Software Requirements Specification for Sparrow Search Algorithm-Based Traffic Congestion Control System Version 1.0

Jam Wilder
Department of Computer Science
Central Washington University
jamarius.wilder@cwu.edu

***Abstract*—This Software Requirements Specification (SRS) document follows the guidelines of IEEE Std 830-1998 to describe the requirements for a Sparrow Search Algorithm (SSA) system for traffic congestion optimization, with plans for future CUDA acceleration. The system aims to provide efficient route optimization in urban traffic networks by leveraging swarm intelligence, with future potential for parallel computation capabilities. This document addresses the functional and non-functional requirements, system constraints, and interface specifications required for implementation. All CUDA-related requirements are planned for future implementation phases and are not part of the initial system delivery.**

CONTENTS

I. Introduction

A. Purpose

This SRS document outlines the requirements for a Sparrow Search Algorithm (SSA) implementation targeting traffic congestion control, with plans for future CUDA acceleration. The document is intended for use by developers, testers, and stakeholders involved in the implementation and deployment of the traffic optimization system.

B. Scope

The software system described in this document, hereafter referred to as the "SSA Traffic Optimizer," provides an implementation of the Sparrow Search Algorithm to find optimal routes through congested traffic networks. The system includes:

- Graph generation and representation of road networks
- SSA-based route optimization engine
- Plans for future CUDA-accelerated parallelization of SSA
- Visualization tools for network analysis and route display
- Performance analysis and statistics generation

The system does not include:

- Real-time traffic monitoring hardware
- Integration with vehicle navigation systems
- Traffic light control hardware interfaces
- Public transportation scheduling

C. Definitions, Acronyms, and Abbreviations

- **SSA** - Sparrow Search Algorithm, a swarm intelligence metaheuristic
- **CUDA** - Compute Unified Device Architecture, NVIDIA's parallel computing platform
- **GPU** - Graphics Processing Unit
- **Node** - Representation of an intersection or point of interest in the traffic network
- **Edge** - Connection between two nodes, representing a road segment
- **Weight** - Value assigned to an edge, representing travel time, distance, or other metrics
- **Route** - Sequence of nodes representing a path through the network
- **Density** - Ratio of existing edges to potential edges in the graph
- **Producer** - Role in SSA representing sparrows that search for food sources (solutions)
- **Scrounger** - Role in SSA representing sparrows that follow producers
- **Scout** - Role in SSA representing sparrows that detect danger and maintain diversity

D. References

1) IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE, 1998.
2) Xue, J., & Shen, B. (2020). "A novel swarm intelligence optimization approach: Sparrow search algorithm." Systems Science & Control Engineering, 8(1), 22-34.
3) NVIDIA CUDA Programming Guide, Version 11.0, NVIDIA Corporation, 2020.
4) Wilder, J., "Sparrow Search Algorithm for Traffic Congestion Control: A CUDA-Enhanced Approach," Central Washington University, 2024.

E. Overview

The remainder of this document is organized as follows: Section 2 provides an overall description of the system, including product perspective, functions, user characteristics, constraints, and assumptions. Section 3 details the specific requirements, both functional and non-functional. Section 4 covers verification approaches, and Section 5 contains appendices and supporting information.

II. Overall Description

A. Product Perspective

The SSA Traffic Optimizer is a standalone system that can operate independently or integrate with existing traffic management systems. It builds upon existing research in swarm intelligence algorithms and lays the groundwork for future parallel computing extensions to provide an efficient solution for traffic congestion problems.

1) System Interfaces: The system consists of the following major components:

- **Graph Generation Module**: Creates and maintains traffic network graphs
- **SSA Core Engine**: Implements the Sparrow Search Algorithm
- **Visualization Module**: Generates graphical representations of results
- **Data Import/Export Interface**: Allows interchange with other systems
- **CUDA Parallel Processing Module**: (Planned for future implementation only) Will accelerate computation using GPU

2) User Interfaces: The system provides:

- Command-line interface for configuration and execution
- Generated visual outputs (PNG files) for analysis
- CSV-based data exchange format

3) Hardware Interfaces: The system requires:

- Sufficient system memory for graph representation
- Standard input/output devices
- CUDA-compatible NVIDIA GPU (for future CUDA implementation)

4) Software Interfaces: The system interfaces with:

- Python runtime environment (version 3.8+)
- NumPy, Matplotlib, NetworkX libraries
- Standard C/C++ development environment
- CUDA runtime library (version 10.0+, for future implementation)

5) Communication Interfaces: The system uses:

- File-based data exchange
- Standard output for logs and progress information

6) Memory Constraints: The system must efficiently manage memory, particularly:

- Host memory for large graph representation
- (Future) GPU global memory for parallel computations
- (Future) Optimization of memory transfers between host and device

B. Product Functions

The system's primary functions include:

- **Graph Generation**: Create random or predefined traffic network graphs with configurable parameters
- **Route Optimization**: Implement SSA to find optimal routes through the network
- **Result Visualization**: Generate visual representations of optimized routes and search patterns
- **Statistical Analysis**: Calculate and report performance metrics and optimization statistics
- **Parameter Tuning**: Allow configuration of algorithm parameters for optimization
- **Parallel Computation**: (Future) Utilize CUDA for acceleration of fitness evaluation and position updates

C. User Characteristics

The system is designed for:

- **Traffic Engineers**: Professionals analyzing traffic patterns and designing network improvements
- **Researchers**: Individuals studying optimization algorithms and traffic management
- **Software Developers**: Engineers integrating the system with other traffic solutions

Users are expected to have:

- Basic understanding of graph theory and optimization concepts
- Familiarity with command-line interfaces
- Knowledge of traffic network characteristics

D. Constraints

The system development and operation are subject to the following constraints:

- **Algorithm Limitations**: SSA may not guarantee global optimality for all inputs
- **Performance Constraints**: Must handle graphs with at least 100 nodes efficiently
- **Development Tools**: Must be compatible with GCC and Python toolchains
- **Interoperability**: Must support standard file formats for data exchange
- **Future Hardware Dependencies**: Future CUDA implementation will require NVIDIA GPU with CUDA support and NVCC

E. Assumptions and Dependencies

The system assumes:

- Traffic network can be represented as a weighted directed graph
- Users have access to CUDA-capable hardware
- Edge weights accurately represent travel costs (time, distance, etc.)
- Graph structure is relatively static during optimization

Dependencies include:

- Python environment with NumPy, Matplotlib, and NetworkX
- C/C++ compiler with C17 standard support
- Future dependency: CUDA Toolkit (10.0+)

F. Apportionment of Requirements

Requirements may be prioritized for implementation across multiple development phases:

- **Phase 1**: Core SSA implementation and basic visualization
- **Phase 2**: Advanced visualization and analysis tools
- **Phase 3**: Integration capabilities and extensions
- **Phase 4**: CUDA acceleration and performance optimization (Future)

III. Specific Requirements

A. External Interfaces

1) User Interfaces:

1) The system shall provide a command-line interface for configuration and execution
2) The system shall generate PNG image files visualizing:
   - Traffic network with optimized route
   - Node visit frequency histograms
   - Visit matrix heatmaps
   - Buildings and traffic jam representations
3) The system shall accept parameters for:
   - Graph size and density
   - Algorithm iterations and population size
   - Output file names and destinations
4) The system shall provide a help command that documents all available options

2) Hardware Interfaces:

1) The system shall primarily use CPU computation
2) The system shall be designed to allow for future GPU acceleration
3) (Future) The system shall utilize NVIDIA CUDA-compatible GPUs
4) (Future) The system shall detect and utilize available GPU compute capabilities
5) (Future) The system shall optimize memory usage based on available GPU resources

3) Software Interfaces:

1) The system shall interface with Python 3.8+ and required libraries
2) The system shall provide a C API for potential integration with other systems
3) The system shall support file-based data exchange using standardized formats
4) (Future) The system shall interface with the CUDA runtime (version 10.0+)

B. Functional Requirements

1) Graph Generation:

1) The system shall generate random traffic networks as weighted directed graphs
2) The system shall allow specification of node count and edge density
3) The system shall assign meaningful weights to edges based on simulated distance
4) The system shall ensure connectivity of the generated graph
5) The system shall load existing graphs from CSV-formatted adjacency matrices
6) The system shall generate node coordinates for visualization
7) The system shall save generated graphs to file in standard formats

2) Sparrow Search Algorithm Implementation:

1) The system shall implement the SSA with producer, scrounger, and scout behaviors
2) The system shall maintain a population of candidate solutions (routes)
3) The system shall evaluate solution fitness based on route total weight
4) The system shall perform position updates according to SSA rules
5) The system shall track the global best solution across iterations
6) The system shall implement convergence criteria based on iteration count or quality
7) The system shall allow configuration of key algorithm parameters:
   - Population size
   - Producer percentage
   - Maximum iterations
   - Danger awareness probability

3) Future CUDA Acceleration:

1) (Future) The system shall implement parallel fitness evaluation using CUDA
2) (Future) The system shall implement parallel position updates using CUDA
3) (Future) The system shall optimize memory transfers between host and device
4) (Future) The system shall utilize appropriate memory types (global, shared) for performance
5) (Future) The system shall implement efficient thread and block organization
6) (Future) The system shall properly synchronize parallel operations
7) (Future) The system shall fall back to CPU implementation when CUDA is unavailable

4) Visualization and Analysis:

1) The system shall generate network visualizations with the optimized route highlighted
2) The system shall generate histograms of node visit frequencies
3) The system shall generate heatmaps of the visit matrix
4) The system shall visualize traffic jams and buildings on the map
5) The system shall provide statistical analysis of the optimization process
6) The system shall output performance metrics and route quality information
7) The system shall allow customization of visualization parameters

5) Data Management:

1) The system shall save all generated data to files in appropriate formats
2) The system shall maintain consistency across related data files
3) The system shall implement appropriate error handling for file operations
4) The system shall validate input data for correctness and consistency
5) The system shall implement cleanup operations for temporary data

C. Performance Requirements

1) The system shall process graphs with up to 100 nodes within 60 seconds on recommended hardware
2) The system shall support at least 200 iterations with population size up to 100
3) The system shall generate visualizations for large graphs within 30 seconds
4) The system shall maintain optimization quality across graph sizes
5) (Future) The system shall achieve at least 5x speedup with CUDA compared to CPU-only implementation
6) (Future) The system shall use no more than 2GB of GPU memory for graphs up to 100 nodes

D. Design Constraints

1) The system shall be implemented using C/C++ for core algorithms
2) The system shall use Python and appropriate libraries for visualization
3) The system shall follow a modular architecture for component separation
4) The system shall follow file format standards for data interchange
5) (Future) The system shall use CUDA C/C++ for GPU acceleration
6) (Future) The system shall comply with standard development practices for parallelized code

E. Software System Attributes

1) Reliability:

1) The system shall handle invalid inputs gracefully with appropriate error messages
2) The system shall verify the consistency of generated solutions

3) The system shall implement appropriate memory management to prevent leaks or corruption
4) The system shall provide diagnostics for troubleshooting
5) The system shall validate all configuration parameters

2) Availability:

1) The system shall provide alternative methods when preferred approaches fail
2) The system shall implement graceful degradation when GPU resources are limited
3) The system shall support execution in CPU-only environments with reduced performance

3) Security:

1) The system shall validate all input files before processing
2) The system shall implement appropriate file permission handling
3) The system shall prevent buffer overflows and other memory-related vulnerabilities

4) Maintainability:

1) The system shall follow consistent coding standards
2) The system shall include appropriate documentation in the source code
3) The system shall implement modular design with clear separation of concerns
4) The system shall provide diagnostic features for identifying issues
5) The system shall include a comprehensive testing framework

5) Portability:

1) The system shall run on Windows, Linux, and macOS platforms
2) The system shall support multiple CUDA compute capabilities (5.0+)
3) The system shall use standard libraries to minimize platform dependencies
4) The system shall implement platform-specific code in isolated modules
5) The system shall use a portable build system

IV. Verification

A. Test Approach

The system shall be verified using:

- Unit tests for individual components
- Integration tests for component interactions
- Performance tests for optimization
- (Future) Performance tests for CUDA acceleration
- Validation tests comparing results to known optimal solutions
- Stress tests with large graphs and extreme parameters

B. Test Requirements

1) The system shall correctly generate connected graphs of specified size and density
2) The system shall find valid routes through all test graphs
3) (Future) The system shall demonstrate performance improvement with CUDA acceleration
4) The system shall produce correct visualizations for all test cases
5) The system shall handle both small (10 nodes) and large (100+ nodes) graphs
6) The system shall validate parameter boundaries and handle invalid inputs
7) The system shall maintain numerical stability across iterations

V. Supporting Information

A. Appendices

1) Appendix A: Sparrow Search Algorithm: The Sparrow Search Algorithm is a swarm intelligence metaheuristic inspired by the foraging and anti-predator behaviors of sparrows. Key components include:

- **Producer Behavior**: Leading sparrows that explore the solution space
- **Scrounger Behavior**: Following sparrows that exploit promising areas
- **Scout Behavior**: Alert sparrows that monitor for dangers and maintain diversity
- **Danger Response**: Random jumps to avoid local optima

2) Appendix B: File Formats:

a) Graph Format (CSV):

```
n
w11,w12,...,w1n
w21,w22,...,w2n
...
wn1,wn2,...,wnn
```

b) Coordinates Format (CSV):

```
node,x,y
0,0.123,0.456
1,0.789,0.012
...
```

c) Route Format (Text):

```
node_id_1
node_id_2
...
node_id_n
```

3) Appendix C: Future CUDA Implementation Considerations: Future CUDA implementation should consider:

- Thread organization for fitness evaluation
- Memory coalescing for efficient access patterns
- Shared memory usage for frequently accessed data
- Synchronization points to maintain algorithm integrity
- Host-device transfer optimization

B. Index

- Performance requirements - Section 3.3
- Sparrow Search Algorithm - Appendix A
- Visualization - Section 3.2.4