RevDecode:
Enhancing Binary Function Matching with Context-Aware Graph Representations and Relevance Decoding

Tongwei Ren[1], Ronghan Che[1], Guin R. Gilman[1], Lorenzo De Carli[2], Robert J. Walls[1]
Affiliations: [1]Worcester Polytechnic Institute, [2]University of Calgary
Contacts: {tren, rche, grgilman, rjwalls}@wpi.edu, lorenzo.decarli@ucalgary.ca

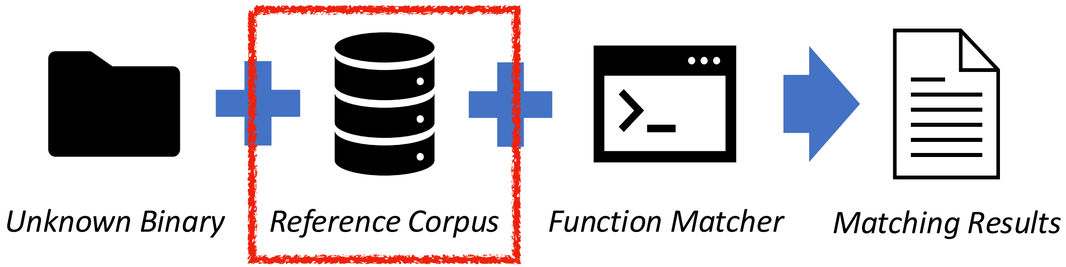
Problem Definition

- What is **Function matching**?

Contact: tren@wpi.edu

Problem Definition

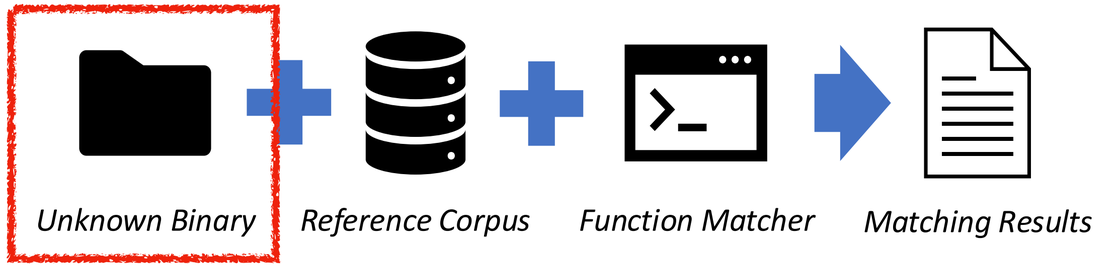
- What is **Function matching**?



Contact: tren@wpi.edu

Problem Definition

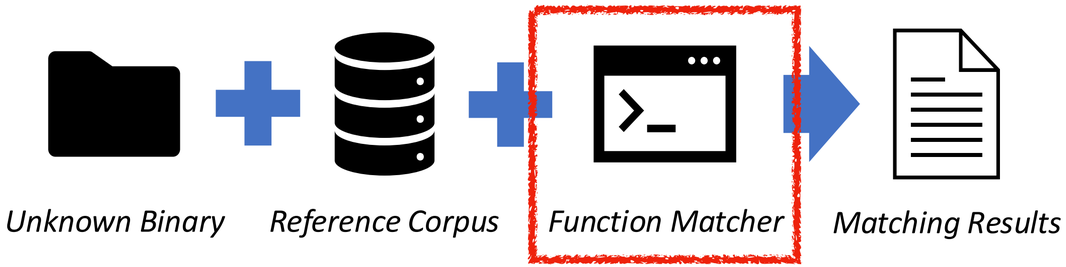
- What is **Function matching**?



Contact: tren@wpi.edu

Problem Definition

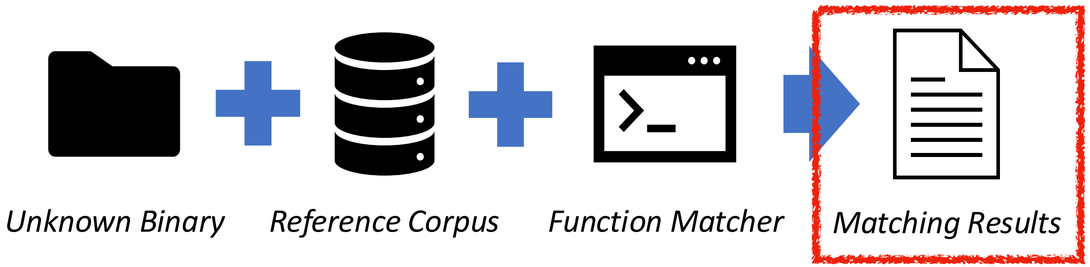
- What is **Function matching**?



Contact: tren@wpi.edu

Problem Definition

- What is **Function matching**?



Contact: tren@wpi.edu

Problem Definition

- What is **Function matching**?

- **It's useful in Reverse Engineering** for:
  - Identifying libraries in embedded firmware
  - Isolating vulnerable functions
  - Understanding code reuse in proprietary codebases

Contact: tren@wpi.edu

Problem Definition

- What is **Function matching**?

- **It's useful in Reverse Engineering** for:
  - Identifying libraries in embedded firmware
  - Isolating vulnerable functions
  - Understanding code reuse in proprietary codebases

- **It's hard** because of:
  - Compilation settings
  - Optimization levels
  - Version differences

Contact: tren@wpi.edu

Existing Approaches

- Solutions to the function matching problem, e.g. BSim, discovRE, SAFE, Gemini, use a variety of techniques:
  - Control flow structures and gadgets
  - Instruction and function embeddings
  - Signature matching
- Generally, these approaches generate abstract representations of functions and produce a similarity score, which measures how similar the unknown function to some other function in the corpus.

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Similarity vs. Relevance**
  - Matchers focus on **similarity**, but reverse engineers prioritize meaningful insights (**relevance**)

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Similarity vs. Relevance**
  - Matchers focus on **similarity**, but reverse engineers prioritize meaningful insights (**relevance**)
  - For example:
    - The unknown function: *custom_exts_copy* from *libssl-3.5.0*

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Similarity vs. Relevance**
  - Matchers focus on **similarity**, but reverse engineers prioritize meaningful insights (**relevance**)
  - For example:
    - The unknown function: *custom_exts_copy* from *libssl-3.5.0*
    - Reference corpus doesn't contain *libssl-3.5.0,* but it does contain *libssl-3.0.2*

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Similarity vs. Relevance**
  - Matchers focus on **similarity**, but reverse engineers prioritize meaningful insights (**relevance**)
  - For example:
    - The unknown function: *custom_exts_copy* from *libssl-3.5.0*
    - Reference corpus doesn't contain *libssl-3.5.0,* but it does contain *libssl-3.0.2*
    - *elf32_arm_size_stubs* from *libbfd-2.30* in the corpus is most structural similar and has 0.68 similarity score returned by discovRE

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Similarity vs. Relevance**
  - Matchers focus on **similarity**, but reverse engineers prioritize meaningful insights (**relevance**)
  - For example:
    - The unknown function: *custom_exts_copy* from *libssl-3.5.0*
    - Reference corpus doesn't contain *libssl-3.5.0,* but it does contain *libssl-3.0.2*
    - *elf32_arm_size_stubs* from *libbfd-2.30* in the corpus is most structural similar and has 0.68 similarity score returned by discovRE
    - An older version from *libssl-3.0.2* in the corpus is more meaningful for reverse engineers, but it only has a 0.44 similarity score

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Similarity vs. Relevance**
  - Matchers focus on **similarity**, but reverse engineers prioritize meaningful insights (**relevance**)
  - For example:
    - The unknown function: *custom_exts_copy* from *libssl-3.5.0*
    - Reference corpus doesn't contain *libssl-3.5.0,* but it does contain *libssl-3.0.2*
    - *elf32_arm_size_stubs* from *libbfd-2.30* in the corpus is most structural similar and has 0.68 similarity score returned by discovRE
    - An older version from *libssl-3.0.2* in the corpus is more meaningful for reverse engineers, but it only has a 0.44 similarity score
    - Even though this older version has lower similarity score, it is more relevant than *elf32_arm_size_stubs* from *libbfd-2.30*

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Incomplete Corpora**
  - Impractical to have exhaustive function database
  - E.g. Corpus includes the *libbfd-1.0* to *3.0*, but the unknown binary has functions from *libbfd-3.5*

Contact: tren@wpi.edu

Limitations in Existing Approaches

- **Incomplete Corpora**
  - Impractical to have exhaustive function database
  - E.g. Corpus includes the *libbfd-1.0* to *3.0*, but the unknown binary has functions from *libbfd-3.5*
- **Ambiguous Matches**
  - Common or similar functions difficult to distinguish
  - E.g. The same function from different versions of the same library

Contact: tren@wpi.edu

RevDecode

Key Idea

Relevance Decoding from Context

- Identify function relevance by leveraging surrounding contextual information
- Context includes interdependencies and contextual relationships—from both individual functions and their binary environment
- Relevance decoding builds on the observation that existing matchers overlook the surrounding binary context—such as preceding and succeeding code segments—which provides insights into function relationships

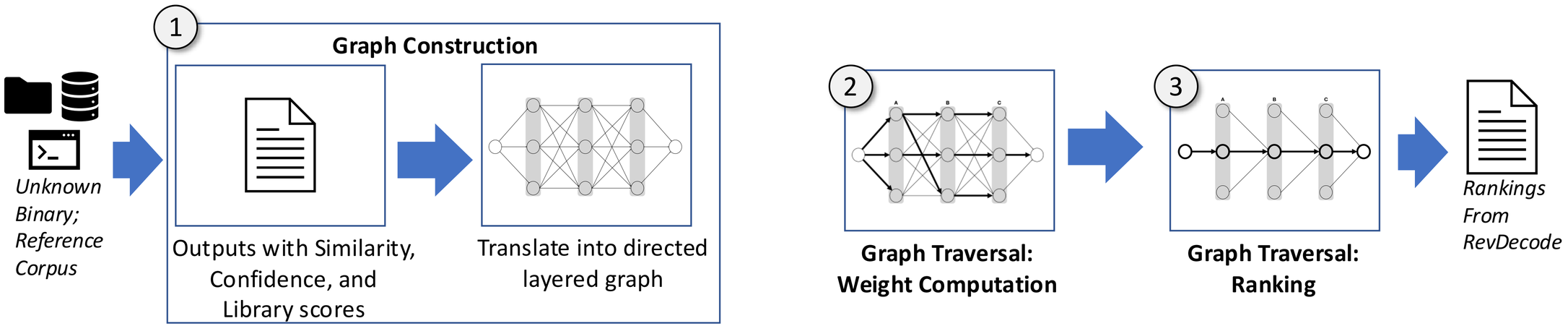
Contact: tren@wpi.edu

Key Idea
RevDecode

- Employs a graph representation to capture contextual signals and improve matching accuracy
- Not a new function-matching technique but a framework designed to enhance the accuracy of any underlying function matcher
- Bridges the gap between similarity and relevance, offering a more comprehensive solution to function matching challenges.

Contact: tren@wpi.edu

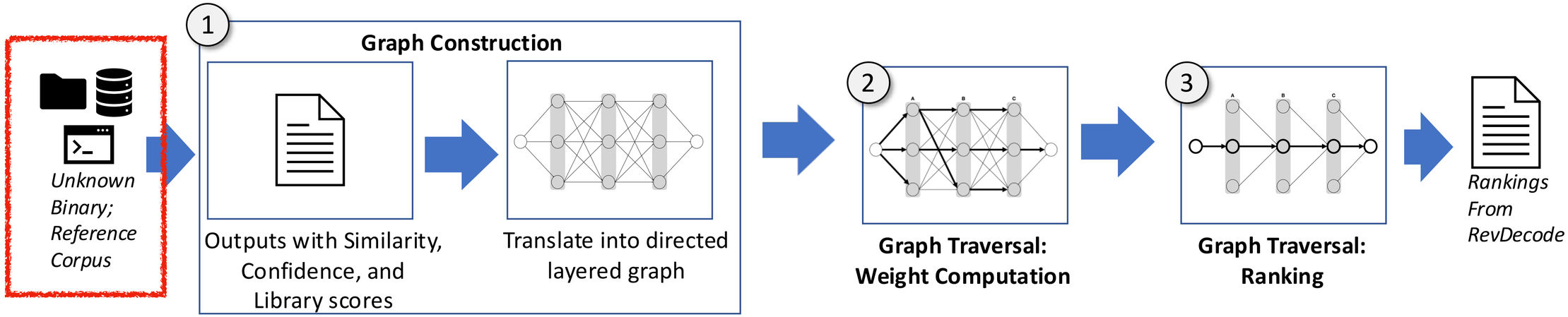
RevDecode Overview



Contact: tren@wpi.edu

RevDecode Overview

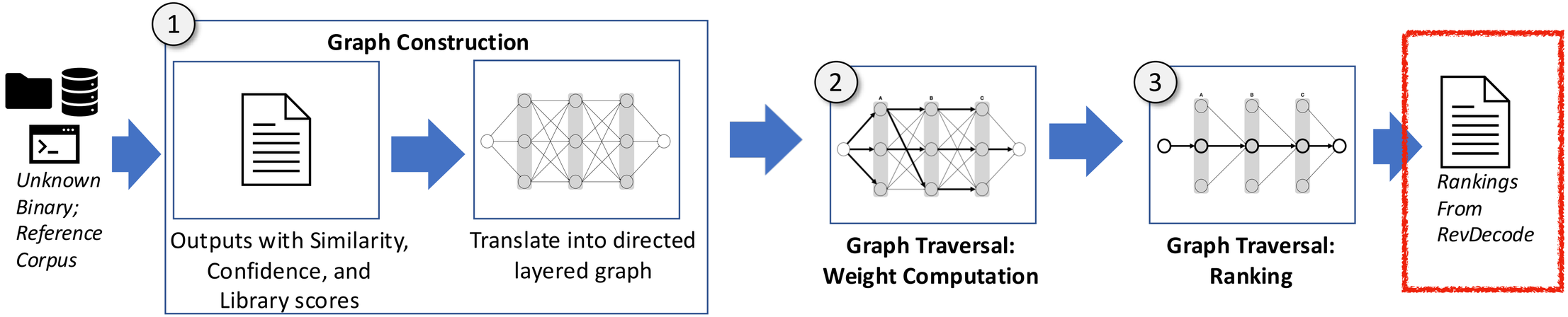
- **Input**: Unknown binary, reference corpus



Contact: tren@wpi.edu

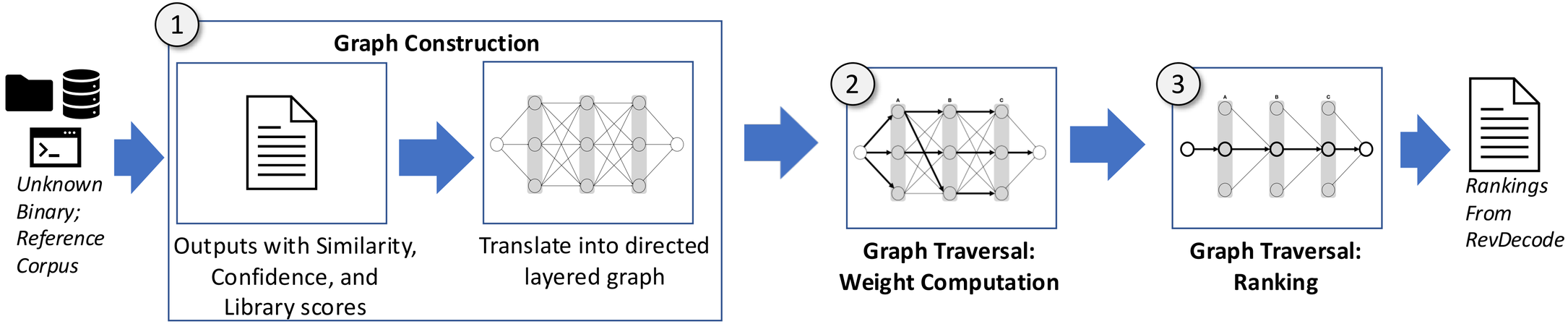
RevDecode Overview

- **Input**: Unknown binary, reference corpus
- **Output**: A ranking of matches for each unknown function



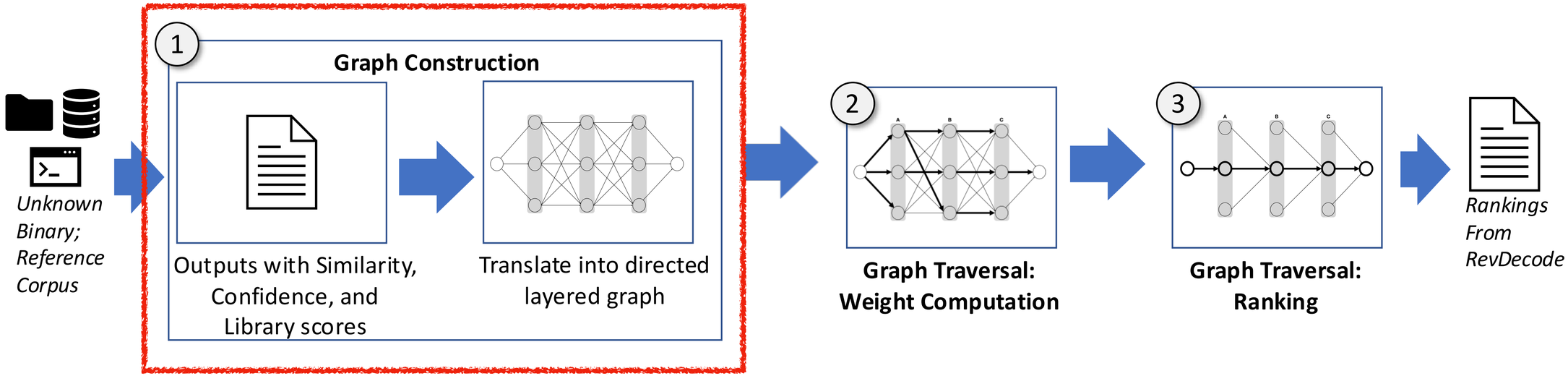
RevDecode Overview

- **Input**: Unknown binary, reference corpus
- **Output**: A ranking of matches for each unknown function
- **Three phases**:
  1. Graph Construction
  2. Graph Traversal: Weight Computation
  3. Graph Traversal: Ranking



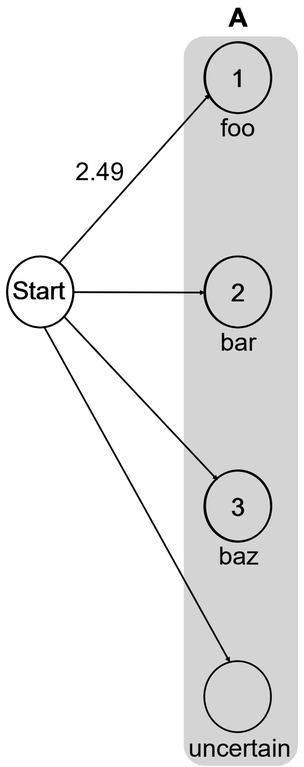
RevDecode Overview

- **Input**: Unknown binary, reference corpus
- **Output**: A ranking of matches for each unknown function
- **Three phases**:
  1. Graph Construction
  2. Graph Traversal: Weight Computation
  3. Graph Traversal: Ranking



Phase 1
Graph Construction

- **Idea:** Translating function matcher outputs to a directed layered graph representation and encode function interdependencies into the graph by setting **connections** between nodes and **weights** on edges

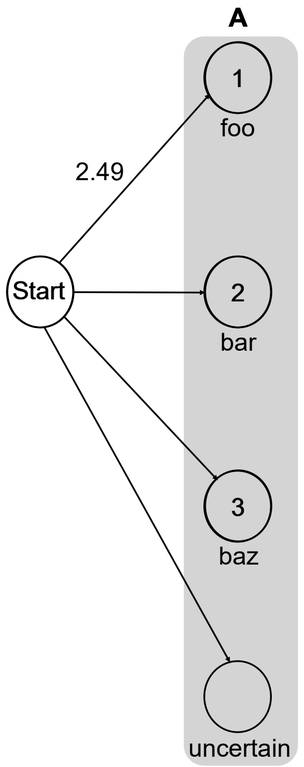


Contact: tren@wpi.edu

Phase 1

Graph Construction

- **Idea:** Translating function matcher outputs to a directed layered graph representation and encode function interdependencies into the graph by setting **connections** between nodes and **weights** on edges
- **Columns:** Each column represents an unknown function from the unknown binary and follows the function sequence in the binary

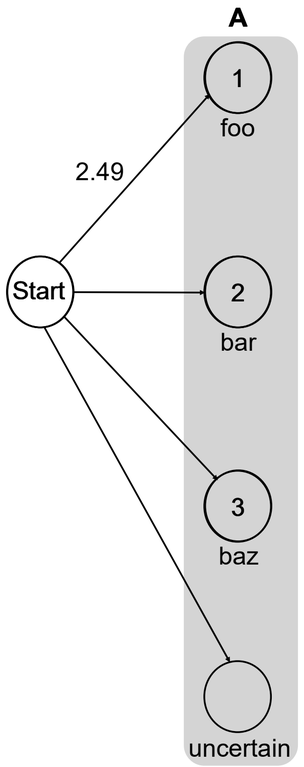


Contact: tren@wpi.edu

Phase 1

Graph Construction

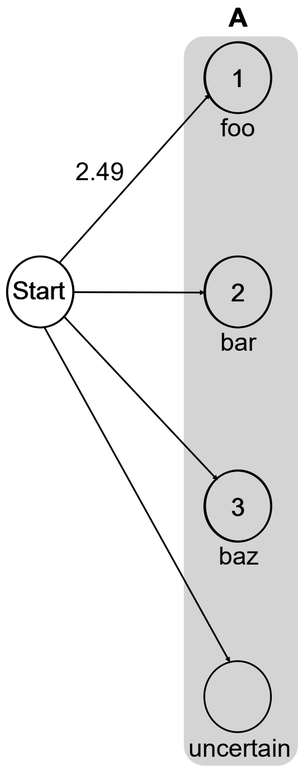
- **Idea:** Translating function matcher outputs to a directed layered graph representation and encode function interdependencies into the graph by setting **connections** between nodes and **weights** on edges
- **Columns:** Each column represents an unknown function from the unknown binary and follows the function sequence in the binary
- **Nodes:** Each node in each column represents a match returned by function matcher



Phase 1

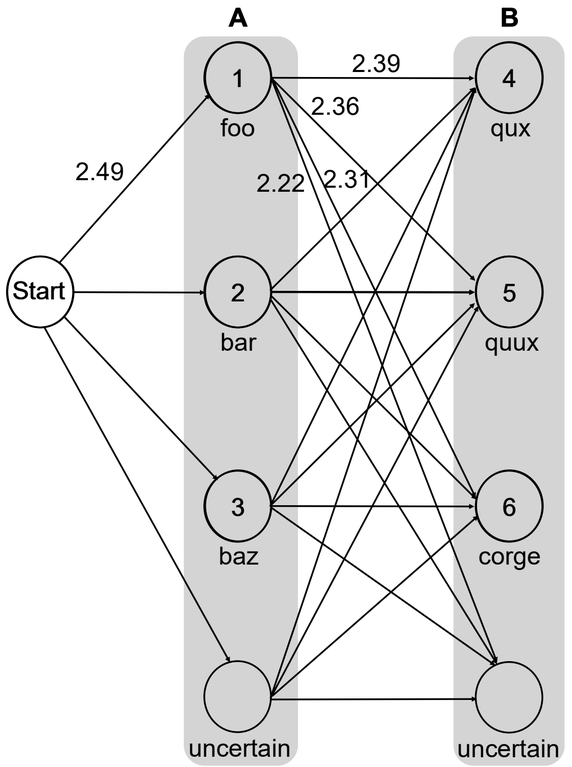
Graph Construction

- **Idea:** Translating function matcher outputs to a directed layered graph representation and encode function interdependencies into the graph by setting **connections** between nodes and **weights** on edges
- **Columns:** Each column represents an unknown function from the unknown binary and follows the function sequence in the binary
- **Nodes:** Each node in each column represents a match returned by function matcher
- Uncertain nodes manage incomplete corpora



Phase 1

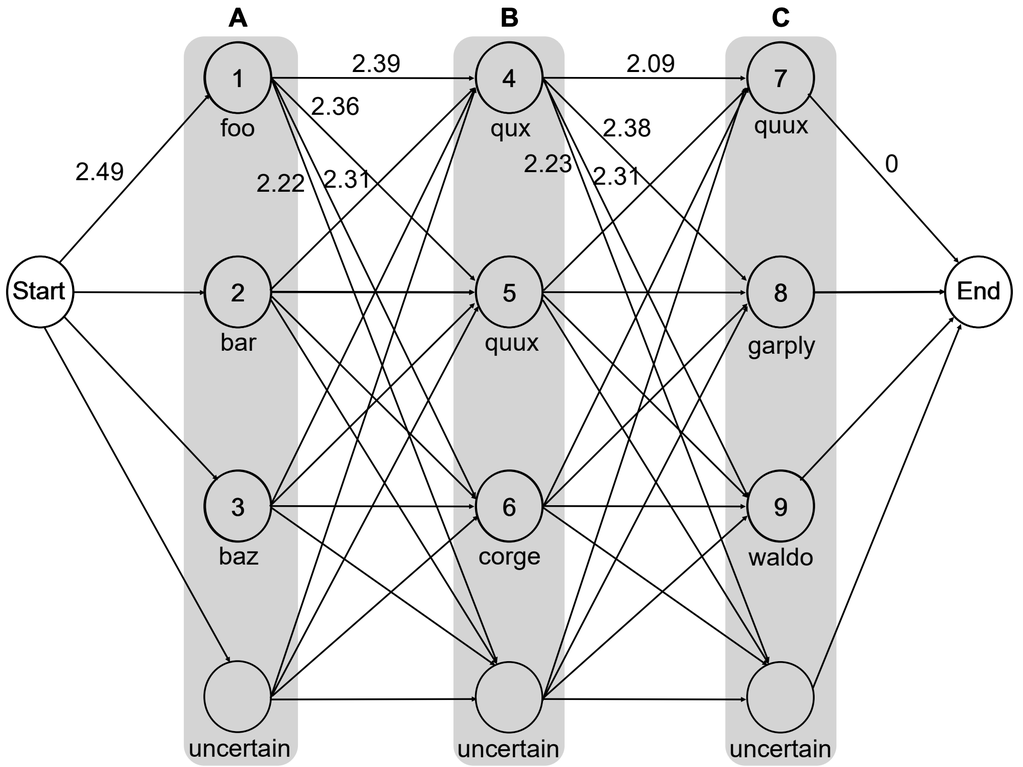
Graph Construction



Phase 1

Graph Construction

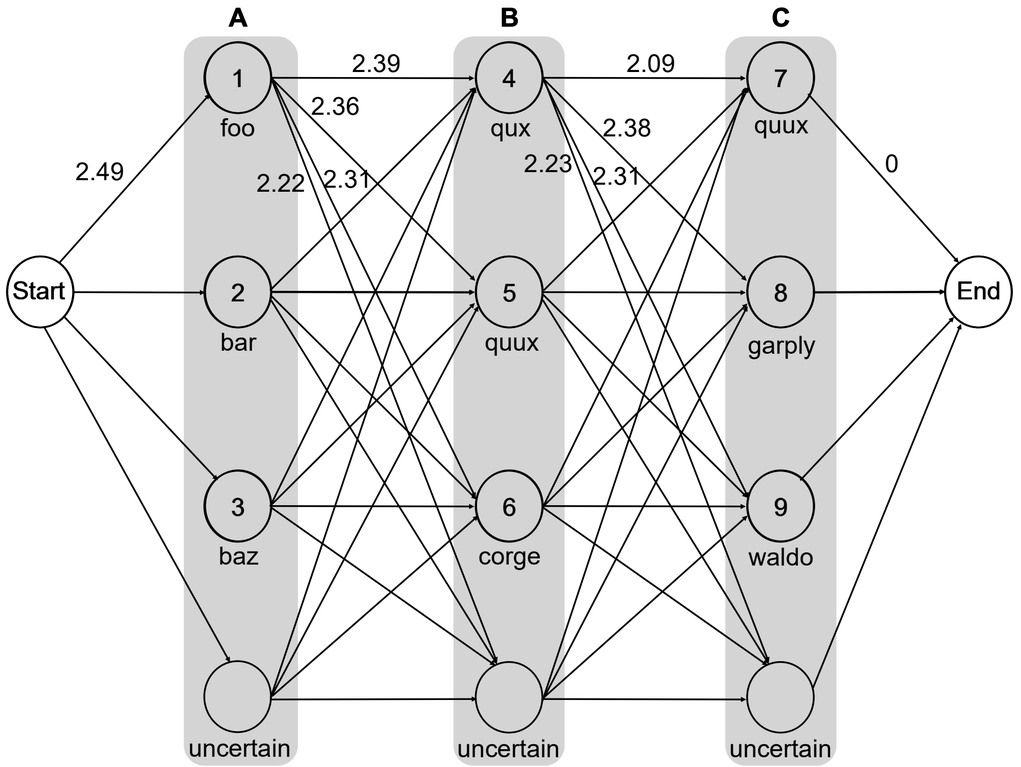
- **Weights** are the combination of:
  - **Similarity scores:** A measure of similarity between two functions, provided by existing function matcher



Phase 1

Graph Construction

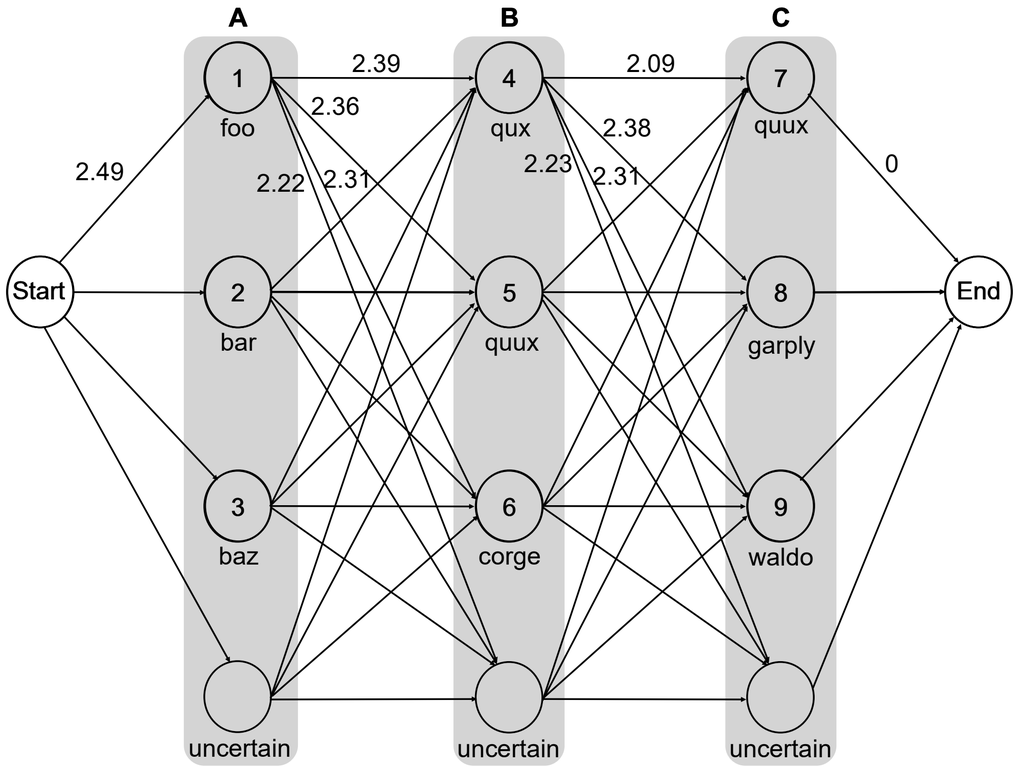
- **Weights** are the combination of:
  - **Similarity scores:** A measure of similarity between two functions, provided by existing function matcher
  - **Confidence scores:** More features are matched, more unique matched features are, higher confidence score it gets



Phase 1

Graph Construction

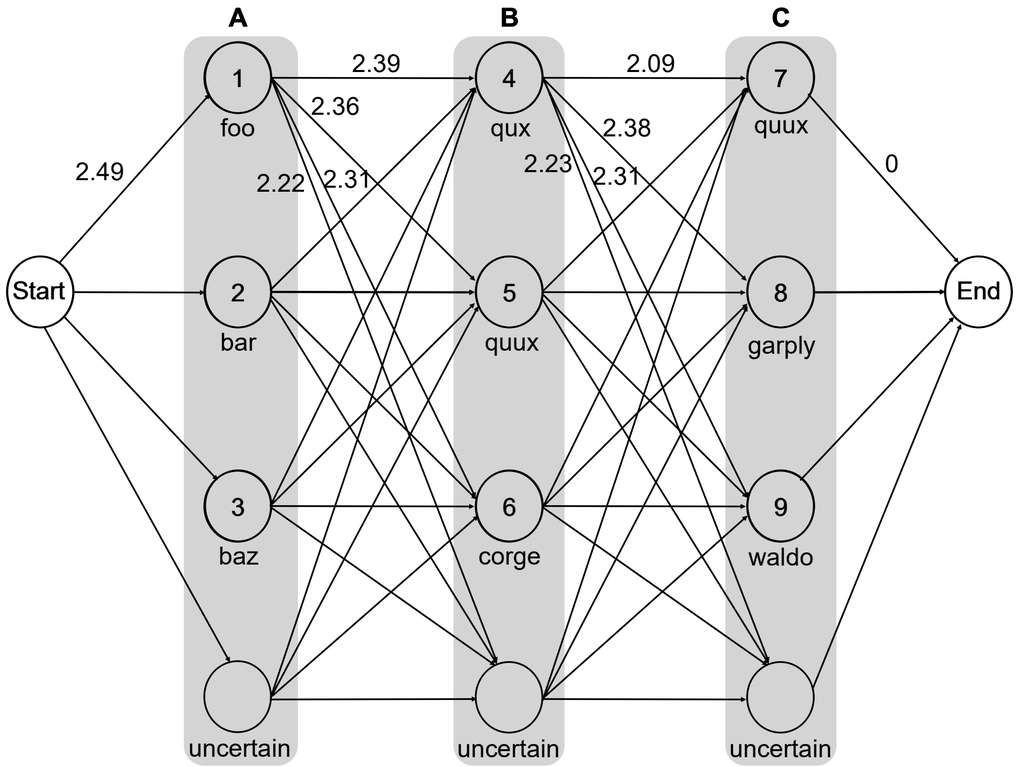
- **Weights** are the combination of:
  - **Similarity scores:** A measure of similarity between two functions, provided by existing function matcher
  - **Confidence scores:** More features are matched, more unique matched features are, higher confidence score it gets
  - **Library scores:** More unique the match's library is in the corpus, higher the library score it gets



Phase 1

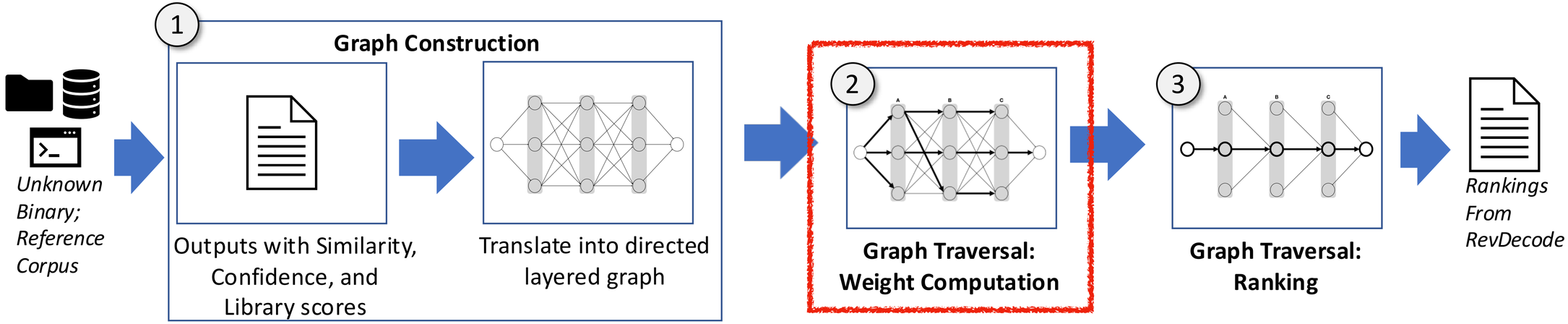
Graph Construction

- **Weights** are the combination of:
  - **Similarity scores:** A measure of similarity between two functions, provided by existing function matcher
  - **Confidence scores:** More features are matched, more unique matched features are, higher confidence score it gets
  - **Library scores:** More unique the match's library is in the corpus, higher the library score it gets
  - **Adjacency scores:** A value that quantifies the contextual relationship between two matches connected by an edge



RevDecode Overview

- **Input**: Unknown binary, reference corpus
- **Output**: A ranking of matches for each unknown function
- **Three phases**:
  1. Graph Construction
  2. Graph Traversal: Weight Computation
  3. Graph Traversal: Ranking

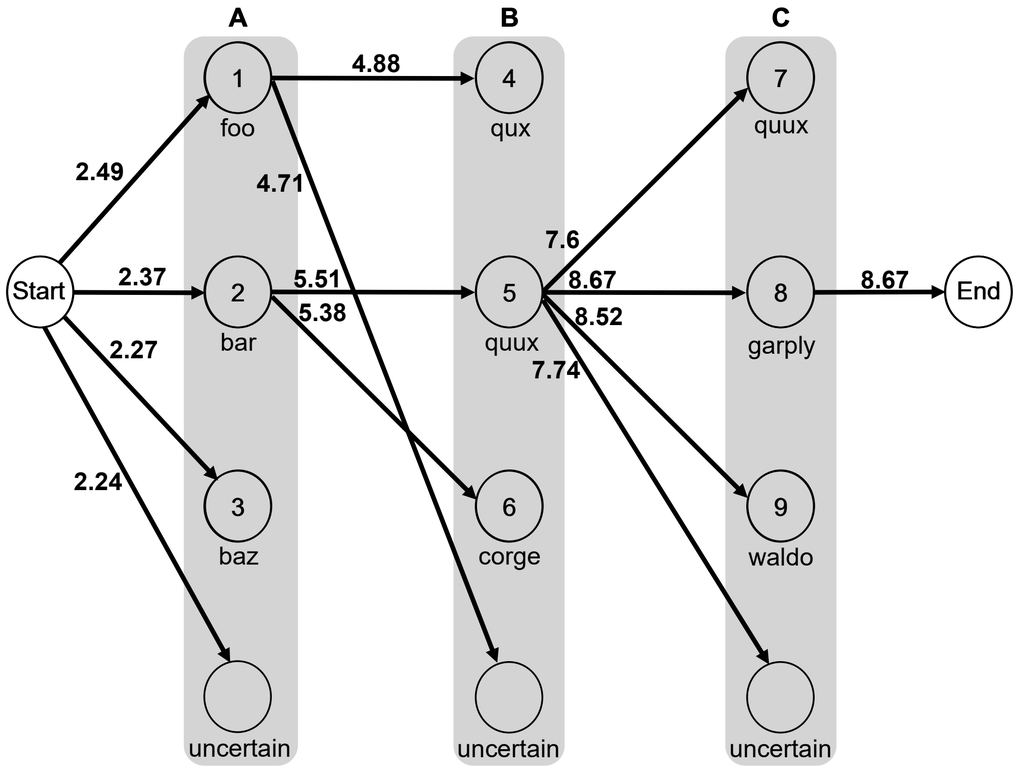


Contact: tren@wpi.edu

Phase 2

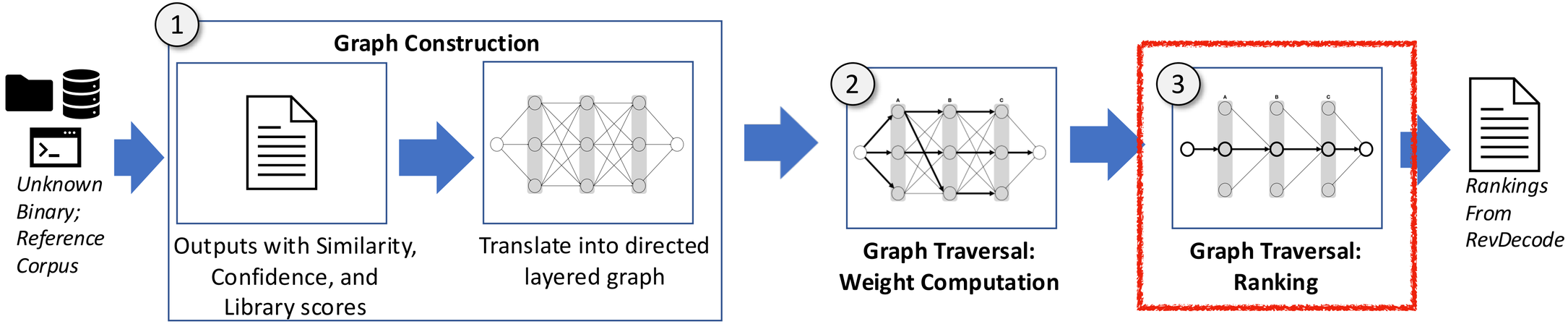
Graph Traversal: Weight Computation

- **Idea**: Identify all nodes that lie on at least one of the maximum weight paths
- **Viterbi-inspired forward pass** for efficient path computation



RevDecode Overview

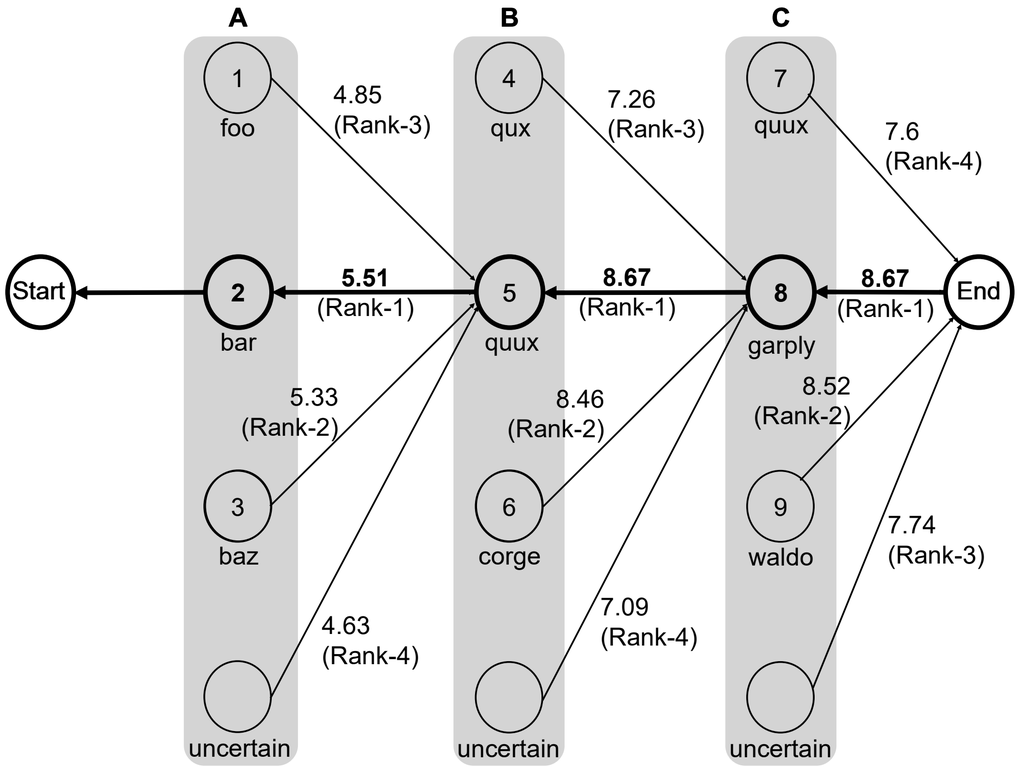
- **Input**: Unknown binary, reference corpus
- **Output**: A ranking of matches for each unknown function
- **Three phases**:
  1. Graph Construction
  2. Graph Traversal: Weight Computation
  3. Graph Traversal: Ranking



Phase 3

Graph Traversal: Ranking

- **Idea**: Nodes ranked based on their proximity to a maximum weight path
- **Backward pass** starts from end node
- **Output**: A ranking of matches for each unknown function



Evaluation

Evaluation Methodology

- **Datasets**:
  - General-purpose: 132 binaries, 262486 functions, 11 billions function matches.
  - Synthetic Frankenbinaries: 168 libraries, 107643 functions, 5.8 billions function matches
- **Metrics:** Tie-aware Normalized Discounted Cumulative Gain (NDCG), a measure of ranking quality
- **Matchers Evaluated:** BSim[1], discovRE[2], SAFE[3], Gemini[4]

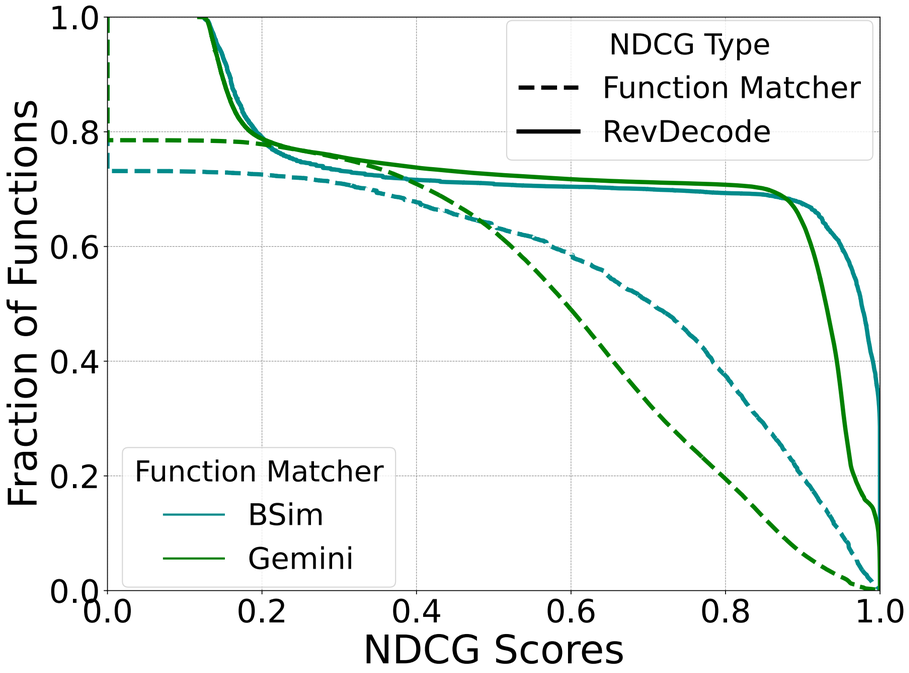
[1] Ghidra. https://ghidra-sre.org, 2019
[2] Sebastian Eschweiler, Khaled Yakdan, and Elmar Gerhards-Padilla. discovre: Efficient cross-architecture identification of bugs in binary code. 02 2016.
[3] Luca Massarelli, Giuseppe Antonio Di Luna, Fabio Petroni, Roberto Baldoni, and Leonardo Querzoni. Safe: Self-attentive function embeddings for binary similarity. Lecture Notes in Computer Science, page 309–329, 2019.
[4] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. Neural network-based graph embed- ding for cross-platform binary code similarity detection. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, page 363–376, New York, NY, USA, 2017. Association for Computing Machinery.

Contact: tren@wpi.edu

Evaluation Results

Ranking Effectiveness

- RevDecode significantly improves the quality of function rankings produced by existing matchers across both the general-purpose and frankenbinary datasets.
- General-purpose binaries: 56.3%-97.3% rankings are improved
- Frankenbinaries: 72.3%-98.8% rankings are improved



Contact: tren@wpi.edu

Evaluation Results

Ranking Effectiveness

- **Why improves?**

  By leveraging the graph structure and contextual signals from nearby strong matches**:**

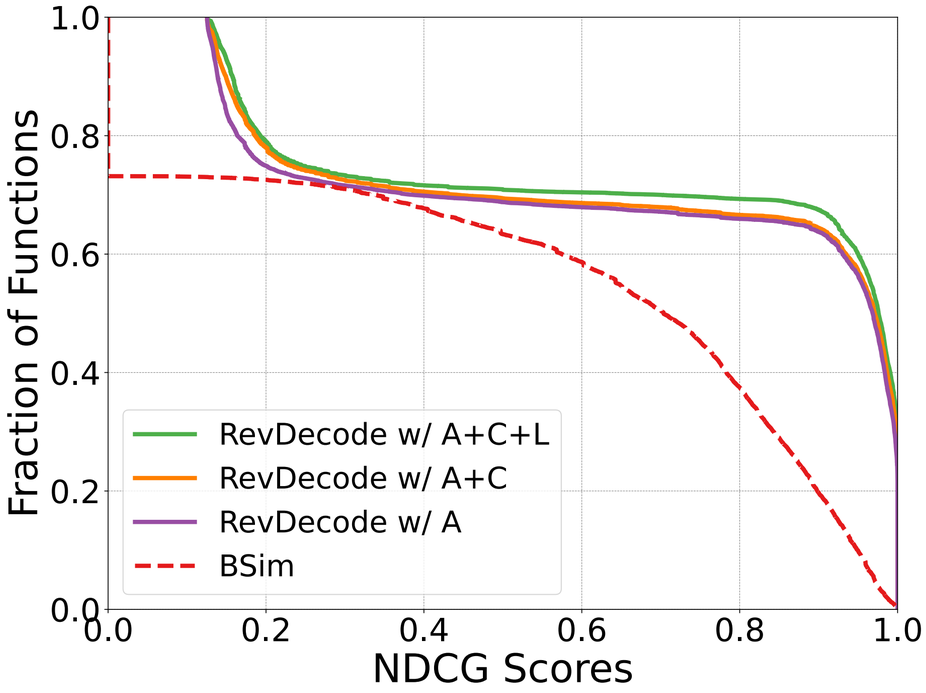
  - **Recovering relevant matches** that are structurally dissimilar due to version or compilation differences.
    - E.g. BSim, which misidentifies *uncompress* from *libz*(Ubuntu 20.04) as *get_absolute_expression* from *as-2.24*. RevDecode corrects the ranking, promoting the *uncompress* from *libz*(Ubuntu 14.04) from rank 187 to the top position because of the *libz* context provided by surrounding *libz* matches
  - **Resolving ambiguity** among functions that lack distinctive features
    - E.g. In the general-purpose dataset, 70,720 of the 202,058 improved BSim rankings (35%) fall into this category

Contact: tren@wpi.edu

Evaluation Results

Ranking Effectiveness

- **Ablation Study**: All components in weight are useful for the ranking refining. E.g. BSim's average NDCG is improved from:
  - 0.27 to 0.42 with similarity and adjacency scores
  - Then to 0.44 with confidence scores
  - Finally to 0.46 with library scores
- Uncertain node plays an important role in mitigating irrelevant context chains and led to a 7% decrease in context-chain-induced ranking errors.



Contact: tren@wpi.edu

Parallelizing Graph Traversal

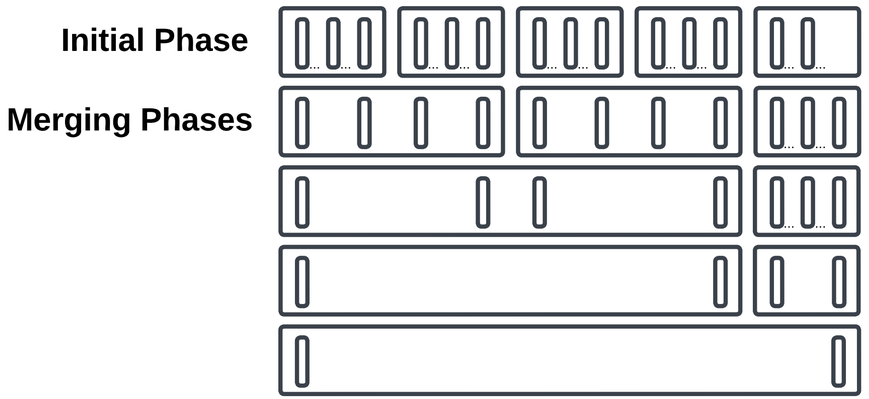
Challenges in Parallelizing RevDecode

- **Motivation:** Be scalable for large graph size (e.g. Unoptimized CPU-based implementation takes 40mins for an unknown binary having 900 functions, each function has 250 matches.)
- **Solution:** Parallel graph traversal algorithms
- **Challenges:**
  - **Sequential Dependencies:** Layer computation depends on previous layers, limiting naive parallelization
  - **Resource Utilization:** Efficient division of workload essential for maximum GPU resource use
  - **Memory Dependence:** Frequent accesses to cumulative weight matrix could create bottlenecks; memory access patterns require careful design

Contact: tren@wpi.edu

Parallel Traversal Algorithms

- **Naive Traversal:** Serves as a baseline, launching a single GPU thread block and walking the graph layer by layer, no additional optimization
- **Fine-Grained Traversal:** Parallel edge computations, efficient GPU utilization
- **Segment-Based Estimation Traversal:** Divide-and-conquer strategy for large graphs; trades slight accuracy loss for significant performance gain

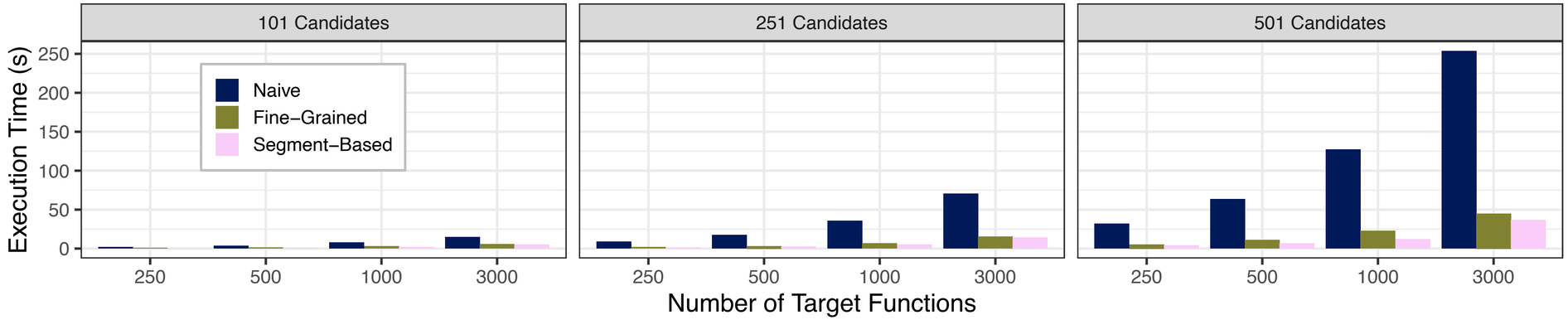


Contact: tren@wpi.edu

Evaluation Results

Parallel Algorithms Performance

- **Fine-Grained Traversal:**
  - Over 40x speedup relative to a simple CPU implementation
  - ~5x speedup over naive GPU-based traversal
- **Segment-Based Traversal:**
  - 1.39x speedup for total runtime, verse fine-grained traversal,
  - Additional ~3.8x speedup for forward-pass time
  - The cost is 0.003 NDCG reduction



Contact: tren@wpi.edu

Summary

Summary

- Relevance decoding identifies function relevance by leveraging surrounding contextual information.
- RevDecode significantly improves function matcher results by:
  - Converting results optimization problem to a graph exploration problem
  - Leveraging functions' interdependencies and contextual signals during the graph exploration process
- GPU acceleration ensures the scalability by parallelizing the graph traversal algorithm
- Evaluation demonstrates that RevDecode significantly enhances function matcher performance by reducing ambiguities and improving ranking accuracy

Contact: tren@wpi.edu

Thanks for listening!

RevDecode:
Enhancing Binary Function Matching with Context-Aware Graph Representations and Relevance Decoding

Tongwei Ren[1], Ronghan Che[1], Guin R. Gilman[1], Lorenzo De Carli[2], Robert J. Walls[1]
Affiliations: [1]Worcester Polytechnic Institute, [2]University of Calgary
Contacts: {tren, rche, grgilman, rjwalls}@wpi.edu, lorenzo.decarli@ucalgary.ca

WPI