International Institute of Information Technology Bangalore

Micron – IIITB
Bi-Weekly Presentation

Date: November 29nd, 2023

By:
- Prof. Madhav Rao
- Saket Gurjar (iMTech)
- Anshul M (iMTech)
- Vinay Rayapati (MTech)

Agenda

- Things done
- Action plan
- Interfacing with DB
- Graph embedding
- Wire mask
- Summary

Things Done

Inputs:- Spice netlists are being used as input to the parser. These netlists are at present being generated from Cadence Virtuoso + PVS LVS, post manual PNR, and LVS spice extraction.

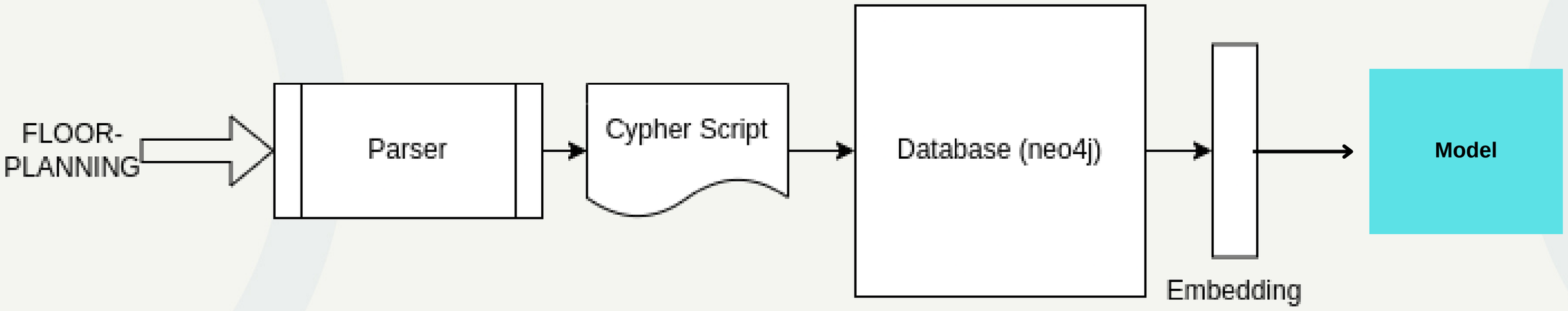
Parser:- The parser is built in the sense of extracting the necessary information from these netlists, such as the names, connections, sizes, and shapes. Necessary changes can be made in the future to the parser, to parse a blackboxed input spice netlist. (This change can be done in no time) . The output of the parser is a cypher script, which is given as an input to the neo4j.

Neo4j:- At present, the nodes are built with the properties of sizes, shapes, coordinates, and connections. The other constraints will be considered in a later part of the development. This is done for the ease of debugging and developing a prototype model that can place the macros with the right connections as a start. The output of the neo4j is used to produce an embedding vector using the embedding techniques.

Action Plan

- **Step 1 : Get Calibre LVS running. For now, spice netlists extracted from PVS LVS.**
- **Step 2 : Make necessary changes to parser after Calibre is running with black-boxing.**
- **Step 3 : Decide on what model to use after embedding.**
- **Step 4 : Model Training without any additional constraints.**
- **Step 5 : Integration of constraints into the pipeline**

Interfacing with DB



Interfacing with DB - Parser

- Parser will take in the Spice Netlist as input and give the cypher script that will generate the same graph .
- Mainly reads connectivity information and any other features about each macro like size, shape, etc.

Interfacing with DB - Cypher Script

- Cypher is a declarative graph query language that allows for expressive and efficient data querying in a property graph.
- It is the language that is primarily used in Neo4J to query the database.
- This will mainly contain connectivity and features of individual macros.
- Additionally, properties about the connections (wires) can also be added.

Interfacing with DB - Database

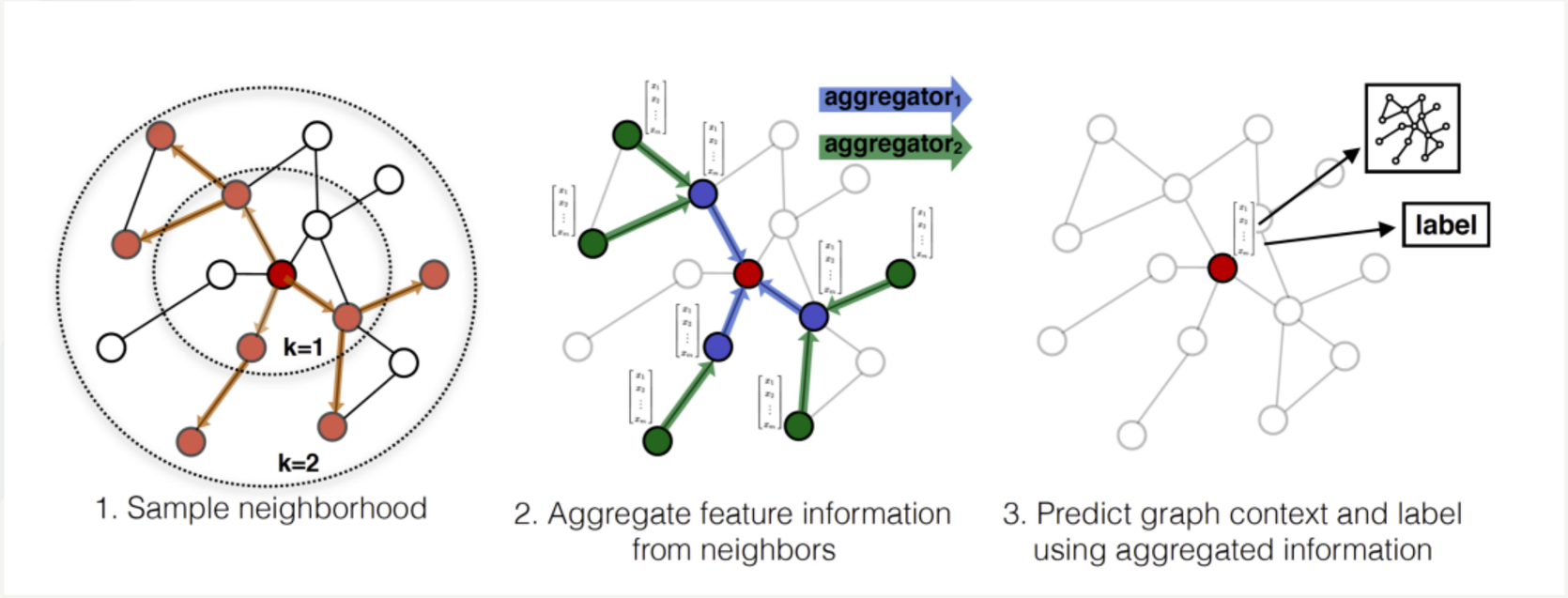
- Neo4J will be used as the database to store and query the graph data coming from the parser.
- Some of its useful features include:
  - Built and designed specifically for storing graph data
  - Easy to add properties to nodes and edges.
  - Good visualisation tools available (eg. Neo4J Bloom).
  - Has in-built Data-Science library for processing tasks.

Interfacing with DB - Embedding

- The graph data needs to be processed into form of vectors that the downstream ML models can understand.
- Similar to text-to-vector embedding used in NLP. (Graph-to-vector in this case)
- Several algorithms that could be used here.
  - Currently FastRP and GraphSAGE are explored.
  - Working on finding better algorithms.
- Gives output of n (embedding dimension) vectors of floats.

Graph Embedding - GraphSAGE

- **Inductive Algorithm that uses the node feature information.**
- **Trains Aggregate functions and set of weights via supervised and unsupervised methods.**



GraphSAGE paper : https://arxiv.org/pdf/1706.02216.pdf

Graph Embedding - GraphSAGE

$$
\begin{algorithm}
\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}\,; \\
\textbf{for } k = 1 \ldots K \textbf{ do} \\
\quad \textbf{for } v \in \mathcal{V} \textbf{ do} \\
\quad\quad \mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}); \\
\quad\quad \mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right) \\
\quad \textbf{end} \\
\quad \mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V} \\
\textbf{end} \\
\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}
\end{algorithm}
$$

- **Forward Propagation (after training)**
  - Embedding vectors initialised to feature vectors of each nodes.
  - Vector computed from aggregation of neighbours. This is concatenated with vector at the current node.
  - Forward Pass through non-linearity.
  - Normalisation

GraphSAGE paper : https://arxiv.org/pdf/1706.02216.pdf

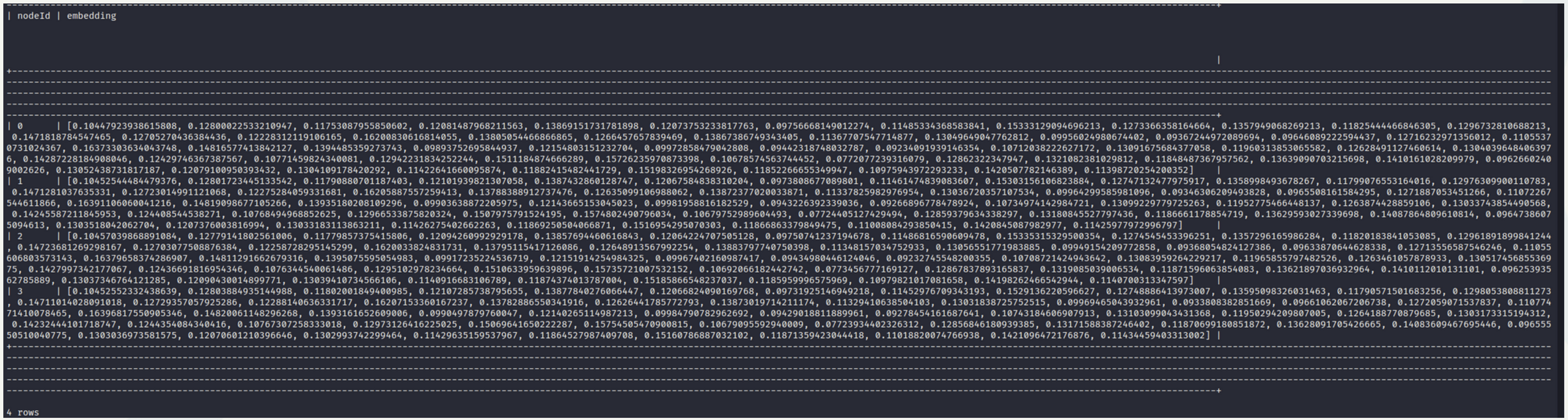
Graph Embedding - GraphSAGE

- **Training Phase**
  - Can be done in supervised or unsupervised manner
  - Nodes would be labelled and cross-entropy loss can be computed in supervised learning
  - Similarity measure used as loss function in case of unsupervised learning.
  - We plan to modify this to suit our needs. (i.e. include parameters that would apply constraints)

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right)$$

GraphSAGE paper : https://arxiv.org/pdf/1706.02216.pdf

Graph Embedding - GraphSAGE



nodeId	embedding
0	[0.10447923938615808, 0.12800022533210947, 0.11753087955850602, 0.12081487968211563, 0.13869151731781898, 0.12073753233817763, 0.09756668149012274, 0.11485334368583841, 0.15333129094696213, 0.12733663581646664, 0.1357949068269213, 0.11825444466846305, 0.1296732810688213, 0.1471818784547465, 0.12705270436384436, 0.12228312119106165, 0.16200830616814055, 0.13805054466866865, 0.1266457657839469, 0.13867386749343405, 0.11367707547714877, 0.13049649047762812, 0.09956024980674402, 0.09367244972089694, 0.09646089222594437, 0.12716232971356012, 0.11055370731024367, 0.16373303634043748, 0.14816577413842127, 0.1394485359273743, 0.09893752695844937, 0.12154803151232704, 0.09972858479042808, 0.09442318748032787, 0.09234091939146354, 0.10712038222627172, 0.13091675684377058, 0.11960313853065582, 0.12628491127460614, 0.13040396484063976, 0.14287228184908046, 0.12429746367387567, 0.10771459824340081, 0.12942231834252244, 0.1511184874666289, 0.15726235970873398, 0.10678574563744452, 0.0772077239316079, 0.12862322347947, 0.1321082381029812, 0.11848487367957562, 0.13639090703215698, 0.1410161028209979, 0.09626602409002626, 0.13052438731817187, 0.12079100950393432, 0.13041091784202292, 0.11422641660095874, 0.11882415482441729, 0.15198326954268926, 0.11852266655349947, 0.10975943972293233, 0.1420507782146389, 0.11398720254200352]
1	[0.10452544484479376, 0.12801723445133542, 0.11790880701187403, 0.12101939821307058, 0.1387432860128747, 0.12067584838310204, 0.0973808677089801, 0.11461474839083607, 0.15303156106823884, 0.12747132477975917, 0.1358998493678267, 0.11799076553164016, 0.12976309900110783, 0.1471281037635331, 0.12723014991121068, 0.12275284059331681, 0.16205887557259413, 0.13788388912737476, 0.12635099106988062, 0.13872377020033871, 0.11337825982976954, 0.13036720357107534, 0.09964299585981096, 0.09346306209493828, 0.0965508161584295, 0.1271887053451266, 0.11072267544611866, 0.16391106060041216, 0.14819098677105266, 0.1393518020810929 6, 0.09903638872205975, 0.12143665153045023, 0.09981958816182529, 0.0943226392339036, 0.09266896778478924, 0.10734974142984721, 0.13099229779725263, 0.11952775466448137, 0.12638744288591 06, 0.13033743854490568, 0.14245587211845953, 0.1244085445382 71, 0.10768494968852625, 0.12966533875820324, 0.150797579152 4195, 0.1574802490796034, 0.10679752989604493, 0.07724405127429494, 0.12859379634338297, 0.13180845527797436, 0.11866611788547 19, 0.13629593027339698, 0.14087864809610814, 0.096473860750 94613, 0.13035180420627 04, 0.12073760038169 94, 0.13033183113863211, 0.11426275402662263, 0.11869250504066871, 0.15169542950703 03, 0.11866863379849475, 0.11008084293850415, 0.14208450879829 77, 0.11425977972996797]
2	[0.10457039868891084, 0.12779141802561006, 0.11779857375415806, 0.12094260992929178, 0.13857694460616843, 0.12064224707505128, 0.09750741237194678, 0.11486816590609478, 0.15335315329500354, 0.1274545453396251, 0.1357296165986284, 0.11820183841053085, 0.12961891899841244, 0.14723681269298167, 0.12703077508876384, 0.12258728295145299, 0.16200338248 31731, 0.13795115417126086, 0.12648913567992254, 0.13883797740750398, 0.11348157034752933, 0.13056551771983885, 0.09949154209772858, 0.09368054824127386, 0.09633870644628338, 0.12713556587546246, 0.11055606803573143, 0.16379658374286907, 0.14811291662679316, 0.13950755950549 83, 0.09917235224536719, 0.12151914254984325, 0.09967402160987417, 0.09434980446124046, 0.09232745548200355, 0.10708721424943642, 0.13083959264229217, 0.11965855797482526, 0.12634610578789 33, 0.13051745685536975, 0.14279973421770 67, 0.12436691816954346, 0.10763445400614 86, 0.12951029782346 64, 0.15106339959 63896, 0.15735721007532152, 0.10692066182442742, 0.0773456777169127, 0.12867837893165837, 0.13190850390065 34, 0.11871596063854083, 0.13621897036932964, 0.14101120101311 01, 0.09625393562785889, 0.13037346764121285, 0.12090430014899771, 0.13039410734566106, 0.11409166831067 89, 0.11874374013787004, 0.15185866548237037, 0.11859599965759 69, 0.10979821017081658, 0.14198262466542944, 0.11407003133475 97]
3	[0.10452552332438639, 0.12803884935144988, 0.11802001849400985, 0.12107285738795655, 0.13877840276066447, 0.12066824090169768, 0.09731925146949218, 0.11452976709343193, 0.15291362205966 27, 0.12748886413973007, 0.13595098326031463, 0.11790571501683256, 0.12980538088112 73, 0.14711014028091018, 0.12729357057925286, 0.12288140636331717, 0.16207153360167237, 0.13782886550341916, 0.12626441785772793, 0.13873019714211174, 0.11329410638504103, 0.13031838725752515, 0.09969465043932961, 0.09338083828515669, 0.09661062067206738, 0.12720590715378 37, 0.11077471410078465, 0.16396817550905346, 0.14820061148296268, 0.13931616526090 06, 0.09904978797600 47, 0.12140265114987213, 0.09984790782962692, 0.09429018811889961, 0.09278454161687641, 0.10743184606907913, 0.13103099043431368, 0.11950294209807005, 0.12641887708796 85, 0.13031733151943 12, 0.14232444101718747, 0.12443540843404 16, 0.10767307258333018, 0.12973126416225025, 0.15069641650222287, 0.15754505470900815, 0.10679095592940009, 0.07723934402326312, 0.12856846180939385, 0.13171588387246402, 0.11870699180851872, 0.13628091705426665, 0.14083609467695446, 0.09655550510040775, 0.13030369735815 75, 0.12070601210396646, 0.13029937429994 64, 0.11429635159537967, 0.11864527987409708, 0.15160786887032102, 0.11871359423044418, 0.11018820074766938, 0.14210964721768 76, 0.11434459403313002]

4 rows

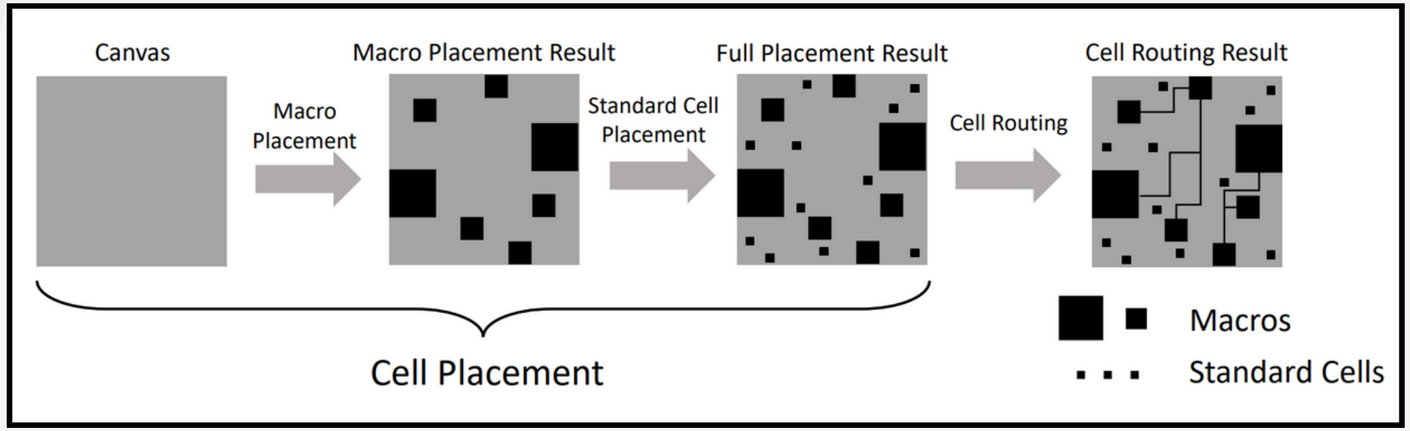
WireMask - Black Box Optimization

- A similar tool was explored that almost aligns with our problem statement. WireMask-BBO is a technique for Macro Placement. This approach uses the same constraints i.e., the Half Perimeter Wire Length (HPWL) and Congestion parameters for optimization.
- Analytical methods for simultaneous placement of MACROs and Standard Cells often cause overlapping and to fix this a simpler solution involved dividing the chip canvas into discrete grids and placing the MACROs step by step, checking for the most optimized results

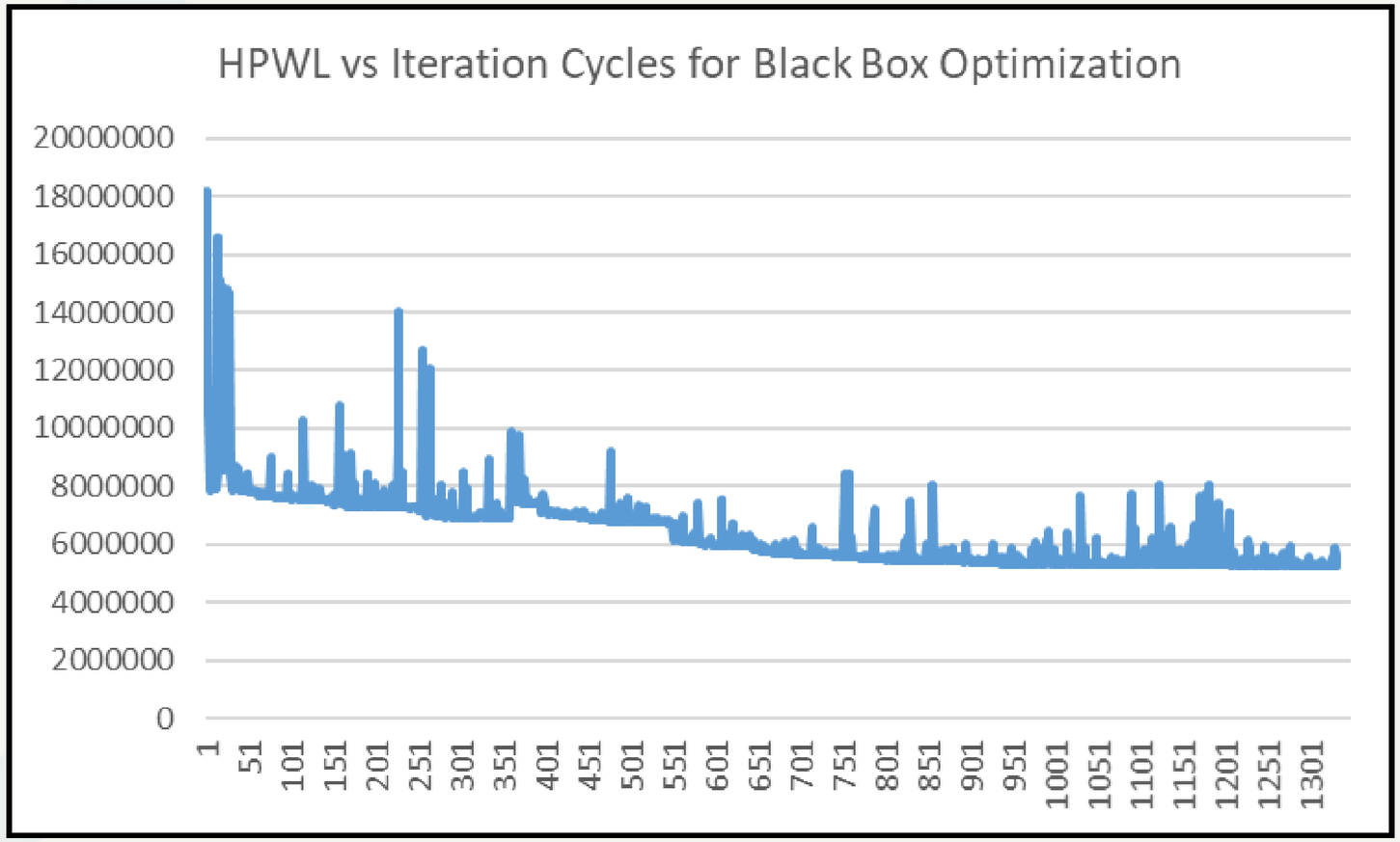
https://github.com/lamda-bbo/WireMask-BBO - for improved optimization over MASKPlace
https://github.com/laiyao1/maskplace - For visualization

WireMask - Black Box Optimization

- We can use the WireMask to record the increment if HPWL by moving a macro to each candidate grid on the chip canvas.
- The macros in the solution can be sequentially adjusted to the nearest solution.
- This algorithm can be further optimized with the prior knowledge of best datasets using ML techniques



WireMask - Black Box Optimization



The graph on the right shows a gradual descent of HPWL for an example circuit common to other optimization project like Maskplace, DREAMplace etc. and WireMask was observed to have slower settling results but an optimized HPWL value for a longer run.

Summary of overall Progress

- The parser for the cypher script is now built and running.
- Neo4j graph database is built with the necessary properties
- Working on embedding vectors produced using graph sage embedding technique
- Steps for Dataset generation using Virtuoso and Calibre were explored.
- WireMask Black Box Optimization was surveyed.
- Working to get the calibre lvs up and running
- Working on the type of model to use post-embedding

Any specific request to Micron

- Confirmation to proceed with the current flow.