factor (0, 1]



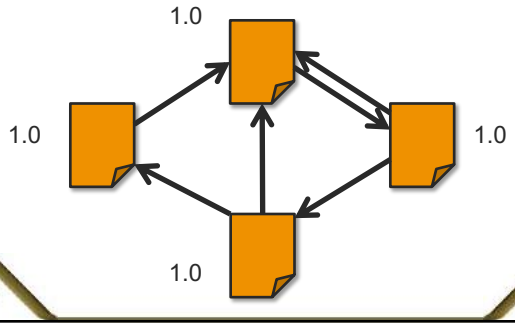
PageRank: An Example (cont.)

$$R(u) = d \sum_{v \in B_u} \frac{R(v)}{N_v} = d(R(v_1) * \frac{1}{N_{v_1}} + R(v_2) * \frac{1}{N_{v_2}} + ... + R(v_k) \frac{1}{N_{v_k}})$$

- To compute page rank for page u, we sum the weighted PageRanks of all incoming pages v_i (i=1,2…k), multiplied with a damping factor d which can be set between 0 and 1.
- By using the damping factor, the extend of PageRank benefits for a page by another page linking to it is reduced by a factor d.

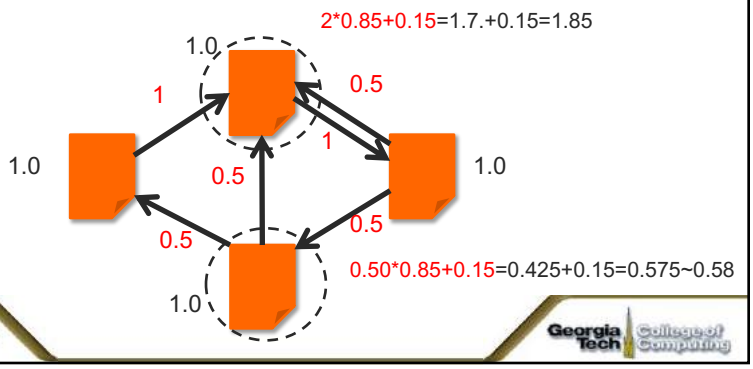
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute certain portion of its current rank ($rank_p$ / | $neighbors_p$ |) to its (outgoing) neighbors (d=0.85)
3. Set each page's rank to 0.15 + 0.85 × contribs



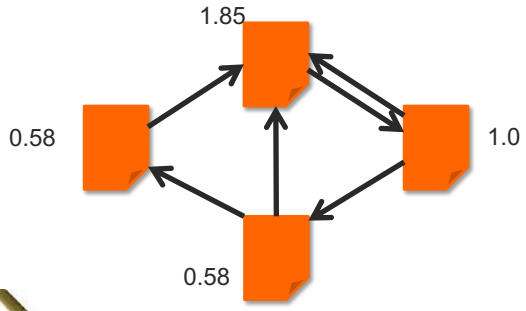
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



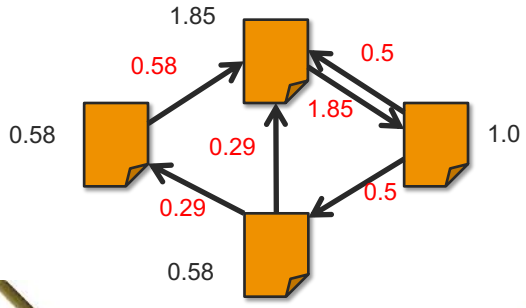
Algorithm

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



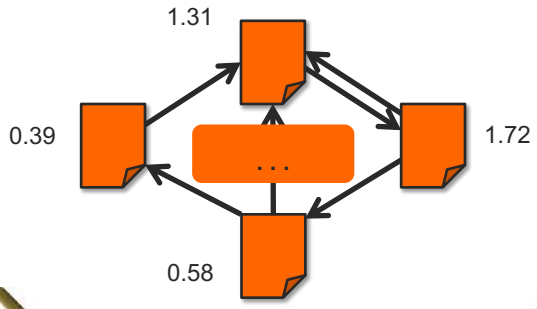
Algorithm (2nd iteration)

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



Algorithm

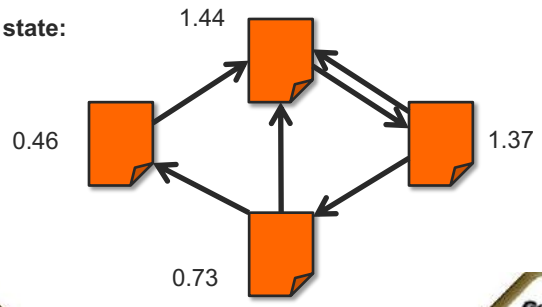
1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs



Algorithm (final state)

1. Start each page at a rank of 1
2. On each iteration, have page p contribute $rank_p$ / | $neighbors_p$ | to its neighbors
3. Set each page's rank to 0.15 + 0.85 × contribs

Final state:



What's wrong with the strawman implementation?

- No consideration on memory capacity
- No consideration on Performance
  - how to store the vertex set and the edge set to make the access to vertices and edges fast (super-efficient)? → index, cache …
- Concurrent Access and Consistency
- Recovery in the presence of hardware or software failures
- …

What's wrong with the strawman implementation?

- Assumption that a matrix representation can fit into the memory may not scale to big graph
- Problem 1: Memory Resource Constraint
  - Access graph in row by row layout in memory and on disk: no flexibility to scale
  - Strong Assumption: one row must fit into the memory at the minimum
  - Algorithm execution may incur out of memory (OOM) error
    - when a high degree vertex and its adjacency list plus intermediate result cannot fit into the available working memory

What's wrong with the strawman implementation?

- Assumption that a matrix representation can fit into the memory may not scale to big graph
- Problem 2: Poor Parallelism due to degree skewedness
  - Parallel processing using vertex by vertex (i.e., row by row layout) may lead to load imbalance
  - Vertex centric parallelization uses Strong Assumption: vertices has similar degree → not true in real graphs
  - Multi-thread execution may take much longer to run

Graph Computation: How hard can it be?

- **Algorithm complexity really matters!**
  - Run-time of $O(n^2) \sim O(n^m)$ on a trillion node graph
- **Increased size imposes greater challenge . . .**
  - ★ Latency
  - ★ resource contention
- **Difficult to parallelize (data/computation)**
  - irregular data access increases latency
  - skewed data distribution creates bottlenecks
    - high degree vertices
    - highly skewed edge weight distribution
    - giant component v.s. tiny component

Real-world Big Graphs (Highly skewed)

Graph	Type	#Vertices	#Edges	AvgDeg	MaxIn	MaxOut
Yahoo	directed	1.4B	6.6B	4.7	7.6M	2.5K
uk-union	directed	133.6M	5.5B	41.22	6.4M	22.4K
uk-2007-05	directed	105.9M	3.7B	35.31	975.4K	15.4K
Twitter	directed	41.7M	1.5B	35.25	770.1K	3.0M
Facebook	undirected	5.2M	47.2M	18.04	1.1K	1.1K
DBLPS	undirected	1.3M	32.0M	40.67	1.7K	1.7K
DBLPM	undirected	0.96M	10.1M	21.12	1.0K	1.0K
Last.fm	undirected	2.5M	42.8M	34.23	33.2K	33.2K

How do we store and process graphs?

- Conventional approach is to store and compute graph analysis in-memory.
  - **SHARED-MEMORY**
    - Parallel Random Access Machine (PRAM)
    - data in globally-shared memory
    - implicit communication by updating memory
    - fast-random access
  - **DISTRIBUTED-MEMORY**
    - Bulk Synchronous Parallel (BSP)
    - data distributed to local, private memory
    - explicit communication by sending messages
    - easier to scale by adding more machines

Possible Solution Ideas

- Problem 1
  - **What would you do if the graph and its intermediate computation results cannot fit into the available memory ?**
- Problem 2
  - **What would you do if the graph is highly skewed ?**

External Memory Solutions

- Remote Memory
  - In the same cluster
  - In the same network
- Key Innovations
  - Data Partitioning
  - Communication/Messaging cost/optimization
- Secondary Storage
  - External memory
    - PCM …
  - SSD/HDD
- Key Innovations
  - Data Placement
  - Sequential v.s. Random Access
  - Parallel processing

Execution Time: Apach Hama v.s. GraphMap

Dataset	total execution time (sec)					
	SSSP		CC		PageRank	
	Hama	Graph Map	Hama	Graph Map	Hama	Graph Map
hollywood-2011	108.776	18.347	177.854	39.365	268.474	111.466
orkut	108.744	21.345	195.841	54.383	286.054	111.46
cit-Patents	27.693	12.337	24.646	12.335	30.688	18.353
soc-LiveJournal1	48.697	18.346	60.734	33.357	75.76	39.369
uk-2005 (8.6GB)	Fail	156.49	Fail	706.329	Fail	573.964
twitter (24GB)	Fail	150.486	Fail	303.653	Fail	1492.966

12GB DRAM per node of a cluster of size 21 nodes compared to 252GB distributed shared memory

Analysis

- Hama fails for large graphs with more than 900M edges while GraphMap still works
- Note that, in all the cases, GraphMap is faster (up to 6 times) than Apach Hama

Performance Comparison

- Existing distributed graph systems are all in-memory systems.
- GraphMap uses local disk at each node of the cluster
- WebUK: 8.6GB, Twitter: 24G

System	Setting	CC (sec)		PageRank (sec per iteration)		Type
		twitter	uk-2005(*uk-2007)	twitter	uk-2005(*uk-2007)	
GraphMap on Hadoop	21 nodes (21x4=84 cores, 21x12=252GB RAM)	304	706	149	57	Out-of-core
Hama on Hadoop	21 nodes (21x4=84 cores 21x12=252GB RAM)	Fail	Fail	Fail	Fail	In-memory
GraphX on Spark	16 nodes (16x8=128 cores 16x68=1088GB RAM)	251	800*	21	23*	In-memory
GraphLab 2.2 (PowerGraph)	16 nodes (16x8=128 cores 16x68=1088GB RAM)	244	714*	12	42*	In-memory
Giraph 1.1 on Hadoop	16 nodes (16x8=128 cores 16x68=1088GB RAM)	200	Fail*	30	62*	In-memory
Giraph++ on Hadoop	10 nodes (10x8=80 cores 10x32=320GB RAM)	No result reported	723	No result reported	89	In-memory

5x DRAM per node

- GraphMap: 12GB DRAM per node of a cluster of size 21 nodes and 252GB distributed shared memory

Example Problems: A Summary

- Problem 1
  - **What would you do if the graph and its intermediate computation results cannot fit into the available memory ?**
- Problem 2
  - **What would you do if the graph is highly skewed ?**

What's wrong with the simple strawman implementation?

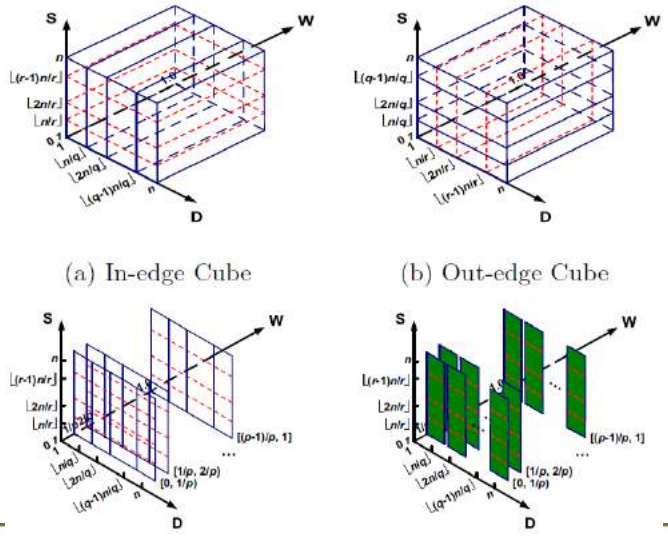
- Lack of graph abstractions that promote Parallel Computation
  - Some vertex has very short computation time in each iteration and other vertex has relatively long computation time
  - CPU time is very short compared to data access time
  - …

Skewedness: Novel Data Structures

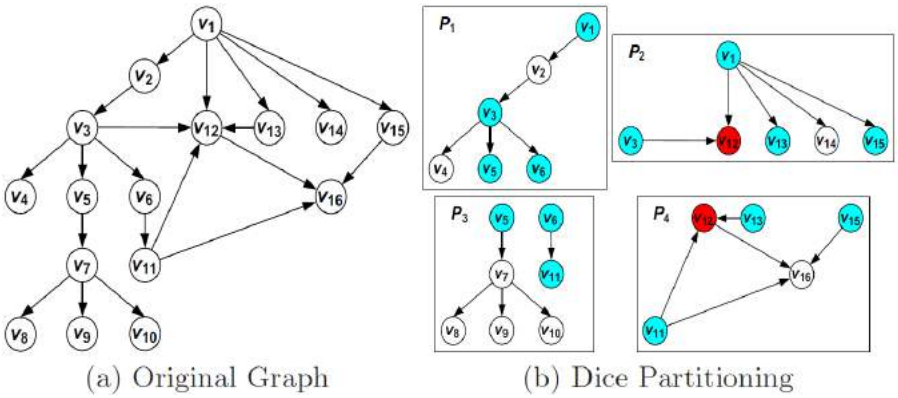
- Vertex Degree Skewedness
- Vertices have skewed neighbors (both in-edge and out-edge)
- Edge Weight Skewedness
- Some edges have tiny weights and others significant weights

Multi-level Hierarchical Graph Parallel Abstractions

[Zhou,Liu VLDB2015]



Dice Partitioning: An Example

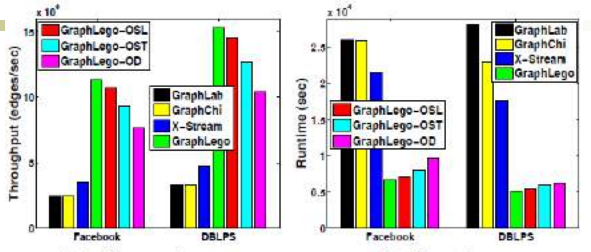


SVP v1,v2,v3, v5,v6,v7, v11, v12, v15

DVP: v2,v3,v4,v5,v6, v7,v8,v9,v10,v11, v12,v13,v14,v15.v16

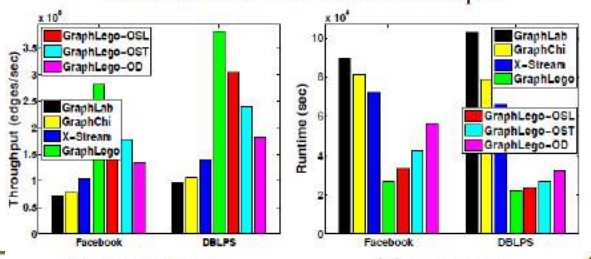
[Zhou,Liu VLDB2015]

Execution Efficiency on Multiple Graphs



(a) Throughput (b) Runtime

Diffusion Kernel on Two Real Graphs



(a) Throughput (b) Runtime

Inc-Cluster on Two Real Graphs

[Zhou,Liu VLDB2015]

Parallel Graph Processing @ DiSL, Georgia Tech

- Developing Multi-tier Graph Parallel Abstractions
  - Top Tier: Subgraph centric
  - Bottom Tier: : Vertex centric or Edge centric
- **New Frameworks for Parallel Graph Processing**
  - Single Machine Solutions
    - GraphLego [ACM HPDC 2015] → Parallel Abstractions
    - GraphTwist [VLDB2015] → Two Tier Optimizations
  - Distributed Approaches
    - GraphMap [IEEE SC 2015]
    - PathGraph [IEEE SC 2014], joint with Dr. P.Yuan@HUST





GPUs | Multicore | Clusters | Science Clouds | Commercial Clouds

Other Complications:

- API
  - No flexible and easy to use and standardized API for graph algorithms
- Concurrency
  - No concurrency control
  - Single user runtime, one job on a big graph dataset each time
  - Open problem: How can multiple users access / modify a big graph file at the same time consistently
- Recovery and Reliability
  - Can lose data in a crash, expensive, start multi-iteration job from scratch today!
  - Can leave operations half done → how to recover in the presence of failure?
- Security + Privacy

Other Complications:

- Small Graphs v.s. Big Graphs
  - Small graph: can fit whole graph and its processing in main memory
  - Big Graph: cannot fit the whole graph and its processing in main memory
- Simple Graphs v.s. Multigraphs
  - **Simple graph**: allow only one edge per pair of vertices
  - **Multigraph**: allow multiple parallel edges between pairs of vertices
- Singe Graph Computation v.s. Multiplication of Graphs

Graph Processing Challenges

- **Two broad classes of problems:**
  - **Graph queries** to find matching subgraphs
  - **Iterative Algorithms** to find clusters, orderings, paths, patterns, …
- **Graph Kernel**
  - traversal, shortest path algorithms, flow algorithms, spanning tree algorithms, topological sort, …
- **Many factors can influence the choices of graph analytic algorithms and performance optimization techniques**
  - graph sparsity (edge/vertex ratio), Static/dynamic, diameter, graph heterogeneity,
  - weighted/unweighted (and weight distribution), vertex degree distribution,
  - directed/undirected, simple/multi/hyper graph,
  - problem size, granularity of computation (vertices/edges), problem-specific or domain specific characteristics

Technical Challenges in Big Data Services

Three Case Studies:

- Big Graph Processing
- Discrete Optimization Problems
  - Need rethinking of how the software defined intelligence is delivered
- **Deep Learning → next lecture**
  - Need innovation in how deep neural networks should be constructed in neurons, stackable deep learning layers and their interactions with flexibility, adaptablity, customizability, and extensibility

Discrete Optimization Problems: Some Examples you know

- ❑ **Finding top k most influential people in a large social network**
- ❑ **Finding top k most interesting blogs from sea of blogs**
- ❑ **Finding top k best matches of keyword search**
- ❑ **Finding top k locations to place sensors to gain the best coverage**
- ❑ **…**

(1) Discrete Optimization Problems

finding the right information

Finding the Most influential people

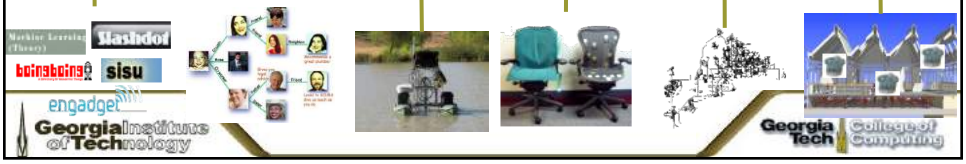
monitoring algal blooms

predicting postures

detecting contaminations

reducing energy consumption

each app requires its own utility function with the same greedy algo & optimization
→ Implementing case by case is time consuming, doesn't scale! ☹



Example: Discrete Optimization Problems

- **Social Influence Analysis**
  - **How to discover the most influential k people for marketing of a product/an idea/an innovation on a social network?**
  - Want maximum effect of advertisement **with limited marketing budget**
- **Sensor placements**
  - Maximize information coverage with budget constraints (Krause *et al.* 2005a)
  - Maximize information coverage at minimum communication cost (Krause *et al.* 2006)
  - **Example: Where should we place sensors to quickly detect fresh water contamination?**
    - Water sensors are expensive and only be placed in limited locations on the network
    - Want get most useful information **at lowest cost**

Find top K most influential people in a social network

Given: a finite set V of people in a social network graph, and the **social influence utility function F**

Want: find a subset A of V such that

NP-hard!

$$\mathcal{A}^* \subseteq \mathcal{V}$$
$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})$$

We can use Greedy algorithm to get an approximation:

how well can this simple heuristic do?

Greedy algorithm:

Start with A = {}

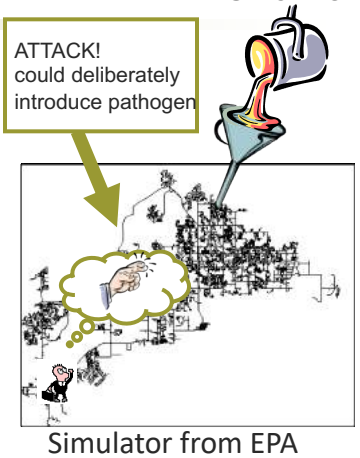
For i = 1 to k

s* := argmax_s F(A ∪ {s})

A := A ∪ {s*}

Water distribution networks

- Water distribution in a city → very complex system
- Pathogens in water can affect thousands (or millions) of people
- Currently: Add chlorine to the source and hope for the best



Simulator from EPA

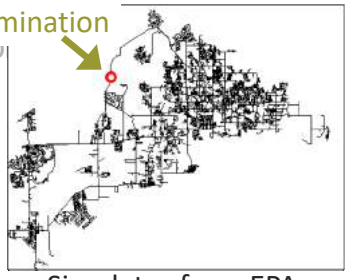
Source: Guestrin, Krause@IJCAI 2009

Example: Sensor Placement

[Leskovec, Guestrin, VanBriesen, Faloutsos, J Wat Res Mgt '08]

Contamination of drinking water could affect millions of people

Contamination



Simulator from EPA

Hach Sensor

~$14K

Place sensors to detect contaminations

"Battle of the Water Sensor Networks" competition

Where should we place sensors to quickly detect contamination?

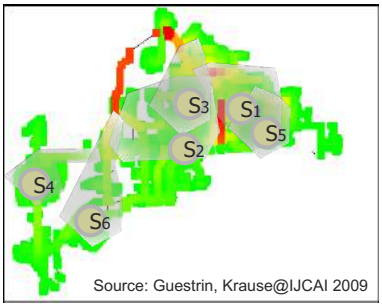
Sensor placement

Given: finite set V of locations, sensing quality F

Want:

$$\mathcal{A}^* \subseteq \mathcal{V} \quad \text{such that}$$
$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})$$

NP-hard!



Source: Guestrin, Krause@IJCAI 2009

Greedy algorithm:

Start with A = {}

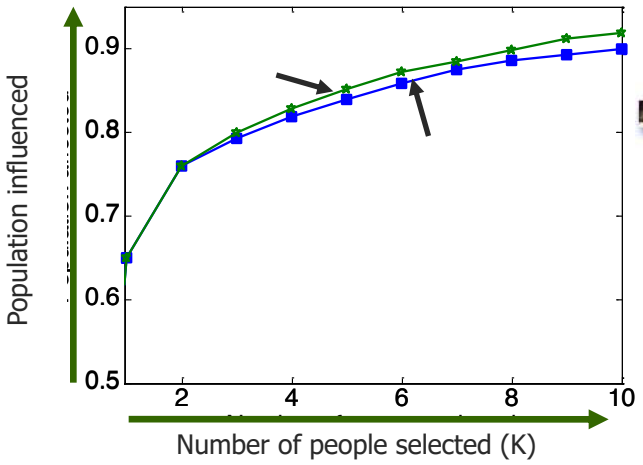
For i = 1 to k

s* := $\operatorname{argmax}_s F(A \cup \{s\})$

A := A ∪ {s*}

How well can this simple heuristic do?

Performance of greedy algorithm



Small subset of a social network

Why?

Greedy score empirically close to optimal if F is monotonic and submodular.

Theorem on submodularity

Theorem [Nemhauser et al '78]

Suppose F is *monotonic* and *submodular*. Then greedy algorithm gives constant factor approximation: $F(A_{greedy}) \geq \underbrace{(1-1/e)}_{\sim 63\%} \max_{|A| \leq k} F(A)$

- Greedy algorithm gives near-optimal solution!
- In general, guarantees best possible unless P = NP!

G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, Maximizing submodular set functions, *Mathematical Programming*, 1978.

Building a sensing chair

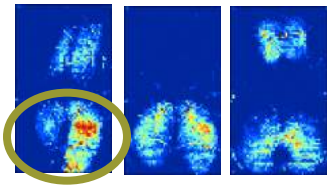
[Mutlu, Krause, Forlizzi, G., Hodgins '07]

- People sit a lot
- Activity recognition in assistive technologies
- Seating pressure as user interface

equipped with 1 sensor per cm^2!

sheet costs $6,000! ☹

can we get similar accuracy with fewer, cheaper sensors?

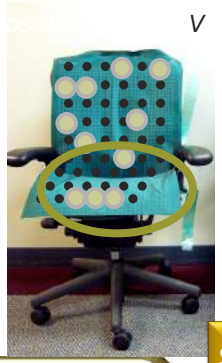


82% accuracy on 10 postures! [Tan et al]

How to place sensors on a chair?

- Predict posture Y; Possible locations V
- Goal: **minimize uncertainty** in prediction ⇔ **maximize information gain** (IG):

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} \mathrm{IG}(Y; \mathcal{A})$$



Theorem: information gain is submodular!* [Krause, G. '05]

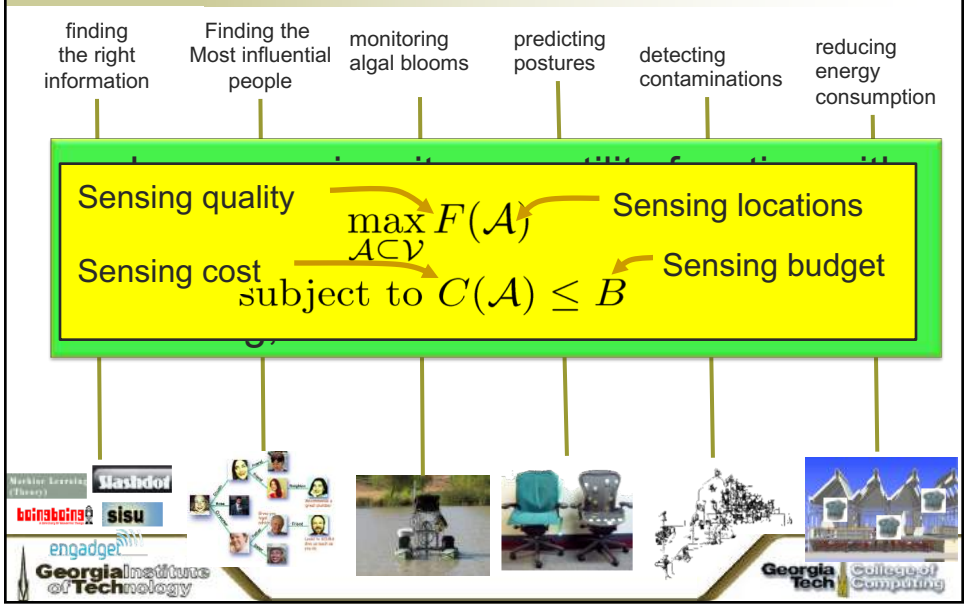
	Accuracy	Cost
Before	82%	**$6,000** ☹
After		

random placement: 53%; uniform Placement: 73%

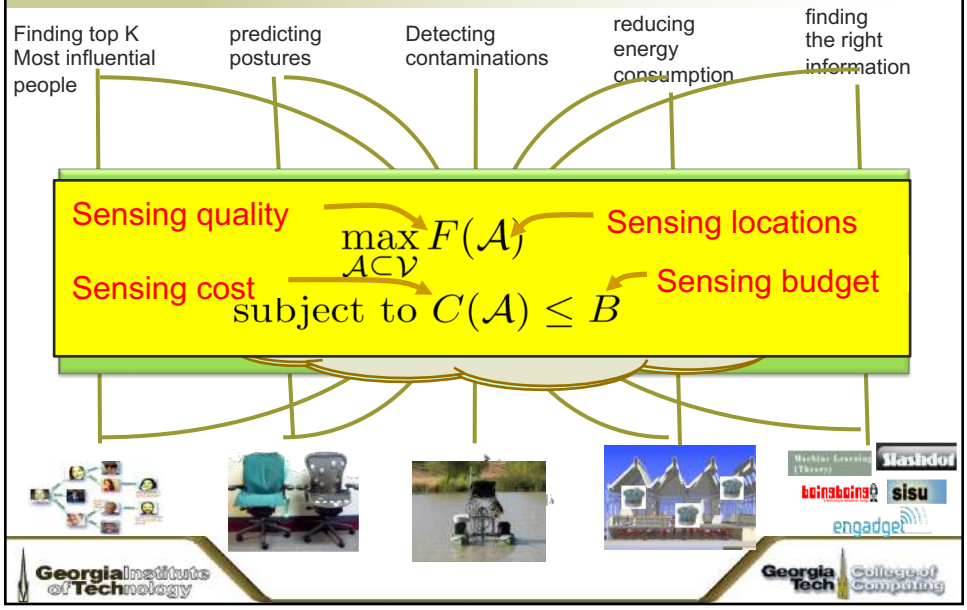
similar accuracy at <2% of cost!

Source: [Mutlu, Krause, Forlizzi, G., Hodgins '07]

Analytics As a Service: Discrete Optimization



Data analysis as a service: the quest for programmable algorithm abstractions



Technical Challenges in Big Data Services

Three Case Studies:

- Big Graph Processing
- Discrete Optimization Problems
  - Need rethinking of how the software defined intelligence is delivered
- **Deep Learning**
  - Need innovation in how deep neural networks should be constructed in neurons, stackable deep learning layers and their interactions with other AI/Search Algorithms for real world applications

Georgia Institute of Technology | Georgia Tech College of Computing

ImageNet

http://image-net.org (2009)

- Over 15M labeled high resolution images
- Roughly 22K categories
- Collected from web and labeled by Amazon Mechanical Turk

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC): software programs compete to correctly classify and detect objects and scenes

Vision is so critical to how we understand the world (Human's perspective)
It's hard to imagine any intelligent computer of the future without it
(Human's perception of how machine can learn)

Georgia Institute of Technology | Georgia Tech College of Computing

Datasets and computer vision

UIUC Cars (2004)
S. Agarwal, A. Awan, D. Roth

CMU/VASC Faces (1998)
H. Rowley, S. Baluja, T. Kanade

FERET Faces (1998)
P. Phillips, H. Wechsler, J. Huang, P. Raus

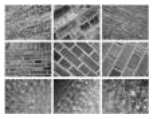
COIL Objects (1996)
S. Nene, S. Nayar, H. Murase

MNIST digits (1998-10)
Y LeCun & C. Cortes

KTH human action (2004)
I. Leptev & B. Caputo

Sign Language (2008)
P. Buehler, M. Everingham, A. Zisserman

Segmentation (2001)
D. Martin, C. Fowlkes, D. Tal, J. Malik.



3D Textures (2005)
S. Lazebnik, C. Schmid, J. Ponce

CuRRET Textures (1999)
K. Dana B. Van Ginneken S. Nayar J. Koenderink

CAVIAR Tracking (2005)
R. Fisher, J. Santos-Victor J. Crowley

Middlebury Stereo (2002)
D. Scharstein R. Szeliski

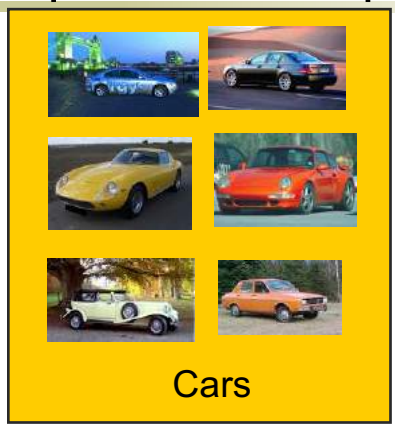
Storage Challenges

- A large amount of Data
  - ImageNet (update on April 30, 2010)
    - 21,841 categories
    - **14,197,122 images**
    - **Growing**
- Estimation of ImageNet data size
  - Storage: compressed format, e.g., jpg (Typical size: 150 KB)
    - **14,197,122 x 150 KB = 1.5 TB (Storage)**
  - In-memory: matrix format (Typical size: 1MB)
    - **14,197,122 x 1 MB = 13.54 TB (In-memory)**

ImageNet In practice (**Less than 1/10**)
Only a small subset of ImageNet is widely used:
1,000 categories, 1,281,167 images, **140GB**

[Yanzhao Wu, Ling Liu et al 2019]

Supervised Deep learning

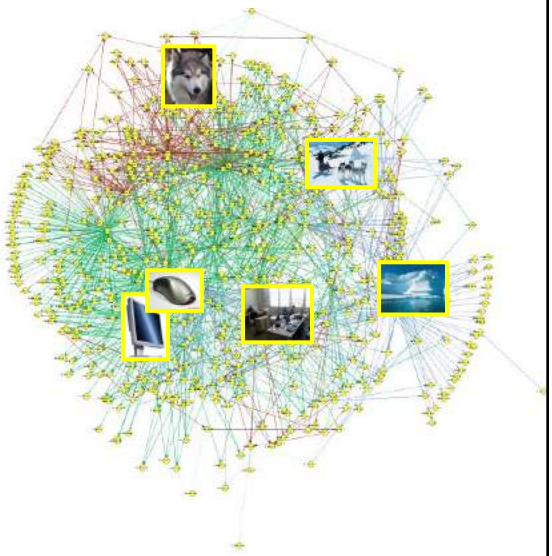


Georgia Institute of Technology

Georgia Tech College of Computing

IMAGENET
is a knowledge ontology

- Taxonomy
- Partonomy
- The "social network" of visual concepts
  - **Prior knowledge**
  - **Context**
  - Hidden knowledge and structure among visual concepts

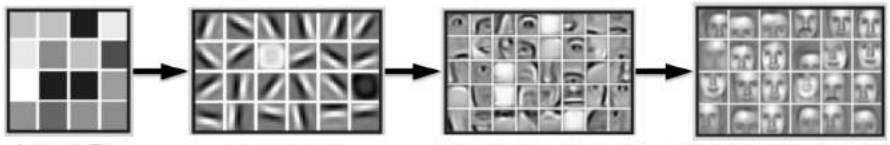


Georgia Institute of Technology

Tech

Enabling the machine to learn the feature representations from data autonomously

Deep-learning neural networks use layers of increasingly complex rules to extract and learn feature hierarchies and categorize complex features, e.g., shapes such as faces, in multi-steps and multi-iterations.



Layer 1: The computer identifies pixels of light and dark.

Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.

pixels

edges

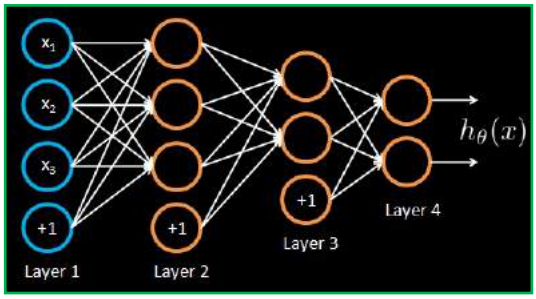
object parts (combination of edges)

object models

N. Jones "The Learning Machines," Nature, 2014

Training a neural network

Example 4 layer neural network with 2 output units:



Given training set $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots$.

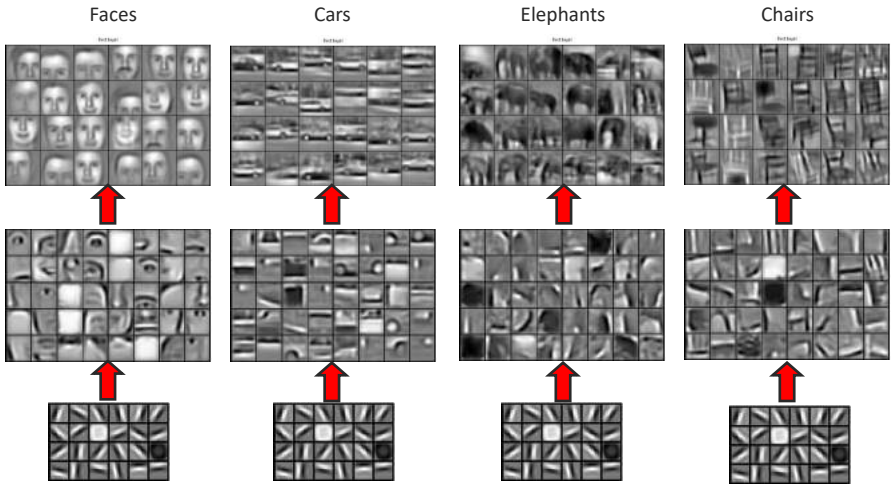
Adjust parameters θ (for every node) to make: $h_\theta(x_i) \sim y_i$

$h_\theta(x_i) \approx y_i$

(Use gradient descent. "Backpropagation" algorithm. Susceptible to local optima.)

Courtesy: Andrew Ng

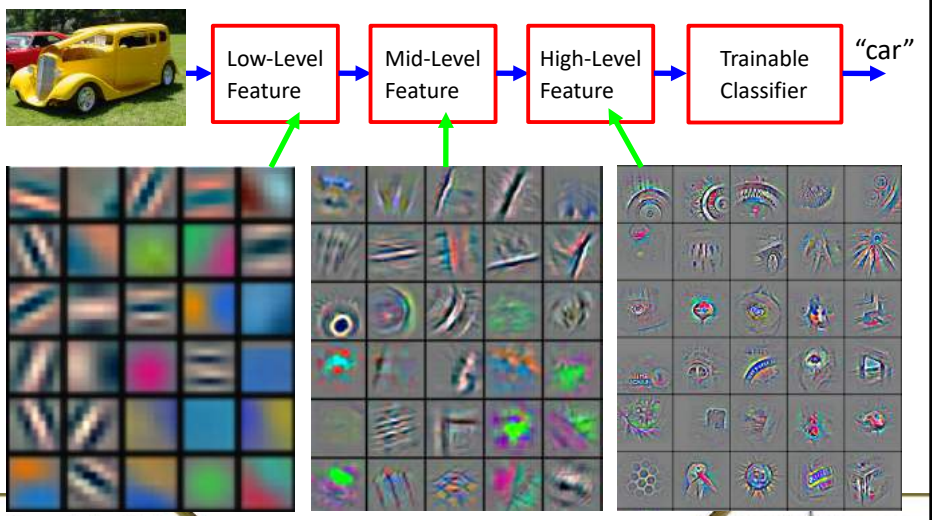
Learning of object parts



Georgia Institute of Technology

Georgia Tech College of Computing

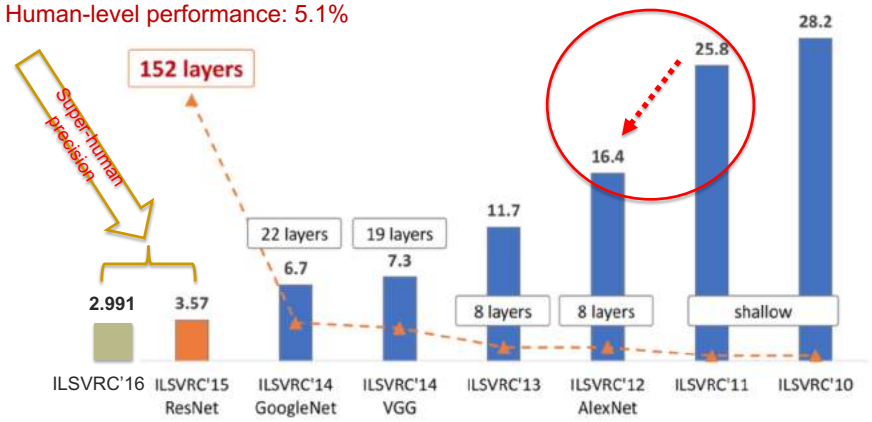
Deep Learning = Hierarchical Compositionality



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

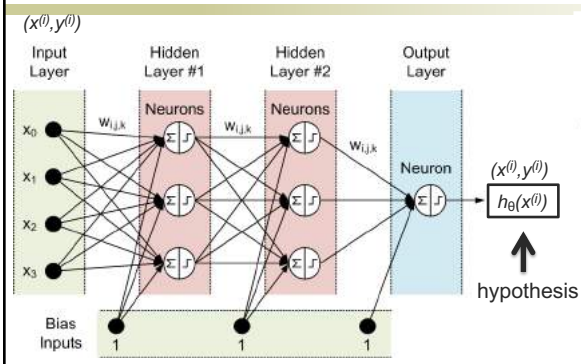
Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

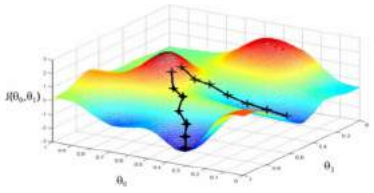
Achieving Human-Level Performance on ImageNet Classification



https://www.researchgate.net/figure/The-evolution-of-the-winning-entries-on-the-ImageNet-Large-Scale-Visual-Recognition_fig1_321896881

Deep Feed Forward Neural Networks





Here is the algorithm:

Learninag rat

```
Repeat until convergence {
    Wj = Wj - λ ∂F(Wj)/∂Wj
}
```

derivative w.r.t. w_j

Given training set (x_1, y_1), (x_2, y_2), (x_3, y_3), …. Learning is adjusting parameters θ (for every node), i.e., adjusting the $w_{i,j}$'s such that the cost function $J(\boldsymbol{\theta})$ is minimized, and to make: $h_\theta(x_i) \sim y_i$

$$J(\theta_0,\theta_1,\ldots,\theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

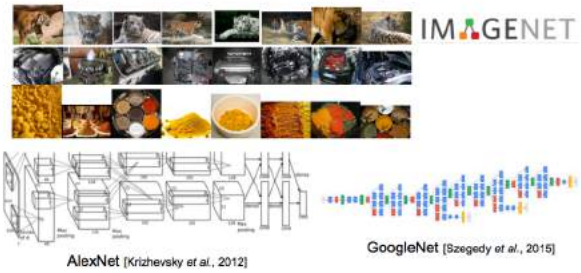
$$E(W) = \frac{1}{2}(\hat{y} - y)^2$$

$$\Delta w_i = \alpha \frac{\partial E}{\partial w_i}$$

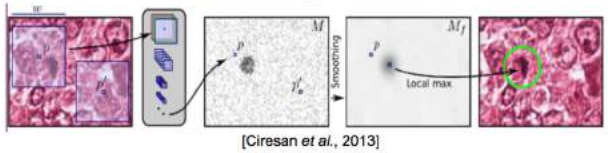
$$w_{i,t+1} = w_{i,t} + \Delta w_i$$

Deep Neural Network

- Object recognition
  - Accuracy close to or better than human performance: 5.1%



- Mitosis detection in Breast Cancer Histology Images
  - Win the MICCAI Grand Challenge 2013



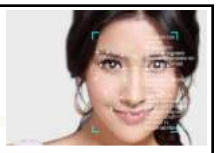
[Ciresan et al., 2013]

source: Le

Georg

College of Computing

Deep Neural Networks: Initial Success in Practice



Face Recognition

Speech data

Siri by Apple 2010

Image data

Amazon Go

Schema data

AlphaGo Google 2015

Video data

Self-Driving By Uber, Tesla

Georgia Institute of Technology

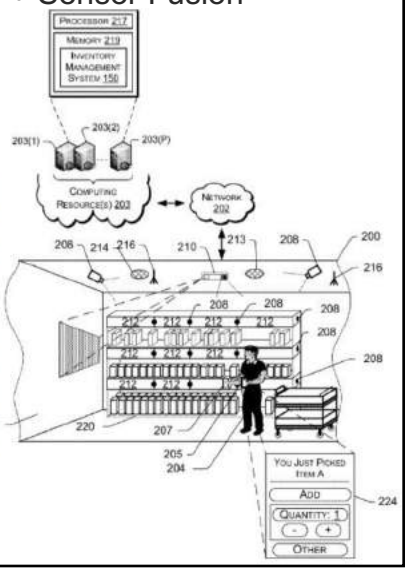
Georgia Tech College of Computing

Integrating Deep Learning with Other Computing Technologies

- *Amazon Go*
  - Deep Learning + Computer Vision + Sensor Fusion + HPC
- *Google Alpha Go*
  - Deep Learning + Computer Vision + Monte Carlo Tree Search + Supervise Learning + Reinforcement Learning + HPC
- *Big Graph*
  - *Graph/network/knowledge embedding (if time permits)*

Amazon Go

- Deep Learning + Computer Vision + Sensor Fusion Powered Shopping Experience



Amazon Go

- Deep Learning + Computer Vision + Sensor Fusion Powered Shopping Experience

Alpha Go (Google)

[Alan Fern, OSU, Alpha-go]

- Deep Learning + Monte Carlo Tree Search + HPC + Powered by Big Data (huge dataset)

Deep Mind's AlphaGo vs. Lee Sedol (2016)

AlphaGo

- Deep Learning + Monte Carlo Tree Search + HPC
- Learn from 30 million expert moves and self play
- Highly parallel search implementation
- 48 CPUs, 8 GPUs (scaling to 1,202 CPUs, 176 GPUs)

March 2016 :
AlphaGo beats Lee Sedol 4-1

Mastering the game of Go with deep neural networks and tree search Nature, 529, January 2016.

Evolution of Machine Intelligence

(an example)

Garry Kasparov vs. Deep Blue (1997)

Watson vs. Ken Jennings (2011)

Deep Mind's AlphaGo vs. Lee Sedol (2016)

March 2016 : AlphaGo beats Lee Sedol 4-1

A Brief History of Computer Go

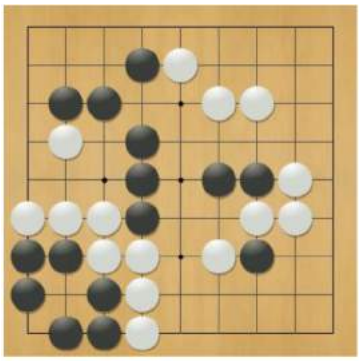
[Alan Fern, OSU, Alpha-go]

- 1997: Super human Chess w/ Alpha-Beta + Fast Computer
- 2005: Computer Go is impossible!

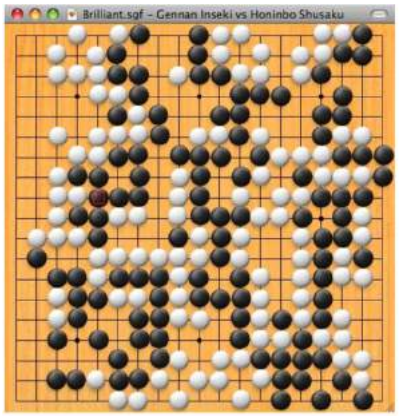
- 2006: Monte-Carlo Tree Search applied to 9x9 Go (bit of learning)
- 2007: Human master level achieved at 9x9 Go (bit more learning)
- 2008: Human grandmaster level achieved at 9x9 Go (even more)

- 2012: Zen program beats former international champion Takemiya Masaki with only 4 stone handicap in **19x19**
- 2015: DeepMind's AlphaGo Defeats European Champion 5-0 (lots of learning)
- 2016 (march): AlphaGo beats Lee Sedol 4-1 (lots of learning + huge dataset)
  - Learn from 30 million expert moves and self play
  - Highly parallel search implementation
  - 48 CPUs, 8 GPUs (scaling to 1,202 CPUs, 176 GPUs)

Why is Computer Go a grand challenge task for AI?



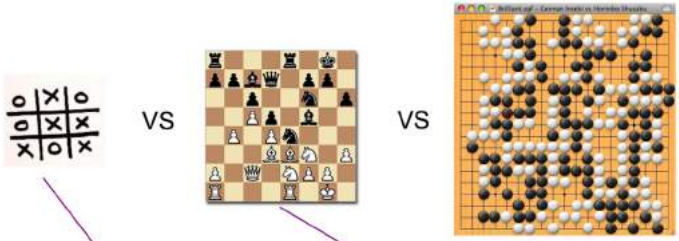
9x9 (smallest board)

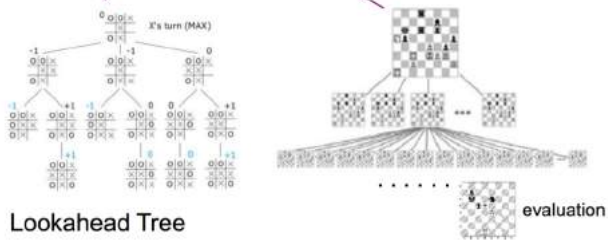


19x19 (standard board)

[Alan Fern, OSU, Alpha-go]

Why is Computer Go a grand challenge task for AI?

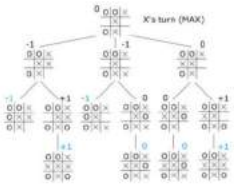




Lookahead Tree

[Alan Fern, OSU, Alpha-go]

Why is Computer Go a grand challenge task for AI?



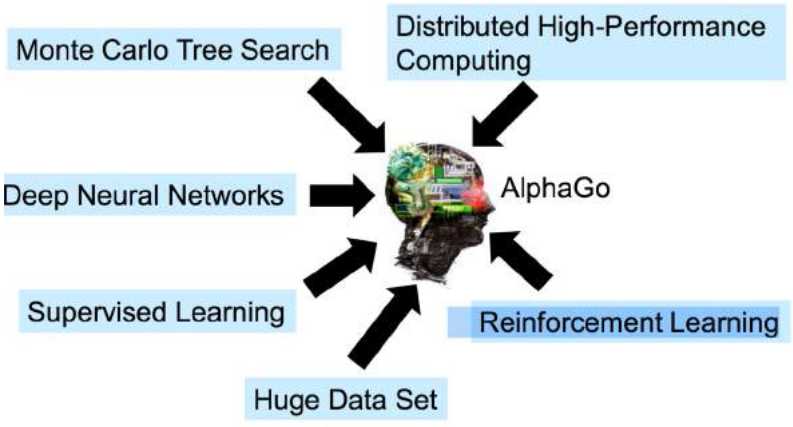
MiniMax Tree

- Branching Factor
  - Chess ≈ 35
  - Go ≈ 250
- Required search depth
  - Chess ≈ 14
  - Go ≈ much larger
- Leaf Evaluation Function
  - Chess – good hand-coded function
  - Go – no good hand-coded function

[Alan Fern, OSU, Alpha-go]

[Alan Fern, OSU, Alpha-go]

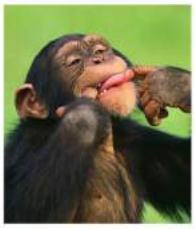
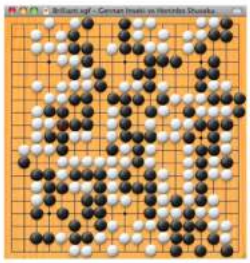
Arsenal of AlphaGo



Mastering the game of Go with deep neural networks and tree search
Nature, 529, January 2016.

Monte Carlo Tree Search

Idea #1: board evaluation function via random rollouts

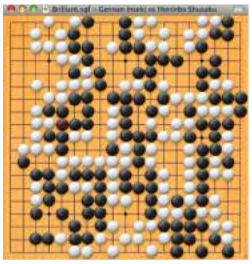

Evaluation Function:
- play many random games
- evaluation is fraction of games won by current player
- surprisingly effective

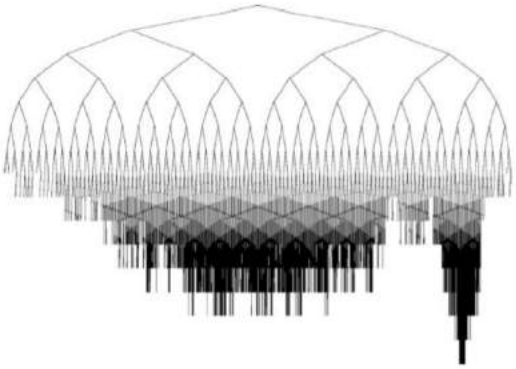
Even better if use rollouts that select better than random moves

[Alan Fern, OSU, Alpha-go]

Monte Carlo Tree Search

Idea #2: selective tree expansion





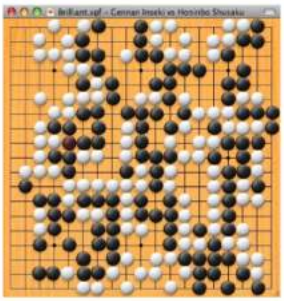
Non-uniform tree growth

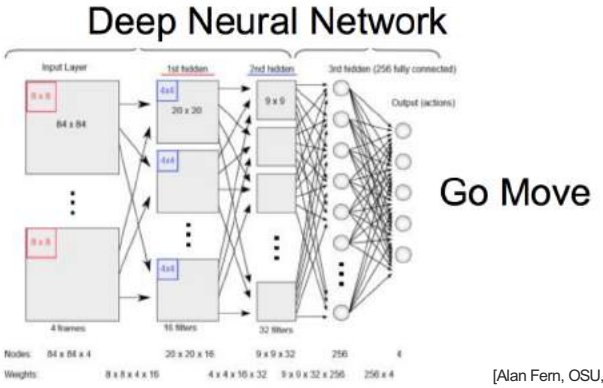
[Alan Fern, OSU, Alpha-go]

DNN Learning to Predict Good Moves

State-of-the-Art Performance: very fast GPU implementations allow training giant networks (millions of parameters) on massive data sets

Could a Deep NN learn to predict expert Go moves by looking at board position? Yes!
Idea: treat Go board as an image—use modern computer vision

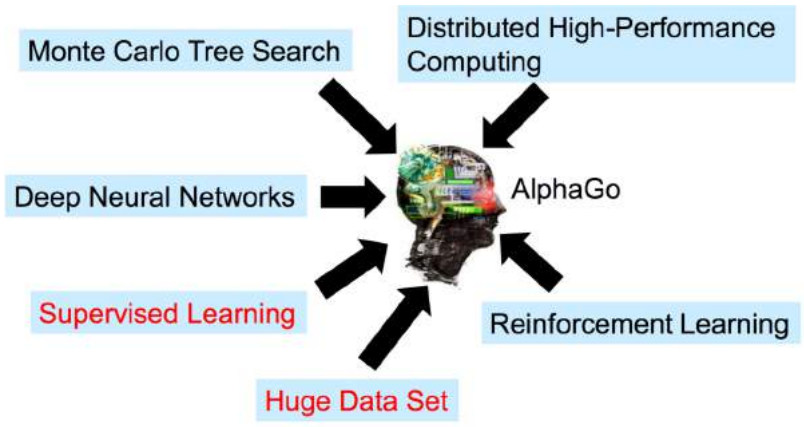




[Alan Fern, OSU, Alpha-go]

[Alan Fern, OSU, Alpha-go]

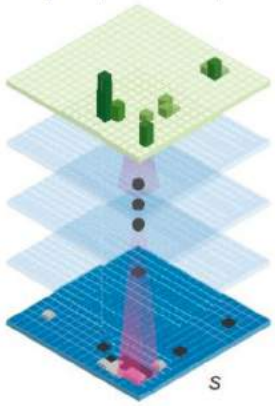
Arsenal of AlphaGo



Mastering the game of Go with deep neural networks and tree search
Nature, 529, January 2016.

Supervised Learning for Go

Output: probability of each move



Deep NN Internal Layers

Trained for 3 weeks on 30 million expert moves

- 57% prediction accuracy!

Input: Board Position

[Alan Fern, OSU, Alpha-go]

[Alan Fern, OSU, Alpha-go]

Arsenal of AlphaGo

Monte Carlo Tree Search

Distributed High-Performance Computing

Deep Neural Networks

AlphaGo

Supervised Learning

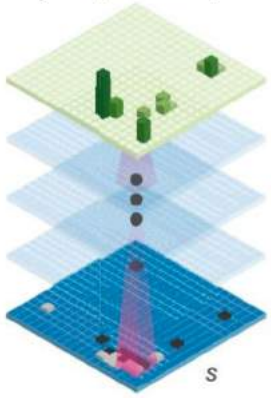
Reinforcement Learning

Huge Data Set

Mastering the game of Go with deep neural networks and tree search
Nature, 529, January 2016.

Reinforcement Learning for Go

Output: probability of each move

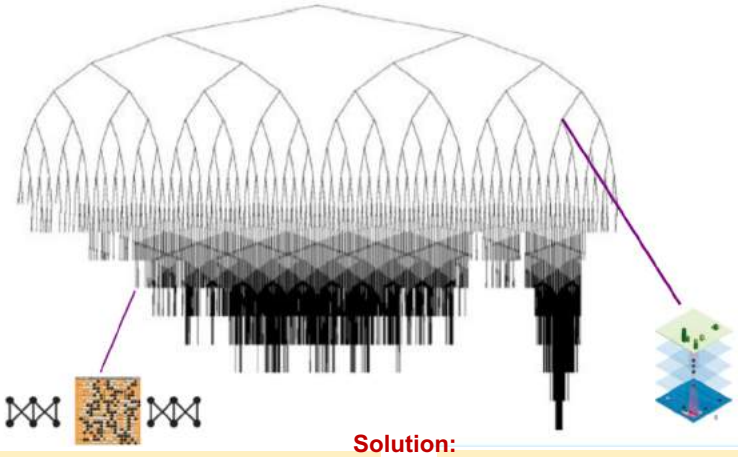


Input: Board Position

- Start with Deep NN from supervised learning.
- Continue to train network via self play.
- AlphaGo did this for months.
- 80% win rate against the original supervised Deep NN
- 85% win rate against best prior tree search method!
- Still not close to professional level

[Alan Fern, OSU, Alpha-go]

Monte Carlo Tree Search



Solution:

use smaller networks (less accurate but fast)

Use expensive network to guide tree expansion

[Alan Fern, OSU, Alpha-go]

Alpha Go: Summary

- Deep Learning + Monte Carlo Tree Search + HPC
- Learn from 30 million expert moves and self play
- Highly parallel search implementation
- 48 CPUs, 8 GPUs (scaling to 1,202 CPUs, 176 GPUs)

2015 :
AlphaGo beats European Champ (5-0)

lots of self play

March 2016 :
AlphaGo beats Lee Sedol (4-1)

[Alan Fern, OSU, Alpha-go]

Big Data Driven Deep Learning

- The idea of combining search with learning is very general and widely applicable
- Deep Networks are leading to advances in many areas of Big Data Analytics and machine intelligence
  - ComputerVision
  - SpeechProcessing
  - NaturalLanguageProcessing
  - Bioinformatics
  - Robotics
- It is a very exciting time to be working in Big Data Intelligence

[Alan Fern, OSU, Alpha-go]