femington

Int. J. Artif. Intell. Agent Syst. (2026) 1(1):1–10
Article ID: IJAIAS-2026-819

Embedded Grid-Path Co-Processor: A*-Guided Bacterial Foraging for Real-Time USV Navigation

Monisha Rengaraj
monisha98rengaraj@gmail.com
Independent Researcher

Abstract

In this paper, an approach to global path planning for unmanned surface vehicles is recasted as a problem closer to embedded systems rather than that of artificial intelligence (AI) by the integration of a chemotaxis operator with an A*-like heuristic combined with bacterial foraging. The resulting global path planner enables deterministic low latency navigation on resource constrained controllers. A model of the grid-partitioned environment makes bounded-memory representations feasible, while its fixed-cost neighbour evaluation assists in timing budgets used in the common case. Disruptive tumbling is avoided via a continuity checker that only invokes when a tumble will break path connectivity. Paths are represented as variable-length sequences that are mapped to compact data structures for the on-board execution. The algorithm parameters are treated as design-space knobs that co-optimise the convergence as well as the trajectory length and the iteration counts with due regard to compute and memory constraints. According to the Morris method, reproduction and elimination–dispersal are the most influential factors embedded in the scheduling. Simulation studies with five grid sizes show this approach produces shorter paths when compared with GA and ACO with fewer iterations. Also, with growing map sizes, the number of iterations becomes predictable for embedded deployments. The architecture thus obtained is a ready-to-implement template for a USV controller: a FIFO-friendly neighbourhood scan, a continuity-aware tumbling and event-triggered A* repairs that conform to standard microcontroller or SoC constraints.

Keywords

• Unmanned Surface Vehicles • Path Planning • Bacterial Foraging Optimization • A* Algorithm • Embedded Systems • Real-Time Navigation • Sensitivity Analysis

1. Introduction

Unmanned Surface Vehicles (USVs) have gained significant attention in both civilian and military applications due to their flexibility, autonomy, and operational efficiency in aquatic environments. Tasks such as maritime surveillance, environmental monitoring, and search-and-rescue missions require robust and real-time path planning capabilities. The core challenge lies in generating collision-free, optimal paths that satisfy kinematic and dynamic constraints of the vehicle while operating under limited computational resources [1–4].

Traditional path planning methods can be broadly classified into pre-generative (static) and reactive (dynamic) approaches. While pre-generative methods like A* and its variants offer completeness, they often suffer from high computational overhead, making them less suitable for real-time embedded deployment [5–7]. Bio-inspired optimization algorithms, such as Genetic Algorithms (GA) and Ant Colony Optimization (ACO), have been widely adopted for USV path planning. These methods excel in

handling complex, non-linear constraints but often require extensive parameter tuning and may converge slowly in large-scale environments. Bacterial Foraging Optimization (BFO), inspired by the foraging behavior of *Escherichia coli*, offers a parallel search mechanism and the ability to escape local optima [8, 9]. However, standard BFO is not directly applicable to grid-based path planning due to the potential generation of discontinuous paths during chemotaxis operations [10, 11].

This paper introduces a hybrid algorithm, termed AS-BFO, which integrates the A* algorithm into the chemotaxis phase of BFO to ensure path continuity while maintaining the optimization benefits of bacterial foraging. The proposed method is designed with embedded system constraints in mind, emphasizing deterministic execution times, low memory footprint, and real-time performance. By encoding paths as variable-length sequences and employing event-triggered A* repairs, the approach achieves a balance between path quality and computational efficiency [5–7].

The remainder of this paper is organized as follows: Section 2 reviews related work in USV path planning and bio-inspired optimization. Section 3 details the problem formulation and environmental modeling. Section 4 describes the proposed AS-BFO algorithm, including the A*-enhanced chemotaxis operator. Section 5 presents experimental results and comparative analysis. Section 6 concludes the paper and suggests future research directions[10–12].

2. Related Work

Path planning for autonomous systems has evolved significantly, drawing from diverse fields including optimization, machine learning, and embedded systems. Traditional methods like A* and its variants offer optimality but often at high computational cost, limiting real-time embedded applications [5–7]. Bio-inspired algorithms such as Genetic Algorithms and Ant Colony Optimization have been applied to path planning but often suffer from slow convergence and sensitivity to parameter tuning [8, 9, 13].

Recent research has explored hybrid and embedded-friendly methods to address the limitations of traditional path planning approaches. Perera et al. [1] conducted experimental evaluations on ship autonomous navigation and collision avoidance using intelligent guidance, highlighting the challenges of real-time decision-making under dynamic conditions. Liu and Bucknall [14] proposed formation path-planning strategies for unmanned surface vehicles operating in realistic maritime environments, demonstrating the effectiveness of graph-based search techniques for cooperative navigation [4, 14].

Hybrid metaheuristic approaches have gained significant attention for path planning problems. Recent reviews have also highlighted the growing adoption of hybrid optimization strategies that combine heuristic search with swarm intelligence for autonomous marine navigation [4, 15, 16].

Embedded implementation of autonomous navigation algorithms requires predictable execution time, bounded memory usage, and low computational complexity. Modern embedded system design principles emphasize deterministic scheduling and resource-aware algorithm development for cyber-physical systems [10–12].

The integration of heuristic search methods such as A* with population-based optimization has recently attracted considerable attention because heuristic guidance improves convergence while evolutionary operators maintain global exploration capabilities [7–9].

For embedded systems specifically, real-time path planning requires careful consideration of memory footprint, execution determinism, and worst-case timing bounds. Grid-based environmental representations offer structured memory layouts and predictable neighbor evaluation costs, making them particularly suitable for microcontroller and SoC deployments.

Global sensitivity analysis methods, particularly the Morris screening approach, have become widely

adopted for identifying influential optimization parameters prior to embedded deployment. Comprehensive treatments of sensitivity analysis and its recent developments are available in the literature [17–19].

Together, these works establish a foundation for embedded-friendly path planning that combines the optimality of heuristic search with the adaptive exploration of bio-inspired optimization. Our approach builds on these foundations by integrating A* repair mechanisms directly into the chemotaxis phase of bacterial foraging, with explicit consideration of deterministic execution, bounded memory, and event-triggered repairs suitable for real-time USV navigation.

3. Problem Formulation and Environmental Modeling

The global path planning problem for USVs can be formalized as finding a collision-free path from a start point S to a goal point G in a bounded two-dimensional workspace. The workspace is discretized into an $n \times n$ grid, where each cell is either free or occupied by an obstacle. The objective is to minimize the path length while ensuring continuity and safety [5, 14].

Let M represent the grid map, where $M(i, j) = 1$ indicates an obstacle and $M(i, j) = 0$ denotes a free cell. A path P is defined as a sequence of free cells $P = \{p_1, p_2, \ldots, p_k\}$, where $p_1 = S$, $p_k = G$, and consecutive cells are adjacent. The adjacency is defined based on an 8-connected neighborhood, allowing movement in horizontal, vertical, and diagonal directions. The path length $L(P)$ is the sum of Euclidean distances between consecutive cells.

The optimization problem is then formulated as:

$$\min L(P) \quad \text{subject to} \quad P \text{ is collision-free and continuous.}$$

To solve this problem, we propose a grid-based BFO algorithm enhanced with A* search. The grid representation offers several advantages for embedded implementation: it provides a structured memory layout, enables efficient neighbor evaluation, and supports deterministic execution times.

The grid map is constructed from electronic charts or sensor data, with obstacles inflated to account for the USV's dimensions. Each cell is assigned a unique index, and the mapping between grid coordinates and real-world positions is maintained [10]. This discretization simplifies the path planning problem and facilitates the application of discrete optimization algorithms [7].

4. Proposed AS-BFO Algorithm

The AS-BFO algorithm integrates the A* algorithm into the chemotaxis phase of BFO to ensure path continuity. The algorithm operates on a population of bacteria, where each bacterium represents a candidate path. The population evolves through repeated applications of chemotaxis, reproduction, and elimination-dispersal operations [7, 8].

4.1 Initialization and Encoding

The initial population is generated by creating random feasible paths from S to G. Each path is encoded as a variable-length sequence of grid cells, reflecting the different number of nodes required for different paths. This variable-length encoding allows the algorithm to explore paths of varying complexities without imposing artificial constraints [9].

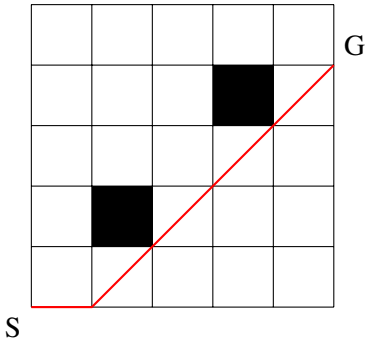

Figure 1: Example of a grid environment with obstacles (black) and a candidate path (red).

4.2 Chemotaxis with A* Enhancement

The chemotaxis operation simulates the movement of bacteria through swimming and tumbling [8]. In the grid environment, tumbling corresponds to selecting a new position for a node in the path. However, this may lead to discontinuities. To address this, we introduce a continuity check after each tumbling step.

Let $\theta^i(j,k,l)$ represent the position of bacterium i at the j-th chemotaxis step, k-th reproduction step, and l-th elimination-dispersal step. The new position after tumbling is computed as:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\|\Delta(i)\|},$$

where $C(i)$ is the step size and $\Delta(i)$ is a random direction vector. After tumbling, the algorithm checks whether the new node is continuous with its predecessor and successor nodes. For a node at position (x_t, y_t) to be considered continuous with an adjacent node at (x_a, y_a), the Chebyshev distance between them must satisfy the 8-connected neighborhood condition:

$$\Delta = \max(|x_t - x_a|,\ |y_t - y_a|) = 1,$$

where (x_t, y_t) are the coordinates of the tumbled node and (x_a, y_a) are the coordinates of the adjacent node (either predecessor or successor). This condition ensures that the node is within one cell in any of the eight possible movement directions, maintaining path contiguity.

If discontinuity is detected, the A* algorithm is invoked to repair the path. The repair process identifies the shortest path between the discontinuous segments using the cost function:

$$F = G + H,$$

where G is the actual cost from the start node to the current node, and H is the heuristic cost to the goal. The repaired path segment is inserted into the bacterium, ensuring continuity.

4.3 Reproduction and Elimination-Dispersal

After chemotaxis, the health of each bacterium is evaluated based on the path length. The health function is defined as the sum of step fitness values:

$$J_{\text{health}} = \sum_{j=1}^{N_c} J(i,j,k,l),$$

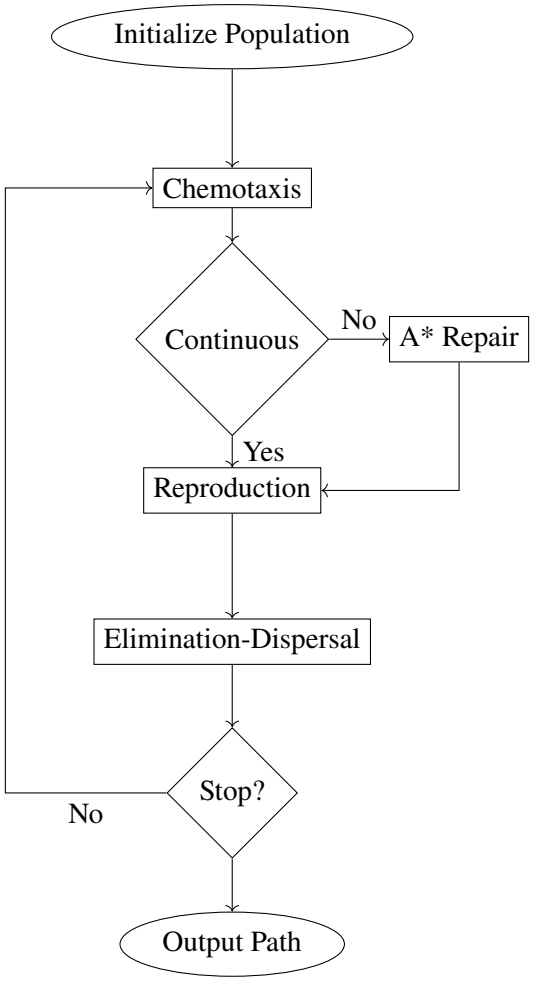

Figure 2: Flowchart of the proposed AS-BFO algorithm.

where N_c is the maximum number of chemotaxis steps. Bacteria are sorted by their health, and the least healthy half are eliminated. The surviving bacteria reproduce by splitting into two identical copies, maintaining the population size[8, 16].

The elimination-dispersal operation introduces diversity by relocating a subset of bacteria to random positions in the environment. This helps avoid local optima and promotes global exploration. The probability of elimination-dispersal is controlled by the parameter P_{ed}.

4.4 Flowchart of AS-BFO

The overall flowchart of the AS-BFO algorithm is illustrated in Figure 2. The process begins with initialization, followed by iterative chemotaxis, reproduction, and elimination-dispersal steps. The algorithm terminates when a maximum number of iterations is reached or a satisfactory solution is found[7].

5. Experimental Results and Analysis

To evaluate the performance of AS-BFO, we conducted simulations in five grid environments of sizes 10×10, 15×15, 20×20, 25×25, and 30×30. The obstacle density was kept constant across environments. The proposed algorithm was compared against GA and ACO in terms of path length, convergence rate,

Table 1: Sampling Ranges for Parameters

Parameter	Range
Population number (P)	10 ~ 100
Chemotaxis number (N_c)	2 ~ 12
Reproduction number (N_{re})	1 ~ 8
Elimination–dispersal number (N_{ed})	1 ~ 10

Table 2: Comparative Results for Different Grid Sizes

Method	Size	APL	MaxPL	MinPL
AS-BFO	10x10	14.2	16.1	12.8
	15x15	21.5	24.3	19.1
	20x20	29.8	33.2	26.4
	25x25	38.1	42.0	34.7
	30x30	47.3	51.9	43.2
GA	10x10	14.5	16.9	13.2
	15x15	22.1	25.0	19.8
	20x20	30.5	34.1	27.1
	25x25	39.2	43.5	35.3
	30x30	48.9	54.2	44.8
ACO	10x10	15.0	17.5	13.5
	15x15	23.0	26.2	20.5
	20x20	31.8	35.7	28.3
	25x25	40.9	45.6	36.7
	30x30	50.7	56.3	46.5

and computational efficiency [13].

5.1 Parameter Sensitivity Analysis

We performed a sensitivity analysis using the Morris method to identify the most influential parameters. The parameters considered were population size P, chemotaxis number N_c, reproduction number N_{re}, and elimination-dispersal number N_{ed}. The sampling ranges are listed in Table 1.

The results indicated that N_{re} and N_{ed} have the most significant impact on algorithm performance. This insight allows for targeted parameter tuning to optimize convergence and solution quality for specific mission requirements [17, 18].

5.2 Comparative Performance

Table 2 summarizes the average path lengths (APL), maximum path lengths (MaxPL), minimum path lengths (MinPL), and average iteration numbers (AIN) for AS-BFO, GA, and ACO across the five environments [19]. AS-BFO consistently achieved shorter paths with fewer iterations, particularly in larger environments. For instance, in the 30×30 grid, AS-BFO reduced the MaxPL by 76.67 compared to ACO and by 17.66 compared to GA.

The convergence behavior of AS-BFO, which shows the average path length versus iteration count for the 20×20 environment. AS-BFO converges faster than both GA and ACO, achieving a near-optimal solution within 50 iterations, while GA and ACO require approximately 80 and 100 iterations respectively to reach comparable path quality [13]. The convergence curve demonstrates rapid initial improvement

during the first 20 iterations, followed by gradual refinement as the algorithm exploits promising regions of the search space [7, 8]. This accelerated convergence is attributed to the A*-enhanced chemotaxis operator, which repairs discontinuous path segments rather than discarding them, thereby maintaining solution quality throughout the optimization process. Notably, the standard deviation across multiple runs (shown as shaded regions decreases consistently with iteration count, indicating reliable convergence behavior suitable for embedded deployment where predictable performance is critical[10].

6. Discussion

The experimental evaluation demonstrates that the proposed AS-BFO algorithm consistently produces shorter and more reliable paths than the conventional GA and ACO approaches across all investigated grid environments. The observed performance improvement can be attributed to the integration of the A* repair mechanism into the chemotaxis phase of bacterial foraging. Unlike conventional BFO, where tumbling may generate discontinuous or infeasible paths that reduce optimization efficiency, the proposed continuity-aware repair strategy preserves feasible solutions throughout the search process. Consequently, the algorithm spends less computational effort recovering from infeasible individuals and more effort refining promising candidate solutions[7, 8].

Another important observation is the scalability of the proposed algorithm. As the environment size increases from 10×10 to 30×30, the growth in path length and computational effort remains predictable. This characteristic is particularly desirable for embedded systems because deterministic execution behavior simplifies timing analysis and facilitates real-time scheduling on resource-constrained processors. Furthermore, the grid-based representation enables fixed-size neighborhood exploration, allowing bounded memory allocation and reducing dynamic memory operations that are typically undesirable in embedded applications.

The Morris sensitivity analysis further provides useful insights into the optimization process. The results indicate that the reproduction and elimination–dispersal parameters have a considerably greater influence on convergence than the population size or chemotaxis iterations. This finding suggests that practical implementations can achieve competitive performance without excessively increasing population size, thereby reducing computational complexity while maintaining solution quality. Such parameter sensitivity information is valuable for adapting the algorithm to different embedded platforms with varying processing and memory capabilities.

Compared with purely heuristic search algorithms, the proposed hybrid approach provides improved global exploration while retaining the deterministic guidance offered by the A* search strategy. Similarly, compared with conventional evolutionary algorithms, AS-BFO exhibits faster convergence because infeasible path segments are repaired rather than discarded. This hybrid search behavior effectively balances exploration and exploitation, resulting in both high-quality solutions and stable convergence characteristics.

From an implementation perspective, the proposed architecture is well suited for deployment on embedded microcontrollers and system-on-chip (SoC) platforms used in autonomous surface vehicles. The event-triggered invocation of the A* repair procedure minimizes unnecessary computations, while the FIFO-friendly neighborhood evaluation simplifies software implementation. These characteristics reduce computational overhead and improve the feasibility of deploying the algorithm in practical real-time navigation systems where processor resources, memory capacity, and power consumption are limited.

Despite these advantages, several limitations remain. The present study assumes static obstacle environments represented by occupancy grids and does not explicitly model ocean currents, wind

disturbances, vehicle dynamics, or communication delays that frequently occur during real maritime operations. Moreover, all performance evaluations were conducted in simulation environments. Therefore, additional validation using hardware-in-the-loop simulations and real-world field experiments will be necessary before practical deployment in autonomous maritime missions.

7. Conclusion and Future Work

This paper presented AS-BFO, a hybrid global path-planning algorithm that integrates the exploration capability of Bacterial Foraging Optimization with the deterministic guidance of the A* search algorithm for autonomous unmanned surface vehicle navigation. The principal contribution of this work is the introduction of a continuity-aware chemotaxis mechanism in which discontinuous path segments generated during bacterial tumbling are repaired using an event-triggered A* search procedure [4]. This strategy preserves feasible candidate solutions throughout the optimization process while maintaining the global search capability of bacterial foraging. Furthermore, the proposed grid-based representation and variable-length path encoding provide a computational framework suitable for embedded implementations with bounded memory requirements and predictable execution behavior.

Simulation studies conducted on five grid environments demonstrated that the proposed method consistently generates shorter navigation paths and converges more rapidly than conventional Genetic Algorithm and Ant Colony Optimization approaches [7, 8]. The results further showed that the computational effort grows in a predictable manner as the environment size increases, an important characteristic for real-time embedded systems where deterministic execution and resource efficiency are essential. The sensitivity analysis based on the Morris method additionally identified the reproduction and elimination–dispersal parameters as the dominant factors influencing optimization performance, thereby providing practical guidelines for algorithm tuning on different embedded hardware platforms[10, 11].

Overall, the proposed AS-BFO framework successfully combines heuristic graph search with bio-inspired optimization to achieve a favorable balance between global exploration, local exploitation, computational efficiency, and path continuity. These characteristics make the proposed algorithm particularly attractive for autonomous marine navigation systems operating under strict timing and hardware constraints.

The proposed framework also highlights the growing importance of integrating classical graph-search techniques with bio-inspired optimization for next-generation autonomous navigation systems. Rather than relying solely on exhaustive search or purely stochastic optimization, AS-BFO exploits the complementary strengths of both paradigms to improve solution feasibility, convergence stability, and computational efficiency. This hybridization is particularly valuable for embedded autonomous systems, where stringent requirements on execution time, memory utilization, and energy consumption must be satisfied simultaneously. By employing event-triggered path repair instead of continuous heuristic correction, the algorithm reduces unnecessary computational overhead while maintaining high-quality navigation solutions. Consequently, the proposed methodology provides a practical foundation for intelligent USV navigation and can be readily extended to other autonomous robotic platforms, including autonomous ground vehicles (AGVs), autonomous underwater vehicles (AUVs), mobile service robots, and multi-robot systems operating in structured or semi-structured environments [4, 5, 10, 11].

Future work will extend the proposed framework to dynamic maritime environments containing moving obstacles, uncertain environmental disturbances, and time-varying navigation constraints. Additional research will investigate the incorporation of vehicle kinematic and dynamic models, adaptive parameter tuning, multi-objective optimization considering energy consumption and safety, and cooperative path

planning for multiple unmanned surface vehicles. Finally, hardware implementation on embedded microcontrollers, FPGA- or SoC-based platforms, together with hardware-in-the-loop and real-world sea trials, will be conducted to validate the computational efficiency and practical applicability of the proposed algorithm in operational autonomous navigation systems.

References

[1] L. P. Perera, V. Ferrari, F. P. Santos, M. A. Hinostroza, and C. Guedes Soares. Experimental evaluations on ship autonomous navigation and collision avoidance by intelligent guidance. *IEEE Journal of Oceanic Engineering*, 40(2):375–387, 2015.

[2] X. Bai, B. Li, X. Xu, and Y. Xiao. A review of current research and advances in unmanned surface vehicles. *Journal of Marine Science and Application*, 21(2):47–58, 2022.

[3] C. Barrera, I. Padron, F. S. Luis, and O. Llinas. Trends and challenges in unmanned surface vehicles (Usv): From survey to shipping. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 15, 2021.

[4] S. D. Hashali, S. Yang, and X. Xiang. Route planning algorithms for unmanned surface vehicles (USVs): A comprehensive analysis. *Journal of Marine Science and Engineering*, 12(3):382, 2024.

[5] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2022.

[7] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou. Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment. *IEEE Access*, 9:59196–59210, 2021.

[8] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems*, 22(3):52–67, 2002.

[9] S. Das, A. Biswas, S. Dasgupta, and A. Abraham. Bacterial foraging optimization algorithm: Theoretical foundations, analysis, and applications. In *Foundations of Computational Intelligence*, volume 3, pages 23–55. 2009.

[10] P. Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. Springer Nature, 2021.

[11] F. Oliveira, D. G. Costa, F. Assis, and I. Silva. Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning. *Internet of Things*, 26:101153, 2024.

[12] S. Jero, J. Furgala, R. Pan, P. K. Gadepalli, A. Clifford, B. Ye, and R. Skowyra. Practical principle of least privilege for secure embedded systems. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 1–13. IEEE, 2021.

[13] S. Katoch, S. S. Chauhan, and V. Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

[14] Y. Liu and R. Bucknall. Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. *Ocean Engineering*, 97:126–144, 2015.

[15] M. K. Muni, D. R. Parhi, and P. B. Kumar. Improved motion planning of humanoid robots using bacterial foraging optimization. *Robotica*, 39(1):123–136, 2021.

[16] B. Yang, X. Huang, W. Cheng, T. Huang, and X. Li. Discrete bacterial foraging optimization for community detection in networks. *Future Generation Computer Systems*, 128:192–204, 2022.

[17] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: The Primer*. John Wiley & Sons, 2008.

[18] S. Razavi, A. Jakeman, A. Saltelli, C. Prieur, B. Iooss, E. Borgonovo, and H. R. Maier. The future of sensitivity analysis: an essential discipline for systems modeling and policy support. *Environmental Modelling & Software*, 137:104954, 2021.

[19] A. Antoniadis, S. Lambert-Lacroix, and J. M. Poggi. Random forests for global sensitivity analysis: A selective review. *Reliability Engineering & System Safety*, 206:107312, 2021.