Hardware-Aware Static Optimization of Hyperdimensional Computations

Pu (Luke) Yi and Sara Achour

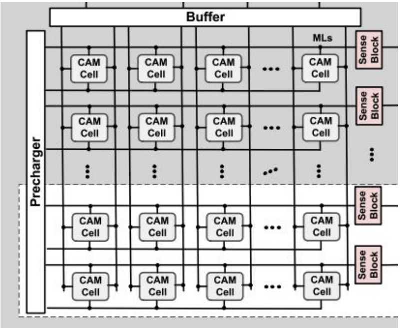
SystemX ALLIANCE
Stanford University

ACCESS
AI Chip Center for Emerging Smart Systems
智能晶片與系統研發中心

Emerging Hardware Technologies

Emerging hardware technologies promise to revolutionize computation

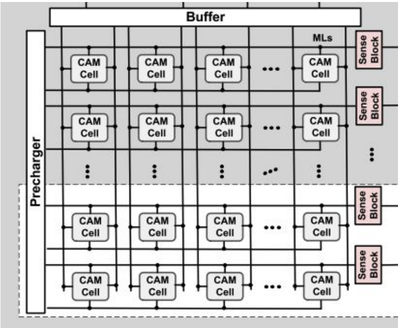
- Improved performance and energy consumption

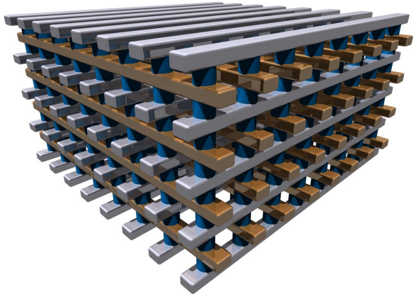


Emerging Hardware Technologies

Emerging hardware technologies promise to revolutionize computation

- Improved performance and energy consumption
- Higher storage density and faster memory access times

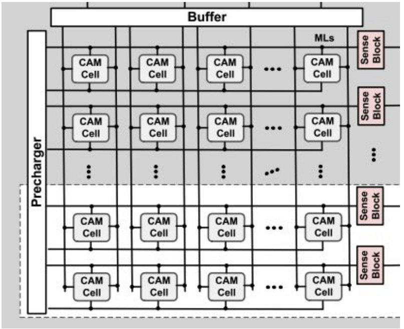


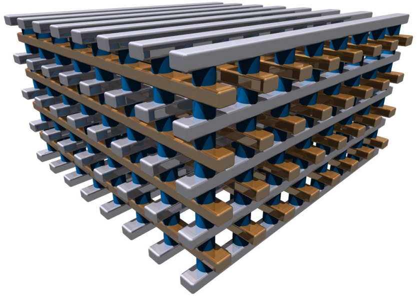


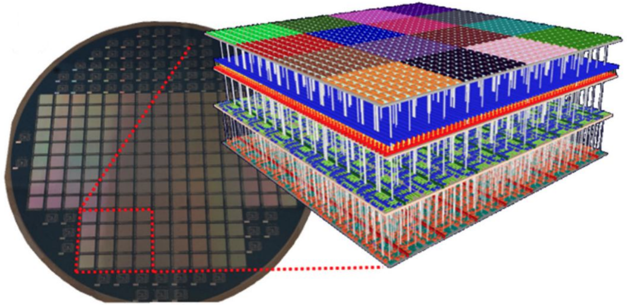
Emerging Hardware Technologies

Emerging hardware technologies promise to revolutionize computation

- Improved performance and energy consumption
- Higher storage density and faster memory access times
- Dense 3D interconnect; significantly higher bandwidths



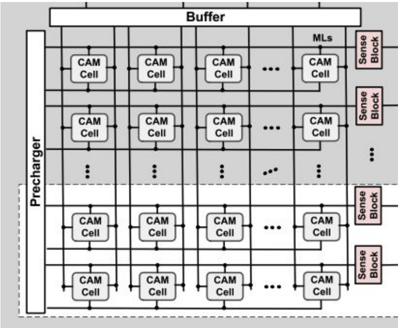


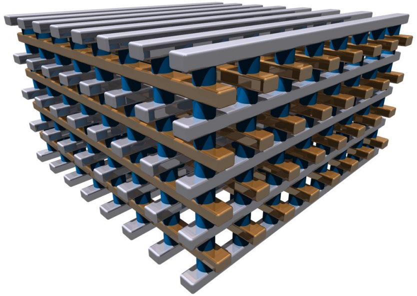


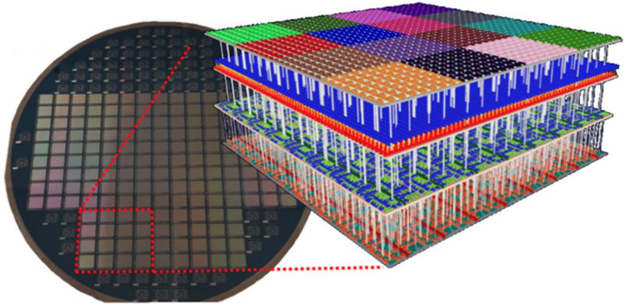
Emerging Hardware Technologies

These emerging technologies are highly promising, but are less reliable than conventional memory/compute substrates and occasionally corrupt bits.

Why does this occur? Conformational changes in materials, static errors from immature fabrication processes, sensitivities to the environment.

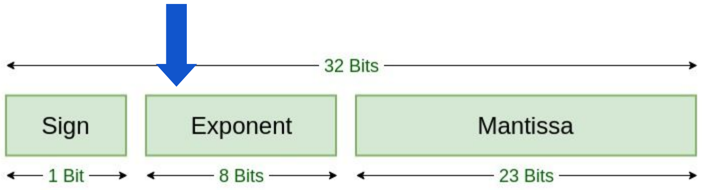


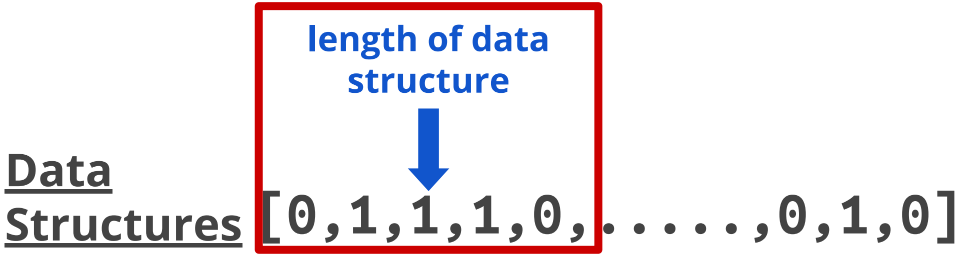




Performing Computation on Emerging Hardware Technologies

Performing classical computation on these hardware substrates is challenging because *where* corruptions occur in the program & the program data has a substantial impact on the computed result.





In classical computation, some bits are more important than others...

Classical computations are highly sensitive to bit corruptions if the "wrong" bits are corrupted.

In classical computation, some bits are more important than others...

Classical computations are highly sensitive to bit corruptions if the "wrong" bits are corrupted.

Over the years a number of error mitigation techniques have been developed to work around the problem.

E.g., error-correcting codes, precise/approximate data partitioning, redundant computation.

In classical computation, some bits are more important than others...

Classical computations are highly sensitive to bit corruptions if the "wrong" bits are corrupted.

These mitigations introduce hardware and software overheads that affect projected energy/performance improvements.

In classical computation, some bits are more important than others...

Classical computations are highly sensitive to bit corruptions if the "wrong" bits are corrupted.

What else can be done? We can change *how* we perform computation.

In classical computation, some bits are more important than others…

Classical computations are highly sensitive to bit corruptions if the "wrong" bits are corrupted.

What else can be done? We can change *how* we perform computation.

This work focuses on *hyperdimensional computation*, a computational model that is naturally resilient to error.

What is hyperdimensional computation?

Hyperdimensional computing (HDC) is a highly error-resilient novel computational paradigm that originated from the cognitive science community.

[0,1,1,1,0,..........,0,1,0]

The basic unit of information is a *hypervector*, a high-dimensional binary vector.[1]

1. This talk overviews Binary Spatter Code (BSC), a variant of HDC that works with dense binary hypervectors.

What is hyperdimensional computation?

In HD computing, information is evenly distributed across bits, so all bits are equally important (or unimportant) to the computation.

[0,1,1,1,0,..........,0,1,0]

What is hyperdimensional computation?

In HD computing, information is evenly distributed across bits, so all bits are equally important (or unimportant) to the computation.

[0,1,0,1,0,..........,0,0,0]

Doesn't matter where a bit corruption occurs!

What is hyperdimensional computation?

In HD computing, information is encoded in the hamming distances between hypervectors.

What is hyperdimensional computation?

In HD computing, information is encoded in the hamming distances between hypervectors.

```
[0,1,0,1,0,...........,0,0,0]
[0,1,1,1,0,...........,0,1,0]
```

Many bit corruptions are required to make a non-negligible change in the distance. **Highly error resilient!**

Hyperdimensional Computation

Hyperdimensional Computation (HDC) can perform a variety of tasks

- *Data structures* - construction and querying of database, graph, tree, finite automata, etc.
- *Processing tasks* - information retrieval, load balancing, analogical reasoning
- *Machine learning* - time-series data classification/edge/low-power

Heim at a First Glance

We present ***Heim***, the first static analysis-based optimizer for optimizing hyperdimensional computations to execute with acceptable accuracy on emerging hardware technologies.

Heim at a First Glance

We present ***Heim***, the first static analysis-based optimizer for optimizing hyperdimensional computations to execute with acceptable accuracy on emerging hardware technologies.

Heim provides **accuracy guarantees** over all data structures and queries captured in a user-provided specification, while **minimizing resource usage**

Heim at a First Glance

We present ***Heim***, the first static analysis-based optimizer for optimizing hyperdimensional computations to execute with acceptable accuracy on emerging hardware technologies.

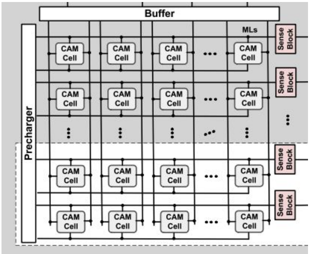
Heim provides **accuracy guarantees** over all data structures and queries captured in a user-provided specification, while **minimizing resource usage**

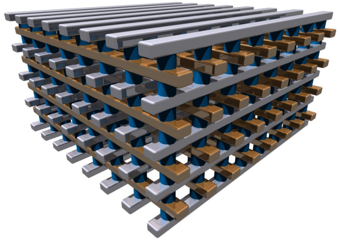
Heim's analysis is ***static***, and completes in milliseconds at compile-time.

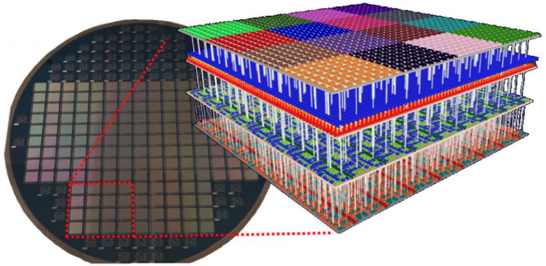
Heim at a First Glance

We present ***Heim***, the first static analysis-based optimizer for optimizing hyperdimensional computations to execute with acceptable accuracy on emerging hardware technologies.

Heim is **hardware-aware** and optimizes computations to execute accurately on emerging hardware technologies.







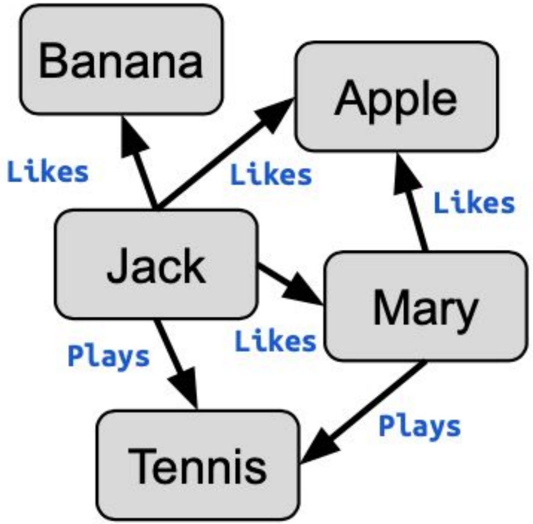
Hyperdimensional Computation by Example

Knowledge Graph Data structure. directed graph with labeled edges and nodes.

Node Label = “concept”

Edge Label = “relation”

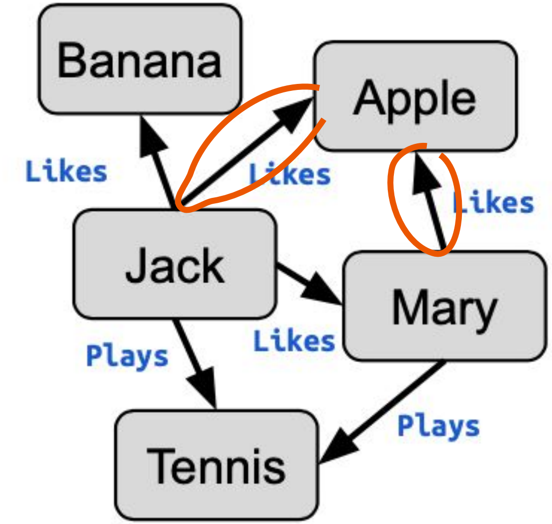
Edge Direction = “interaction”



Student knowledge graph

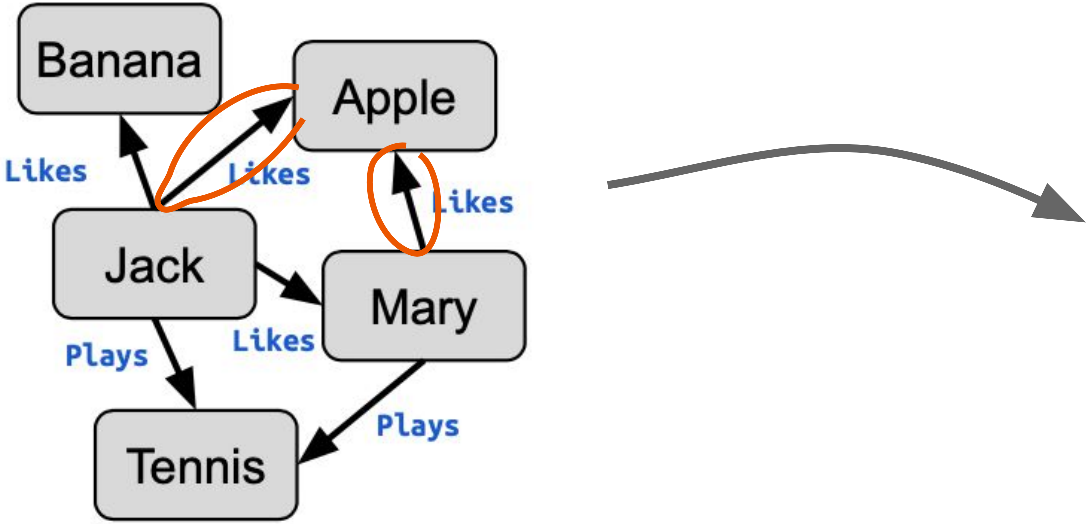
Edge Queries. Ask about relationships between nodes, or concepts.

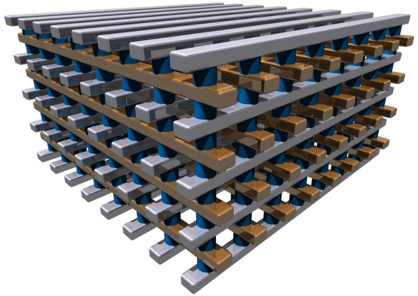
Query. How many students like apples?



Result.
Two students
# of students with "likes" relations that point to the Apple concept.

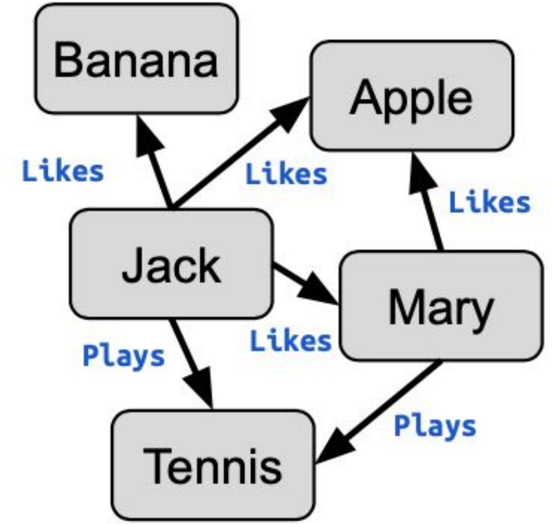
We want to execute the "apples" query on a piece of hardware that stores information in an **Two-Bit-Per-Cell Resistive RAM (RRAM) storage array**, an information-dense emerging memory technology that is prone to error.





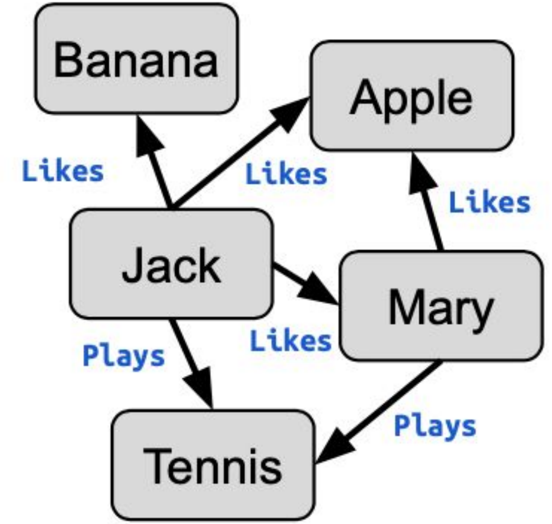
per-bit error rate of 2.15%

Data structure. directed graph with labeled edges and nodes.



How do we encode this data structure using HD computing?

Data structure. directed graph with labeled edges and nodes.



How do we encode this data structure using HD computing?

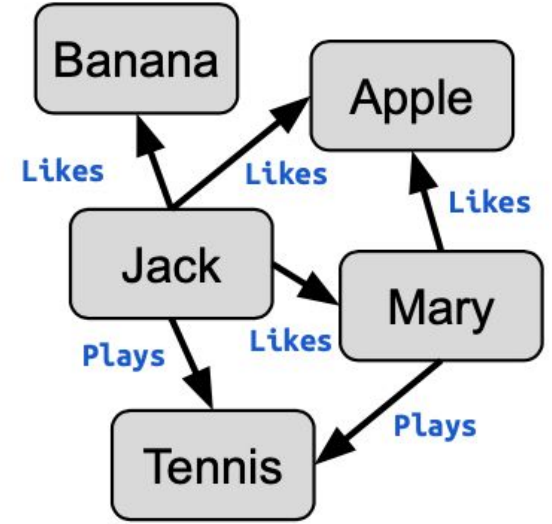
First, we need to construct the *atomic elements* of knowledge graph:

Relations = {likes,plays}

Concepts = { jack, mary, apple, tennis, banana}

Interactions= {act,target}

Data structure. directed graph with labeled edges and nodes.



How do we encode this data structure using HD computing?

First, we need to construct the *atomic elements* of knowledge graph:

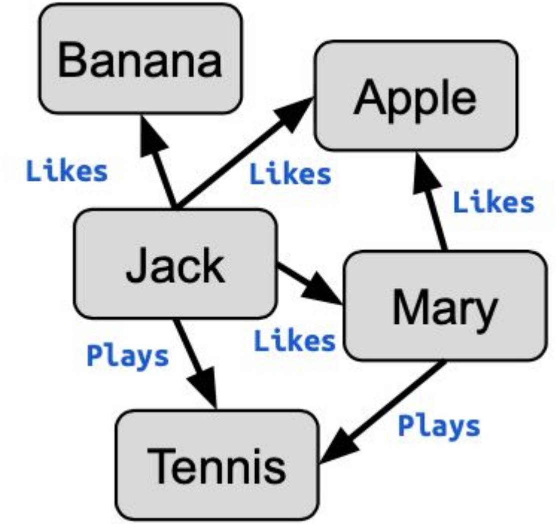
Relations = {likes,plays}

Concepts = { jack, mary, apple, tennis, banana}

Interactions= {act,target}

How do we do this?

We generate a random binary vector, or atomic hypervector, for each type of node label (concept), edge label (relation), and edge direction in the knowledge graph.



Relations = {likes,plays}

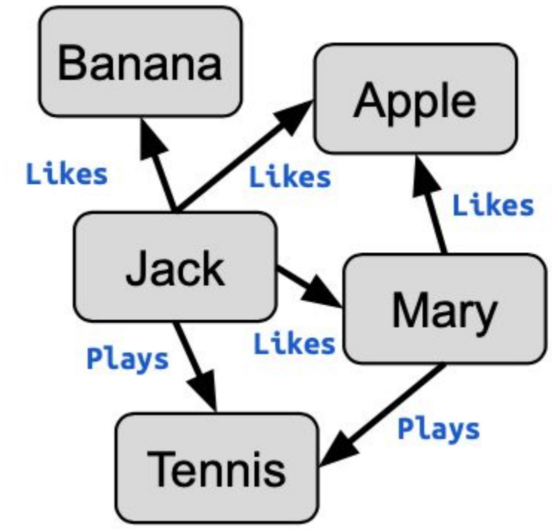
Concepts = { jack, mary, apple, tennis, banana}

Interactions= {act,target}

apple

[0,1,1,1,0,..........,0,1,0]

We generate a random binary vector, or hypervector, for each type of node label (concept), edge label (relation), and edge direction in the knowledge graph.



apple

[0,1,1,1,0,..........,0,1,0]

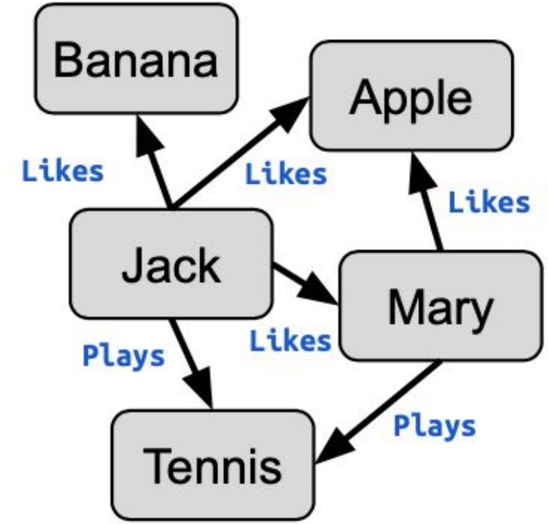
Hamming distance (HD) between atomic hypervectors is large!

Conceptually, makes sense. The "apples" node and "plays" are not related at all.

plays

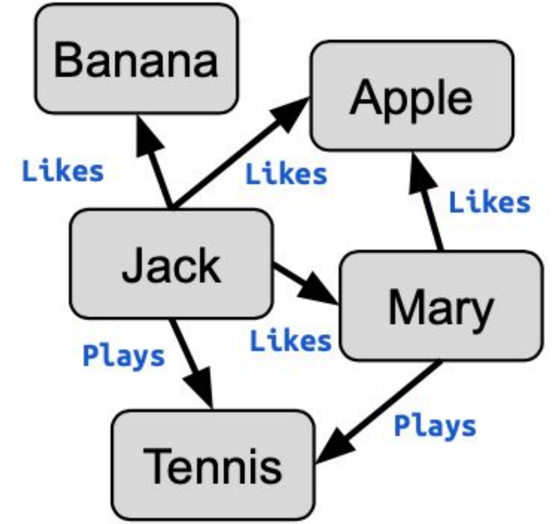
[1,0,0,1,1,..........,1,0,0]

Data structure. directed graph with labeled edges and nodes.



Now we're ready to encode the data structure as a hypervector using HD computing.

Data structure. directed graph with labeled edges and nodes.

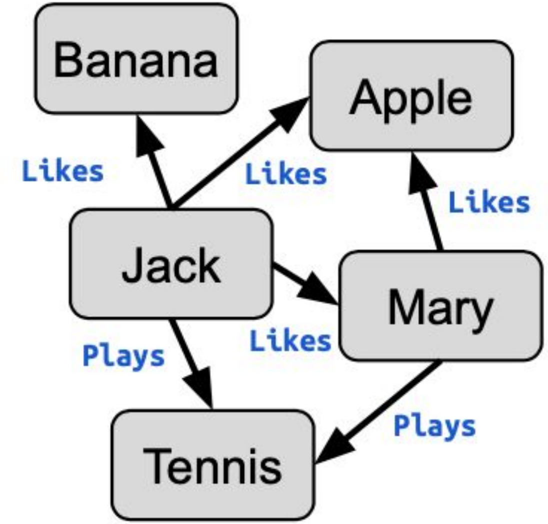


Now we're ready to encode the data structure as a hypervector using HD computing.

We will be building the data structure from the bottom-up.

Edges → edge sets → graph

Data structure. directed graph with labeled edges and nodes.



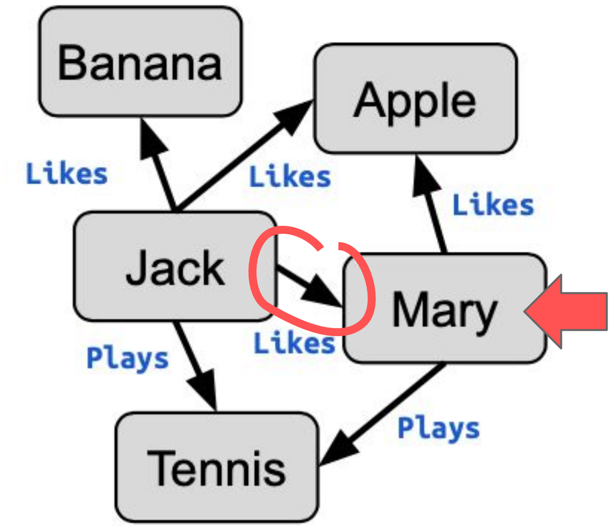
Now we're ready to encode the data structure as a hypervector using HD computing.

How do we encode information? We will compute over the atomic hypervectors!

Interactions= {act,target}

Concepts= { jack, mary, apple, tennis, banana}

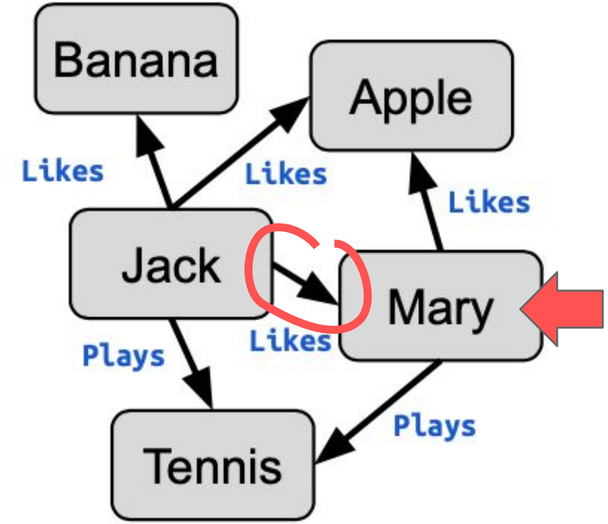
Relations= {likes,plays}



We want to construct each labeled, directed edge relative to a particular node

<target,likes,jack> points to **Mary** node

Interactions= {act,target}   Concepts= { jack, mary, apple, tennis, banana}   Relations= {likes,plays}



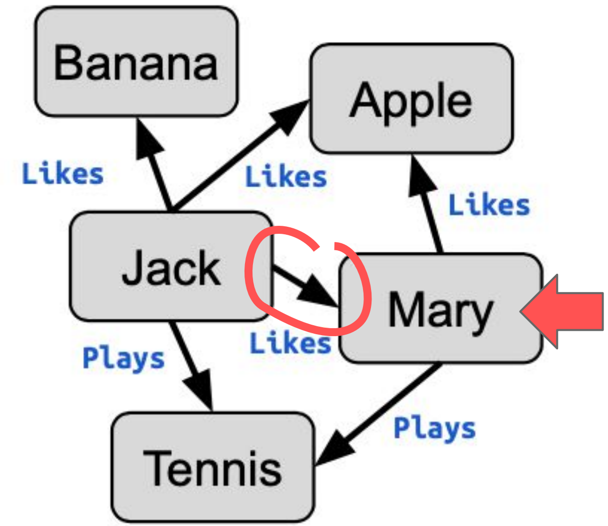
<target,likes,jack> points to Mary node

hv1 = target ⊙ likes ⊙ jack

binding operation
XOR

For each node, we construct each graph edge by binding together the interaction, relation, and concept hypervectors.

Interactions= {act,target} Concepts= { jack, mary, apple, tennis, banana} Relations= {likes,plays}



<target,likes,jack> points to Mary node

$$hv1 = target \odot likes \odot jack$$

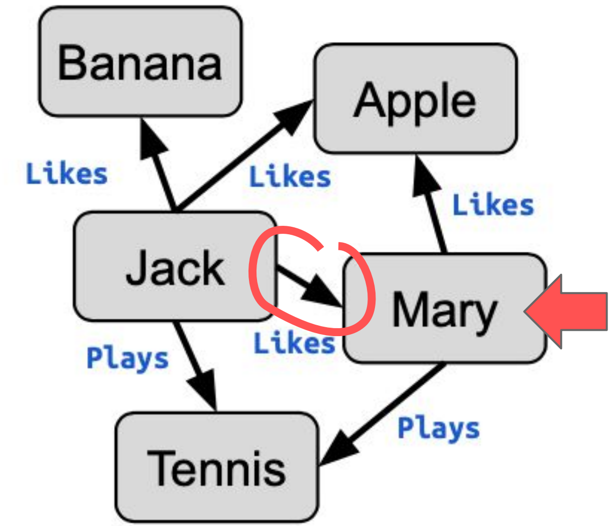
Binding creates a hypervector that is dissimilar to the input hypervecvectors

HD(**hv1**,**target**) is large

HD(**hv1**,**likes**) is large

HD(**hv1**,**jack**) is large

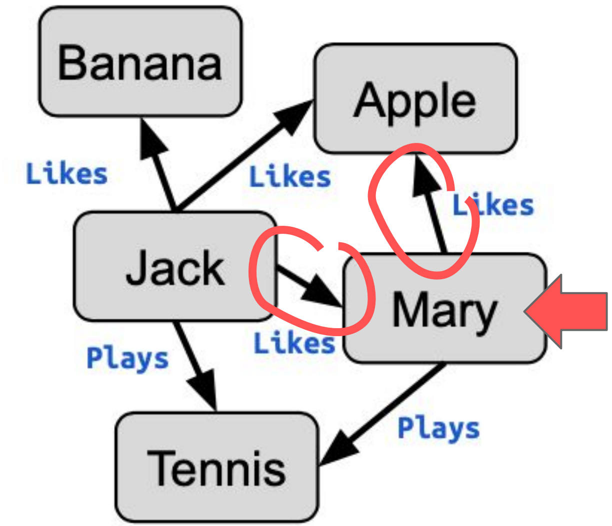
Interactions= {act,target} Concepts= { jack, mary, apple, tennis, banana} Relations= {likes,plays}



hv1 = target ⊙ likes ⊙ jack <target,likes,jack>

We construct a edge hypervector for each edge that is connected to the mary node.

Interactions= {act,target}   Concepts= { jack, mary, apple, tennis, banana}   Relations= {likes,plays}

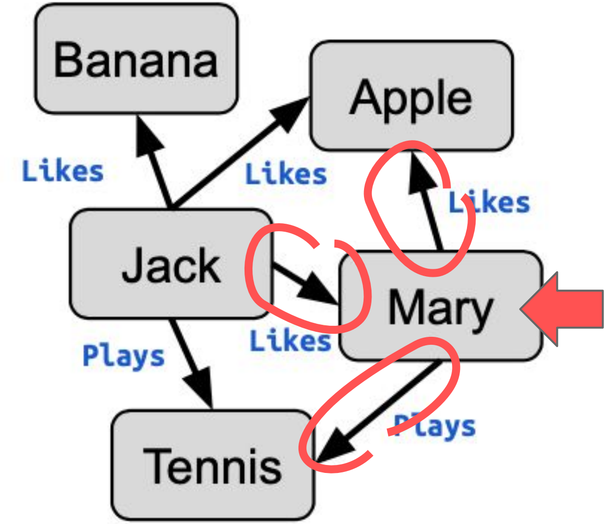


hv1 = target ⊙ likes ⊙ jack   <target,likes,jack>

hv2 = act ⊙ likes ⊙ apple   <act,likes,apple>

We construct a edge hypervector for each edge that is connected to the mary node.

Interactions= {act,target}   Concepts= { jack, mary, apple, tennis, banana}   Relations= {likes,plays}



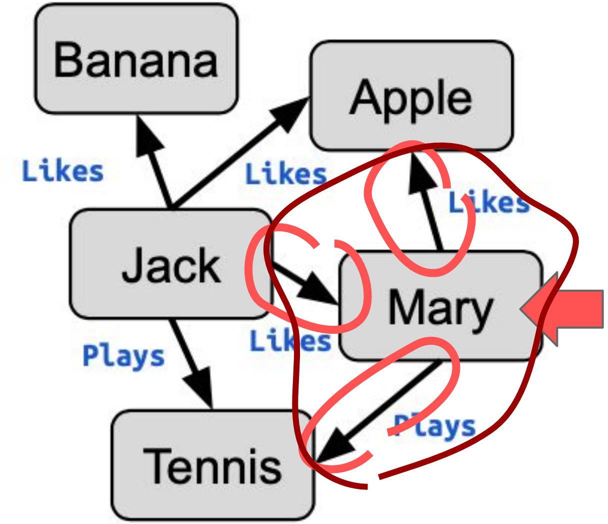
hv1 = target ⊙ likes ⊙ jack   <target,likes,jack>

hv2 = act ⊙ likes ⊙ apple   <act,likes,apple>

hv3 = act ⊙ plays ⊙ tennis   <act,plays,tennis>

We construct a edge hypervector for each edge that is connected to the mary node.

Interactions= {act,target} Concepts= { jack, mary, apple, tennis, banana} Relations= {likes,plays}



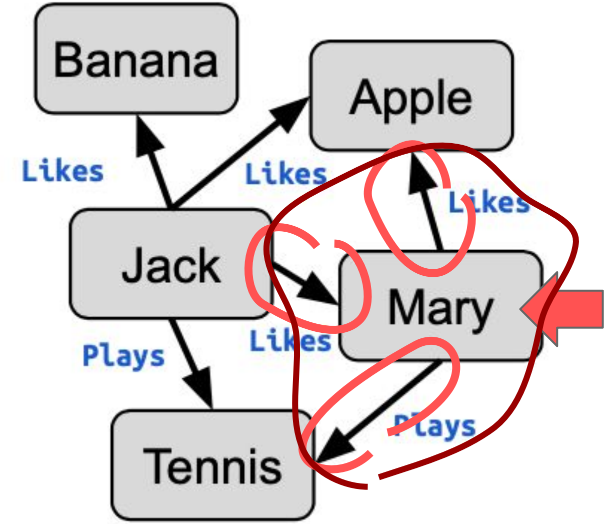
We next want to create a set of edges that are connected to the mary node.

{ <target,likes,jack>, <act,likes,apple>, <act,plays,tennis> }

Interactions= {act,target}

Concepts= { jack, mary, apple, tennis, banana}

Relations= {likes,plays}



We next want to create a set of edges that are connected to the mary node.

{ <target,likes,jack>, <act,likes,apple>, <act,plays,tennis> }

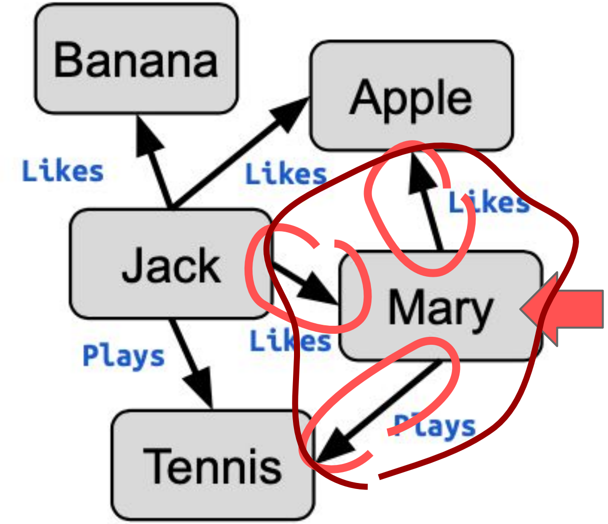
hv_mary = hv1 + hv2 + hv3

bundling operation

Bitwise Majority

To accomplish this, we bundle the edge hypervectors together

Interactions= {act,target} Concepts= { jack, mary, apple, tennis, banana} Relations= {likes,plays}



We next want to create a set of edges that are connected to the mary node.

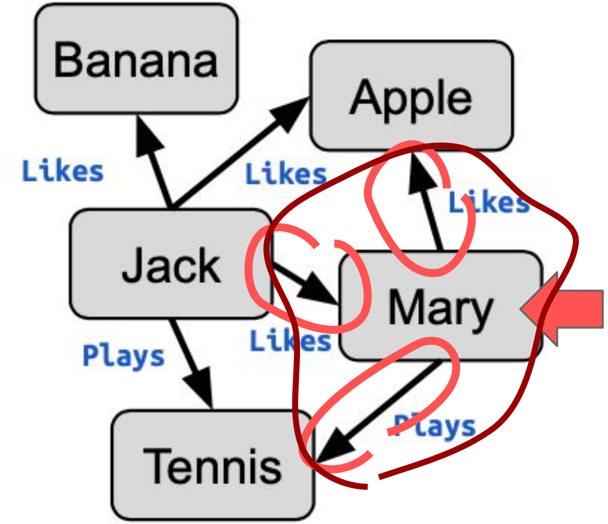
{ <target,likes,jack>, <act,likes,apple>, <act,plays,tennis> }

hv_mary = hv1 + hv2 + hv3

Bundling creates a hypervector similar to the input hypervectors.

HD(**hv_mary**,**hv1**) is small
HD(**hv_mary**,**hv2**) is small
HD(**hv_mary**,**hv3**) is small

Interactions= {act,target} Concepts= { jack, mary, apple, tennis, banana} Relations= {likes,plays}



We next want to create a set of edges that are connected to the mary node.

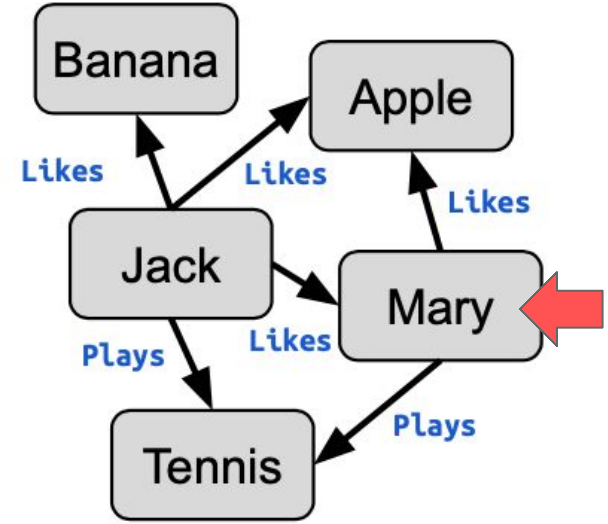
{ <target,likes,jack>, <act,likes,apple>, <act,plays,tennis> }

hv_mary = hv1 + hv2 + hv3

We can use the hamming distance to query if an edge belongs to an edge set!

HD(**hv_mary**, **act** ⊙ **likes** ⊙ **apple**) -> ***small, in set***
HD(**hv_mary**, **act** ⊙ **likes** ⊙ **apple**) -> ***large, NOT in set***

Interactions= {act,target} Concepts= { jack, mary, apple, tennis, banana} Relations= {likes,plays}



Next, we build a edge set hypervector for each node in the graph.

hv_mary = hv1 + hv2 + hv3

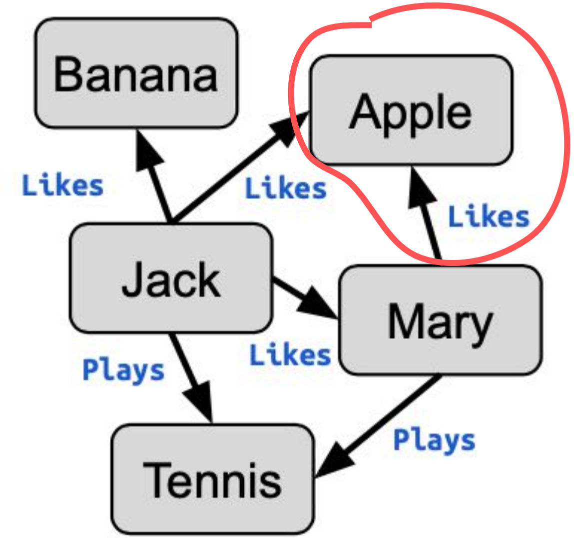
hv_jack = hv4 + hv5 + hv6 + hv7

hv_tennis = hv8 + hv9

hv_banana = hv10

hv_apple = hv11 + hv12

Interactions= {act,target}   Concepts= { jack, mary, apple, tennis, banana}   Relations= {likes,plays}



Now we can query for edges. Let's test for the "likes-apples edge":

<act,likes,apple>   act ⊙ likes ⊙ apple

hv_mary = hv1 + hv2 + hv3

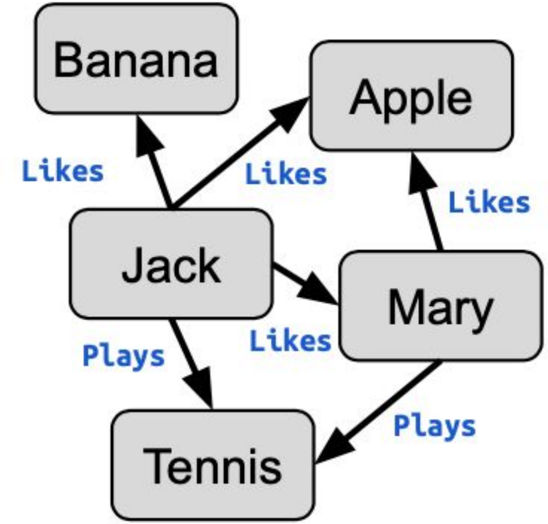
hv_jack = hv4 + hv5 + hv6 + hv7

hv_tennis = hv8 + hv9

hv_banana = hv10

hv_apple = hv11 + hv12

Interactions= {act,target}   Concepts= { jack, mary, apple, tennis, banana}   Relations= {likes,plays}



We then perform test for an edge by computing the hamming distance between query edge and each edge set.

HD(hv_mary, act ⊙ likes ⊙ apple)

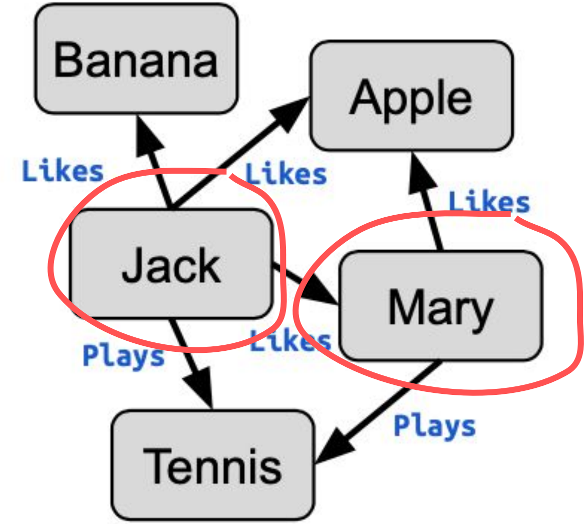
HD(hv_jack, act ⊙ likes ⊙ apple)

HD(hv_tennis, act ⊙ likes ⊙ apple)

HD(hv_banana, act ⊙ likes ⊙ apple)

HD(hv_apple, act ⊙ likes ⊙ apple)

Interactions= {act,target}   Concepts= { jack, mary, apple, tennis, banana}   Relations= {likes,plays}



If the hamming distance is small, the edge is contained within the node's edge set.

HD(hv_mary, act ⊙ likes ⊙ apple) -> small distance, in set!

HD(hv_jack, act ⊙ likes ⊙ apple) -> small distance, in set!

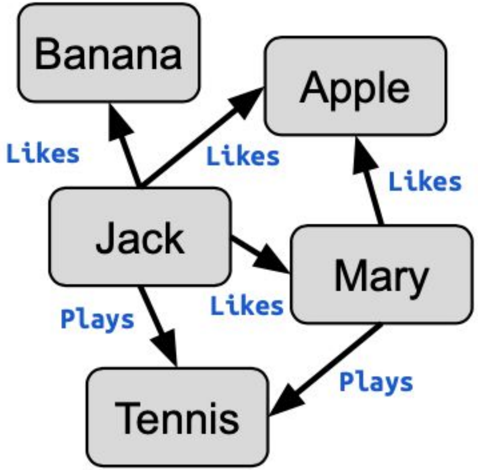
HD(hv_tennis, act ⊙ likes ⊙ apple)

HD(hv_banana, act ⊙ likes ⊙ apple)

HD(hv_apple, act ⊙ likes ⊙ apple)

So, there is some missing information..

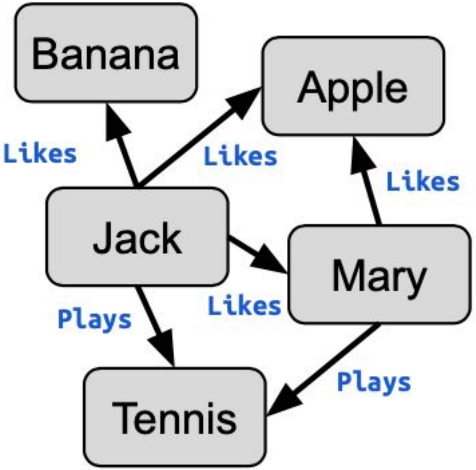
What distance threshold should we use? How do we distinguish between a small and large distance?



So, there is some missing information..

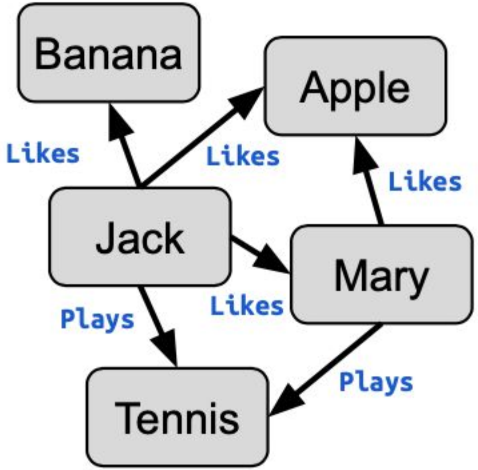
What distance threshold should we use? How do we distinguish between a small and large distance?

How big are these hypervectors exactly?



How are the thresholds and size set currently?

Currently, practitioners dynamically tune the parameters with Monte Carlo simulations



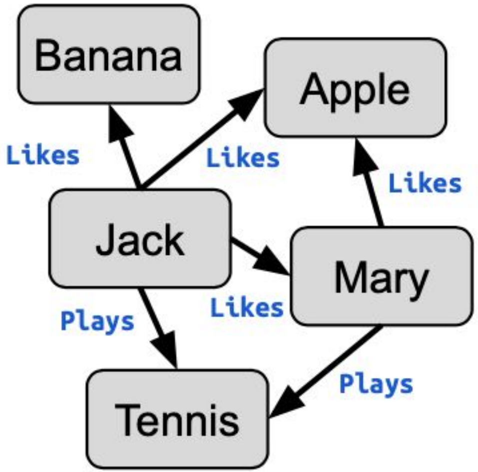
Lack of accuracy guarantees

May not generalize well

Computationally intensive

How are the thresholds and size set currently?

Currently, practitioners dynamically tune the parameters with Monte Carlo simulations



Lack of accuracy guarantees

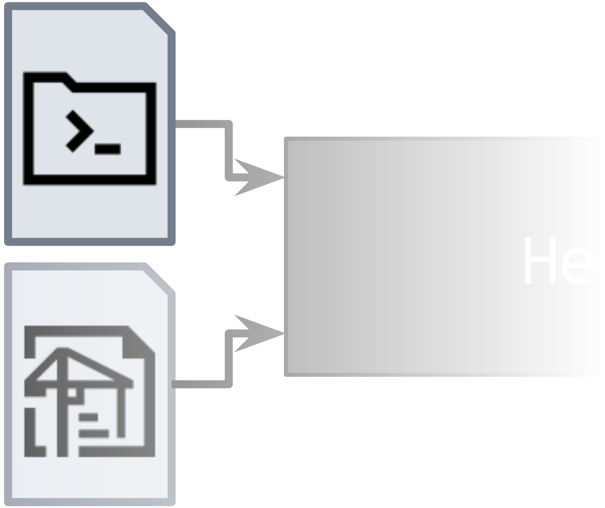
May not generalize well

Computationally intensive

We present Heim, a static optimizer that analytically derives the size and distance thresholds for an HD computation.

Static parameter optimization with Heim

Overview of Heim's System Structure



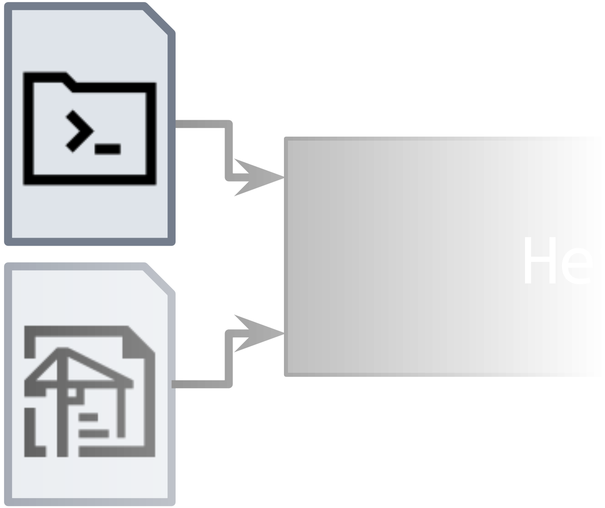
Heim program specification

```
spec {
    abs-data query = prod(interactions,relations,concepts);
    abs-data ds = sum(4,prod(interactions,relations,concepts));
    thr-query(query, ds, 1, 0.99, 0.01, 0.01);
}
```

Knowledge graph specification

Heim works with a program specification that describes the space of HD data structures to analyze and the desired query accuracies.

Overview of Heim’s System Structure



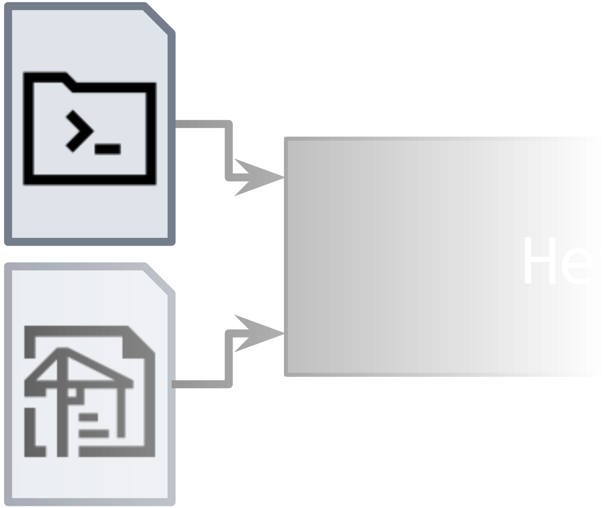
Heim program specification

```
spec {
    abs-data query = prod(interactions,relations,concepts);
    abs-data ds = sum(4,prod(interactions,relations,concepts));
    thr-query(query, ds, 1, 0.99, 0.01, 0.01);
}
```

Query edges with 99% accuracy.

Heim chooses a hypervector size and a set of distance thresholds that satisfies all query accuracy constraints in the specification.

Overview of Heim's System Structure



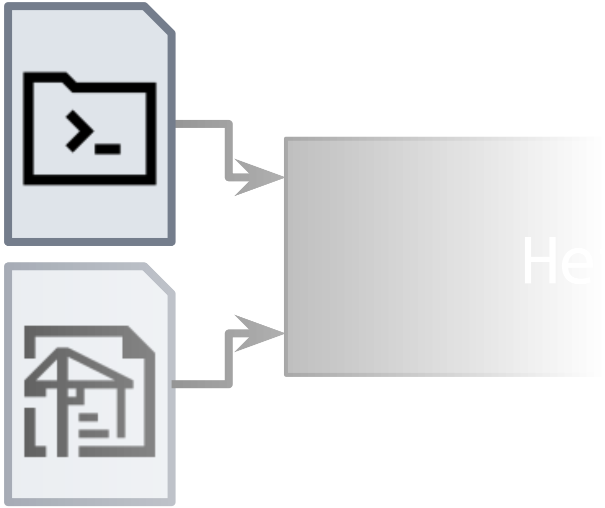
Heim program specification

```
spec {
    abs-data query = prod(interactions,relations,concepts);
    abs-data ds = sum(4,prod(interactions,relations,concepts));
    thr-query(query, ds, 1, 0.99, 0.01, 0.01);
}
```

Query edges with 99% accuracy.

Heim's accuracy guarantee: on expectation, the accuracy of each described query will converge to the accuracy specified by the most restrictive accuracy constraint.

Overview of Heim's System Structure



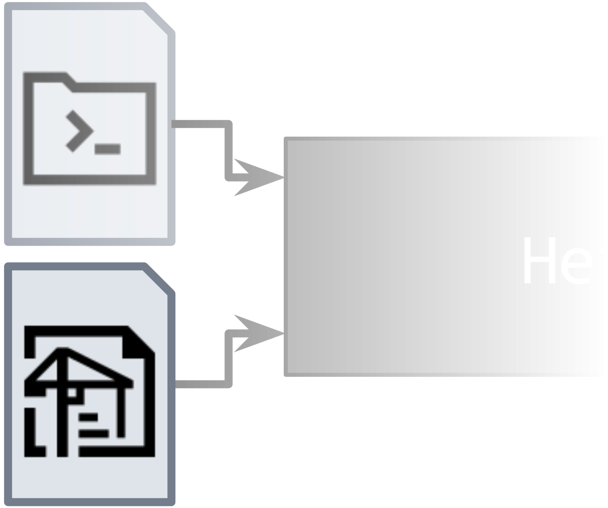
Heim program specification

```
spec {
  abs-data query = prod(interactions,relations,concepts);
  abs-data ds = sum(4,prod(interactions,relations,concepts));
  thr-query(query, ds, 1, 0.99, 0.01, 0.01);
}
```

Edge queries on graphs with maximum node cardinality of 4.

Heim ensures this ***accuracy guarantee*** holds over all queries and data structures described in the Heim specification.

Overview of Heim's System Structure



Two-bits-per-cell Resistive Memory Architecture

```
hardware-model {
  mem codebook = 0.00;
  mem item-mem = 0.0215;
  op bind = 0.00;
  op bundle = 0.00;
}
```

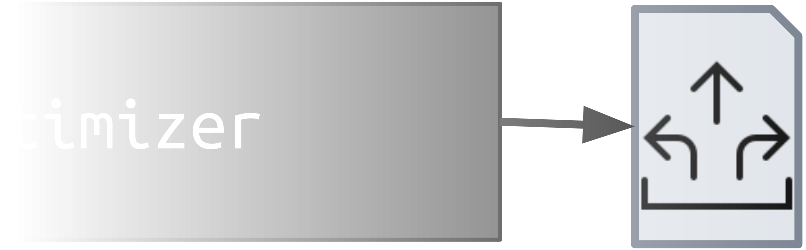
Edge sets stored in resistive memory

Heim accepts a hardware specification which specifies the bit error rates of different storage and compute elements in the hardware architecture. Heim's accuracy guarantees hold in the presence of hardware error.

Overview of Heim's System Structure

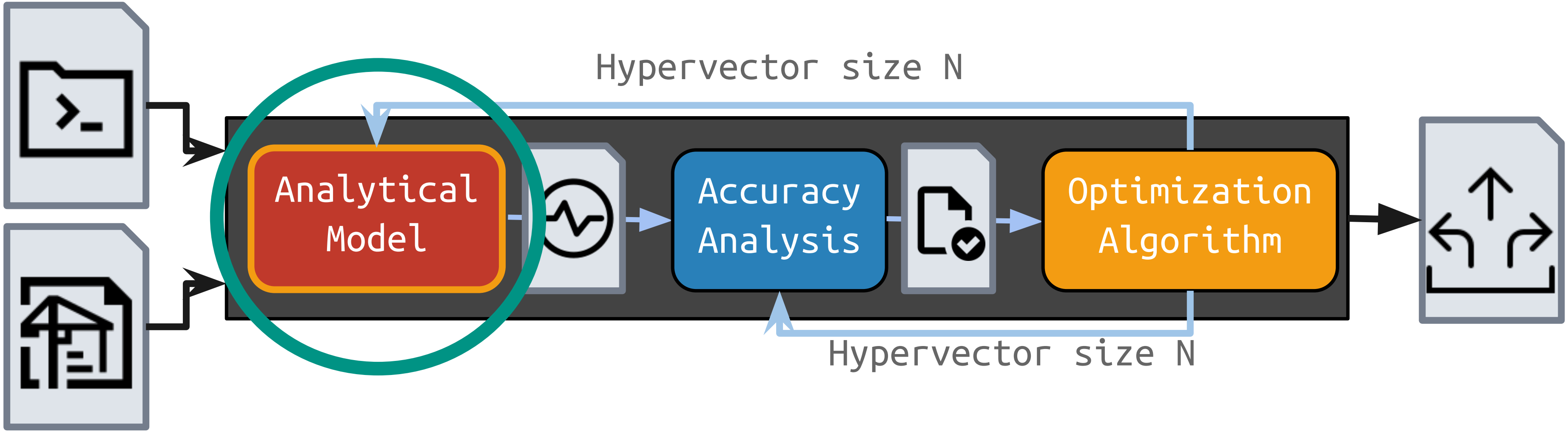
optimal hypervector size(s)
N = 173, 1060, …

optimal distance thresholds
thr = 0.45, 0.413, …



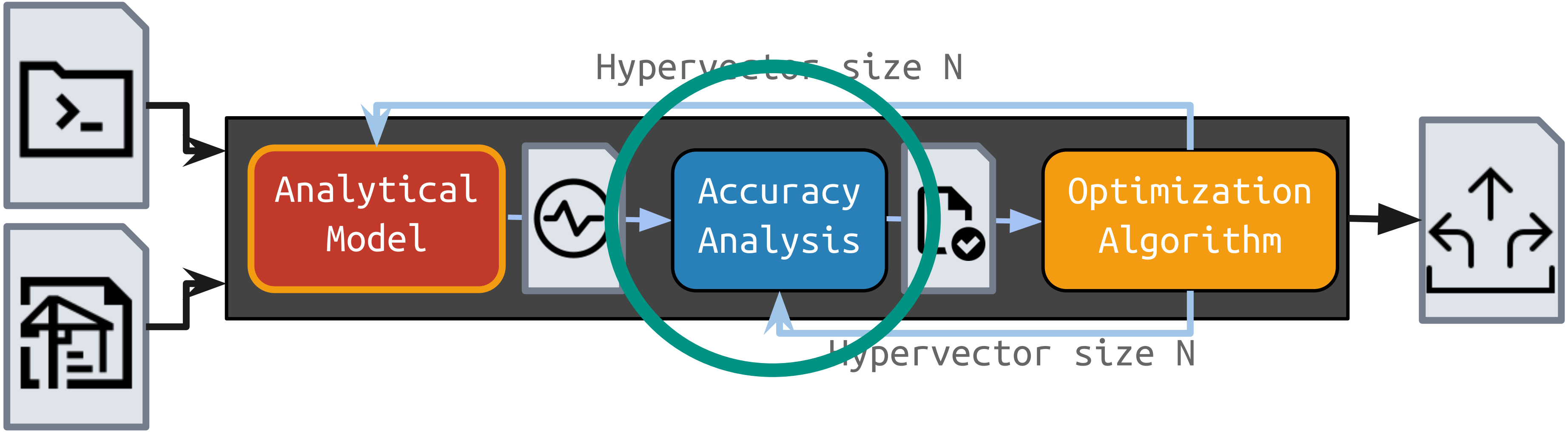
Heim analytically derives the optimal distance thresholds and the minimum hypervector size required to meet the target accuracy constraints on the target hardware.

Overview of Heim's System Structure



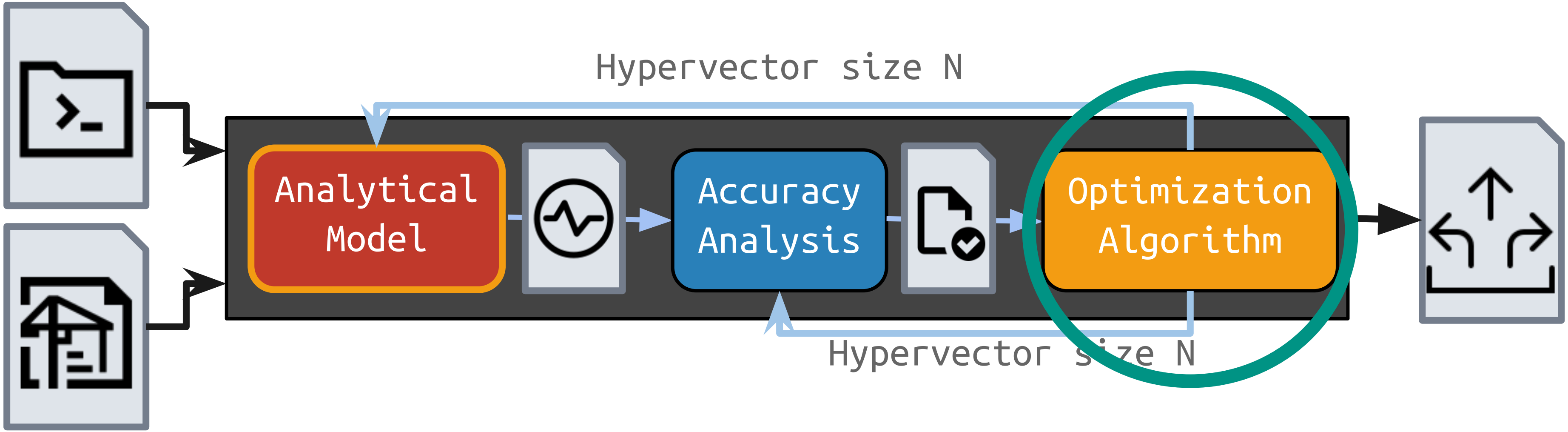
Heim's implementation consists of an analytical model of hypervector-query hamming distances for the given data structure, parametrized over hypervector size.

Overview of Heim’s System Structure



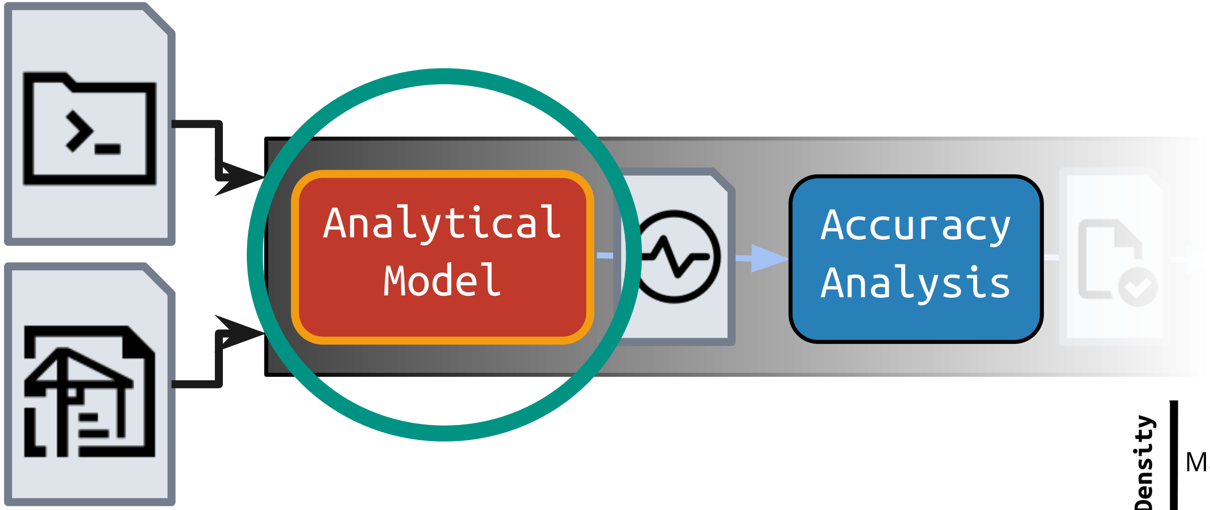
..an accuracy analysis that derives query accuracies from the analytical model.

Overview of Heim’s System Structure

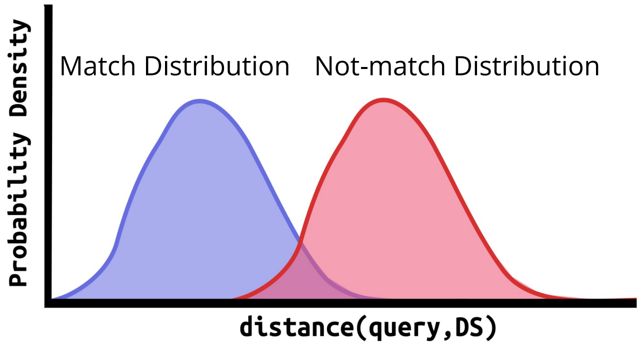


..and an optimization algorithm that uses the accuracy analysis to find the smallest hypervector size that delivers the desired accuracy.

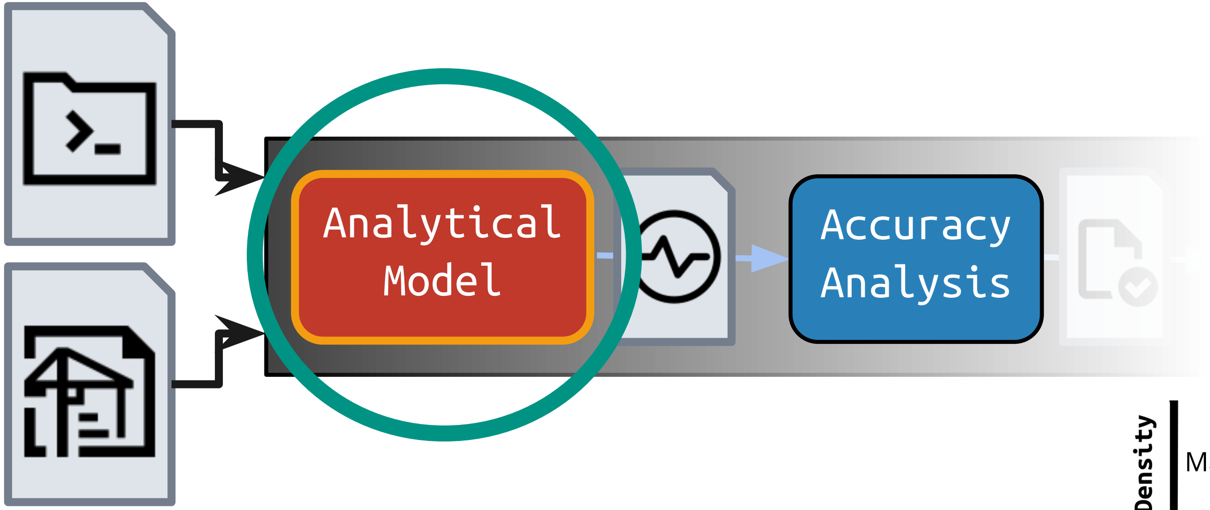
Overview of Heim's System Structure



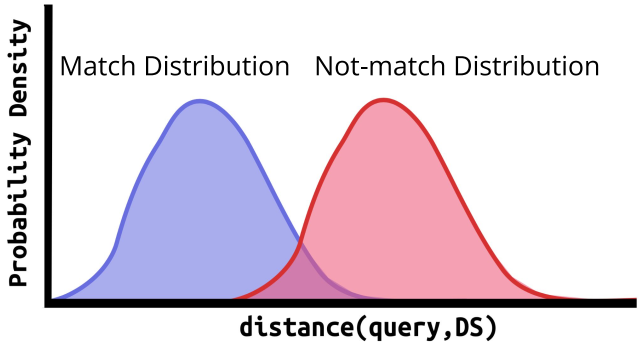
The analytical model precisely models the distribution of hamming distances for matching and non-matching queries.



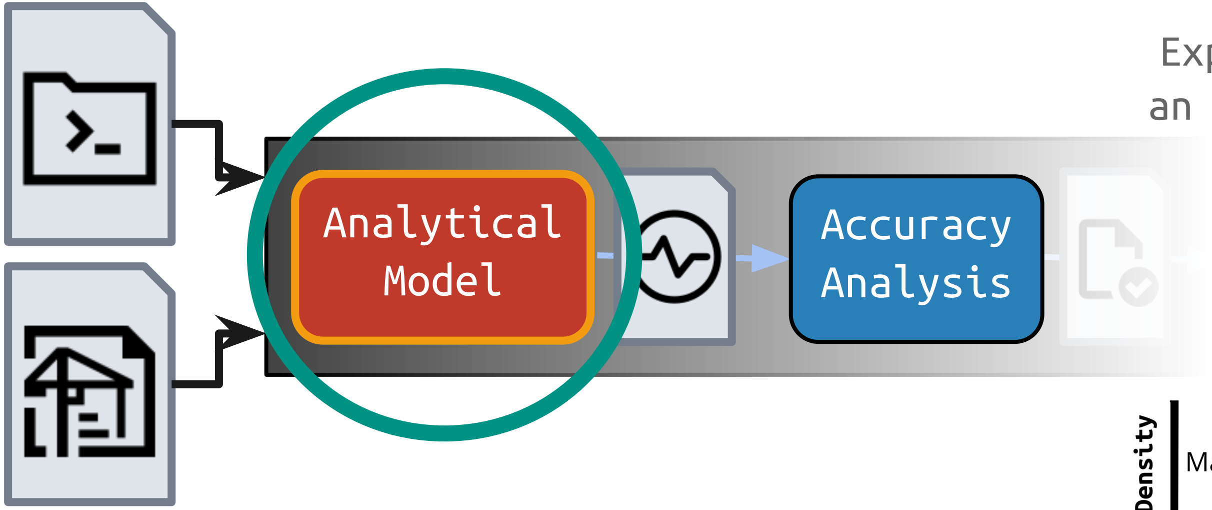
Overview of Heim's System Structure



We observe distance distributions because there is non-determinism introduced by atomic hypervector sampling and hardware error.

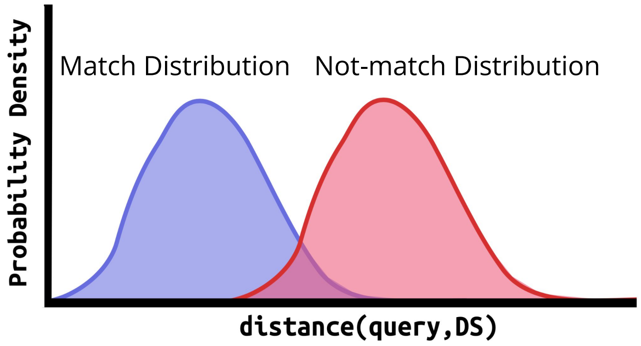


Overview of Heim's System Structure



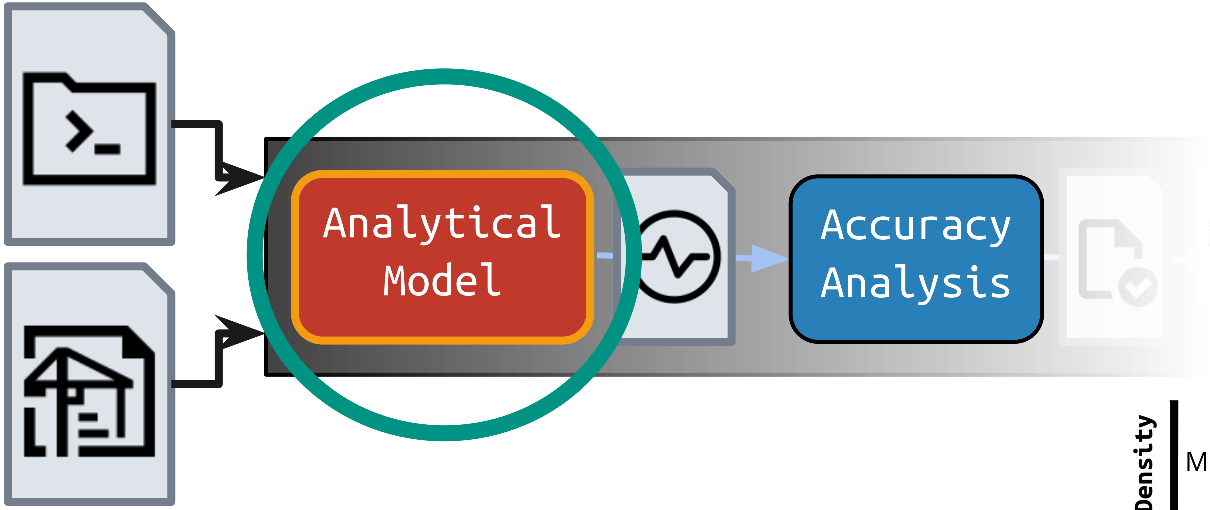
Expected mean distance from an edge to a matching m-edge set

$$M(\{c_1\},\{c_1,c_2,\ldots,c_m\}) = \frac{1}{2} - \frac{\binom{m-1}{\frac{m-1}{2}}}{2^m}$$



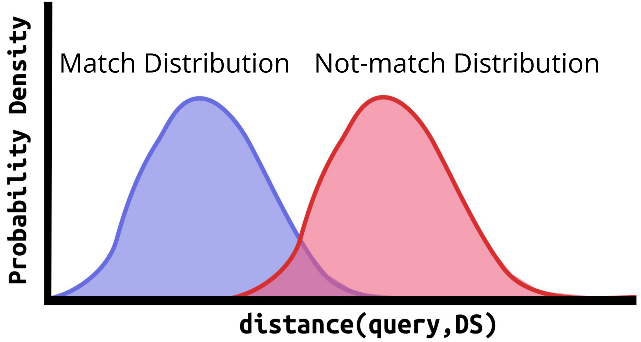
The distributions are normally distributed. We derive closed-form formulas for the mean and variance of the matching/non-matching queries, parametrized over hypervector size.

Overview of Heim's System Structure



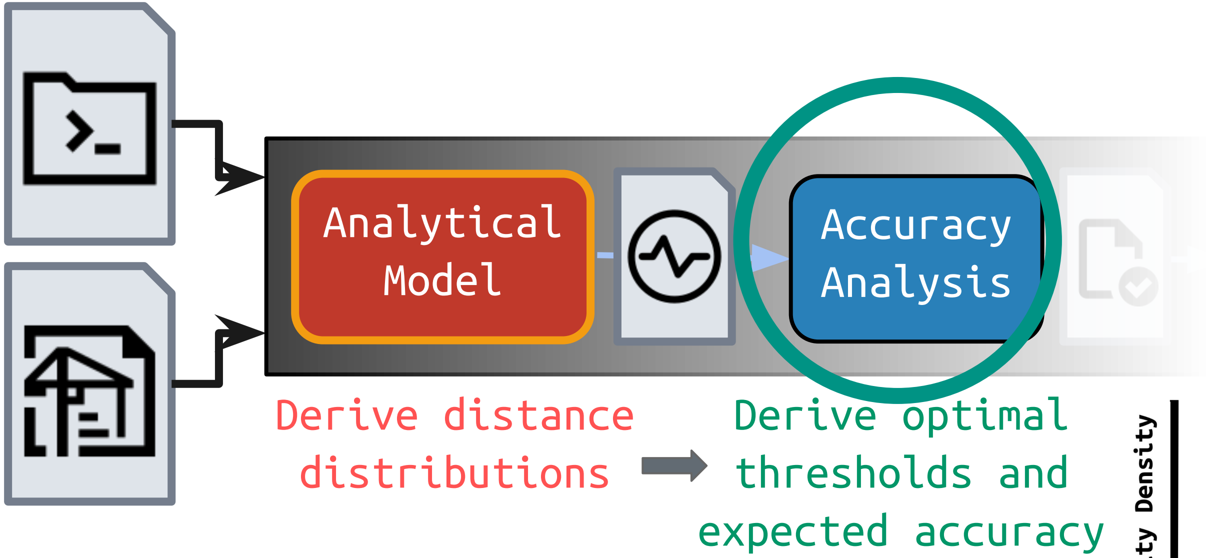
An upper bound of
bit flip rate

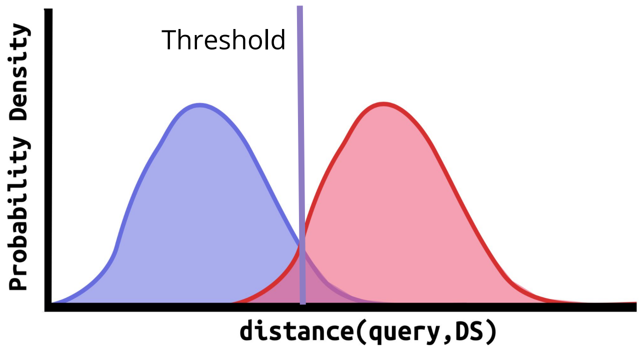
$$p = 1 - \left[\prod_{op \in Op} (1 - err(\mathrm{op})) \right]$$



To model hardware error, an upper bound on the bit flip probability is derived from the hardware spec and applied to the analytical model.

Overview of Heim's System Structure





To perform accuracy analysis, we derive closed-form formulas for the expected accuracy and compute the optimal threshold from these formulae.

Summary of Theoretical Formulations

Formulation	reference	description
WTA-*acc*, $w=1$ (6)	[Frady et al. 2018]	WTA accuracy for exactly one winner $w=1$
WTA-*acc* (10)	Section 5.4	WTA accuracy for more than one winner $w>1$
WTA-*prob* (12)	Section 5.4	probability of the w positives being in top t.
QDS I (14)	[Kanerva et al. 1997]	single-element sum-of-product set membership
QDS II (15)	[Kleyko et al. 2016]	subset sum-of-product set membership
QDS III (17)	Section 6.6	single-element product-of-sum set membership
Hardware Error (20)	Section 6.8	incorporation of hardware error

To develop this analysis, we apply results from the theoretical cognitive science community and contribute new derivations.

How does Heim perform?

Sound parameter optimization with Heim

We evaluated Heim on five hyperdimensional computing-based data structures, parametrized with five different sizes.

Benchmark data structures

benchmark	query type	data structure and query sizes
set	threshold	50k-100k element sets, 1 element/query
db-match	threshold	5k-10k fields/record, 50-100 records, m fields/query
kgraph	threshold	1-100k edges/concept, 100k+10 concepts, 800k-1000k edges, 1 edge/query, 2 relations
nfa	threshold	recognizes *str* with length k, 1-k character strings/query, 26 letters/alphabet
	query: matches are substrings of *str*, non-matches are partial substrings of *str*	
db-analogy	winner-take-all (WTA)	$k/2$-k fields/record, 50m-100m records, 1 analogy query
	query: select rows a,b where $\langle k,v \rangle \in a$, $\langle k,v' \rangle \in b$, infer v from item memory and v'	

benchmark	size parameters	benchmark sizes				
set	k	1	2	3	4	5
db-match	(k,m)	(1,2)	(2,4)	(3,6)	(4,8)	(5,10)
kgraph	k	1	2	3	4	5
nfa	k	6	8	10	12	14
db-analogy	(k,m)	(4,1)	(8,2)	(12,3)	(16,4)	(20,5)

Sound parameter optimization with Heim

We evaluated Heim on five hyperdimensional computing-based data structures, parametrized with five different sizes.

We parametrize data structures with different sizes.

benchmark	**query type**	**data structure and query sizes**
set	threshold	50k-100k element sets, 1 element/query
db-match	threshold	5k-10k fields/record, 50-100 records, m fields/query
kgraph	threshold	1-100k edges/concept, 100k+10 concepts, 800k-1000k edges, 1 edge/query, 2 relations
nfa	threshold	recognizes *str* with length k, 1-k character strings/query, 26 letters/alphabet
	query: matches are substrings of *str*, non-matches are partial substrings of *str*	
db-analogy	winner-take-all (WTA)	k/2-k fields/record, 50m-100m records, 1 analogy query
	query: select rows a,b where $\langle k,v \rangle \in a$, $\langle k,v' \rangle \in b$, infer v from item memory and v'	

benchmark	**size parameters**	**benchmark sizes**				
set	k	1	2	3	4	5
db-match	(k,m)	(1,2)	(2,4)	(3,6)	(4,8)	(5,10)
kgraph	k	1	2	3	4	5
nfa	k	6	8	10	12	14
db-analogy	(k,m)	(4,1)	(8,2)	(12,3)	(16,4)	(20,5)

Sound parameter optimization with Heim

We evaluated Heim on five hyperdimensional computing-based data structures, parametrized with five different sizes.

We parametrize data structures with different sizes.

benchmark	query type	data structure and query sizes
set	threshold	50k-100k element sets, 1 element/query
db-match	threshold	5k-10k fields/record, 50-100 records, m fields/query
kgraph	threshold	1-100k edges/concept, 100k+10 concepts, 800k-1000k edges, 1 edge/query, 2 relations
nfa	threshold	recognizes *str* with length k, 1-k character strings/query, 26 letters/alphabet
	query: matches are substrings of *str*, non-matches are partial substrings of *str*	
db-analogy	winner-take-all (WTA)	$k/2$-k fields/record, 50m-100m records, 1 analogy query
	query: select rows a,b where $\langle k,v\rangle \in a$, $\langle k,v'\rangle \in b$, infer v from item memory and v'	

benchmark	size parameters	benchmark sizes				
set	k	1	2	3	4	5
db-match	(k,m)	(1,2)	(2,4)	(3,6)	(4,8)	(5,10)
kgraph	k	1	2	3	4	5
nfa	k	6	8	10	12	14
db-analogy	(k,m)	(4,1)	(8,2)	(12,3)	(16,4)	(20,5)

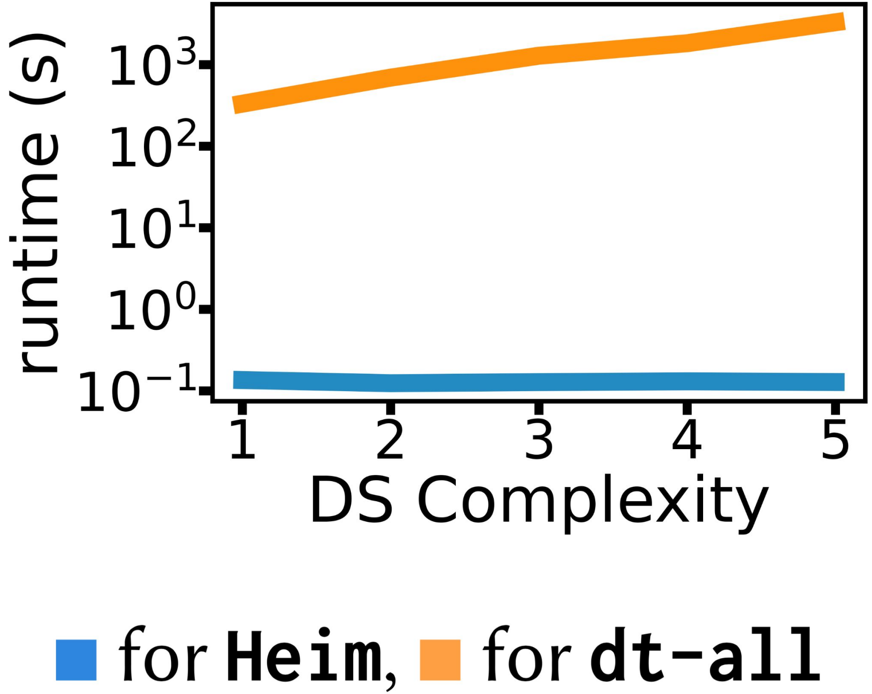
We compared against dynamic tuning (**dt-all**) and other baselines, and configured all methods to deliver **99% query accuracy**.

Optimization Runtime

Heim runs in milliseconds, and is orders of magnitude faster than dynamic tuning-based approaches. (**2-5 orders of magnitude faster**)

Optimization runtime for knowledge graph

Y-axis is log-scale

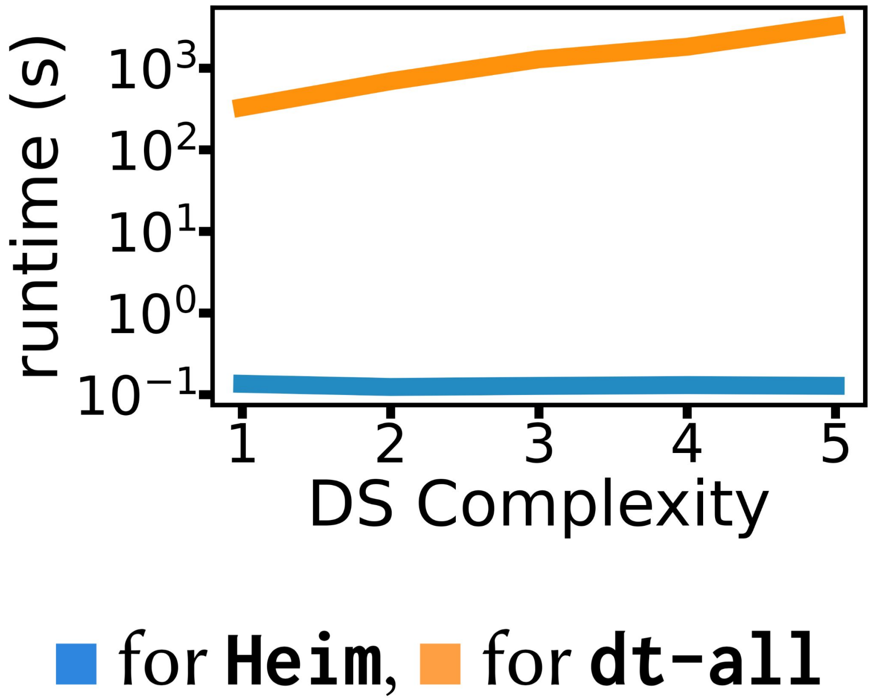


Optimization Runtime

Heim runs in milliseconds, and is orders of magnitude faster than dynamic tuning-based approaches. (**2-5 orders of magnitude faster**)

Optimization runtime for knowledge graph

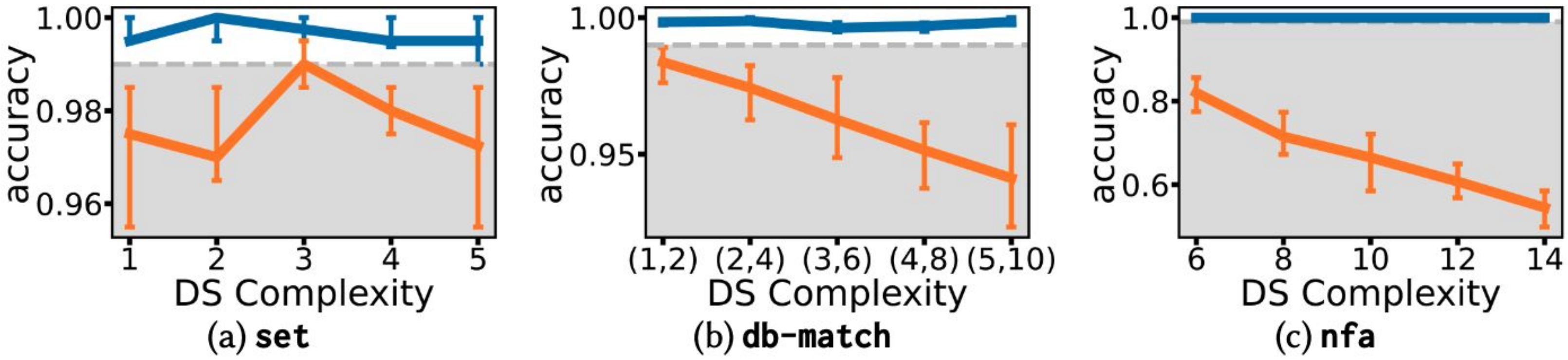
Y-axis is log-scale



Heim's analysis is analytical, the runtime does not depend on the data structure size

Accuracy

Heim-derived thresholds and sizes consistently meet accuracy target (above shaded region).

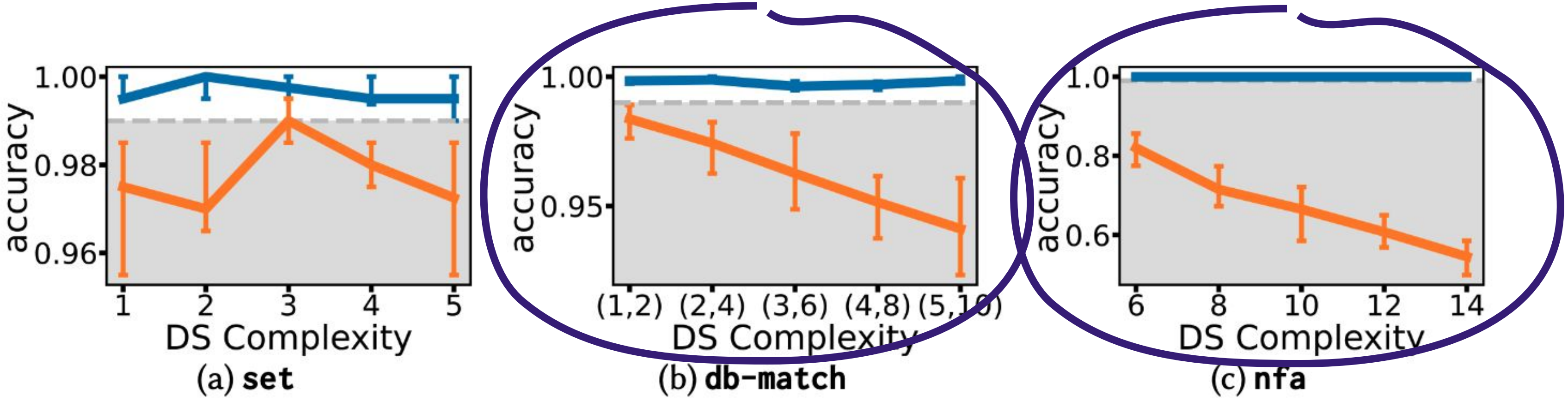


We sample 100 random data structures for each benchmark
Y-axis is median accuracy, error bars are quartiles.

■ for **Heim**, ■ for **dt-all**

Accuracy

For several benchmarks, Heim is able to find parameterizations that dynamic tuning cannot find.

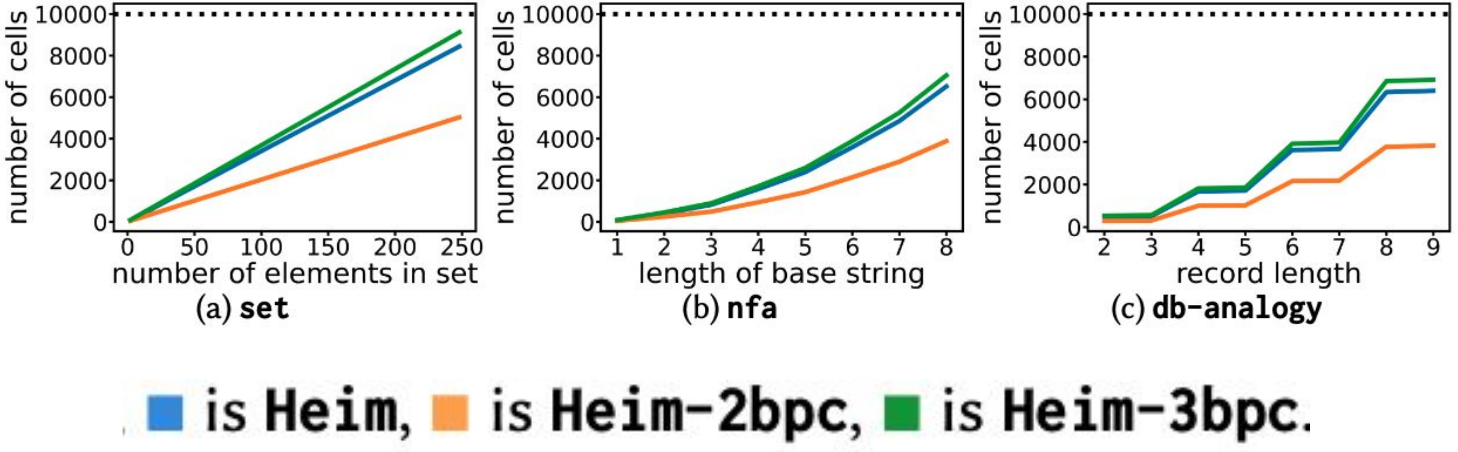


We sample 100 random data structures for each benchmark
Y-axis is median accuracy, error bars are quartiles.

■ for **Heim**, ■ for **dt-all**

Optimal RRAM Density at Iso-Accuracy

We can perform perform hardware-aware parameter optimization, and use the derived parameterizations to systematically analyze the tradeoffs between different device technologies / usages of device technologies.



99% accuracy

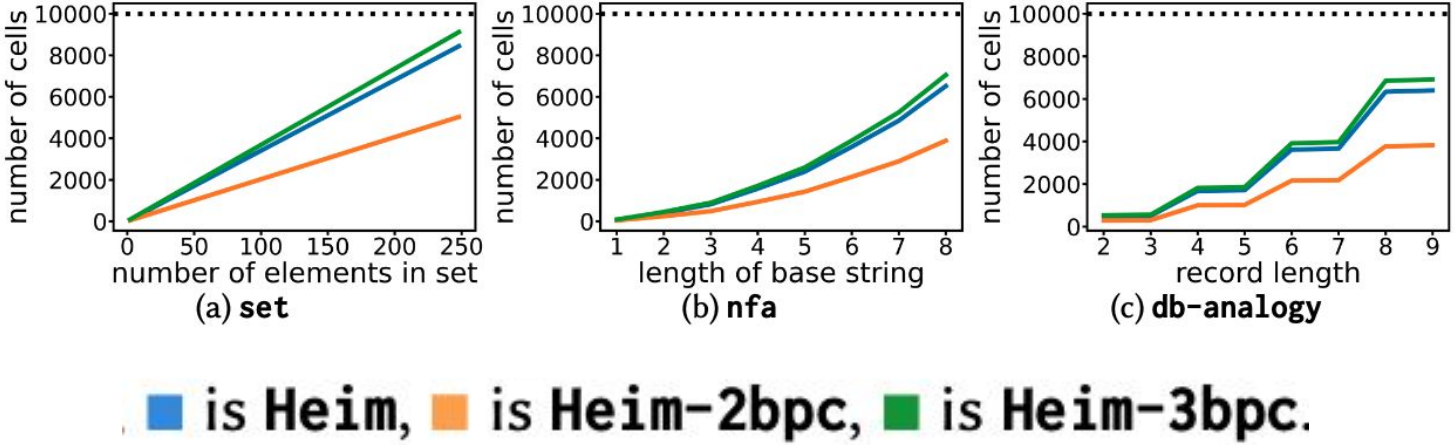
3BPC BER[1]
0.1273

2BPC BER[1]
0.0215

1. Anjiang Wei, Akash Levy, Pu (Luke) Yi, Robert Radway, Priyanka Raina, Subhasish Mitra, and Sara Achour. 2023. PBA: Percentile-Based Level Allocation for Multiple-Bits-Per-Cell RRAM. In ICCAD.

Optimal RRAM Density at Iso-Accuracy

We can perform perform hardware-aware parameter optimization, and use the derived parameterizations to systematically analyze the tradeoffs between different device technologies / usages of device technologies.



99% accuracy

3BPC BER[1]
0.1273

2BPC BER[1]
0.0215

In this case study, 2BPC ReRAM outperforms 3BPC ReRAM, 3BPC ReRAM worse than conventional memory at iso-accuracy across all hardware-optimized data structures. **Systematic applications-to-devices analysis!**

1. Anjiang Wei, Akash Levy, Pu (Luke) Yi, Robert Radway, Priyanka Raina, Subhasish Mitra, and Sara Achour. 2023. PBA: Percentile-Based Level Allocation for Multiple-Bits-Per-Cell RRAM. In ICCAD.

Conclusion

We presented Heim, the first static analysis framework for hyper-dimensional computation that minimizes the resource usage in presence of hardware error

- Heim achieves better accuracy than dynamic tuning and offers guarantees, and is orders of magnitude faster
- Heim enables iso-accuracy systematic application-to-device analysis

We envision Heim as a sound core and basis for future analyses that may be unsound but apply to more practical applications