Wave-Based Search Algorithm in Semantic and Extended Networks, Parallelism and Applications

Merab Pkhovelishvili
Faculty of Informatics and Control Systems
Georgian Technical University
Tbilisi, Georgia
e-mail: m.pkhovelishvili@gtu.ge

Natela Archvadze
Department of Computer Sciences
Faculty of Exact and Natural Sciences
Ivane Javakhishvili Tbilisi State University
Tbilisi, Georgia
e-mail: natela.archvadze@tsu.ge

Lia Shetsiruli
Visiting Professor
Grigor Robakidze University
Batumi, Georgia
e-mail: lika77u@gmail.com

***Abstract*—This paper explores the theoretical and practical aspects of the Wave-Based Search algorithm in modern, complex graph structures. An abstract model of wave search is introduced, based on the Breadth-First Search (BFS) method, yet offering a high degree of parallelism. The algorithm is discussed in the context of tree structures, semantic networks, and extended graphs, where parallel processing of wavefronts and results folding is possible. The paper also presents implementation examples using the F# programming language.**

***Keywords*—Wave-Based Search, Breadth-First Search, Semantic Networks, Parallel Algorithms, Subnetwork Integration.**

I. Introduction

In modern data structures—particularly semantic networks and complex graph models—efficient information retrieval is a critical task. Traditional search algorithms, such as Depth-First Search (DFS) and Breadth-First Search (BFS), provide a strong foundation but are inherently limited by their sequential execution nature.

Wave-Based Search is a parallelized version of BFS that divides the search process into waves, or levels. This structure enables the use of parallel processing resources, especially when dealing with complex or segmented structures.

This study examines the principles of Wave-Based Search, its parallelization capabilities, and its application in semantic networks and subnetworks. It also provides formal justification and performance analysis, along with practical examples and an implementation in F#.

II. Related Work

In one of the earliest works on wave search, Archvadze and Pkhovelishvili (2012) discussed the tree-structured representation of the Georgian language dictionary and its further processing using functional programming tools.

In recent years, wavefront-based parallel search algorithms have received increasing attention, particularly in large-scale and highly connected graphs. Liu et al. (2024) introduced PASGAL—a parallel-optimized graph analytics framework that employs a wavefront-style approach with vertical granularity control (VGC) to address performance challenges in wide-diameter graphs.

Zhang et al. (2023) developed Parallel Clustered BFS (C-BFS), which combines bit-level optimization with multi-threaded support to significantly reduce latency in landmark labeling and shortest-path queries. This model effectively extends wave-based search to multi-source scenarios.

Bilo et al. (2023) examined the deterministic performance of Bidirectional BFS in real-world networks, demonstrating that sublinear performance can be achieved through proper wave scheduling.

Collectively, these studies confirm the high potential of the wave-based approach for both parallel graph processing and knowledge-based systems, aligning with the purpose of the present paper: to demonstrate the effective use of wave-based search in semantic and extended networks through subnetwork integration.

These studies confirm the high potential of wave-based approaches for both parallel graph processing and knowledge-based systems, aligning with the purpose of the present paper.

III. Motivation

The core motivation of this study arises from a fundamental question: Why do traditional search methods fail to operate efficiently in semantic systems?

Traditional graph search algorithms assume structural graphs where connections are syntactic rather than semantic. They are effective when:

- The graph is standardized (e.g., road networks).
- Connections represent clear physical or logical routes.
- Node semantics are irrelevant to the search process.

However, semantic networks differ: nodes are connected not only structurally but also by meaning and context. For example, the concept “cat” may connect to “animal”, “pet”, or “predator”. In such networks, traditional algorithms face key limitations:

- They cannot account for meaning or context.
- They fail to address polysemy or the intensity of associations.
- They operate sequentially, lacking parallelism, which makes them inefficient for large semantic graphs.

Modern applications where faster and parallel semantic search is required include:

1. AI Assistants (e.g., Siri, Alexa, Google Assistant).
2. Large Language Models (e.g., GPT, BERT, LLaMA).
3. Semantic Search Engines (e.g., Google Scholar, Semantic Scholar).
4. Medical Decision Support Systems.
5. Cybersecurity and Intelligence Analysis.

IV. Wave-Based Search in Tree Structures

WBS is rooted in BFS but enhances parallelism by evaluating nodes in waves simultaneously.

Core Principle.

The structure of wave progression can be illustrated as follows (see Fig.):

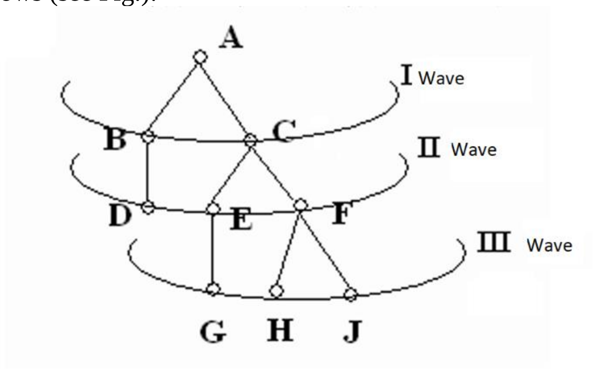

Fig. Waves during search:

- A – root
- B, C – first wave nodes
- D, E, F – second wave
- G, H, J – third wave, and so on

The search starts at the root node. If the target value is found at the root, the search ends successfully. Otherwise, the process continues to the first wave—the child nodes directly connected to the root. The key advantage here is that all nodes within the current wave can be evaluated simultaneously, significantly improving performance.

If the target is not found in the first wave, the search proceeds to the second wave—children of the first wave nodes. This process continues until the target is found or all leaf nodes are exhausted.

- Search begins at the root node.
- If the target is found at the root, the process terminates.
- Otherwise, the first wave (child nodes) is evaluated in parallel.
- If unsuccessful, the second wave (children of the first wave) is processed, and so on.

Example.

In a tree structure, a root node A connects to children B and C. They form the first wave.

The next wave includes their children, such as D, E, F.

The third wave may contain G, H, J, and so forth.

Key Innovation.

- Semantic relevance determines the propagation and intensity of the wave.
- Waves do not expand in all directions but selectively follow the strongest paths.
- Processes across neighboring nodes can occur in parallel.

This is analogous to a droplet falling into water: the ripple spreads outward, but not uniformly; it propagates more strongly where resistance is lower or semantic relevance is higher.

Wave-Based Search is a method for parallel information retrieval in tree structures, based on the principles of Breadth-First Search (BFS). Unlike traditional BFS, it emphasizes the parallel evaluation of each wave, making it particularly suited for multi-core or clustered architectures.

A. Parallelization Capabilities

All nodes within a given wave are independent of each other, which allows for parallel processing. This approach is especially effective when searching large trees or when real-time responsiveness is required.

The Wave-Based Search method can be extended to semantic networks by leveraging the concept of **subnetworks**. This enables the representation and processing of complex knowledge structures in a distributed and parallelized manner.

V. Functional Programming and Parallelism

Functional programming languages (e.g., F#, Haskell, OCaml) are well-suited for implementing WBS:

- Recursion: Each wave depends on the previous, aligning naturally with recursive functions.
- Immutability: Reduces side effects, making parallel execution safe and efficient.
- Built-in Parallelism: Libraries such as Parallel.map in F# or par in Haskell enable straightforward workload distribution.

Thus, functional programming offers both conceptual and technical compatibility with WBS.

VI. Implementation Example in F#: Wave-Based Search in Tree Structures

```
open System.Threading.Tasks

type Tree<'T> =
| Node of 'T * List<Tree<'T>>

// Parallel Wave Search
```

```
let rec waveSearchParallel
            (predicate: 'T -> bool)
          (tree: Tree<'T>) =let rec searchWave wave =
match wave with
| [] -> None
| _ ->
// in parallel to check Current Wave
let results =
waves
|> List.map (fun (Node(value, _)) -> Task.Run(fun () -> if predicate value then Some value else None))
|> Task.WhenAll
|> fun t -> t.Result |> Array.choose id

if results.Length > 0 then
Some(results.[0]) // First Result Return
else
let nextWave =
waves
|> List.collect (fun (Node(_, children)) -> children)
searchWave nextWave
searchWave [tree]
```

Data usage Example:

```
let exampleTree =
Node(1, [
Node(2, [Node(5, []); Node(6, [])]);
Node(3, [Node(7, [])]);
Node(4, [])
])

let found = waveSearchParallel ((=) 7) exampleTree
match found with
| Some v -> printfn "Value found: %d" v
| None -> printfn "Value not found"
```

VII. Wave-Based SeDetailarch in Semantic Networks

Originally developed for tree-like structures, the Wave-Based Search method can be extended to semantic networks. This approach enables efficient depth-limited search within interconnected knowledge structures, especially in multiprocessor environments.

Also known as **multi-level parallel search**, Wave-Based Search begins at a predefined point in the network (a root or starting node) and proceeds in successive **waves** or layers, each including adjacent nodes. Each wave contains all previously unvisited nodes that are topologically equidistant from the starting point. Unlike depth-first search, this method allows all nodes within a layer to be evaluated in parallel.

A. *Algorithmic Constraints for Semantic Networks*

To adapt Wave-Based Search to semantic networks, additional constraints must be implemented:

- **Depth limitation**: A maximum number of waves (or levels) must be defined to prevent uncontrolled traversal.
- **Visited node marking**: Each visited node must be properly marked to avoid infinite loops in cyclic graphs.
- **Subnetwork integration** (as per Woods): Nodes may belong to subgraphs with differing structures or semantics, and the integration of these subgraphs must be accounted for.

Woods [5,6] introduced the concept of subnets—portions of semantic networks representing distinct structures or domains. This enables specialized processing:

- Subnetworks may represent distinct knowledge domains or linguistic layers (e.g., morphological analysis).
- Subnetworks are processed independently of the main network and therefore in parallel.
- Once processed, the results from subnetworks are reintegrated into the main network.

Example 1:
In a linguistic semantic network, the main graph may represent the syntactic structure of a sentence, while subnetworks handle the morphological or lexical analysis of individual words.
*Example 2***:**
In a monolingual dictionary's semantic network, subnetworks may contain multilingual translations or contextual explanations, often with distinct internal structures.
Illustrative Example.
Start node: “Toy”

- First wave: Toy → Child; Toy → Game; Toy → Design.
- Second wave: Child → Motor Skills; Game → Communication; Design → Visual Perception.

All nodes within a given wave are processed simultaneously, ensuring broad coverage with contextual relevance.

B. Benefits of Parallel Subnetwork Processing

In Woods’ model, the independence of subnetworks aligns well with modern multi-core and distributed architectures. Each subnetwork can be processed simultaneously, which:

- Enhances performance in knowledge-based systems
- Enables advanced, reasoning-driven analysis in complex domains
- Supports scalability in artificial intelligence and natural language processing applications

C. F# Example (Simplified): Wave-Based Search with Subnetworks

```
let rec waveSearch graph startNode maxDepth matchFn =
let rec searchWave currentWave depth visited =
if depth > maxDepth || List.isEmpty currentWave then
None
else
match currentWave |> List. tryFind matchFn with
| Some node -> Some node
| None ->
let nextWave =
currentWave
|> List.collect (fun node -> graph.[node])
|> List.filter (fun n -> not (Set.contains n visited))
let visited' = Set.union visited (Set.ofList nextWave)
searchWave nextWave (depth + 1) visited'

searchWave [startNode] 0 (Set.singleton startNode)
```

Thus, Wave-Based Search—enhanced with subnetwork support and executed in parallel environments—becomes a powerful tool for **multicomponent search and inference** in semantic systems. The structurally diverse subnetwork model introduced by Woods enables expressive and generalized knowledge representation, particularly in **natural language processing** and **expert systems**.

Detailed Explanation of Constraints and Their Implementation

Content:

- Constraints define rules to prune the search space
- Examples include:
- Avoiding revisiting already explored nodes
- Preventing cycles and contradictory paths
- Enforcing semantic consistency between connected nodes
- Implemented via:
- State tracking mechanisms
- Rule-based filters
- Threshold limits on search depth or breadth
- Result: more focused and logically sound search process

This slide provides a closer look at the constraints applied in the wave-based search algorithm. These rules help prune the search space by avoiding revisiting nodes and preventing cycles, which could lead to infinite loops or logical contradictions. Additionally, semantic consistency rules ensure that only meaningful connections are explored. Implementation often involves state tracking, rule-based filters, and limits on how deep or broad the search can go, resulting in a more efficient and reliable search process.

VIII. Real-World Value of the Wave-Based Search Algorithm

Better Performance. Wave-based search avoids exhaustive traversal of irrelevant nodes by prioritizing semantically strong connections. This significantly reduces computational overhead compared to traditional BFS.

High Scalability. The algorithm supports parallel processing of wavefronts, making it suitable for massive semantic graphs and distributed environments such as Big Data and Knowledge Graph systems.

Structural Adaptability. Unlike BFS, the wave mechanism adapts to varying topologies and dynamically evolving networks. It does not rely on uniform graph structures, enabling flexible deployment in non-homogeneous semantic systems.

Example: Real-World Application

User query:

"Recommend a book that develops logical thinking in children."

- BFS might return all books labeled “for children.”
- Wave-based search follows the semantic path:

child → development → logical reasoning → recommended books,

offering context-aware, precise results.

IX. Performance Evaluation and Complexity Analysis

To rigorously assess the efficiency of Wave-Based Search (WBS) compared to classical algorithms such as Breadth-First Search (BFS), we analyze both theoretical and empirical aspects.

A. *Theoretical Complexity*

Both BFS and WBS operate with a worst-case time complexity of:

$$O(V + E)$$

where V is the number of vertices and E is the number of edges. However, the essential difference lies in execution time under parallel conditions. If p processors (or threads) are available, the effective runtime of WBS can be expressed as:

$$T\_WBS \approx (V + E) / p$$

This approximation assumes balanced wavefront sizes and efficient workload distribution. Thus, WBS does not reduce asymptotic complexity, but achieves significant constant-factor improvements in practice, especially when large wavefronts are processed simultaneously.

B. *Computational Example*

Consider a tree with n = 10^6 nodes and an average branching factor of b = 4.

- Sequential BFS: All nodes are processed sequentially, resulting in approximately 10^6 operations.
- WBS with p = 100 cores: Each wavefront contains, on average, 4,000 nodes. These nodes are distributed across 100 cores, leading to only 40 operations per core per wavefront.

This yields an estimated speedup of:

$$S = T\_BFS / T\_WBS \approx p = 100$$

Hence, WBS achieves a 100-fold improvement in processing speed under ideal parallel conditions.

C. *Semantic Network Example*

In a semantic search scenario, suppose a query requires exploring 5 waves, each with 2,000 nodes.

- Sequential BFS: 5 × 2000 = 10,000 node checks.
- WBS with 20 cores: Each wavefront of 2,000 nodes is divided into 100 per core, yielding a total of 5 × 100 = 500 node checks per core.

This represents a 20× reduction in effective processing load compared to BFS.

D. Results Summary

Scenario	Sequential BFS (ops)	WBS with 20 cores	WBS with 100 cores
Large tree (10^6 nodes)	1,000,000	50,000	10,000
Semantic search (5 waves)	10,000	500	200

The results demonstrate that while BFS and WBS share the same theoretical complexity, the parallel wave-based mechanism offers dramatic reductions in execution time. The larger and more interconnected the network, the higher the benefit from parallelization.

X. Conclusion

Wave-Based Search represents a hybrid approach based on BFS and parallel algorithms. Its efficiency is especially evident in semantic networks, where subnetwork processing is required and multiprocessor resources are available.

We recommend using Wave-Based Search when:

- A parallel architecture is available (e.g., HPC or multicore CPUs);
- The semantic network is complex and composed of multiple subsystems;
- The search must be bounded by a defined depth (Depth-Limited Reasoning).

This method enables scalable, efficient, and semantically aware search operations within both tree-structured and interconnected graph-based knowledge systems.

References

[1] N. Archvadze and M. Pkhovelishvili, “Wave-Based Search in Tree Structures Using Functional Programming Techniques”, *GESJ: Computer Science and Telecommunications*, no. 2(34), pp. 59-70, 2012.

[2] Z. Liu, X. Li, J. Huang and G. Yuan, “PASGAL: A Parallel and Scalable Framework for Large-scale Graph Analytics”, *Proceedings of the 39th IEEE International Parallel and Distributed Processing Symposium (IPDPS),* IEEE, 2024. [Online]. Available: https://arxiv.org/abs/2404.17101

[3] Y. Zhang, M. Chen and X. Wu, “Clustered Parallel BFS with Bit-Level Optimization for Shortest Path Labeling”, *Proceedings of the 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT*), ACM, 2023.

[4] D. Bilo, L. Gualà, G. Proietti, “Deterministic Bidirectional BFS in Real-World Networks: Complexity and Bounds”, *International Workshop on Combinatorial Algorithms (IWOCA 2023)*, Lecture Notes in Computer Science, Springer, vol. 14077, pp. 32–43, 2023.

[5] A. William, Woods, *Foundations for Semantic Networks*. Academic Press, 1975.

[6] L. Lortkipanidze, N. Amirezashvili, A. Chutkerashvili, N. Javashvili and L. Samsonadze, “Syntax Annotation of the Georgian Literary Corpus”, *Theoretical Computer Science and General Issues. 11th International Tbilisi Symposium on Logic, Language, and Computation,* TbiLLC 2015, Tbilisi, Georgia, 21-26 September, Revised Selected Papers, pp. 89–97, 2015.