Master Thesis

mathematik
fakultät8

Automated Design and Analysis of Gene Regulatory Networks for Simulating Complex Spatial Patterns

Automatisiertes Design und Analyse von Genregulationsnetzwerken zur Simulation komplexer räumlicher Muster

Loghman Samani
Matrikelnummer: 3585810
st186417@stud.uni-stuttgart.de

January 4, 2025

First Examiner: *Prof. Dr. Nicole Radde*
Second Examiner: *Prof. Dr. Michael Heymann*
Advisor: *M. Sc. Amatus Beyer*

Institute for Stochastics and Applications
Mathematical Modeling and Simulation of Cellular Systems
University of Stuttgart

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to those who have supported me throughout the course of my master's thesis.

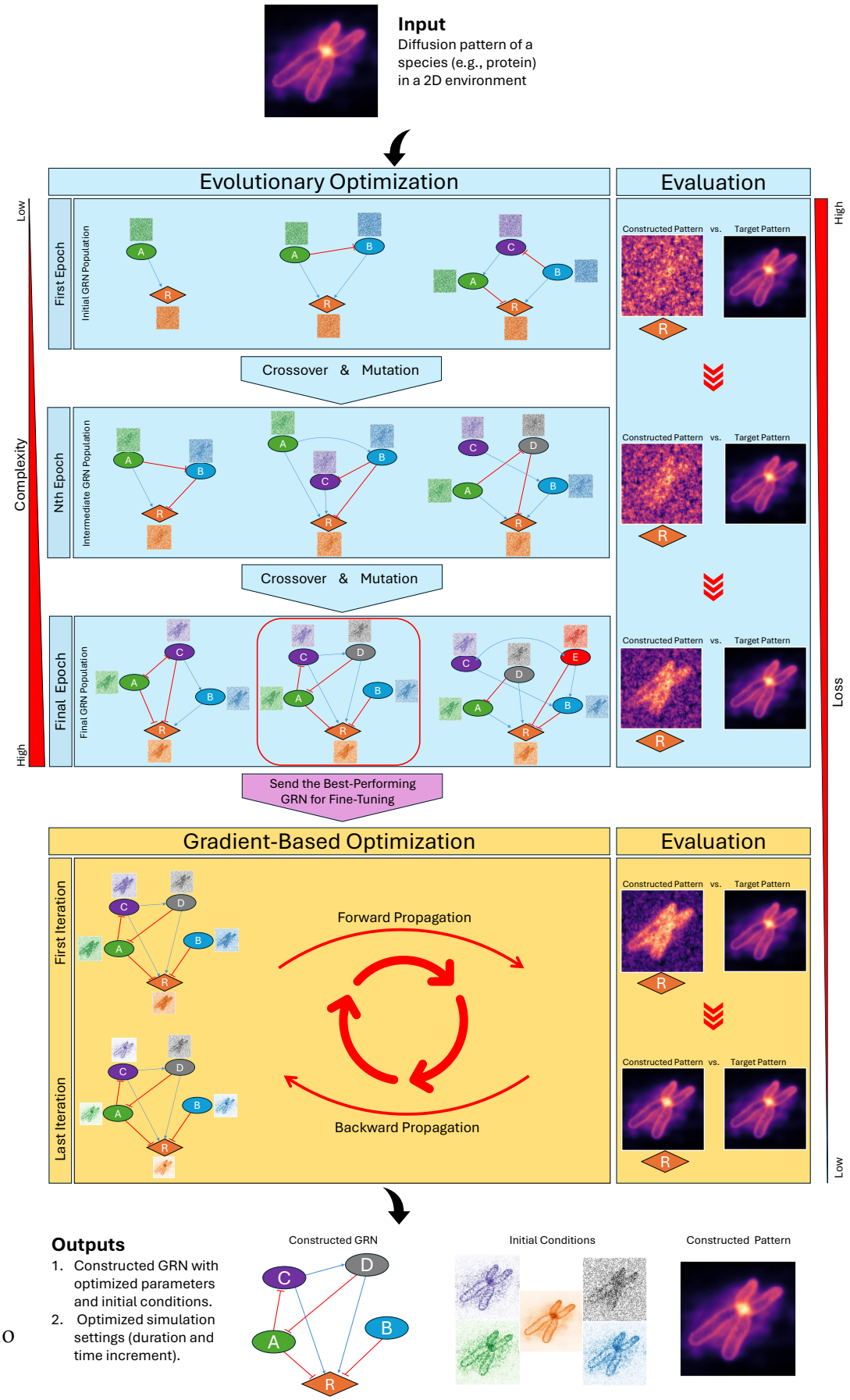
First and foremost, I am deeply grateful to Prof. Dr. rer. nat. Nicole Radde, Professor of Mathematical Modeling and Simulation of Cellular Systems at the Institute for Stochastics and Applications. As the head of the research group where I conducted my thesis work, her expertise, vision, and guidance were invaluable in shaping the direction and quality of this research.

I also extend my heartfelt thanks to my advisor, M. Sc. Amatus Beyer, Research Assistant at the Institute for Stochastics and Applications. His guidance and technical support were instrumental in the successful completion of this work. Amatus's patient advice, constructive feedback, and in-depth knowledge were essential at every stage of my research, from conceptualization to implementation. I am immensely grateful for his dedication and support.

Finally, I would like to acknowledge the resources provided by the bwHPC framework, specifically the bwUniCluster 2.0 system managed by the Steinbuch Centre for Computing (SCC) at Karlsruhe Institute of Technology (KIT). The bwUniCluster 2.0 system, with its extensive computational resources and cooperative infrastructure, enabled the efficient execution of computationally intensive optimization tasks that were critical to this research.

Thank you all for your invaluable support and contributions.

Graphical Abstract


Input
Diffusion pattern of a species (e.g., protein) in a 2D environment
Evolutionary Optimization
Evaluation
First Epoch
Initial GRN Population
Crossover & Mutation
Nth Epoch
Intermediate GRN Population
Crossover & Mutation
Final Epoch
Final GRN Population
Send the Best-Performing GRN for Fine-Tuning
Constructed Pattern vs. Target Pattern
Complexity
Low
High
Loss
High
Low
Gradient-Based Optimization
Evaluation
First Iteration
Last Iteration
Forward Propagation
Backward Propagation
Outputs
1. Constructed GRN with optimized parameters and initial conditions.
2. Optimized simulation settings (duration and time increment).
Constructed GRN
Initial Conditions
Constructed Pattern

The graphical abstract illustrates the workflow of the **GRN-Designer algorithm**, a hybrid optimization system designed to construct gene regulatory networks (GRNs) that replicate a target diffusion pattern.

The system takes as input a two-dimensional reference diffusion pattern (e.g., a protein's diffusion) and uses two optimization methods: **evolutionary optimization** and **gradient-based optimization**.

- **Evolutionary Optimization:** A population of GRNs is randomly initialized each with one gene (by default). Through generations, the GRNs are modified by crossover, mutation, and parameter tuning. This phase optimizes the GRN structure, simulation settings (duration and time increment), and initial conditions of each gene. The best-performing GRN is selected for further refinement.
- **Gradient-Based Optimization:** The selected GRN is fine-tuned to improve accuracy by refining the initial conditions and parameters (e.g., rate constants of gene products and interactions between genes).

Outputs: The system outputs:

- A fully constructed GRN with optimized parameters and initial conditions.
- Final simulation settings (e.g., duration and time increment).

0.1 Abstract

Gene Regulatory Networks (GRNs) are essential frameworks for understanding complex gene interactions and regulatory mechanisms in biological systems. Traditional GRN modeling approaches often rely on deterministic or probabilistic frameworks to simulate gene expression dynamics, but they typically overlook the critical role of spatially heterogeneous initial conditions. Synthetic developmental biology has focused on constructing GRNs to replicate specific spatial patterns, yet most models assume simplified, uniform distributions of gene products, limiting biological realism. This work addresses this limitation by introducing an approach that incorporates variable initial conditions, simulating diverse spatial distributions of gene activity in a two-dimensional environment.

We developed the GRN-Designer algorithm, an end-to-end hybrid optimization framework that combines evolutionary algorithms with gradient-based methods to construct GRNs capable of replicating predefined spatial patterns. By treating the initial expression levels and spatial activation potential of genes as adjustable parameters, the algorithm designs GRN structure and optimizes regulatory parameters to achieve target diffusion patterns. Multiple distinct spatial patterns were tested, each simulated through partial differential equations (PDEs) to model gene expression dynamics.

The algorithm successfully designed GRNs for all target patterns, ranging from simple gradients to intricate shapes, demonstrating flexibility in managing diverse spatial configurations. The constructed GRNs adaptively balanced complexity, tailoring the number of genes and regulatory interactions to match the intricacies of each target pattern. Results indicate that our approach effectively combines computational efficiency with biological realism, producing spatial patterns that closely resemble their intended designs.

This work establishes a robust framework for GRN design, with promising applications in synthetic biology and tissue engineering. By integrating heterogeneous spatial initial conditions, our model advances the realism of GRN simulations. Future enhancements may explore stochastic simulation techniques to account for biological variability, extend the framework to three-dimensional environments, and improve optimization performance to address more complex biological systems.

Keywords: Gene Regulatory Networks, Synthetic Biology, Initial Conditions, Spatial Patterning, Hybrid Optimization, Stochastic Simulation.

Zusammenfassung

Genregulationsnetzwerke (GRNs) sind wesentliche Rahmenwerke für das Verständnis komplexer Geninteraktionen und regulatorischer Mechanismen in biologischen Systemen. Traditionelle Ansätze zur Modellierung von GRNs basieren häufig auf deterministischen oder probabilistischen Modellen zur Simulation von Genexpressionsdynamiken, berücksichtigen jedoch oft nicht die entscheidende Rolle räumlich heterogener Anfangsbedingungen. Die synthetische Entwicklungsbiologie hat sich darauf konzentriert, GRNs zu konstruieren, die spezifische räumliche Muster replizieren, wobei jedoch die meisten Modelle vereinfachte, gleichmäßige Verteilungen von Genprodukten annehmen, was die biologische Realitätsnähe einschränkt. Diese Arbeit geht auf diese Einschränkung ein, indem ein Ansatz eingeführt wird, der variable Anfangsbedingungen berücksichtigt und unterschiedliche räumliche Verteilungen der Genaktivität in einer zweidimensionalen Umgebung simuliert.

Wir haben den GRN-Designer-Algorithmus entwickelt, ein End-to-End-Hybrid-Optimierungsframework, das evolutionäre Algorithmen mit gradientenbasierten Methoden kombiniert, um GRNs zu konstruieren, die in der Lage sind, vordefinierte räumliche Muster zu replizieren. Indem die Anfangsexpressionsniveaus und räumliche Aktivierungspotentiale der Gene als anpassbare Parameter behandelt werden, entwirft der Algorithmus die Struktur von GRNs und optimiert regulatorische Parameter, um Ziel- Diffusionsmuster zu erreichen. Es wurden mehrere unterschiedliche räumliche Muster getestet, die jeweils durch partielle Differentialgleichungen (PDEs) simuliert wurden, um die Dynamik der Genexpression zu modellieren.

Der Algorithmus hat für alle Zielmuster erfolgreich GRNs entworfen, die von einfachen Gradianten bis hin zu komplexen Formen reichen, was die Flexibilität bei der Handhabung unterschiedlicher räumlicher Konfigurationen zeigt. Die konstruierten GRNs passten sich adaptiv an, indem sie die Anzahl der Gene und regulatorischen Wechselwirkungen so anpassten, dass sie den Komplexitäten jedes Zielmusters entsprachen. Die Ergebnisse zeigen, dass unser Ansatz rechnerische Effizienz mit biologischer Realitätsnähe kombiniert und räumliche Muster erzeugt, die ihren beabsichtigten Designs sehr nahe kommen.

Diese Arbeit stellt ein robustes Framework für das Design von GRNs dar, mit vielversprechenden Anwendungen in der synthetischen Biologie und Gewebeengineering. Durch die Integration heterogener räumlicher Anfangsbedingungen wird die Realitätsnähe von GRN-Simulationen weiter verbessert. Zukünftige Erweiterungen könnten stochastische Simulationstechniken zur Berücksichtigung biologischer Variabilität, die Erweiterung des Frameworks auf dreidimensionale Umgebungen sowie die Verbesserung der Optimierungsleistung zur Behandlung komplexerer biologischer Systeme umfassen.

0.2 Introduction

The genome encodes thousands of genes that collectively drive cellular survival and enable a range of complex cellular functions. The specific timing and quantity of gene products present in cells are crucial to sustaining life processes, which are tightly regulated by complex gene regulatory networks (GRNs) [1]. GRNs are intricate assemblies of molecular regulators—including DNA, RNA, and proteins—that govern gene expression, influencing mRNA and protein levels to establish and maintain cellular functions [2, 3, 50]. These networks regulate the expression of genes in a coordinated manner, determining cellular identity, activity, and ultimately the behavior of an organism.

In GRNs, the regulatory interactions between genes can involve various molecules such as transcription factors, RNA molecules, or other proteins, and may include interactions at transcriptional, translational, or post-translational levels. These interactions can act directly or indirectly, modifying cellular processes by enabling or inhibiting the expression of particular genes [8]. For example, transcription factors can activate or repress gene expression by binding to promoter regions of DNA, initiating cascades that lead to the synthesis of specific gene products. This coordinated regulation is essential in multicellular organisms, where GRNs play a critical role in defining cell fate and function, particularly during developmental processes [3].

Multicellular organisms rely on GRNs for the spatial and temporal regulation of gene expression, which is essential for processes like morphogenesis and cell differentiation. Morphogen gradients—a hallmark of early development—provide positional information that guides cell fate by establishing differential exposure of cells to signaling molecules. These gradients enable identical genomes to be expressed differently across cells, allowing cells to adopt specialized functions within a single organism [5–7]. As depicted in Figure 1, GRNs orchestrate the formation of morphogen gradients that spread from localized sources, shaping an organism's structure by guiding cells to respond based on their position within the gradient field [5].

Despite the wealth of molecular data available on early development, the dynamic interactions within GRNs are exceedingly complex due to the vast number of components and potential interactions. The high-dimensional nature of these networks presents significant computational challenges, as modeling GRNs requires accounting for nonlinear interactions and intricate feedback loops [10]. Further, generating predictive models of GRNs to understand developmental dynamics in detail remains challenging and requires advanced computational approaches.

To address the complexities of spatial pattern formation, synthetic developmental biology aims to construct and study synthetic gene regulatory networks (GRNs) that can generate precise spatial patterns in multicellular systems [9]. For example, Mousavi and Lobo recently demonstrated an approach using dual morphogen

sources in a two-dimensional diffusion model, where morphogens introduced from compartment edges create spatial patterns through GRN optimization [9]. Their work highlights how morphogen gradients, combined with evolutionary optimization algorithms, can be systematically designed to achieve specific target structures. However, in biological systems, morphogen sources are not always confined to edges; the spatial distribution of morphogen sources can vary depending on factors such as the type, location, and origin of the gradient [11, 12].

Building on Mousavi and Lobo's model [9], which relies on dual morphogen sources to create target patterns, this research introduces a novel layer of complexity by incorporating spatially varied initial expression levels for each gene or morphogen in the system. Instead of assuming a fixed source location (such as an edge) for morphogen diffusion or gene activation, we introduce a more realistic approach: spatially heterogeneous starting conditions that define the initial expression levels and spatial activation potential of genes across a two-dimensional compartment (or tissue). Throughout this work, we refer to these non-uniform initial expression levels as **initial conditions**, which may include localized activation regions or gradients that simulate varying gene activity or molecular source locations across the system.

By treating initial conditions as a core design parameter, our approach enables a deeper exploration of how spatial variations in the initial expression levels of gene products impact emergent expression gradients and pattern formation within the system. Including these non-uniform starting conditions allows for more realistic modeling of spatial expression dynamics, showing how the initial expression levels and spatial activation potential of genes can affect GRN outputs and ultimately influence cell fate decisions.

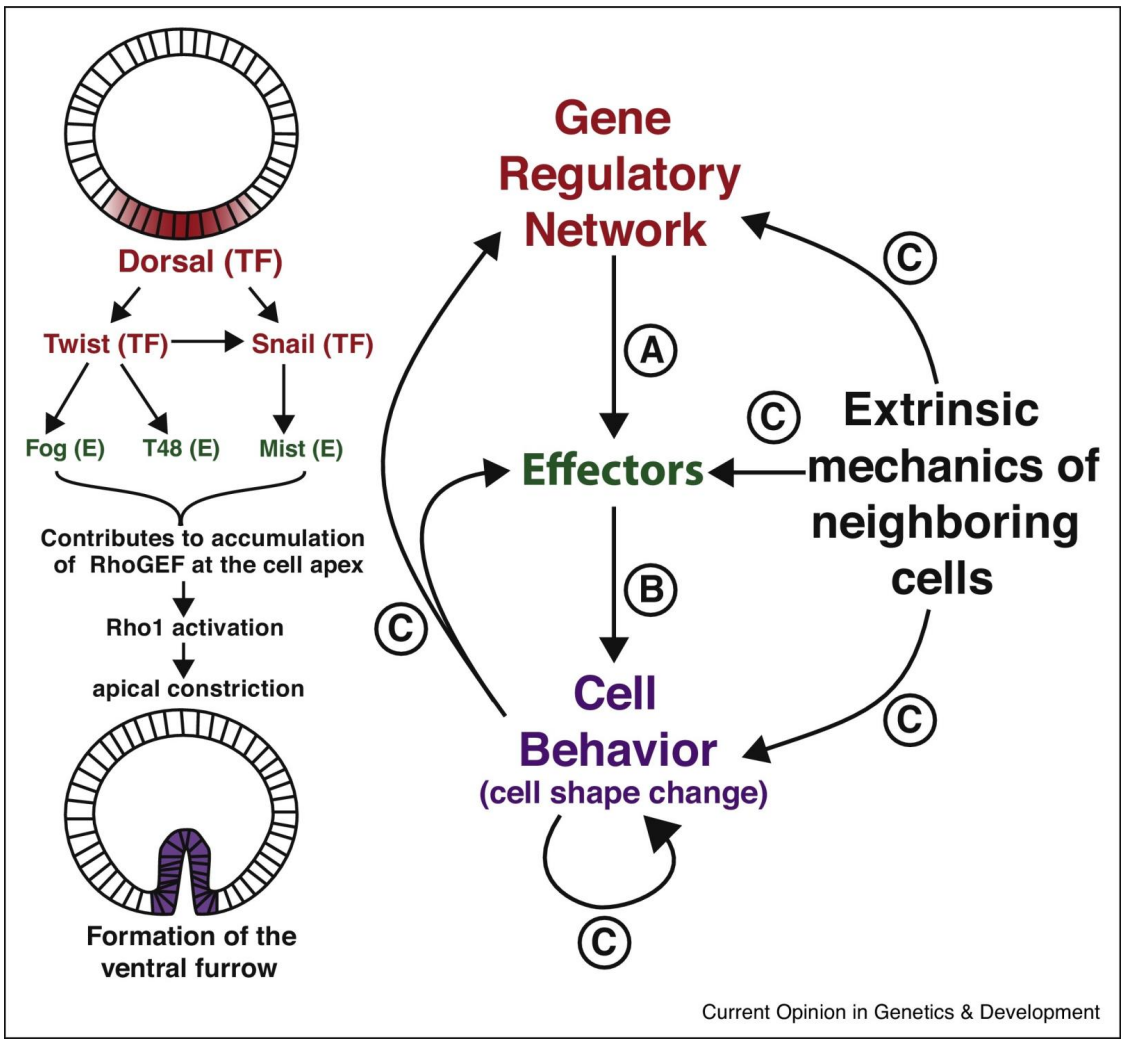


Figure 1: The influence of GRNs, cellular effectors, and neighboring mechanics on tissue morphogenesis.
Morphological structures are pre-patterned by GRNs (red) that turn on a precise set of effector molecules (green). These effectors control cell behaviors, here altering the shape of cells, together forming the final phenotype. (Left) An example of these processes occurring during ventral furrow formation where a nuclear gradient of the transcription factor Dorsal activates downstream transcription factors, which in turn activate effector molecules. These effector molecules accumulate at the apical membrane, causing the cells to apically constrict, and resulting in tissue invagination. (Right) Depiction of the interaction between gene regulatory networks, cellular effectors, and cell behaviors. While networks activate effectors (a), which drive changes in cell behaviors (b), development can also be influenced by signals and mechanical cues coming from neighboring cells in addition to feedback from different parts of the process (c). Adapted from (Smith et al., 2018) [5].

0.3 Literature Review

In recent years, diverse computational approaches have been developed to tackle the complex task of gene regulatory network (GRN) inference. Zhao et al [2]. categorized these methods into three major classes: model-based methods, information theory-based methods, and machine learning-based methods. Model-based methods use mathematical models to describe gene relationships, encompassing Boolean, Bayesian, differential equation, and neural network approaches. Information theory-based methods, such as correlation and mutual information analyses, are widely used to capture gene interactions in an unbiased way. Machine learning-based methods, which include regression, random forest, and boosting algorithms, leverage data structures and advanced computational strategies to predict regulatory relationships in GRNs [2].

A significant challenge in GRN modeling lies in addressing the high complexity of transcriptional programs, particularly with evolving experimental techniques and data expansion [13]. To capture intricate and high-dimensional gene interactions, neural network architectures have emerged as an adaptable solution for GRN inference [10]. Additionally, techniques such as the redundancy silencing and network enhancement technique (RSNET) have been introduced to filter indirect interactions in GRNs, improving accuracy by focusing on direct regulatory relationships [14].

Neural network approaches have further expanded with tools like dynGENIE3, which uses time-series expression data to infer dynamic gene interactions [15]. Similarly, deep learning tools like DeepIMAGER analyze GRNs from single-cell RNA sequencing (scRNA-seq) data, enhancing single-cell data resolution in GRN analysis [16].

Time-series data and advanced machine learning algorithms have also led to GRN inference tools such as MICRAT, which captures regulatory interactions over time for a more dynamic view of gene expression networks [17]. New transformer-based models, exemplified by TRENDY, further enhance GRN prediction by leveraging deep learning techniques to improve performance across simulated and experimental datasets [18].

Single-cell and multi-omic data have played a transformative role in GRN inference, enabling more precise modeling of gene regulation dynamics. By integrating diverse data types, these multi-omic approaches offer a holistic view of cellular processes [19]. Advances in continuous lifelong learning also facilitate GRN model refinement by integrating large-scale data, thereby keeping models current with the latest biological insights [20].

The accuracy of GRN inference algorithms is critical, necessitating rigorous benchmarking studies to evaluate their effectiveness. For instance, Pratapa et al. [21] performed a comprehensive benchmarking of single-cell GRN inference algorithms, highlighting the strengths and limitations of various methods [21]. To address

computational challenges in large datasets, cloud-based parallel evolutionary algorithms have been proposed, which distribute GRN inference tasks across multiple computing resources to enhance scalability and performance [22].

In synthetic biology, robust GRN designs are essential to create reliable gene networks that perform consistently under various conditions. Tools like Monte Carlo-based algorithms and fitness approximation methods quantify GRN robustness, enhancing their stability in synthetic applications [23]. Additionally, new mathematical frameworks have been proposed to assess genetic robustness across environmental and phenotypic conditions, facilitating synthetic GRN designs that meet specific stability criteria [24].

In developmental biology, synthetic approaches are used to engineer GRNs to model and produce specific spatial patterns. Mousavi and Lobo [9] introduced a methodology that uses evolutionary algorithms with morphogen gradients as positional signals, allowing for automatic GRN designs that create targeted two-dimensional spatial patterns [9]. This approach provides a basis for exploring GRN dynamics in controlled environments, which aligns with the goals of synthetic developmental biology and further integrates GRNs into clinical workflows, as demonstrated by the SCENIC workflow [25].

In summary, advancements in GRN inference reflect a multidisciplinary effort that combines model-based methods, machine learning, and high-throughput data analysis. By addressing computational complexity and expanding data sources, these approaches contribute significantly to GRN modeling accuracy, robustness, and application potential in both biological research and synthetic biology.

0.4 Methodology

In this study, we present a novel computational approach for designing gene regulatory networks (GRNs) capable of replicating complex spatial patterns within a multicellular environment. Our methodology involves the construction of GRNs through an optimization framework that iteratively tunes the network's parameters to approximate a predefined target pattern. This framework, termed the **GRN-Designer algorithm**, integrates both evolutionary and gradient-based optimization techniques to navigate the vast search space inherent in GRN design. By combining a global search approach with precise local tuning, the algorithm can effectively model biological processes where spatial distribution and regulatory dynamics play crucial roles in gene expression. In this section, we detail the GRN-Designer algorithm, the hybrid optimization methods it employs, and the specific parameter configurations required to simulate realistic cellular environments and diffusion patterns.

0.4.1 GRN-Designer Algorithm

The GRN-Designer algorithm is developed to automatically construct a gene regulatory network (GRN) with optimally tuned parameters to replicate a target pattern. To achieve this, it combines two different types of optimization algorithms, each from distinct families. The first algorithm is a tailored version of a **genetic algorithm** [26], a population-based metaheuristic method effective for global search [27]. Metaheuristic algorithms are particularly suited for exploring large search spaces and identifying the approximate global optimum in complex domains. The second algorithm is the **Adaptive Moment Estimation (Adam) algorithm** [28], a member of the gradient-based optimization family, which excels at refining solutions to achieve precise convergence within a local optimum [30].

The vast search space for designing a GRN, especially when little prior knowledge of the system is available, necessitates the use of metaheuristic search methods. Metaheuristic algorithms are well-suited to finding a global minimum due to their exploratory capabilities. However, these algorithms often come with a high computational cost and may not fully converge on a finely-tuned solution [31]. Therefore, once an initial GRN configuration is identified and the system reaches a convergence threshold, a gradient-based optimization method is employed to fine-tune the parameters within the global minimum region. Gradient-based methods like Adam leverage gradient descent principles [28, 32, 33] to optimize solutions with high accuracy and computational efficiency.

The GRN-Designer algorithm thus combines the strengths of both optimization techniques: a population-based genetic algorithm for effective global search and Adam optimization for precise local tuning. This combined approach enables the algorithm to construct a GRN with optimized parameters—spanning initial

conditions and rate constants for each molecular species produced by a specific gene present in the system, and simulation parameters which are simulation duration and time increment Δt—to closely replicate the target pattern. Figure 2 illustrates the workflow of the GRN-Designer algorithm.

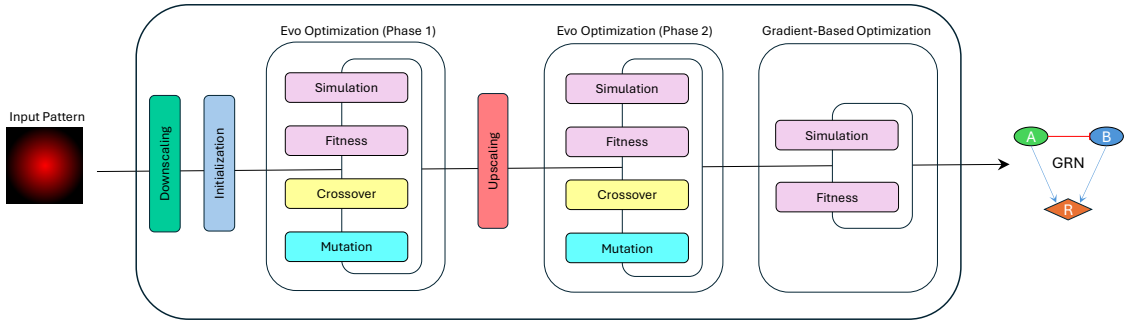


Figure 2: Workflow of the GRN-Designer Algorithm.
Starting with an input pattern, the target is initially downscaled to a reduced resolution $(y, x) \rightarrow (\alpha y, \beta x)$, where α and β are scaling factors between 0 and 1 (green rectangle). Population Initialization (blue rectangle) follows, creating a population of n agents with dimensions $(z, \alpha y, \beta x)$ to match the downscaled target pattern. The algorithm then enters Evolutionary Optimization Phase 1—an evolutionary algorithm incorporating crossover, mutation, and evaluation steps—to begin optimizing the population's fitness. After Phase 1, the population is upscaled to the original target size (z, y, x) (rose rectangle) and undergoes Evolutionary Optimization Phase 2 for further refinement using the same evolutionary strategy. The best agent from the second phase is then passed to a Gradient-Based Optimization block, which uses backpropagation to fine-tune parameters. The final output is a constructed GRN with optimized parameters, capable of reproducing the original target pattern (depicted on the right with green, orange, and blue nodes).

The algorithm starts by down-scaling the target pattern image using third-order interpolation [47], which smooths transitions between pixels. This process is an effort to reduce computational cost by reducing the search domain of the problem.

After down-scaling the target pattern, a population of n agents is initialized (each agent with one gene by default), with each agent representing a potential solution to the problem and subject to optimization. Each agent is structured as a three-dimensional matrix (Figure 3) containing all the essential parameters for the optimization process. For each agent, the parameters described below are initialized by sampling from a uniform distribution, $P \sim U(a, b)$, where P represents the randomly initialized parameter drawn from the uniform distribution

U with range $[a, b)$. These parameters are then stored in designated indices within the agent matrix, ensuring each agent has the necessary information to effectively represent a viable solution.

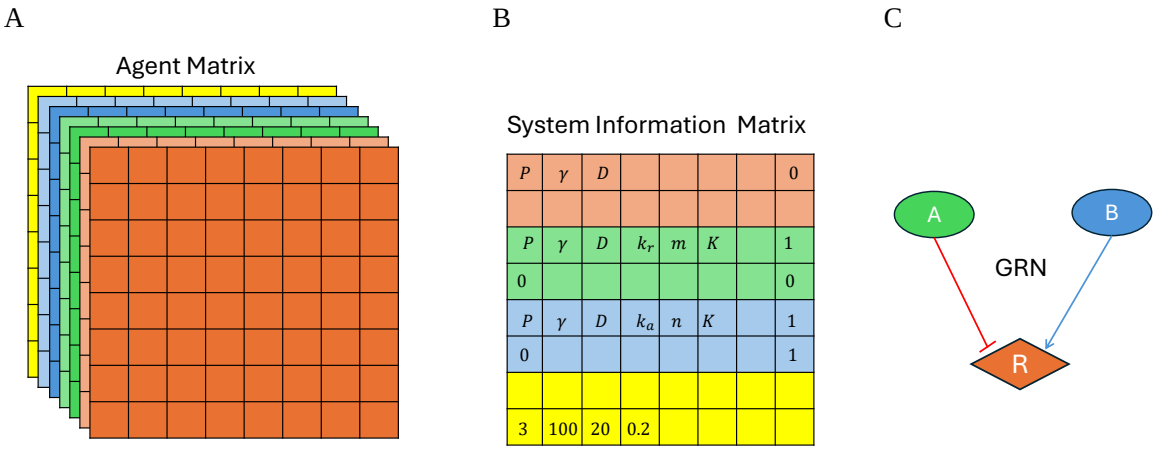


Figure 3: Structure of an agent.
A) The agent matrix, a three-dimensional matrix representing an agent with three genes, labeled as R, A, and B. Each gene is associated with two matrices: a Diffusion Matrix (shown in orange, green, and blue for species R, A, and B, respectively) that stores spatial diffusion patterns, and an Initial Condition Matrix (in light orange, light green, and light blue for species R, A, and B) that defines the initial conditions where the genes present in the system are active and capable of producing molecular species. B) System information matrix (positioned as the last two-dimensional slice within the three-dimensional agent matrix, highlighted in yellow), contains specific parameters and simulation settings. For each species, there are two rows of data: the first row includes the production (P), degradation (γ), and diffusion (D) rate constants, along with interaction parameters (k_a & k_r, n & m, and K), representing interaction rate constants. The end of the first row indicates the number of interactions for each gene, while the second row contains indices of interacting genes and the type of interaction (0 for inhibition, 1 for activation). The final row of the System Information Matrix holds general simulation settings, including the number of genes (3), maximum number of simulation iterations (100), total simulation duration (20), and time increment ($\Delta t = 0.2$). C) The GRN structure for this agent, where arrows indicate activation and inhibition interactions among genes: gene A inhibits R, and B activates R, encapsulating the interaction parameters defined in the System Information Matrix.

- **Initial Conditions**

  For each gene, an initial conditions matrix is defined with dimensions matching those of the target pattern. Each pixel in this matrix corresponds to a specific area containing a defined number of cells ($n \times n$ cells), where both the pixel's length and width represent n cells. These initial conditions describe the **starting levels of gene expression**, reflecting the spatial activation potential of genes at the beginning of the simulation.

  The production of each gene's product in a pixel depends on the intensity of the gene's specific pattern at that location. As a result, **gene expression occurs only in regions where the pattern intensity is non-zero**, and regions with zero intensity do not express the gene. This spatially restricted activation allows for the emergence of localized or gradient-based patterns of gene expression across the culture.

  By adjusting the initial expression levels of genes and the intensities of their patterns, the resulting spatial and temporal behaviors of gene regulatory dynamics can be tailored to mimic diverse biological phenomena. These initial conditions influence the diffusion and interaction dynamics of the gene products, reflecting real biological variability.

  Thus, while the gene regulatory network (GRN) governs the interactions between genes across the system, the activation of specific genes is spatially restricted by their patterns. This heterogeneity in initial conditions and pattern-driven activation enables flexible and realistic modeling of complex biological processes, such as tissue patterning, morphogenesis, and cellular differentiation.

- **Rate Constants for Gene Products and Interactions**

  In a gene regulatory network (GRN), molecular species represent the activity of specific genes or groups of genes. Each molecular species is defined by three key rate constants: the production rate (P), the degradation rate (γ), and the diffusion rate (D).

  Regulatory interactions within the GRN are described using the Hill equation and are characterized by three constants: the binding constant (k_a *or* k_r), the dissociation constant (K), and the Hill coefficient (n *or* m). These constants define the strength and nature of the interaction between molecular species.

  By adjusting these constants, we can precisely control the behavior of molecular species and their interactions within the network.

- **Simulation Settings**

  To accurately model the diffusion pattern, it is essential to specify both the simulation duration (the total time over which the diffusion process

is simulated) and Δt (the time step, representing the speed of change in the system at each iteration). These parameters are crucial because they determine how the diffusion unfolds over time and how precisely we capture changes in the system. During the optimization process, these simulation settings are also tuned to achieve the best match with the target pattern.

Once the initial population of agents is generated with randomly assigned parameters, the algorithm proceeds through successive optimization stages to refine each agent's configuration. The first of these stages is evolutionary optimization (phase one), where agents are iteratively improved through genetic algorithm principles to approach the global optimum.

0.4.1.1 Evolutionary Optimization

Genetic algorithms (GAs) are a class of evolutionary algorithms inspired by the principles of natural selection, first formalized by John Holland in the 1970s [34]. These metaheuristic optimization techniques aim to find globally optimal solutions by simulating the processes of evolution through selection, crossover, and mutation. In a typical GA, a population of agents is maintained, with each agent representing a potential solution to the problem. The evolutionary cycle involves selecting agents for reproduction and manipulating them through crossover and mutation to generate the next generation. The quality of each agent is evaluated using a fitness function, which quantifies how well the agent adapts to the problem environment [35].

Despite their adaptability, classical GAs often struggle to balance exploration (searching for new solutions) and exploitation (refining existing solutions) effectively. This balance is crucial for avoiding premature convergence and ensuring a thorough search of the solution space [26, 36–38]. To address these limitations, a specialized version of the GA called the **Genetic Algorithm Based on Natural Selection (GABONST)**, was developed by M. A. Albadr et al. [26]. GABONST introduces modifications that help to better balance exploration and exploitation, allowing the algorithm to more effectively overcome the common pitfalls of traditional GAs.

In this project, we adapt and extend the GABONST approach to address our specific problem of designing gene regulatory networks (GRNs) capable of producing a target diffusion pattern. This modified version of GABONST is customized to enhance performance in GRN optimization, providing a tailored solution to our unique requirements.

Figure 4 illustrates the workflow of the evolutionary algorithm. It begins by evaluating each agent within the initialized population. For each agent, this evaluation involves simulating the system to generate a two-dimensional diffusion pattern. This pattern is subsequently compared to the target diffusion pattern, with the

difference quantified as the agent's fitness—reflecting how closely its solution approximates the desired target.

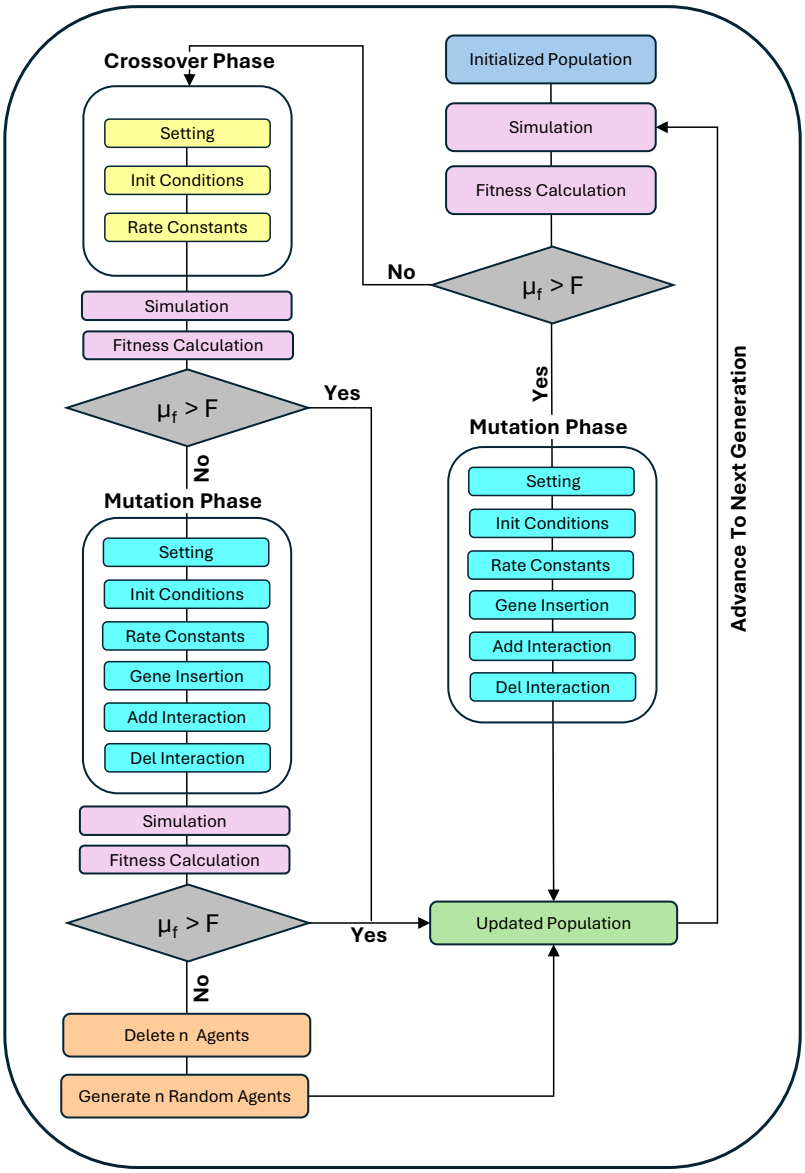


Figure 4: Workflow of the evolutionary algorithm.
The algorithm starts with the evaluation of agents in the randomly initialized population (blue rectangle) by simulation, and their fitness is calculated (purple rectangles). Agents are then divided based on whether their fitness F is greater or less than the population mean fitness μ_f (gray rhombuses). Agents with fitness values below the mean ($F < \mu_f$) undergo mutation, where parameters—simulation setting , initial conditions, rate constants, gene insertion and interaction insertion or deletion—are modified (cyan rectangles). The mutated agents are added to the next-generation pool (green rectangle). Agents with fitness above or equal to the mean ($F \geq \mu_f$) proceed to crossover, where they pair with a top-performing agent. After crossover, they are re-evaluated, and agents with improved fitness ($F < \mu_f$) are sent to the next-generation pool. Those that do not meet this criterion undergo further mutation and re-evaluation. Agents that still fail to improve are removed, and new agents are randomly generated to replace them (orange rectangles). This process of evaluation, mutation, and crossover repeats for k iterations, progressively refining the agent population to achieve optimal performance.

Based on the mean fitness $\bar{f}$ of the population, calculated as follows, the population is divided into two subsets:

$$\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f_i \tag{0.1}$$

$$P_{\text{low}},\ P_{\text{high}} = \{a_i \mid f_i < \bar{f}\},\ \{a_i \mid f_i \geq \bar{f}\} \tag{0.2}$$

The mean fitness of the population, $\bar{f}$, is used to partition the population into two groups: P_{low}, containing agents with fitness values below the mean (those that represent relatively better solutions), and P_{high}, containing agents with fitness values equal to or above the mean.

Agents in P_{low} undergo a set of mutation processes to maintain dversity in the population and are then transferred to the next generation pool. In contrast, agents in P_{high} are given two opportunities to improve their fitness [26].

- **Crossover**

  The first opportunity involves a crossover operation with an elite agent selected from P_{low}. This crossover allows the high-fitness agent to inherit information from a well-qualified agent, potentially enhancing its parameters. After crossover, each agent in P_{high} is re-evaluated. Agents that meet the criterion $f_i < \bar{f}$ are transferred to the next generation pool.

- **Mutation**

  Agents that do not improve through crossover proceed to a mutation process as a second opportunity for adaptation. After mutation, agents are re-evaluated, and those that still fail to meet the mean fitness criterion are removed. To maintain population size, these eliminated agents are replaced by new, randomly initialized agents.

This cycle continues until the algorithm reaches a predefined maximum number of generations or achieves a specific fitness threshold.

0.4.1.1.1 Simulation of GRNs

The simulation models the interactions of genes within a two-dimensional environment. These interactions are described using nonlinear partial differential equations (PDEs) that account for time-dependent dynamics and spatial diffusion of gene products across the simulation grid.

In this gene regulatory network (GRN), each gene (except the reporter gene, R) can activate or inhibit other genes through its products (e.g., transcription factors)

and can also be activated or inhibited by them. The reporter gene, however, is regulated by other genes but does not regulate any gene in return. These regulatory interactions are represented using the Hill equation, which captures the nonlinear effects of activation and inhibition.

Each gene's initial expression is determined by its spatial expression potential, which is defined as the initial condition across the grid. **A gene is active only in regions where its initial condition is greater than zero**. This initial condition serves as the starting point for the simulation, and from there, gene products diffuse, interact, and evolve over time. The spatial distribution and levels of initial expression significantly influence the system's dynamics.

Gene products can diffuse across the grid unless their diffusion coefficient is set to zero to simulate localized expression. Additionally, gene products decay over time at a rate specific to each gene. The concentrations of gene products are governed by the following equation:

$$\frac{\partial g_i}{\partial t} = F_i(\{g_j\}) - \gamma_i g_i + D_i \nabla^2 g_i \tag{0.3}$$

Here:

- $F_i(\{g_j\})$: Regulatory function representing the net production rate of g_i based on interactions with other genes g_j.
- $\gamma_i g_i$: Decay term, where γ_i is the decay rate of g_i.
- $D_i \nabla^2 g_i$: Diffusion term, where D_i is the diffusion coefficient and ∇^2 is the Laplacian operator modeling spatial diffusion.

The regulatory function $F_i(\{g_j\})$ models the effects of activators and repressors using the Hill equation. To simplify the model, the effects of activators and repressors are assumed to be additive, meaning each independently contributes to the regulation of the target gene. This function is expressed as:

$$F_i(\{g_j\}) = P_i \rho_i(x, y) + \sum_{j \in A_i} k_{a_{ij}} \frac{g_j^{n_{ij}}}{K_{ij}^{n_{ij}} + g_j^{n_{ij}}} - \sum_{k \in R_i} k_{r_{ik}} \frac{g_k^{m_{ik}}}{K_{ik}^{m_{ik}} + g_k^{m_{ik}}} \tag{0.4}$$

Where:

- P_i: Baseline production rate constant for gene i.
- $\rho_i(x, y)$: Initial spatial activation potential of gene i at position (x, y).
- $k_{a_{ij}}$: Contribution of activator j to the activation rate of gene i.
- $k_{r_{ik}}$: Contribution of repressor k to the repression rate of gene i.

- A_i: Set of genes activating gene i.
- R_i: Set of genes repressing gene i.
- K_{ij}: Dissociation constant (threshold) for activator j.
- K_{ik}: Dissociation constant (threshold) for repressor k.
- n_{ij}: Hill coefficient indicating the cooperativity of activators.
- m_{ik}: Hill coefficient indicating the cooperativity of repressors.

Boundary conditions are applied to ensure no gene product escapes the simulation grid. These no-flux (Neumann) boundary conditions are expressed as:

$$\frac{\partial g_i}{\partial n} = 0 \tag{0.5}$$

Here, n represents the outward normal vector at the boundary.
In practice:

- Interior grid cells interact with their four immediate neighbors (top, bottom, left, right).
- Edge grid cells interact with three neighbors.
- Corner grid cells interact with two neighbors.

These boundary conditions prevent material loss from the grid, ensuring the conservation of mass within the simulation region.

0.4.1.1.2 Fitness Function

To evaluate the performance of each agent in the population, each agent is used to simulate the system and produce a pattern associated with the reporter gene, referred to as the constructed pattern. This constructed pattern is then compared to a predefined target pattern. This comparison, known as fitness calculation, quantitatively evaluates the performance of each agent. The fitness metric used in the algorithm combines the mean squared error (MSE) and the structural similarity index (SSIM), providing a balanced measure of accuracy and structural alignment between the constructed and target patterns [39].

- Mean Squared Error (MSE) Fitness

  MSE inforces the system to reduce the pixel-wise differences between simulation result and the target pattern [40,41]. The mathematical equation of

MSE is as follows:

$$L_{mse}(\hat{y}, y) = \frac{1}{wh} \sum_{i=1}^{w} \sum_{j=1}^{h} \left(\hat{y}_{(i,j)} - y_{(i,j)} \right)^2 \tag{0.6}$$

Here, $y_{(i,j)}$ and $\hat{y}_{(i,j)}$ are the compared pixels from target and constructed pattern, respectively, and w and h are length and width of the target and constructed pattern.

- Structural Similarity Index (SSIM) Fitness

  since the constructed pattern needs to match the target pattern not only pixel-wise but on perceptual level, the use of a second metric is fundamental for the algorithm. The SSIM is a metric to measure the perceptual difference between two images based on luminance, contrast and texture [42,43]. The equation below illustrate how to calculate this metric:

$$L_{\text{ssim}}(\hat{y}, y) = 1 - \frac{(2\mu_{\hat{y}}\mu_y + C_1)(2\sigma_{\hat{y}y} + C_2)}{(\mu_{\hat{y}}^2 + \mu_y^2 + C_1)(\sigma_{\hat{y}}^2 + \sigma_y^2 + C_2)} \tag{0.7}$$

  Here, μ_y and $\mu_{\hat{y}}$ denote the mean of constructed and target patterns, respectively. σ_y^2 and $\sigma_{\hat{y}}^2$ represent the variances of target and constructed patterns.

  $\sigma_{\hat{y}y}$ indicates the covariance of of $\hat{y}$ and y.

  C_1 and C_2 are small numbers to avoid division by zero.

To have the benefits of both metrics, they can be combined to from the final fitness function as follows:

$$L_{\text{t}}(\hat{y}, y) = \alpha L_{mse}(\hat{y}, y) + \beta L_{\text{ssim}}(\hat{y}, y) \tag{0.8}$$

Here, α and β are hyper parameters which should be choose carefully based on the nature of the target pattern.

0.4.1.1.3 Mutation Processes

To introduce diversity into the population, each agent undergoes a set of mutation processes that apply small, random changes based on a predefined mutation rate. These mutations target specific regions in the agent matrix, modifying key parameters that influence the agent's behavior. Six mutation processes are used to alter various aspects of each agent's parameters:

- Initial Condition Mutation

  In this mutation, the initial conditions for each gene within an agent are randomly modified, which affects the initial expression levels or activation potential of the genes. This alteration can lead to diverse patterns in agent behavior by influencing the spatial and temporal expression dynamics of the genes.

- Simulation Setting Mutation

  This mutation targets hyperparameters that define the simulation timeline, specifically the simulation duration and time increment Δt. These parameters are adjusted based on the mutation rate to introduce variability in the simulation dynamics.

- Species Rate Constant Mutation

  The behavior of each molecular species is defined by a set of rate constants (see 0.4.1.1.1). Each of these rate constants undergoes random modification according to the mutation rate, allowing for changes in species dynamics.

- Gene Insertion Mutation

  This mutation introduces a new gene into the system with a certain mutation probability. Adding new gene increases the complexity of interactions within the agent.

- Interaction Insertion Mutation

  A new regulatory interaction may be added between genes, with both the source (affector) and target (affected gene) selected randomly. The type of interaction (activation or inhibition) is also chosen randomly. This mutation increases the connectivity of the system.

- Interaction Deletion Mutation

  In contrast to interaction insertion, this mutation removes an existing interaction between genes. A randomly selected interaction is deleted, reducing the overall connectivity and potentially simplifying the system's regulatory network.

The first three mutation processes—initial condition mutation, simulation setting mutation, and species rate constant mutation—use a similar approach to alter gene parameters in each agent. For each targeted parameter, a binary mutation mask is generated based on a mutation probability, μ, which determines whether a mutation will occur:

$$M_i = \begin{cases} 1 & \text{if rand}(i) < \mu, \\ 0 & \text{otherwise,} \end{cases} \tag{0.9}$$

where M_i represents the mutation mask for parameter i in the agent. If $M_i = 1$, the parameter will be mutated; if $M_i = 0$, it remains unchanged. The mutation probability μ dictates the likelihood of each parameter being selected for mutation, and $\text{rand}(i)$ represents a randomly generated float from the interval $[0, 1)$.

When a mutation is applied, a new mutant agent is generated using three randomly chosen agents, A_1, A_2, and A_3, from the population. The mutant agent $\tilde{A}$ is calculated as:

$$\tilde{A} = A_1 + F \cdot (A_2 - A_3), \tag{0.10}$$

where F is a real constant in the range $[0, 2]$ that controls the scale of the differential variation [37].

Next, the current agent's selected parameters are updated based on the mutation mask as follows:

$$A'_i = \begin{cases} \tilde{A}_i & \text{if } M_i = 1, \\ A_i & \text{if } M_i = 0, \end{cases} \tag{0.11}$$

ensuring that only parameters indicated by the mask are updated, while other parameters retain their original values.

In the gene insertion mutation, agents undergo mutation at a specific rate. When a new gene is added to an agent, it establishes a connection with an existing gene in the system, chosen randomly. This connection helps integrate the newly added gene and allows it to influence the system dynamics.

The final two mutation processes, interaction deletion mutation and interaction insertion mutation, act in contrast to each other by removing and adding regulatory interactions between genes. These mutations help maintain a balanced system by avoiding excessive complexity while promoting manageable diversity.

0.4.1.1.4 Crossover Processes

In addition to mutation processes that introduce diversity, crossover processes encourage similarity within the system by recombining parameters across agents. Three types of agent parameters undergo crossover, as described below:

- Simulation Setting Crossover

The simulation duration and time increment, Δt, are hyperparameters defining the simulation timeline. These variables undergo crossover in agents with fitness equal to or higher than the average fitness of the population.

- Initial Condition Crossover

  During initial condition crossover, the starting expression levels or activation potential of genes for each agent undergo crossover. This means the initial conditions for gene expression are exchanged between agents, leading to a recombination of the spatial activation patterns and expression dynamics.

- Species Rate Constant Crossover

  Molecular species behaviors are governed by specific rate constants (see 0.4.1.1.1), each of which undergoes crossover in agents with fitness equal to or higher than the population mean.

All three crossover processes use an arithmetic crossover approach [26, 44, 45] to adjust the parameters of the corresponding agent. This process begins by selecting a set of elite agents, which are defined as the agents with the lowest fitness values in the population. For each agent undergoing crossover, an elite agent is randomly selected to blend its properties with the high-fitness agent undergoing crossover. This approach enables the modification of agents with high fitness by incorporating features from elite agents.

To control the degree of influence from the elite agent, a hyperparameter α (within the range $[0, 1]$) is used to generate a crossover factor, β, drawn uniformly from the range $[-\alpha, 1 + \alpha]$. The crossover factor, β, determines the contribution of the elite agent versus the current agent.

$$\beta \sim U(-\alpha, 1 + \alpha) \tag{0.12}$$

The generated recombination factor β is then used to adjust the parameters of the current agent as follows:

$$P' = \beta \cdot P + (1 - \beta) \cdot P_{\text{elite}} \tag{0.13}$$

where:

- P is the parameter from the current agent,
- P_{elite} is the corresponding parameter from the elite agent,
- P' is the updated parameter after crossover.

0.4.1.2 Gradient-Based Optimization

Optimizing an objective function—whether by minimization or maximization—is central to many scientific and engineering applications. When an objective function is differentiable with respect to its parameters, gradient-based methods, such as gradient descent, offer a relatively efficient approach for optimization [32]. These methods rely on calculating first-order partial derivatives for all parameters, a process that has the same computational complexity as simply evaluating the function itself [30]. Common gradient-based algorithms include stochastic gradient descent, RMSProp, and Adam, which have been widely used due to their capacity to reach local minima (or maxima) within a global optimum area in a structured and computationally efficient manner.

In the GRN Designer algorithm, the initial optimization phase is performed with a genetic algorithm to explore a wide global search space, which helps in establishing a preliminary global minimum for the problem. After this broad search, the algorithm applies a more targeted gradient-based optimization approach—specifically the Adam method (Adaptive Moment Estimation). Adam refines the solution by adjusting learning rates for each parameter based on estimated first and second gradient moments, making it particularly efficient for convergence compared to traditional methods such as stochastic gradient descent (SGD) [28].

Once the genetic algorithm shapes the foundational structure of the gene regulatory network (GRN) and arrives at an acceptable preliminary solution [50], the result is passed to the second algorithm phase for fine-tuning. In this gradient-based phase, Adam optimizes specific parameters and initial conditions within the GRN framework [46], focusing only on the single agent with the lowest fitness to improve the solution's precision and accuracy.

The optimization process starts with a forward pass to simulate the system, using the previously defined methods for simulation. The result is compared to the target pattern using the same fitness function—a combination of Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM)—since the function is differentiable. In this context, the forward pass propagates information from the simulation step to the cost function. After the forward pass, backpropagation [32] calculates gradients from the cost function with respect to each system parameter and the initial conditions of the molecular species in the Gene Regulatory Network (GRN).

0.4.1.2.1 Mathematical Formulation of the Cost Function

The cost function L_t, a weighted combination of MSE and SSIM as defined in 0.4.1.1.2, is expressed as:

$$L_{\text{t}}(\hat{y}, y) = \alpha L_{\text{mse}}(\hat{y}, y) + \beta L_{\text{ssim}}(\hat{y}, y) \tag{0.14}$$

where:

- L_{mse}: The Mean Squared Error, capturing pixel-wise differences between $\hat{y}$ and y.
- L_{ssim}: The Structural Similarity Index Measure, capturing structural differences between $\hat{y}$ and y.
- α: A weighting factor for MSE, emphasizing pointwise accuracy.
- β: A weighting factor for SSIM, emphasizing structural similarity.

The gradient of the L_t with respect to $\hat{y}$ is computed as:

$$\frac{\partial L_{\text{t}}}{\partial \hat{y}} = \alpha \cdot \frac{\partial L_{\text{mse}}}{\partial \hat{y}} - \beta \cdot \frac{\partial L_{\text{ssim}}}{\partial \hat{y}} \tag{0.15}$$

This derivative combines the gradients of MSE and SSIM, weighted by α and β.

0.4.1.2.2 Computing Gradients for System Parameters

Using the chain rule, the gradient of the total cost L_{t} with respect to each parameter is computed through successive partial derivatives. For instance, the gradient with respect to the diffusion coefficient for gene i (D_i)is given by:

$$\frac{\partial L_{\text{t}}}{\partial D_i} = \frac{\partial L_{\text{t}}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial \nabla^2 g_i} \cdot \frac{\partial \nabla^2 g_i}{\partial D_i} \tag{0.16}$$

where:

- $\hat{g}_i$: Simulated concentration of the product of gene i.
- $\nabla^2 g_i$: Laplacian of g_i, representing spatial diffusion.
- $\frac{\partial \hat{y}}{\partial \hat{g}_i}$: Gradient of the constructed pattern with respect to the constructed pattern of gene i.
- $\frac{\partial \hat{g}_i}{\partial \nabla^2 g_i}$: Sensitivity of the the constructed pattern to changes in the Laplacian term.
- $\frac{\partial \nabla^2 g_i}{\partial D_i}$: Sensitivity of the Laplacian term to the diffusion coefficient D_i.

Similar expressions are derived for other parameters: production rate (P), degradation rate (γ), binding rate (k_a *or* k_r), Hill coefficient (n *or* m) and dissociation rate (K):

$$\frac{\partial L_t}{\partial P_i} = \frac{\partial L_t}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial P_i} \tag{0.17}$$

$$\frac{\partial L_t}{\partial \gamma_i} = \frac{\partial L_t}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial \gamma_i} \tag{0.18}$$

$$\frac{\partial L_t}{\partial k_{ai} \text{ or } k_{ri}} = \frac{\partial L_t}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial k_{ai} \text{ or } k_{ri}} \tag{0.19}$$

$$\frac{\partial L_t}{\partial n_i \text{ or } m_i} = \frac{\partial L_t}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial n_i \text{ or } m_i} \tag{0.20}$$

$$\frac{\partial L_t}{\partial K_i} = \frac{\partial L_t}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial K_i} \tag{0.21}$$

Finally, the gradient with respect to the initial conditions of gene i (ρ_i) is computed as:

$$\frac{\partial L_t}{\partial \rho_i} = \frac{\partial L_t}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{g}_i} \cdot \frac{\partial \hat{g}_i}{\partial \rho_i} \tag{0.22}$$

Note: These equations provide a general framework, where exact parameter-specific derivatives should replace the symbolic expressions during actual gradient calculations.

0.4.1.2.3 Parameter Updates Using Adam Optimization

Once the gradients are computed, parameters and initial conditions are updated via the Adam optimization algorithm [28]. Adam combines adaptive learning rates with momentum- based updates, using first and second moments of gradients for each parameter. The first and second moment estimates of the gradients are computed as:

$$m_i = \beta_1 m_{i-1} + (1 - \beta_1) \frac{\partial L_i}{\partial \theta} \tag{0.23}$$

$$v_i = \beta_2 v_{i-1} + (1 - \beta_2) (\frac{\partial L_i}{\partial \theta})^2 \tag{0.24}$$

Here:

- $\frac{\partial L_i}{\partial \theta}$: Gradient of the loss with respect to parameter θ at iteration i.

- m_i: First moment estimate.
- v_i: Second moment estimate.
- β_1, β_2: Decay rates (commonly $\beta_1 = 0.9$, $\beta_2 = 0.999$).

Bias-corrected moment estimates are then computed as:

$$\hat{m}_i = \frac{m_i}{1 - \beta_1^i}, \quad \hat{v}_i = \frac{v_i}{1 - \beta_2^i} \tag{0.25}$$

Finally, parameters are updated as:

$$\theta' = \theta - \alpha \cdot \frac{\hat{m}_i}{\sqrt{\hat{v}_i} + \epsilon} \tag{0.26}$$

Where:

- α is the learning rate.
- ϵ is a small constant for numerical stability.

By iteratively applying this process, the constructed pattern gradually aligns with the target pattern. Upon convergence, the final optimized result outputs include the parameters, initial conditions, simulation duration, time step Δt, and GRN structure.

0.5 Results

We evaluated the performance of the GRN Designer algorithm on nine distinct pattern generation tasks. Each target pattern was created with a resolution of 100×100 pixels. To manage computational costs, the simulation was run for a maximum of 100 iterations with a time step $\Delta t \approx 0.1$, resulting in a total simulation time of ≈ 10 (i.e., $\Delta t \times 100 \approx 10$). In the first stage of the algorithm—the evolutionary optimization stage—a fixed number of 100 epochs was applied across all cases without downscaling.In each case, a population of 200 agents, each with a single reporter gene, is initialized. The complexity of the agents is controlled by adjusting the mutation rates (e.g., `species_insertion_mutation_rate`, `connection_insertion_mutation_rate, and connection_deletion_mutation_rate`. After this stage, each constructed gene regulatory network (GRN) was passed to the second stage (gradient-based optimization) for further refinement until reaching a satisfactory convergence.

For these experiments, we started by constructing simpler GRNs with fewer genes and interactions, gradually increasing complexity by adjusting hyperparameters of the system. The number of genes in each GRN ranged from 3 in the simplest case to 12 in the most complex. Table 1 summarizes the complexity details for each constructed GRN, showing the number of genes, interactions, and trainable parameters involved.

Table 1: Details of GRN model complexities.
Each row corresponds to a model tested, with columns indicating the name of the target pattern, the number of genes, the number of interactions (activation or inhibition), and the total number of trainable parameters. For each gene, there is an initial condition matrix of shape $100 \times 100 = 10,000$ parameters, three rate constants P, γ, and D), and a three parameters for each interaction (k_a *or* k_r, K, and n *or* m).

Model	Pattern	Genes	Interactions	Trainable Parameters
1	Gradient	3	4	30,021
2	Rectangle	4	7	40,033
3	Circle	5	8	50,039
4	Infinity	6	10	60,048
5	Rings	7	15	70,066
6	M-Shape	8	13	80,063
7	Chromosome	9	18	90,081
8	DNA Strand	10	21	100,093
9	Biohazard	12	25	120,111

As an example of the process of GRN creation and optimization, Figure 5 illustrates the evolution and resulting data for Model 4. In this model, an infinity-shaped diffusion pattern was provided as the target. The system was initially configured with a single gene (the reporter gene), and through evolutionary optimization, five additional genes were added, resulting in a GRN with six genes. Here, gene R serves as the reporter gene, which is the target for optimization to match the diffusion pattern, while the other five genes (A, B, C, D, and D)are responsible for regulation of the overall GRN behavior.

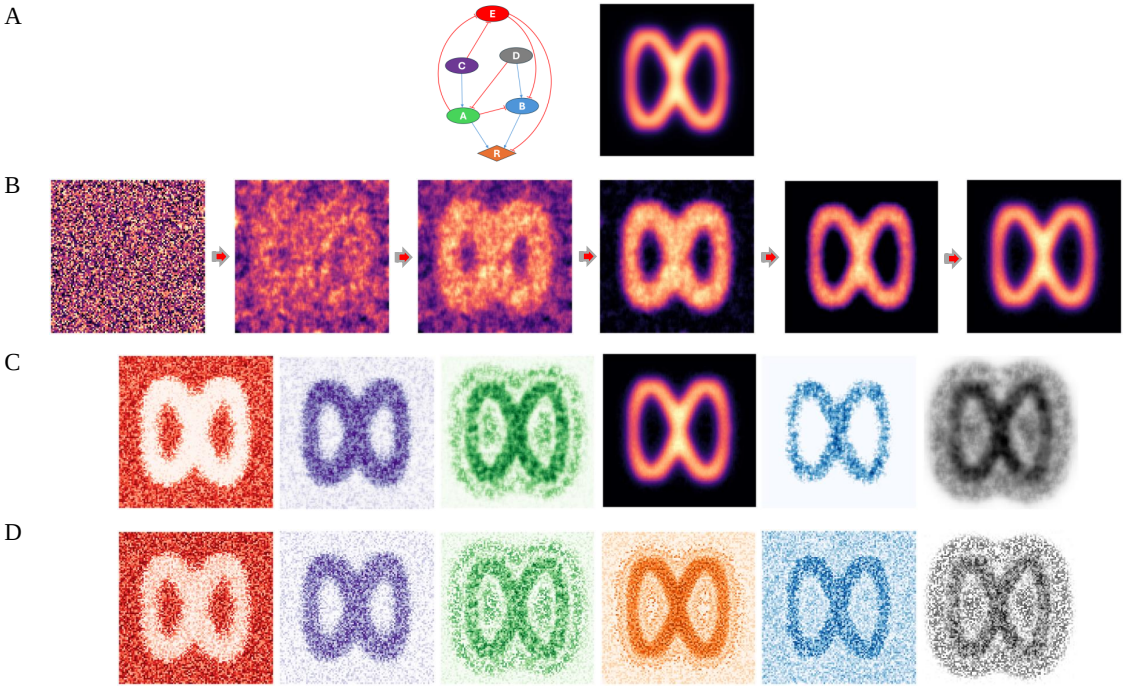


Figure 5: Results of model 4 (infinity symbol).
A) *Left*: Constructed Gene Regulatory Network (GRN) for pattern generation, consisting of 6 genes labeled as R (reporter gene), A, B, C, D, and E. Arrows indicate regulatory interactions, with red lines denoting inhibition and blue lines denoting activation. *Right*: The target pattern provided as input to the GRN Designer algorithm, guiding the optimization of gene parameters and initial conditions for pattern similarity.
B) Sequence of simulation results for the reporter gene R at the endpoint of selected epochs, illustrating the progression toward the optimized pattern. Each stage shows progressive optimization as the system minimizes differences between the constructed pattern and the target pattern.
C) Final diffusion patterns for each gene product, with each color corresponding to its associated gene in the GRN.
D) Optimized initial conditions for each gene, showing the starting configurations after parameter tuning.

In Figure 6, the results of Model 5 are presented, showcasing its ability to produce a rings pattern. Panel A illustrates the target pattern (left) alongside the pattern generated by the reporter gene (R) in the constructed Gene Regulatory Network (GRN) (right). The GRN, shown in Panel B (center), consists of seven interacting genes (A, B, C, D, E, F, and R), which collectively shape the target pattern through a combination of activation (blue arrows) and inhibition (red

arrows).

The optimized initial conditions for each gene are displayed around the GRN in colors corresponding to the genes. These initial conditions, combined with the regulatory interactions within the GRN, play a pivotal role in pattern formation. Genes C, D, and E serve as key up-regulators of the reporter gene R, both directly and indirectly. These genes directly enhance R's expression and also inhibit A, B, and F, which are negative regulators of R. Interestingly, the initial conditions of C, D, and E are similar to the initial condition of R, reflecting their positive influence on R's expression.

Gene F, a down-regulator of R, exhibits a complementary initial condition to R. This relationship aligns with F's role in suppressing R both directly and indirectly by down-regulating C. On the other hand, the initial conditions of gene A generate minimal patterning, suggesting that A has a limited impact on R within this network configuration.

Gene B presents an intriguing case. Despite being a down-regulator of R, its initial condition closely resembles that of R. This apparent contradiction can be attributed to the inhibitory effects exerted on B by genes C and D, both of which are positive regulators of R. These interactions likely neutralize B's negative influence on R, balancing its role within the network.

Overall, the results from Model 5 underscore the intricate interplay between initial conditions and regulatory interactions in achieving the target rings pattern. The GRN's ability to balance activation and inhibition pathways highlights the complexity and robustness of the design. Complementary and neutralizing relationships between certain genes further demonstrate the sophisticated dynamics at play, where individual gene effects are modulated by the broader network.

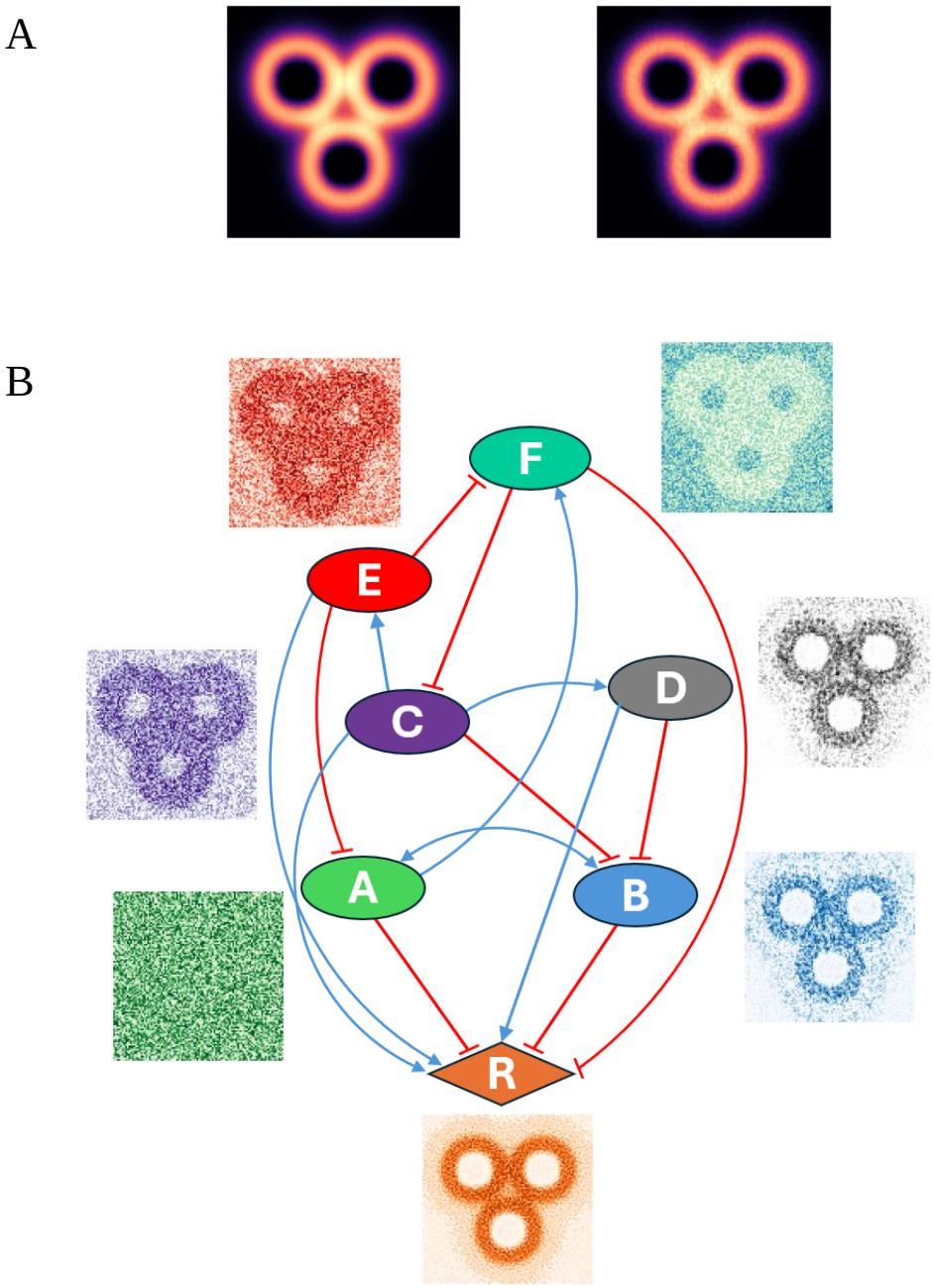


Figure 6: Results of model 5 (rings symbol).
A) Target pattern (left) and the generated pattern for the reporter gene *R* (right). B) The constructed Gene Regulatory Network (GRN) used to generate the pattern, showing seven genes: *R* (reporter gene), *A*, *B*, *C*, *D*, *E*, and *F*. Around the GRN, the optimized initial conditions for each gene are shown with colors matching the genes in the network. Arrows indicate the regulatory interactions: blue arrows show activation, and red arrows show inhibition.

Figure 7 presents the results of three relatively simple GRN models generated by the GRN-Designer. These models, which have fewer genes and interactions compared to the more complex networks explored in this study, allow for a clearer analysis of how the optimization process shapes initial conditions based on the interactions between genes. In each row, the left column shows the target pattern that the model aims to replicate, the middle column depicts the constructed GRN diagram designed to achieve that pattern, and the right column illustrates the constructed pattern produced by the GRN. The Optimized Initial Conditions column (far right) shows the starting spatial distributions for each gene, with color-coded representations that match the nodes in the GRN diagram.

In Model 1 (Gradient), gene B inhibits the reporter gene R, leading to complementary initial conditions for B relative to R. Gene A also inhibits R, but its initial conditions are more similar to those of R, suggesting that the effect of B on R is likely more pronounced than the effect of A on R.

These models demonstrate how even simple GRNs, with carefully optimized initial conditions and targeted interactions, can generate diverse spatial patterns, illustrating the flexibility and power of the GRN-Designer approach.

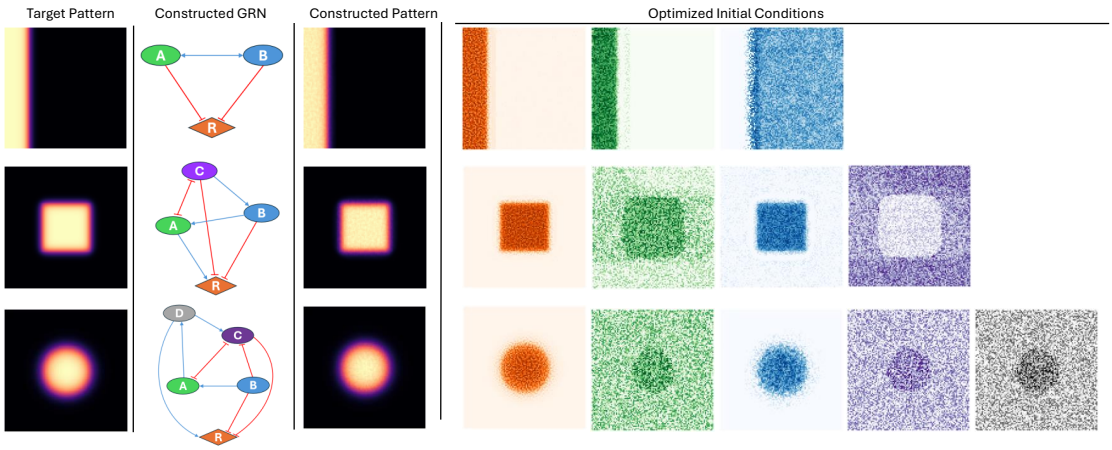


Figure 7: Results of relatively simple models (gradient, rectangle, circle). Examples of relatively simple GRN models generated by GRN-Designer. From top to bottom: Model 1 (gradient), Model 2 (rectangle), and Model 3 (circle). Each row illustrates a distinct target pattern (left), the constructed Gene Regulatory Network (GRN) to achieve that pattern (center), and the resulting pattern produced by the GRN (right). The Optimized Initial Conditions column shows the starting conditions for each gene involved in the GRN, color-coded to match the nodes in the GRN. These simplified models demonstrate how GRNs with a small number of genes and interactions can create diverse spatial patterns.

0.5.1 Results for More Complex Models

For models with increased complexity (Models 6 to 9), we observe a higher number of genes and interactions, leading to richer patterns. In Figure 8, Model 6 (M-shape pattern) demonstrates a eight-gene GRN that combines activation and inhibition interactions to achieve a structured M-shaped pattern. The GRN has thirteen connections, finely tuned to generate the pattern's distinctive structure.

In Models 7, 8, and 9 (shown in Figures 9, 10, and 11, respectively), the GRNs progressively increase in complexity, with up to 12 genes and 25 interactions. Model 7 (chromosome symbol) features intricate interactions between genes, with a carefully balanced mix of activation and inhibition, producing the desired complex structure. Model 8 (DNA-strand), and Model 9 (biohazard symbol) required extensive tuning of initial conditions to achieve intricate shapes, showcasing the algorithm's ability to handle complex target patterns by balancing each gene's influence.

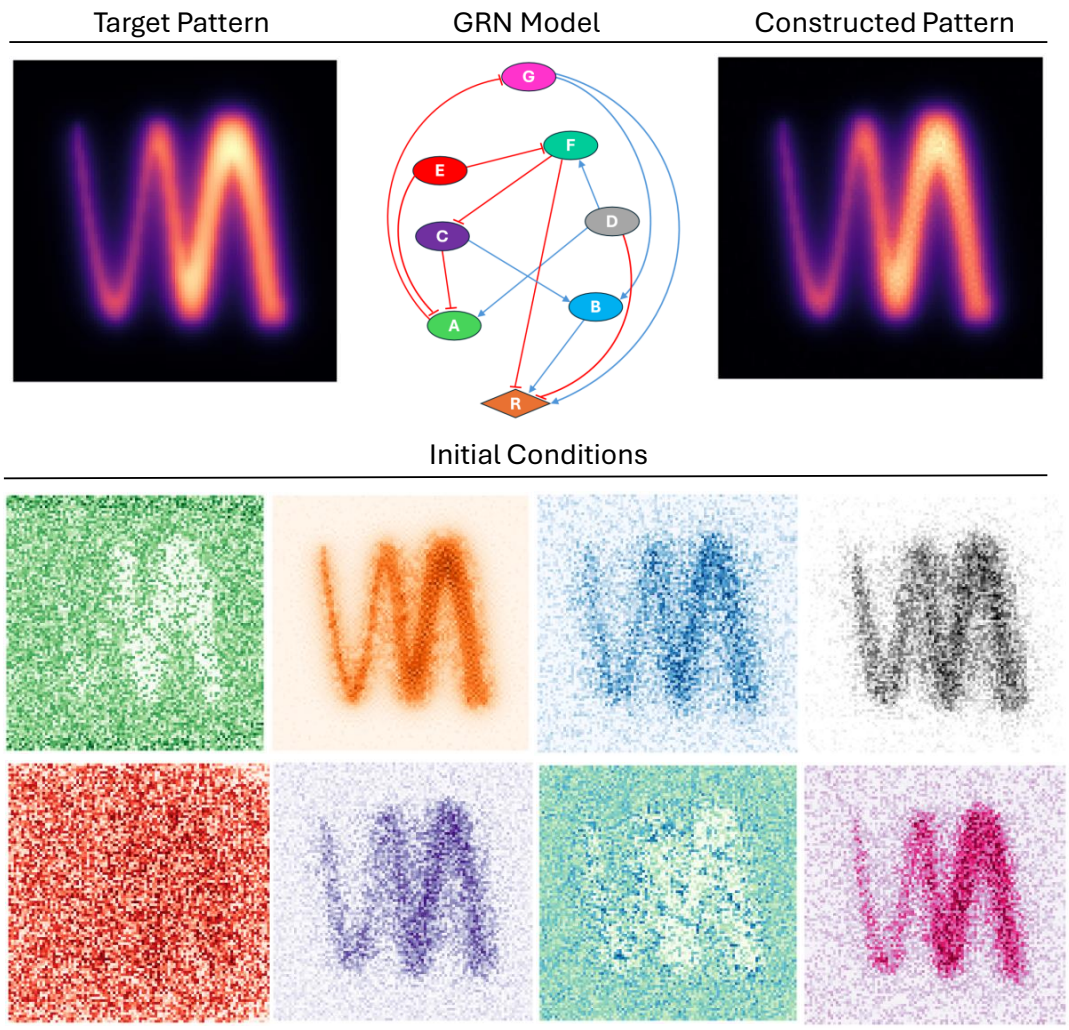


Figure 8: Results of model 6 (M-Shape).
Visualization of a gene regulatory network (GRN) containing eight genes, each influencing the spatial pattern formation through regulatory interactions. Initial conditions are color-coded to match each corresponding gene in the GRN, providing a clear view of the spatial distribution at the start of the simulation.

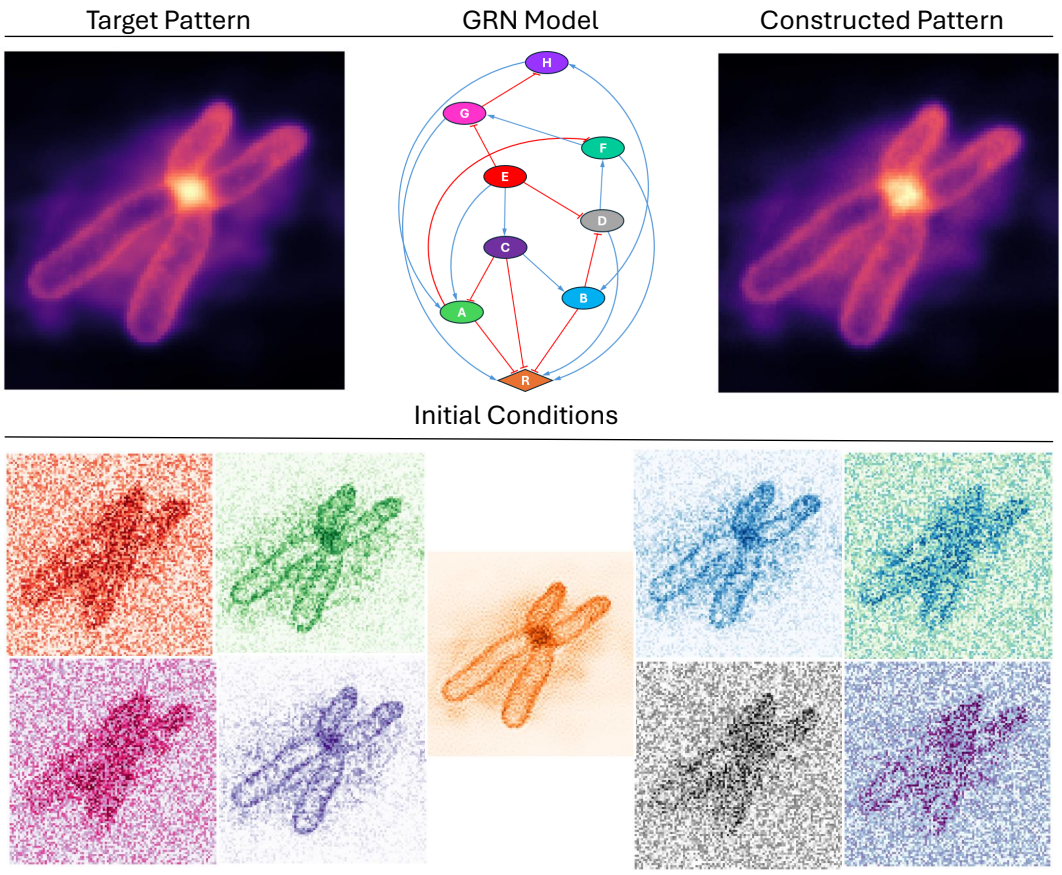


Figure 9: Results of model 7 (chromosome symbol).
This GRN model features 9 genes with a complex network of activation and inhibition interactions (indicated by blue and red lines, respectively). Initial conditions are color-coded to correspond to each gene's influence on the resulting spatial pattern.

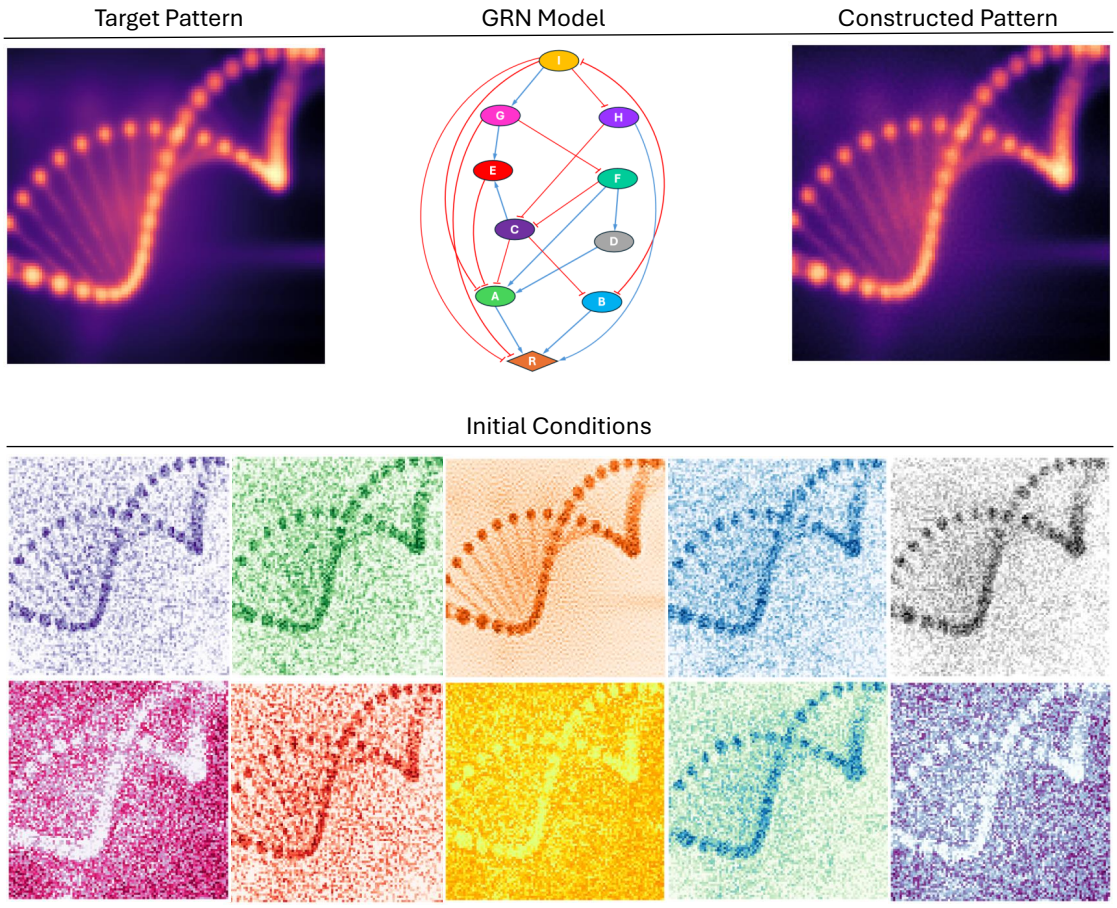


Figure 10: Results of model 8 (DNA-Strand).
This GRN model includes 10 genes, with an extended network of regulatory interactions. Each gene's initial condition, represented by color coding, contributes to generating the constructed spatial pattern shown.

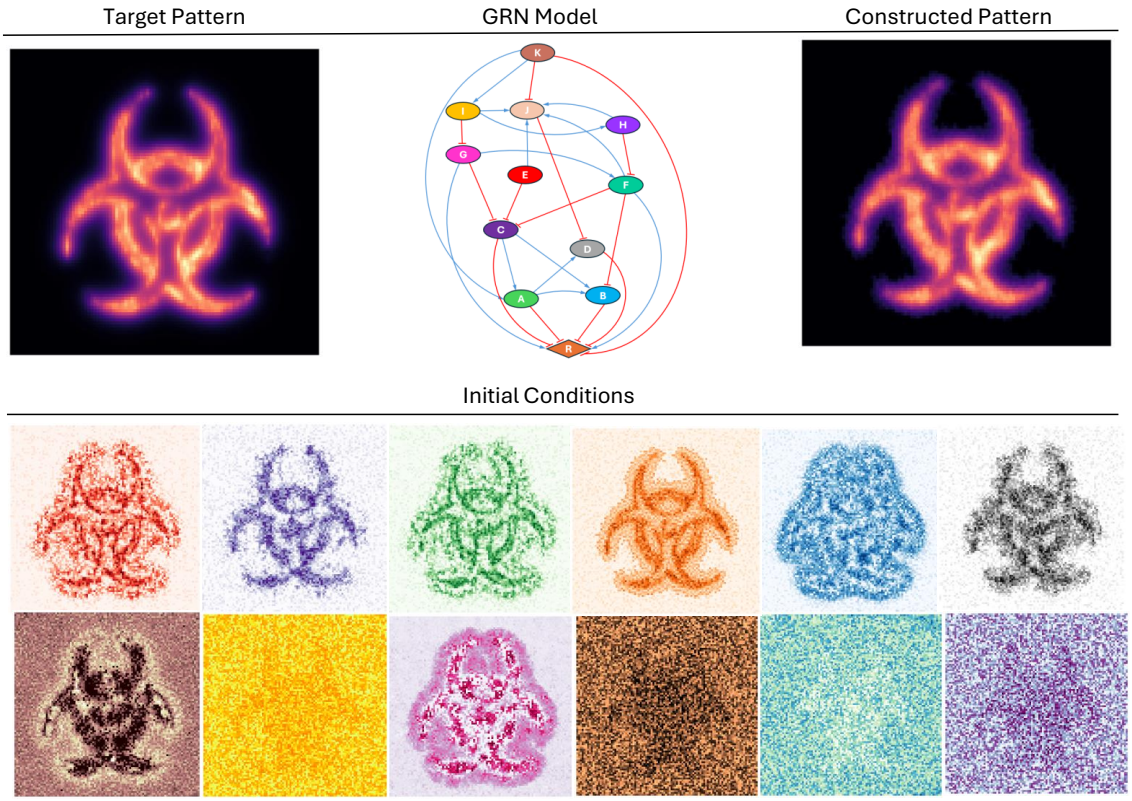


Figure 11: Results of model 9 (biohazard symbol)
The most complex GRN model in this series, with 12 genes interacting through activation (blue lines) and inhibition (red lines). The initial conditions, visualized in color, align with each gene's role in achieving the target pattern.

Figure 12 presents a detailed analysis of the relationship between model complexity and the optimization duration across different models. The figure shows how the number of trainable parameters, which reflects the model's complexity, influences the time required for optimization. As the model complexity increases—due to factors such as the number of genes, initial conditions, and regulatory interactions—the optimization time also increases. This trend underscores the growing computational demands associated with simulating more intricate and detailed GRNs. The observed linear relationship highlights the challenges of scaling up these models for larger, more complex biological systems, where even small increases in complexity can significantly impact the time and resources needed for model optimization.

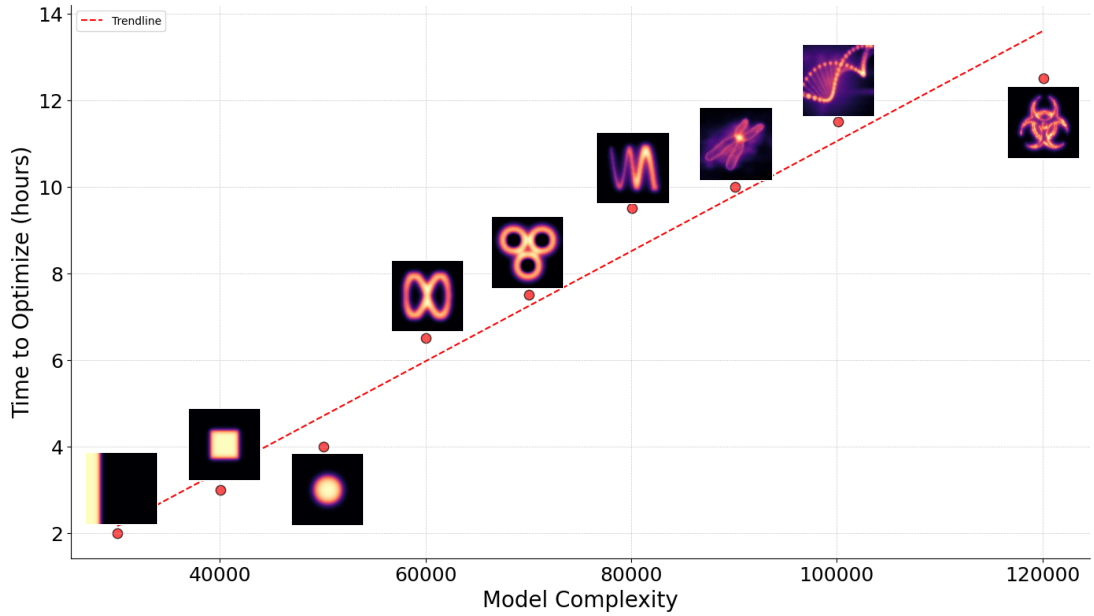


Figure 12: Linear relationship between model complexity and optimization time. Model complexity is quantified by the number of trainable parameters, calculated based on the number of genes, initial conditions, and regulatory interactions. Each gene initial conditions contributes 10,000 parameters due to the 100x100 compartment size, and three reaction rate constants are associated with each gene product. Additionally, each regulatory interaction has three associated rate constants, further contributing to model complexity. The observed trend (shown by the red trendline) indicates a linear increase in optimization time (y-axis) as model complexity (x-axis) increases, highlighting the computational demands of simulating more intricate Gene Regulatory Networks (GRNs). Each data point is annotated with a representative pattern generated by the respective GRN model, illustrating the diversity and complexity of spatial patterns achievable.

0.5.2 Effect of Downscaling on GRN-Designer Performance

As described in the methodology, the GRN-Designer algorithm includes an initial downscaling process that reduces the resolution of each agent in the population. This downscaling is intended to lower the computational cost during the first stage of the evolutionary optimization by using lower-resolution representations of agents, thus enabling faster processing. To evaluate the impact of downscaling on both computational efficiency and solution quality, we tested three models—Model 1 (gradient pattern), Model 2 (rectangle pattern) and Model 6 (M-shape pattern)—with and without downscaling.

In each test, the population consisted of 300 agents, each with a fixed Gene Regulatory Network (GRN) comprising a specific number of genes. These included three genes for Models 1 and 2 (R, A, and B) and seven genes for Model 6 (R, A, B, C, D, E, and F). The evolutionary process was divided into two stages:

- Stage 1: 200 iterations with and without downscaled agent resolution.
- Stage 2: 100 iterations with the original agent resolution.

The algorithm was run twice for each model—once with downscaling and once without—all other parameters remaining fixed.

Results showed that downscaling reduced runtime by nearly 50 percent across all models, without compromising pattern quality. Specifically, Model 1 (gradient) completed in 3,356 seconds with downscaling and 6,221 seconds without; Model 2 (rectangle) completed in 3,426 seconds with downscaling and 6,332 seconds without; and Model 6 (M-shape) completed in 8,193 seconds with downscaling and 13,223 seconds without. Additionally, convergence was faster in models when downscaling was applied, as shown by lower final costs in the loss plots.

Figure 13 illustrates these results. In each case, downscaling not only shortened runtime but also produced lower final costs, indicating better pattern accuracy and improved initial conditions for the reporter gene (R).

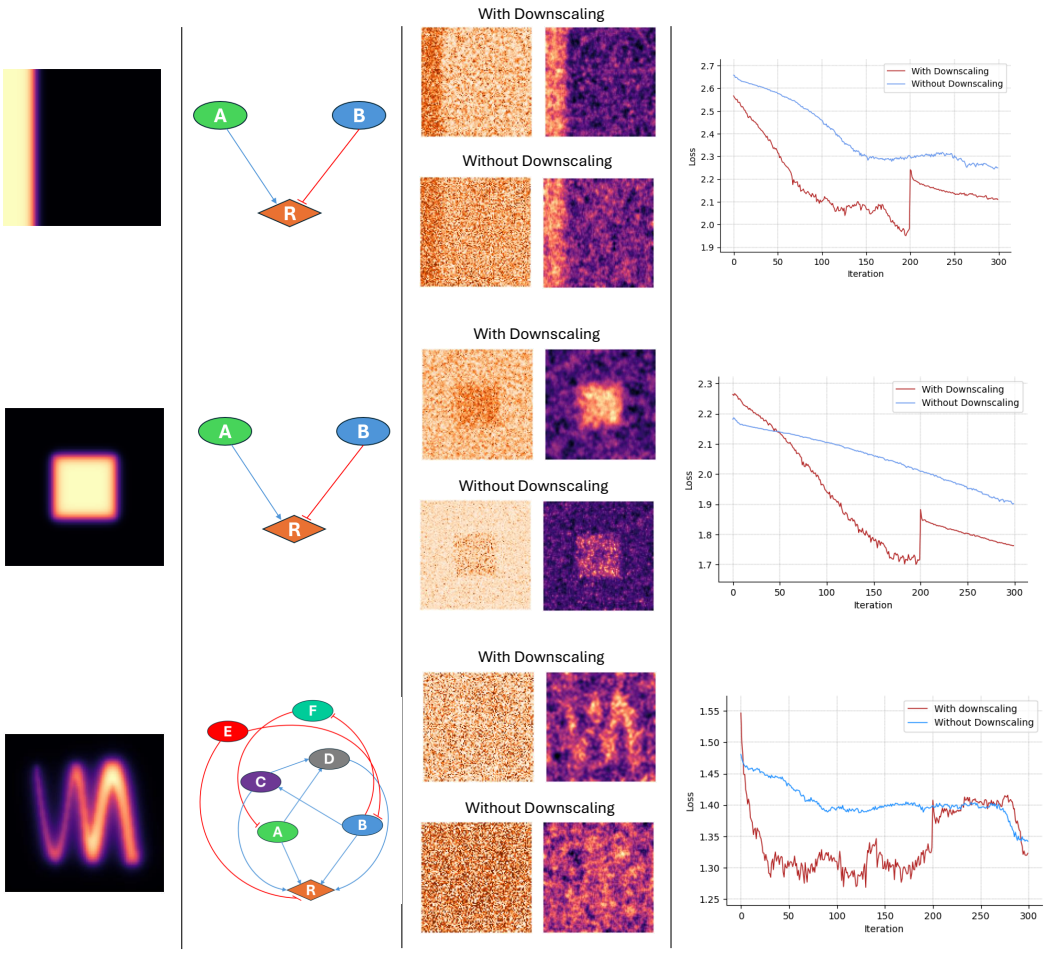


Figure 13: Impact of downscaling on GRN-Designer algorithm performance. The leftmost column displays the target patterns for each model: gradient pattern in the first row, rectangle pattern in the middle, and M-shape pattern in the last row. The second column shows the fixed GRN structure used across both cases (with and without downscaling). The third column provides the initial conditions of the reporter gene (R) on the left, while the right shows the optimized patterns after 300 iterations in each scenario. The final column illustrates the loss reduction over iterations. Results indicate that downscaling leads to faster convergence and lower final costs, resulting in more accurate constructed patterns and optimized initial conditions for R.

0.5.3 Generating the Same Pattern with Different GRN Structures

In this study, we investigated whether the algorithm could construct and optimize gene regulatory networks (GRNs) of varying complexity to generate the same diffusion pattern. The target was a rectangular diffusion pattern, and the algorithm's goal was to construct GRN models where the reporter gene's diffusion pattern closely matched this target.

To achieve GRNs with diverse structures, we used varying hyperparameters to control the model complexity. Table 2 summarizes the key parameters for each model. The simplest GRN consisted of one gene and no interactions, while the most complex contained six genes and nine interactions. For each model, the number of genes was fixed, but the algorithm was free to modify the interactions among the genes to find the optimal configuration.

Table 2: Summary of GRN model monfigurations.
This table provides an overview of the constructed GRN models with varying complexity. The first column indicates the number of genes in each model, which corresponds to the initial gene population size (num_init_genes). The second column lists the total number of interactions allowed in each GRN. The fixed_agent_shape column confirms that the number of genes remained constant during optimization for all models.

# Genes	# Interactions	fixed_agent_shape	num_init_genes
1	0	True	1
2	1	True	2
3	4	True	3
4	5	True	4
5	10	True	5
6	9	True	6

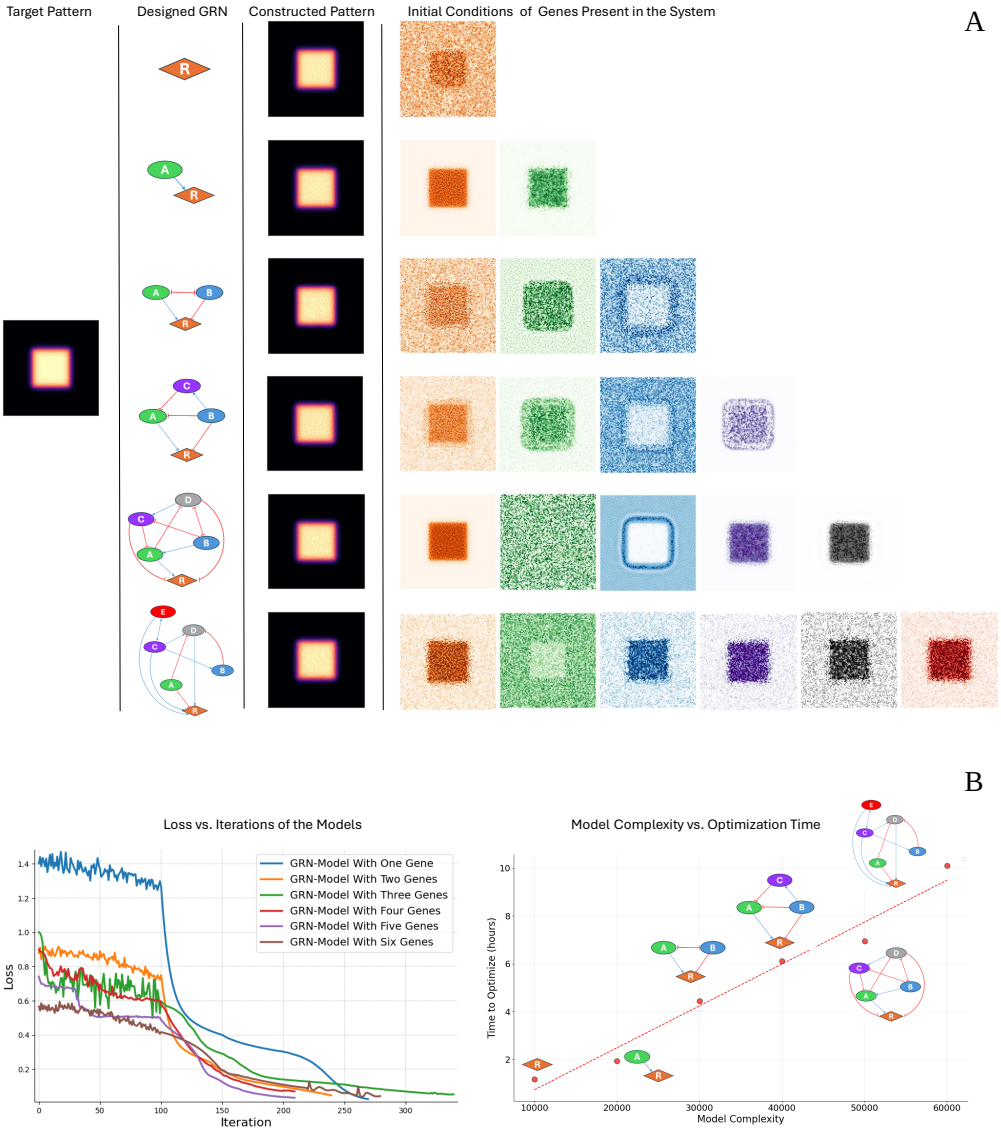


Figure 14: Results of generating the same pattern with different GRN structures. (A) The rectangular target pattern was successfully reproduced by GRNs of varying complexity, ranging from one to six genes. Each GRN structure is displayed alongside its corresponding constructed pattern and the initial gene expression conditions. (B) Left: Loss reduction over optimization iterations. The first 100 iterations exhibit stochastic behavior due to the evolutionary algorithm used for GRN construction. Gradual and smoother loss reduction follows during gradient-based optimization. Right: A linear relationship is observed between model complexity (number of genes and interactions) and optimization time.

0.6 Discussion

Gene Regulatory Networks (GRNs) provide valuable tools for modeling and analyzing biological processes, offering structured representations of gene interactions and their regulatory mechanisms. Over the last two decades, GRN inference has become an active area of research, with the development of advanced algorithms aimed at capturing the complexity of biological systems [48,49]. These algorithms employ diverse methodologies, each with strengths and limitations.

Various computational approaches have been proposed to infer GRNs from gene expression data, including Boolean networks, probabilistic Boolean networks, ordinary differential equations (ODEs), neural networks, Bayesian networks, and dynamic Bayesian networks [50]. Boolean and probabilistic Boolean networks simplify regulatory logic by modeling gene activity as binary states (on/off) but may overlook nuanced interactions. ODE-based models describe gene expression dynamics continuously and can capture rate changes over time, but they rely heavily on accurate parameter estimation, which is often challenging. Neural networks, including deep learning models, excel in modeling complex, non-linear relationships but demand significant computational resources and large datasets. Bayesian and dynamic Bayesian networks effectively incorporate uncertainty and temporal dynamics, but balancing model complexity with interpretability remains a challenge [50].

In this study, we developed a framework for constructing GRNs based on two-dimensional diffusion patterns, optimizing all parameters required to generate target patterns. This work extends the approach proposed by Mousavi and Lobo [9], who demonstrated a system using two diffusible morphogens to guide pattern formation through intermediate genes. Their model constructed GRNs to generate a reporter gene pattern matching a predefined target.

Our GRN-Designer algorithm advances this by incorporating both spatial diffusion and initial condition optimization for each gene and morphogen. Initial expression levels and spatial activation patterns of genes are critical in biological systems, as they establish the foundation for regulatory interactions and spatial organization. By integrating initial condition optimization, our algorithm can construct GRNs tailored to diverse target patterns with adjustable complexity.

The GRN-Designer algorithm combines the strengths of a metaheuristic evolutionary algorithm and a gradient-based optimization algorithm. The evolutionary algorithm is well-suited for exploring large search spaces and identifying near-optimal GRN structures and parameters. However, it alone is insufficient for complete convergence due to the vast number of parameters (e.g., over 30,000 in a simple GRN with three genes). The gradient-based algorithm, while precise in refining parameters, is prone to converging on local optima. By coupling these two approaches, we leverage the exploratory power of the evolutionary algorithm to guide the gradient-based optimization, improving the chances of finding the

global optimum.

We evaluated the GRN-Designer algorithm using nine target patterns of varying complexity. Our models focused on activation and inhibition interactions, each governed by three constants: k_a *or* k_r (binding constant), K (dissociation constant), and n *or* m (Hill coefficient). While this simplified approach enhances computational efficiency, it may not fully capture the intricacies of biological systems, where interactions often involve multi-step regulatory pathways and more complex dynamics. Future iterations could extend this framework to include additional interaction types, enhancing biological fidelity.

The first phase of the algorithm (evolutionary optimization) was limited to 100 iterations due to computational constraints. This restriction may prevent full preliminary convergence, potentially affecting the accuracy of the constructed GRNs. Similarly, we limited the simulation iterations to 100 with a low time step ($\Delta t \approx 0.1$), which influenced the relationship between the initial expression levels (i.e., initial conditions of genes) and their diffusion patterns. With the restricted simulation duration, the diffusion of the gene products were constrained by the initial expression distributions, leading to an overlap between the initial gene expression levels and the observed diffusion patterns.

Additionally, we halted the optimization process when the generated pattern closely resembled the target, even though some models did not fully converge. The loss-versus-iteration plots (see Section 0.8.1) demonstrate this trade-off, where the process became increasingly slow and inefficient. Consequently, the constructed GRNs may not represent the most accurate models.

In several models, optimized initial conditions displayed background noise (e.g., Figure 9). This noise likely arises because the target patterns were generated using the same simulation algorithm, with low levels of gene product diffusion throughout the background. These low concentrations, while not visually apparent in target patterns, influence optimization, particularly in complex GRNs with many trainable parameters. Moreover, incomplete convergence of the models likely contributed to this noise.

In some complex models, the initial conditions of certain genes appeared nearly random, as if they were not optimized (e.g., Figure 6, gene A). This phenomenon is especially evident in complex GRNs with many genes and interactions, where some genes have minimal influence on the reporter gene or overall pattern. These cases warrant further analysis of the interactions and sensitivities of these genes, which can be assessed through interaction constants or sensitivity analysis techniques.

Conversely, in some models, the initial conditions of specific genes complement those of the reporter gene (e.g., Figure 5, gene E). Such patterns are often driven by the nature of regulatory interactions. For instance, in Figure 5, gene E directly inhibits the reporter gene R and indirectly suppresses its activators (A, B, and C). This antagonistic relationship necessitates spatial segregation between gene E and gene R to achieve the desired pattern.

The GRN-Designer algorithm currently operates in a two-dimensional environment, which is suitable for initial studies but does not reflect the full complexity of three-dimensional biological systems. Extending this model to three dimensions [52] would allow for more realistic simulations of biological structures and processes.

Another limitation arises from the use of a simplified variant of the Hill equation to describe gene regulation. In this model, the effects of different regulators on a gene are assumed to be independent and additive. While computationally efficient, this approach is limited in its ability to capture more complex regulatory dynamics. For instance, biological systems often exhibit synergistic interactions, where multiple activators work cooperatively to amplify gene expression, or antagonistic interactions, where repressors and activators compete for the same binding sites [51]. More biologically realistic regulatory models could address these complexities, such as extending the Hill equation to include interaction terms that account for cooperative or competitive binding, using thermodynamic models to simulate transcription factor-DNA interactions.

Additionally, the deterministic nature of the current simulations overlooks the inherent stochasticity of biological systems. Random fluctuations play a significant role in gene expression, especially in cases involving low molecule counts, where stochastic effects can drive phenomena like bistability or oscillations. Incorporating stochastic modeling, such as the Gillespie algorithm or stochastic differential equations, would improve the realism of GRN-Designer simulations by capturing the variability and dynamic behaviors that deterministic models cannot.

Finally, while the GRN-Designer algorithm is efficient, its implementation in Python limits computational speed. Rewriting the algorithm in a lower-level programming language like C++ could significantly enhance performance, enabling simulations of larger and more complex systems.

In summary, this work demonstrates the potential of constructing GRNs based on two-dimensional diffusion patterns with optimized initial conditions. Despite current limitations—such as simplified interaction types, restricted iterations, computational inefficiency, and a two-dimensional framework—the GRN-Designer provides a foundation for further development. Future improvements, including three-dimensional modeling, stochastic simulations, expanded interaction types, and optimized computational implementations, could make the GRN-Designer a powerful tool for addressing complex biological questions.

0.7 Conclusion

This study presents a novel approach for constructing Gene Regulatory Networks (GRNs) to replicate complex spatial patterns in a simulated multicellular environment. Our work builds upon previous efforts in synthetic biology to model

and design GRNs for specific spatial outputs, integrating a new dimension of realism by incorporating initial expression levels (i.e., initial conditions) that reflect spatial distributions of gene activity. This addition is a significant step forward in making GRN models more biologically plausible, as it mirrors the heterogeneous distribution of gene products and morphogens observed in natural systems.

Through the development of the GRN-Designer algorithm, we have demonstrated that a combination of evolutionary and gradient-based optimization techniques can effectively generate GRNs tailored to predefined target patterns. Our results showcase the algorithm's flexibility and robustness, as it successfully constructed GRNs for a diverse set of patterns, ranging from simple gradients to complex shapes, with varying levels of complexity. This hybrid optimization approach allowed the algorithm to navigate the vast parameter space, balancing computational efficiency with biological fidelity.

The current model relies on partial differential equations (PDEs) to simulate gene expression dynamics and diffusion, providing a deterministic framework for GRN design. While this approach has enabled the construction of accurate spatial patterns, biological systems inherently exhibit stochastic behavior, particularly in low-molecule environments where random fluctuations can drive variability in gene expression.

Moreover, while our model successfully constructs GRNs in a two-dimensional framework, real biological systems are inherently three-dimensional, with spatial interactions occurring in more complex tissue structures. Extending the GRN Designer algorithm to support three-dimensional modeling would be a logical next step, allowing for the study of GRNs in configurations that more closely resemble actual multicellular systems. Additionally, further optimizations, such as implementing the algorithm in a faster programming language, could substantially reduce computational costs, making it feasible to model larger and more intricate biological patterns.

In summary, this study advances the field of synthetic developmental biology by offering a flexible and efficient computational tool for designing GRNs based on spatial patterning objectives. By introducing initial conditions as a key design parameter, this work not only enhances our ability to model spatial expression dynamics but also opens new avenues for research in synthetic biology, where the goal is to engineer precise biological patterns for applications in tissue engineering, regenerative medicine, and beyond. Future improvements in the GRN-Designer algorithm, including stochastic simulation, three-dimensional modeling, and opti-

mized performance, will continue to bridge the gap between computational models and biological reality, enabling deeper exploration of gene regulatory processes that underlie development and cellular organization.

Bibliography

[1] Karlebach, G., & Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, *9*(10), 770–780.

[2] Zhao, M., He, W., Tang, J., Zou, Q., & Guo, F. (2021). A comprehensive overview and critical evaluation of gene regulatory network inference technologies. *Briefings in Bioinformatics*, *22*(5), bbab009.

[3] Barbuti, R., Gori, R., Milazzo, P., et al. (2020). A survey of gene regulatory networks modelling methods: from differential equations, to Boolean and qualitative bioinspired models. *Journal of Membrane Computing*, *2*, 207–226.

[4] Chai, L. E., Loh, S. K., Low, S. T., Mohamad, M. S., Deris, S., & Zakaria, Z. (2014). A review on the computational approaches for gene regulatory network construction. *Computers in Biology and Medicine*, *48*, 55-65.

[5] Smith, S. J., Rebeiz, M., & Davidson, L. (2018). From pattern to process: studies at the interface of gene regulatory networks, morphogenesis, and evolution. *Current Opinion in Genetics & Development*, *51*, 103-110.

[6] Levine, M., & Davidson, E. H. (2005). Gene regulatory networks for development. *Proceedings of the National Academy of Sciences*, *102*(14), 4936-4942.

[7] Kicheva, A., & Briscoe, J. (2023). Control of tissue development by morphogens. *Annual Review of Cell and Developmental Biology*, *39*(1), 91-121.

[8] Schlitt, T., & Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, *8*(Suppl 6), S9.

[9] Mousavi, R., & Lobo, D. (2024). Automatic design of gene regulatory mechanisms for spatial pattern formation. *npj Systems Biology and Applications*, *10*, 35.

[10] Shu, H., Zhou, J., Lian, Q., et al. (2021). Modeling gene regulatory networks using neural network architectures. *Nature Computational Science*, *1*, 491–501.

[11] Uzel, S. G. M., Amadi, O. C., Pearl, T. M., Lee, R. T., So, P. T. C., & Kamm, R. D. (2016). Simultaneous or sequential orthogonal gradient formation in a 3D cell culture microfluidic platform. *Small*, *12*(5), 612-622.

[12] Stapornwongkul, K. S., & Vincent, J. P. (2021). Generation of extracellular morphogen gradients: the case for diffusion. *Nature Reviews Genetics*, *22*(6), 393–411.

[13] Pataskar, A., & Tiwari, V. K. (2016). Computational challenges in modeling gene regulatory events. *Transcription*, *7*(5), 188–195.

[14] Jiang, X., & Zhang, X. (2022). RSNET: inferring gene regulatory networks by a redundancy silencing and network enhancement technique. *BMC Bioinformatics*, *23*, 165.

[15] Huynh-Thu, V., & Geurts, P. (2018). dynGENIE3: dynamical GENIE3 for the inference of gene networks from time series expression data. *Scientific Reports*, *8*, 3384.

[16] Zhou, X., Pan, J., Chen, L., Zhang, S., & Chen, Y. (2024). DeepIMAGER: Deeply Analyzing Gene Regulatory Networks from scRNA-seq Data. *Biomolecules*, *14*(7), 766.

[17] Yang, B., Xu, Y., Maxwell, A., Koh, W., Gong, P., & Zhang, C. (2018). MICRAT: a novel algorithm for inferring gene regulatory networks using time series gene expression data. *BMC Systems Biology*, *12*, 19-29.

[18] Tian, X., Patel, Y., & Wang, Y. (2024). TRENDY: Gene Regulatory Network Inference Enhanced by Transformer. *bioRxiv*, 2024-10.

[19] Kim, D., Tran, A., Kim, H. J., Lin, Y., Yang, J. Y. H., & Yang, P. (2023). Gene regulatory network reconstruction: harnessing the power of single-cell multiomic data. *NPJ Systems Biology and Applications*, *9*(1), 51.

[20] Yuan, Q., & Duren, Z. (2023). Continuous lifelong learning for modeling of gene regulation from single cell multiome data by leveraging atlas-scale external data. *bioRxiv*.

[21] Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A., & Murali, T. M. (2020). Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods*, *17*(2), 147-154.

[22] Lee, W. P., Hsiao, Y. T., & Hwang, W. C. (2014). Designing a parallel evolutionary algorithm for inferring gene networks on the cloud computing environment. *BMC Systems Biology*, *8*, 1-19.

[23] Noman, N., Monjo, T., Moscato, P., & Iba, H. (2015). Evolving robust gene regulatory networks. *PLoS ONE*, *10*(1), e0116258.

[24] Chen, B. S., & Lin, Y. P. (2013). A unifying mathematical framework for genetic robustness, environmental robustness, network robustness and their trade-offs on phenotype robustness in biological networks. Part III: synthetic gene networks in synthetic biology. *Evolutionary Bioinformatics*, *9*, EBO-S10686.

[25] Van de Sande, B., Flerin, C., Davie, K., De Waegeneer, M., Hulselmans, G., Aibar, S., ... & Aerts, S. (2020). A scalable SCENIC workflow for single-cell gene regulatory network analysis. *Nature Protocols*, *15*(7), 2247-2276.

[26] Albadr, M. A., Tiun, S., Ayob, M., & Al-Dhief, F. (2020). Genetic algorithm based on natural selection theory for optimization problems. *Symmetry*, *12*(11), 1758.

[27] Halim, A. H., Ismail, I., & Das, S. (2021). Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. *Artificial Intelligence Review*, *54*(3), 2323-2409.

[28] Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[29] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814)*.

[30] Daoud, M. S., Shehab, M., Al-Mimi, H. M., Abualigah, L., Zitar, R. A., & Shambour, M. K. Y. (2023). Gradient-based optimizer (gbo): a review, theory, variants, and applications. *Archives of Computational Methods in Engineering*, *30*(4), 2431-2449.

[31] Friedl, G., & Kuczmann, M. (2014). Population and gradient based optimization techniques, a theoretical overview. *Acta Technica Jaurinensis*, *7*(4), 378-387.

[32] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536.

[33] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, May). On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning* (pp. 1139-1147). PMLR.

[34] Holland, J. (1975). *Adaptation in natural and artificial systems*. Univ. of Michigan Press. Ann Arbor, 7, 390-401.

[35] Razali, N. M., & Geraghty, J. (2011, July). Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the World Congress on Engineering* (Vol. 2, No. 1, pp. 1-6). Hong Kong, China: International Association of Engineers.

[36] Agarwal, P., & Mehta, S. (2014). Nature-inspired algorithms: state-of-art, problems and prospects. *International Journal of Computer Applications*, *100*(14), 14-21.

[37] Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*, 341-359.

[38] Haldurai, L., Madhubala, T., & Rajalakshmi, R. (2016). A study on genetic algorithm and its applications. *International Journal of Computer Science and Engineering*, *4*(10), 139-143.

[39] Metz, J. A. J. (2006). Fitness.

[40] Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2016). Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, *3*(1), 47-57.

[41] Khare, N., Thakur, P. S., Khanna, P., & Ojha, A. (2021, December). Analysis of loss functions for image reconstruction using convolutional autoencoder. In *International Conference on Computer Vision and Image Processing* (pp. 338-349). Cham: Springer International Publishing.

[42] Huang, Y., Song, R., Xu, K., Ye, X., Li, C., & Chen, X. (2020). Deep learning-based inverse scattering with structural similarity loss functions. *IEEE Sensors Journal*, *21*(4), 4900-4907.

[43] Yang, H. H., Yang, C. H. H., & Tsai, Y. C. J. (2020, May). Y-net: Multi-scale feature aggregation network with wavelet structure similarity loss function for single image dehazing. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2628-2632). IEEE.

[44] Köksoy, O., & Yalcinoz, T. (2008). Robust design using Pareto type optimization: a genetic algorithm with arithmetic crossover. *Computers & Industrial Engineering*, *55*(1), 208-218.

[45] Yalcinoz, T., Altun, H., & Uzam, M. (2001, September). Economic dispatch solution using a genetic algorithm based on arithmetic crossover. In *2001 IEEE Porto Power Tech Proceedings (Cat. No. 01EX502)* (Vol. 2, pp. 4-pp). IEEE.

[46] Xu, L., Xie, H., Qin, S. Z. J., Tao, X., & Wang, F. L. (2023). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

[47] Hall, C. A., & Meyer, W. W. (1976). Optimal error bounds for cubic spline interpolation. *Journal of Approximation Theory*, *16*(2), 105–122.

[48] Delgado, F. M., & Gómez-Vela, F. (2019). Computational methods for gene regulatory networks reconstruction and analysis: a review. *Artificial Intelligence in Medicine*, *95*, 133–145.

[49] Marku, M., & Pancaldi, V. (2023). From time-series transcriptomics to gene regulatory networks: A review on inference methods. *PLOS Computational Biology*, *19*(8), e1011254.

[50] Chai, L. E., Loh, S. K., Low, S. T., Mohamad, M. S., Deris, S., & Zakaria, Z. (2014). A review on the computational approaches for gene regulatory network construction. *Computers in Biology and Medicine*, *48*, 55–65.

[51] He, X., Samee, M. A. H., Blatti, C., & Sinha, S. (2010). Thermodynamics-based models of transcriptional regulation by enhancers: the roles of synergistic activation, cooperative binding and short-range repression. *PLoS computational biology*, 6(9), e1000935.

[52] Chen, Y., & Lei, E. (2023, September). Deciphering the contribution of 3D interactions between cis-regulatory elements and promoters to regulate gene expression using graph neural networks. *In Proceedings of the 14th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 1-1).

[53] NumPy Developers. (2024). NumPy: The fundamental package for array computing with Python. Retrieved from https://numpy.org/.

[54] SciPy Developers. (2024). SciPy: Scientific computing tools for Python. Retrieved from https://scipy.org/.

[55] Numba Developers. (2024). Numba: JIT compiler for Python. Retrieved from https://numba.pydata.org/.

[56] PyTorch Developers. (2024). PyTorch: An open-source machine learning framework. Retrieved from https://pytorch.org/.

[57] GitHub. (2024). Algorithm implementation and additional data. Retrieved from https://github.com/LoqmanSamani/grn-designer.

0.8 Appendices

The algorithm is implemented in Python, using common libraries for numerical computations like NumPy [53] and SciPy [54]. The evolutionary optimization part of the system employs the Numba library [55], where just-in-time (JIT) compilation is used to speed up computations. Gradient-based optimization is implemented with the PyTorch deep learning framework [56]. All models were optimized on the BwUniCluster 2.0 server using 40 CPUs. Additional data and the algorithm implementation can be found on the GitHub page [57].

0.8.1 Mathematical Models of Gene Regulatory Networks

In each model, $\rho(x, y)$ represents the initial conditions, and ∇^2 denotes the net diffusion flux. R represents the concentration of the reporter species, while A, B, C, etc., represent the concentrations of other species involved in the gene regulatory network (GRN). For each simulation, a fixed maximum number of iterations (100), a simulation duration of ≈ 10, and a time step $\Delta t \approx 0.1$ were chosen to reduce computational costs. In each model the cost plot is visualized (figure 15-23) which show the progress of the optimization through iterations.

- **Model 1 (Gradient):**

$$\begin{aligned}
\frac{\partial R}{\partial t} &= 0.995 \cdot \rho(x,y) - 0.148 \cdot R - \frac{0.233 \cdot A^{0.205}}{0.987^{0.205} + A^{0.205}} - \frac{0.102 \cdot B^{1.102}}{1.639^{1.102} + B^{1.102}} \\
&\quad + 0.826 \cdot \nabla^2 R \\
\frac{\partial A}{\partial t} &= 0.479 \cdot \rho(x,y) - 0.357 \cdot A + \frac{0.461 \cdot B^{0.647}}{1.066^{0.647} + B^{0.647}} + 0.780 \cdot \nabla^2 A \\
\frac{\partial B}{\partial t} &= 0.015 \cdot \rho(x,y) - 0.997 \cdot B + \frac{0.629 \cdot A^{0.332}}{1.197^{0.332} + A^{0.332}} + 0.996 \cdot \nabla^2 B
\end{aligned}$$

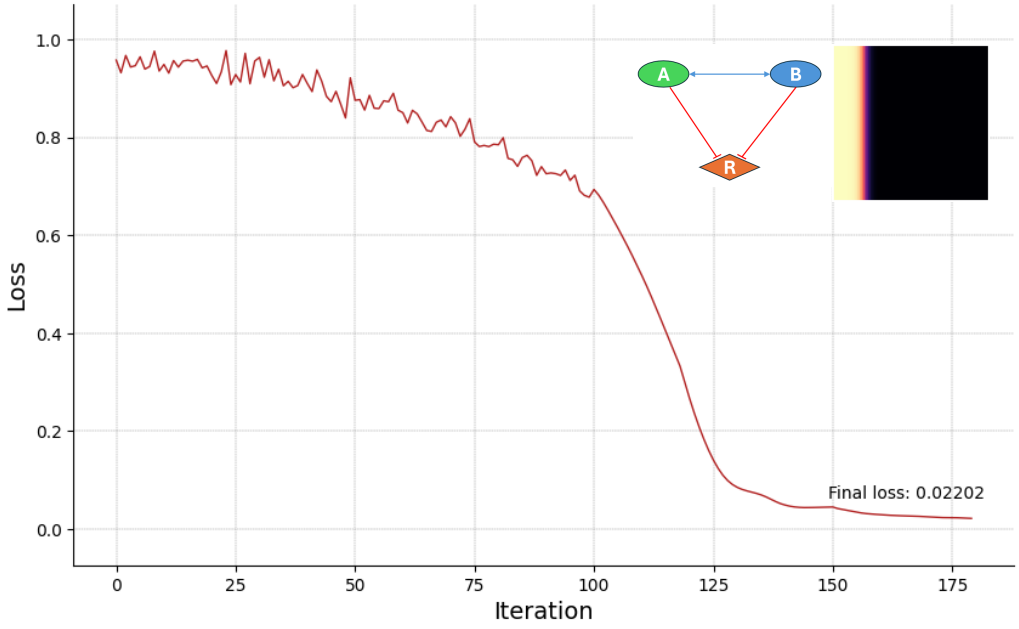


Figure 15: Model 1 (gradient), loss vs. iteration. Total iterations: 180

- **Model 2 (Rectangle):**

$$\frac{\partial R}{\partial t} = 0.998 \cdot \rho(x,y) - 0.229 \cdot R + \frac{0.318 \cdot A^{1.171}}{0.856^{1.171} + A^{1.171}} - \frac{0.492 \cdot B^{0.227}}{0.100^{0.227} + B^{0.227}} - \frac{0.301 \cdot C^{0.700}}{1.168^{0.700} + C^{0.700}} + 0.540 \cdot \nabla^2 R$$

$$\frac{\partial A}{\partial t} = 0.416 \cdot \rho(x,y) - 0.499 \cdot A + \frac{0.546 \cdot B^{0.100}}{0.100^{0.100} + B^{0.100}} - \frac{0.100 \cdot C^{0.283}}{0.415^{0.283} + C^{0.283}} + 0.176 \cdot \nabla^2 A$$

$$\frac{\partial B}{\partial t} = 0.324 \cdot \rho(x,y) - 0.497 \cdot B + \frac{0.501 \cdot C^{0.690}}{0.173^{0.690} + C^{0.690}} + 0.218 \cdot \nabla^2 B$$

$$\frac{\partial C}{\partial t} = 0.552 \cdot \rho(x,y) - 0.706 \cdot C - \frac{0.995 \cdot A^{0.181}}{0.333^{0.181} + A^{0.181}} + 0.067 \cdot \nabla^2 C$$

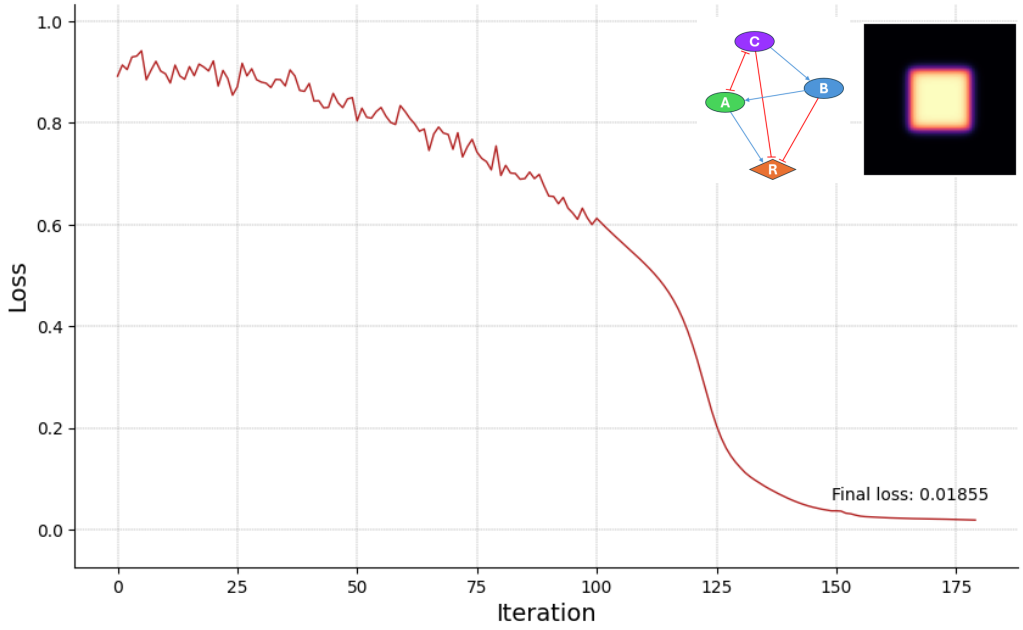


Figure 16: Model 2 (rectangle), loss vs. iteration. Total iterations: 180

- **Model 3 (Circle):**

$$\frac{\partial R}{\partial t} = 0.592 \cdot \rho(x,y) - 0.070 \cdot R + \frac{0.188 \cdot D^{0.856}}{0.766^{0.856} + D^{0.856}} - \frac{0.294 \cdot B^{0.794}}{0.945^{0.794} + B^{0.794}} - \frac{0.759 \cdot C^{0.745}}{0.215^{0.745} + C^{0.745}} + 0.614 \cdot \nabla^2 R$$

$$\frac{\partial A}{\partial t} = 0.412 \cdot \rho(x,y) - 0.418 \cdot A - \frac{0.385 \cdot C^{0.502}}{0.138^{0.502} + C^{0.502}} + 0.114 \cdot \nabla^2 A$$

$$\frac{\partial B}{\partial t} = 0.475 \cdot \rho(x,y) - 0.808 \cdot B + 0.534 \cdot \nabla^2 B$$

$$\frac{\partial C}{\partial t} = 0.432 \cdot \rho(x,y) - 0.348 \cdot C + \frac{0.541 \cdot D^{0.665}}{0.802^{0.665} + D^{0.665}} - \frac{0.609 \cdot A^{0.549}}{0.306^{0.549} + A^{0.549}} - \frac{0.712 \cdot B^{0.863}}{0.127^{0.863} + B^{0.863}} + 0.514 \cdot \nabla^2 C$$

$$\frac{\partial D}{\partial t} = 0.557 \cdot \rho(x,y) - 0.353 \cdot D + \frac{0.539 \cdot A^{0.454}}{0.170^{0.454} + A^{0.454}} + 0.145 \cdot \nabla^2 D$$

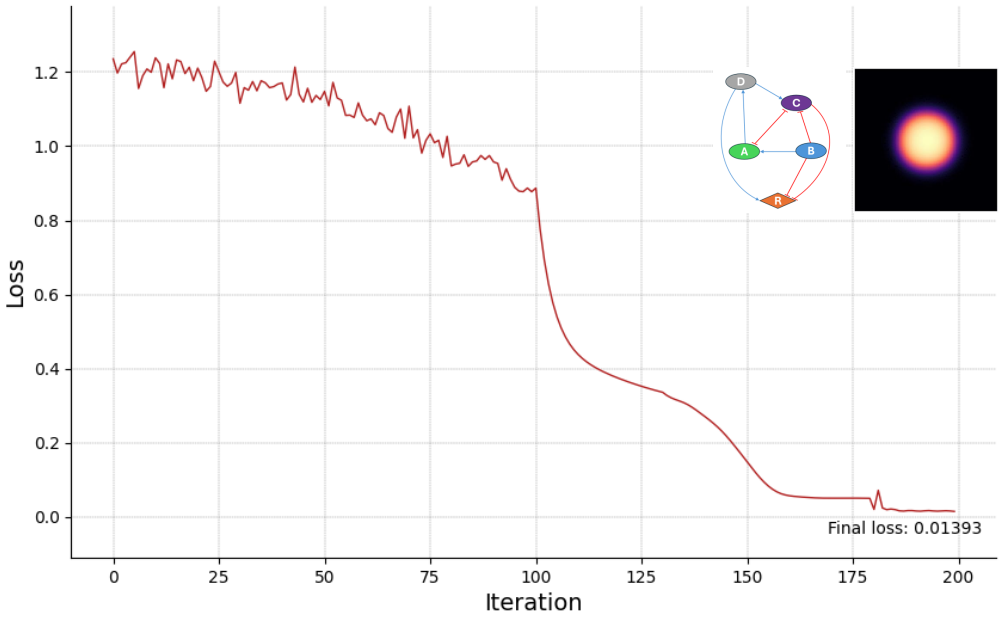


Figure 17: Model 3 (circle), loss vs. iteration. Total iterations: 200

- **Model 4 (Infinity):**

$$\frac{\partial R}{\partial t} = 0.818 \cdot \rho(x,y) - 0.368 \cdot R + \frac{0.745 \cdot A^{0.281}}{0.573^{0.281} + A^{0.281}} + \frac{0.142 \cdot B^{0.830}}{0.759^{0.830} + B^{0.830}} - \frac{0.134 \cdot E^{0.793}}{0.312^{0.793} + E^{0.793}} + 0.976 \cdot \nabla^2 R$$

$$\frac{\partial A}{\partial t} = 0.908 \cdot \rho(x,y) - 0.880 \cdot A + \frac{0.359 \cdot C^{0.466}}{0.530^{0.466} + C^{0.466}} - \frac{0.327 \cdot D^{0.191}}{0.607^{0.191} + D^{0.191}} + 0.488 \nabla^2 A$$

$$\frac{\partial B}{\partial t} = 0.216 \cdot \rho(x,y) - 0.319 \cdot B + \frac{0.649 \cdot D^{0.100}}{0.547^{0.100} + D^{0.100}} - \frac{0.475 \cdot A^{0.507}}{0.123^{0.507} + A^{0.507}} - \frac{0.846 \cdot E^{0.816}}{0.155^{0.816} + E^{0.816}} + 0.041 \cdot \nabla^2 B$$

$$\frac{\partial C}{\partial t} = 0.104 \cdot \rho(x,y) - 0.300 \cdot C + 0.010 \cdot \nabla^2 C$$

$$\frac{\partial D}{\partial t} = 0.909 \cdot \rho(x,y) - 0.273 \cdot D + 0.822 \cdot \nabla^2 D$$

$$\frac{\partial E}{\partial t} = 0.792 \cdot \rho(x,y) - 0.741 \cdot E - \frac{0.278 \cdot A^{0.209}}{1.111^{0.209} + A^{0.209}} - \frac{0.288 \cdot C^{0.722}}{0.925^{0.722} + C^{0.722}} + 0.184 \cdot \nabla^2 E$$

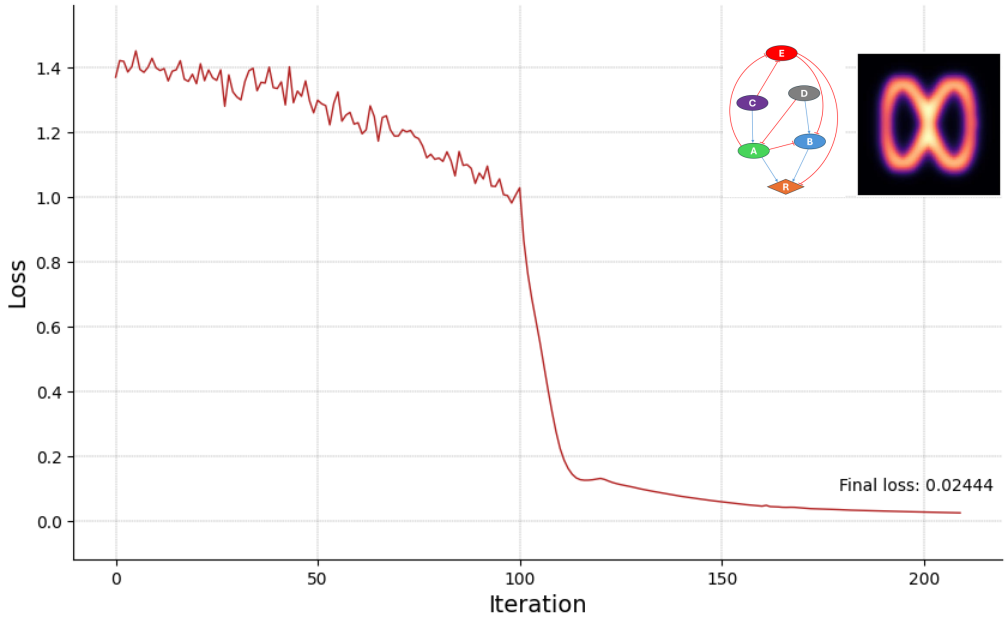


Figure 18: Model 4 (infinity symbol), loss vs. iteration. Total iterations: 210

- **Model 5 (Rings):**

$$\begin{aligned}
\frac{\partial R}{\partial t} &= 0.901 \cdot \rho(x,y) - 0.304 \cdot R + \frac{0.154 \cdot C^{0.283}}{0.647^{0.283} + C^{0.283}} + \frac{0.179 \cdot D^{0.154}}{1.351^{0.154} + D^{0.154}} \\
&\quad + \frac{0.232 \cdot E^{0.202}}{1.216^{0.202} + E^{0.202}} - \frac{0.753 \cdot A^{0.480}}{0.296^{0.480} + A^{0.480}} - \frac{0.830 \cdot B^{0.100}}{0.101^{0.100} + B^{0.100}} \\
&\quad - \frac{0.815 \cdot F^{0.173}}{0.158^{0.173} + F^{0.173}} + 0.885 \cdot \nabla^2 R \\
\frac{\partial A}{\partial t} &= 0.160 \cdot \rho(x,y) - 0.950 \cdot A + \frac{0.186 \cdot B^{0.642}}{0.824^{0.642} + B^{0.642}} - \frac{0.232 \cdot E^{0.202}}{1.216^{0.202} + E^{0.202}} \\
&\quad + 0.921 \cdot \nabla^2 A \\
\frac{\partial B}{\partial t} &= 0.999 \cdot \rho(x,y) - 0.136 \cdot B + \frac{0.174 \cdot A^{0.517}}{0.685^{0.517} + A^{0.517}} - \frac{0.100 \cdot C^{0.432}}{0.425^{0.432} + C^{0.432}} \\
&\quad - \frac{0.100 \cdot D^{0.100}}{0.638^{0.100} + D^{0.100}} + 0.010 \cdot \nabla^2 B \\
\frac{\partial C}{\partial t} &= 0.378 \cdot \rho(x,y) - 0.087 \cdot C - \frac{0.773 \cdot F^{0.976}}{0.796^{0.976} + F^{0.976}} + 0.689 \cdot \nabla^2 C \\
\frac{\partial D}{\partial t} &= 0.450 \cdot \rho(x,y) - 0.010 \cdot D + \frac{0.702 \cdot C^{0.424}}{0.143^{0.424} + C^{0.424}} + 0.010 \cdot \nabla^2 D \\
\frac{\partial E}{\partial t} &= 0.255 \cdot \rho(x,y) - 0.010 \cdot E + \frac{0.242 \cdot C^{0.327}}{0.100^{0.327} + C^{0.327}} + 0.698 \cdot \nabla^2 E \\
\frac{\partial F}{\partial t} &= 0.886 \cdot \rho(x,y) - 0.358 \cdot F + \frac{0.945 \cdot A^{0.825}}{0.193^{0.825} + A^{0.825}} - \frac{1.0507 \cdot E^{0.375}}{0.439^{0.375} + E^{0.375}} \\
&\quad + 0.999 \cdot \nabla^2 F
\end{aligned}$$

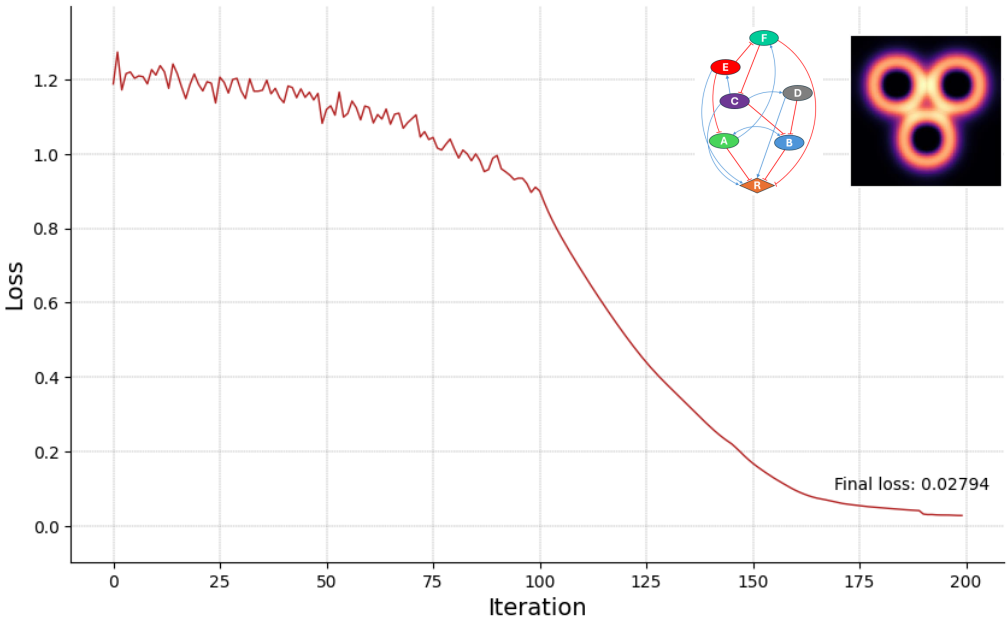


Figure 19: Model 5 (rings symbol), loss vs. iteration. Total iterations: 200

- **Model 6 (M-Shape):**

$$\frac{\partial R}{\partial t} = 0.896 \cdot \rho(x,y) - 0.523 \cdot R + \frac{0.293 \cdot B^{0.209}}{0.330^{0.209} + B^{0.209}} + \frac{1.268 \cdot G^{1.193}}{0.278^{1.193} + G^{1.193}} - \frac{0.738 \cdot D^{0.931}}{0.145^{0.931} + D^{0.931}} - \frac{0.582 \cdot F^{0.718}}{0.665^{0.718} + F^{0.718}} + 0.677 \cdot \nabla^2 R$$

$$\frac{\partial A}{\partial t} = 0.304 \cdot \rho(x,y) - 0.592 \cdot A + \frac{0.209 \cdot D^{0.400}}{0.100^{0.400} + D^{0.400}} - \frac{0.493 \cdot C^{0.744}}{0.899^{0.744} + C^{0.744}} - \frac{0.932 \cdot E^{0.931}}{0.131^{0.931} + E^{0.931}} + 0.387 \cdot A$$

$$\frac{\partial B}{\partial t} = 0.227 \cdot \rho(x,y) - 0.657 \cdot B + \frac{0.115 \cdot C^{0.520}}{0.299^{0.520} + C^{520}} + \frac{0.825 \cdot G^{0.660}}{0.791^{0.660} + G^{0.660}} + 0.015 \cdot \nabla^2 B$$

$$\frac{\partial C}{\partial t} = 0.443 \cdot \rho(x,y) - 0.687 \cdot C - \frac{0.546 \cdot F^{0.606}}{0.103^{0.606} + F^{0.606}} + 0.739 \cdot \nabla^2 C$$

$$\frac{\partial D}{\partial t} = 0.750 \cdot \rho(x,y) - 0.334 \cdot D + 0.291 \cdot \nabla^2 D$$

$$\frac{\partial E}{\partial t} = 0.504 \cdot \rho(x,y) - 0.999 \cdot E + 0.284 \cdot \nabla^2 E$$

$$\frac{\partial F}{\partial t} = 0.390 \cdot \rho(x,y) - 0.999 \cdot F + \frac{0.108 \cdot D^{0.723}}{0.217^{0.723} + D^{0.723}} - \frac{0.859 \cdot E^{0.287}}{0.442^{0.287} + E^{0.287}} + 0.214 \cdot \nabla^2 F$$

$$\frac{\partial G}{\partial t} = 0.390 \cdot \rho(x,y) - 0.948 \cdot G - \frac{0.754 \cdot A^{0.100}}{0.161^{0.100} + A^{0.100}} + 0.556 \cdot \nabla^2 G$$

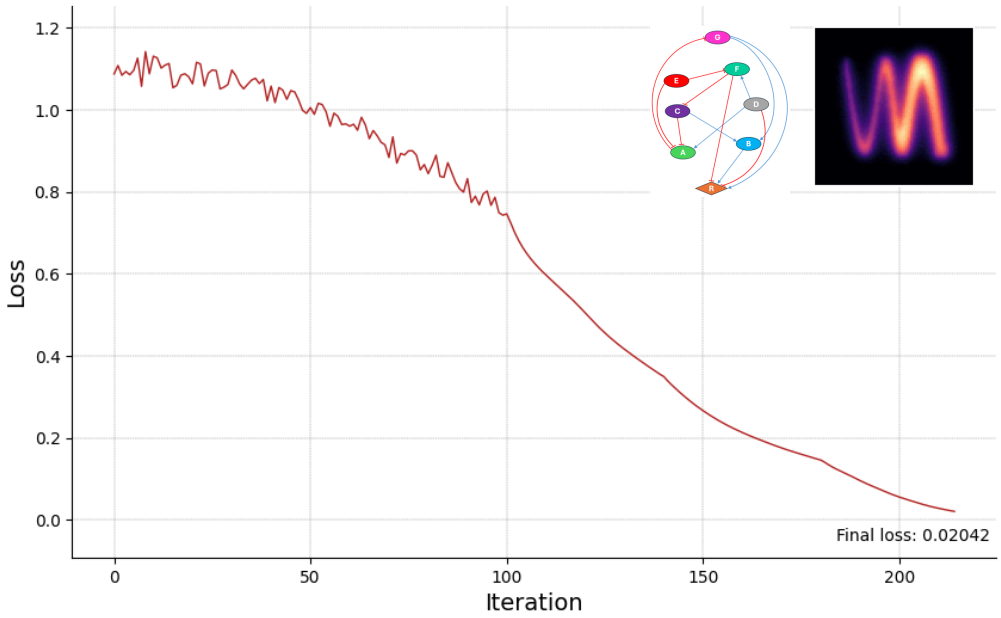


Figure 20: Model 6 (M-Shape), loss vs. iteration. Total iterations: 215.

- **Model 7 (Chromosome):**

$$\begin{aligned}
\frac{\partial R}{\partial t} &= 0.853 \cdot \rho(x,y) - 0.487 \cdot R + \frac{0.100 \cdot D^{0.498}}{0.278^{0.498} + D^{0.498}} + \frac{0.482 \cdot F^{0.100}}{0.860^{0.100} + F^{0.100}} \\
&\quad + \frac{0.464 \cdot G^{0.914}}{0.418^{0.914} + G^{0.914}} - \frac{0.151 \cdot A^{0.865}}{0.756^{0.865} + A^{0.865}} - \frac{0.832 \cdot B^{0.100}}{0.679^{0.100} + B^{0.100}} \\
&\quad - \frac{0.142 \cdot C^{0.115}}{1.132^{0.115} + C^{0.115}} + 0.999 \cdot \nabla^2 R \\
\frac{\partial A}{\partial t} &= 0.680 \cdot \rho(x,y) - 0.999 \cdot A + \frac{0.277 \cdot E^{0.787}}{0.246^{0.787} + E^{0.787}} + \frac{0.164 \cdot H^{0.373}}{1.026^{0.373} + H^{0.373}} \\
&\quad - \frac{0.283 \cdot C^{0.347}}{0.997^{0.347} + C^{0.347}} + 0.100 \cdot \nabla^2 A \\
\frac{\partial B}{\partial t} &= 0.675 \cdot \rho(x,y) - 0.996 \cdot B + \frac{0.100 \cdot H^{0.535}}{0.346^{0.535} + H^{0.535}} + 0.236 \cdot \nabla^2 B \\
\frac{\partial C}{\partial t} &= 0.878 \cdot \rho(x,y) - 0.764 \cdot C + \frac{0.806 \cdot E^{0.820}}{0.483^{0.820} + E^{0.820}} + 0.185 \cdot \nabla^2 C \\
\frac{\partial D}{\partial t} &= 0.570 \cdot \rho(x,y) - 0.938 \cdot D - \frac{0.746 \cdot E^{0.486}}{0.100^{0.486} + E^{0.486}} + 0.535 \cdot \nabla^2 D \\
\frac{\partial E}{\partial t} &= 0.914 \cdot \rho(x,y) - 0.968 \cdot E + 0.023 \cdot \nabla^2 E \\
\frac{\partial F}{\partial t} &= 0.449 \cdot \rho(x,y) - 0.589 \cdot F - \frac{0.447 \cdot A^{0.589}}{0.010^{0.589} + A^{0.589}} + 0.010 \cdot \nabla^2 F \\
\frac{\partial G}{\partial t} &= 0.708 \cdot \rho(x,y) - 0.999 \cdot G - \frac{0.131 \cdot E^{0.272}}{0.121^{0.272} + E^{0.272}} + 0.604 \cdot \nabla^2 G \\
\frac{\partial H}{\partial t} &= 0.816 \cdot \rho(x,y) - 0.896 \cdot H + \frac{0.100 \cdot B^{0.100}}{0.100^{0.100} + B^{0.100}} - \frac{1.066 \cdot G^{0.517}}{0.197^{0.517} + G^{0.517}} \\
&\quad + 0.110 \cdot \nabla^2 H
\end{aligned}$$

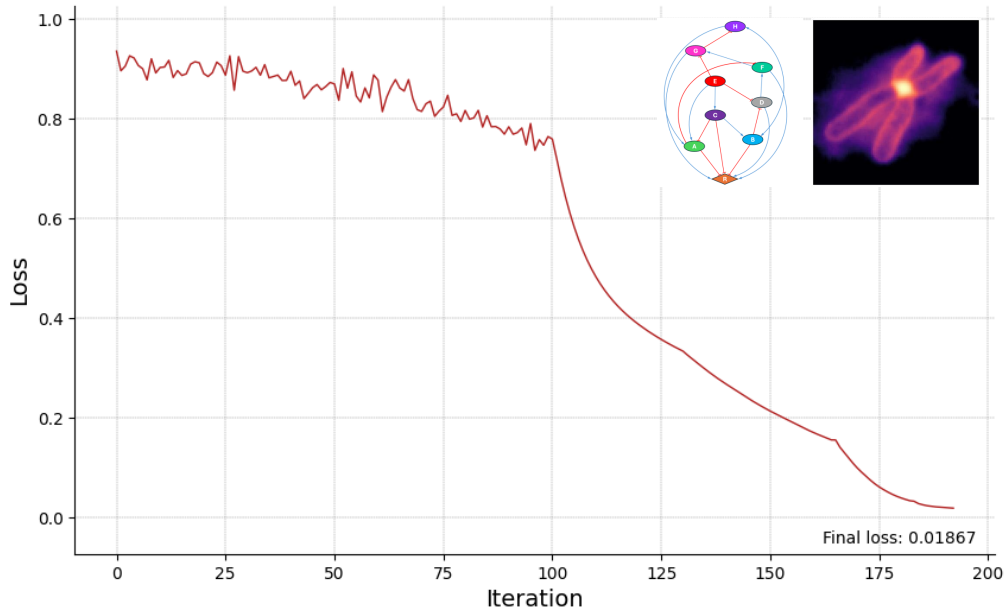


Figure 21: Model 7 (chromosome symbol), loss vs. iteration. Total iterations: 193.

- **Model 8 (DNA-Strand):**

$$\frac{\partial R}{\partial t} = 0.999 \cdot \rho(x,y) - 0.567 \cdot R + \frac{0.504 \cdot A^{0.470}}{0.443^{0.470} + A^{0.470}} + \frac{0.408 \cdot B^{0.935}}{0.834^{0.935} + B^{0.935}} + \frac{0.214 \cdot H^{0.100}}{0.536^{0.100} + H^{0.100}} - \frac{0.100 \cdot I^{1.677}}{1.209^{1.677} + I^{1.677}} - \frac{0.147 \cdot G^{0.672}}{1.293^{0.672} + G^{0.672}} + 0.999 \cdot \nabla^2 R$$

$$\frac{\partial A}{\partial t} = 0.132 \cdot \rho(x,y) - 0.786 \cdot A + \frac{0.743 \cdot D^{0.100}}{0.276^{0.100} + D^{0.100}} + \frac{0.100 \cdot F^{0.755}}{0.995^{0.755} + F^{0.755}} - \frac{0.400 \cdot C^{0.442}}{0.422^{0.442} + C^{0.442}} - \frac{0.600 \cdot E^{0.606}}{0.610^{0.606} + E^{0.606}} - \frac{0.100 \cdot I^{1.677}}{1.209^{1.677} + I^{1.677}} + 0.010 \cdot \nabla^2 A$$

$$\frac{\partial B}{\partial t} = 0.342 \cdot \rho(x,y) - 0.888 \cdot B - \frac{0.349 \cdot C^{0.213}}{0.100^{0.213} + C^{0.127}} - \frac{0.220 \cdot I^{0.531}}{0.425^{0.531} + I^{0.531}} + 0.449 \cdot \nabla^2 B$$

$$\frac{\partial C}{\partial t} = 0.999 \cdot \rho(x,y) - 0.012 \cdot C - \frac{0.642 \cdot F^{0.636}}{1.069^{0.636} + F^{0.636}} - \frac{0.435 \cdot H^{0.499}}{0.983^{0.499} + H^{0.499}} + 0.220 \cdot \nabla^2 C$$

$$\frac{\partial D}{\partial t} = 0.309 \cdot \rho(x,y) - 0.156 \cdot D + \frac{0.845 \cdot F^{0.735}}{0.579^{0.735} + F^{0.735}} + 0.682 \cdot \nabla^2 D$$

$$\frac{\partial E}{\partial t} = 0.591 \cdot \rho(x,y) - 0.268 \cdot E + \frac{0.432 \cdot C^{0.853}}{0.324^{0.853} + C^{0.853}} + \frac{0.902 \cdot G^{0.182}}{0.100^{0.182} + G^{0.182}} + 0.157 \cdot \nabla^2 E$$

$$\frac{\partial F}{\partial t} = 0.531 \cdot \rho(x,y) - 0.101 \cdot F - \frac{0.608 \cdot G^{0.934}}{0.100^{0.934} + G^{0.934}} + 0.310 \cdot \nabla^2 F$$

$$\frac{\partial G}{\partial t} = 0.203 \cdot \rho(x,y) - 0.406 \cdot G + \frac{0.821 \cdot I^{0.338}}{0.100^{0.338} + I^{0.338}} + 0.397 \cdot \nabla^2 G$$

$$\frac{\partial H}{\partial t} = 0.071 \cdot \rho(x,y) - 0.733 \cdot H - \frac{1.074 \cdot I^{0.153}}{0.237^{0.153} + I^{0.153}} + 0.398 \cdot \nabla^2 H$$

$$\frac{\partial I}{\partial t} = 0.045 \cdot \rho(x,y) - 0.804 \cdot I - \frac{0.456 \cdot B^{0.128}}{0.741^{0.128} + B^{0.128}} + 0.535 \cdot \nabla^2 I$$

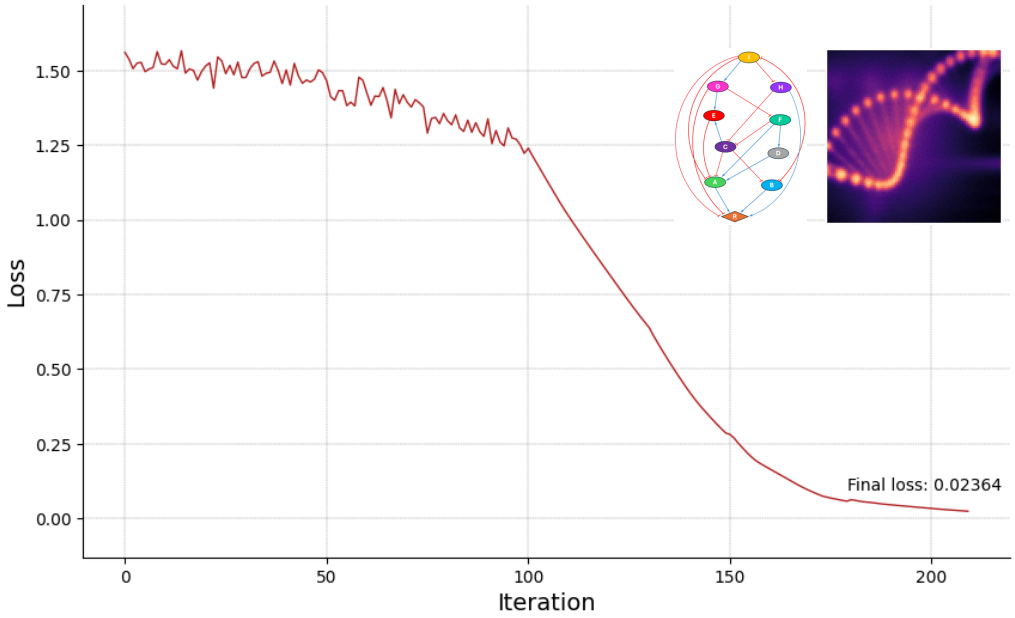


Figure 22: Model 8 (DNA-Strand), loss vs. iteration. Total iterations: 210

- **model 9 (Biohazard):**

$$\frac{\partial R}{\partial t} = 0.883 \cdot \rho(x,y) - 0.418 \cdot R + \frac{0.100 \cdot F^{0.322}}{0.848^{0.322} + F^{0.322}} + \frac{0.746 \cdot G^{0.100}}{0.505^{0.100} + G^{0.100}} - \frac{0.125 \cdot A^{0.215}}{0.538^{0.215} + A^{0.215}} - \frac{0.635 \cdot B^{0.621}}{0.496^{0.621} + B^{0.621}} - \frac{0.100 \cdot C^{0.952}}{0.528^{0.952} + C^{0.952}} - \frac{0.100 \cdot D^{0.794}}{0.175^{0.794} + D^{0.794}} - \frac{0.596 \cdot K^{0.352}}{0.676^{0.352} + K^{0.352}} + 0.999 \cdot \nabla^2 R$$

$$\frac{\partial A}{\partial t} = 0.927 \cdot \rho(x,y) - 0.797 \cdot A + \frac{0.100 \cdot C^{0.545}}{0.466^{0.545} + C^{0.545}} + \frac{0.357 \cdot K^{0.179}}{0.852^{0.179} + K^{0.179}} + 0.010 \cdot \nabla^2 A$$

$$\frac{\partial B}{\partial t} = 0.175 \cdot \rho(x,y) - 0.072 \cdot B + \frac{0.271 \cdot A^{0.681}}{0.330^{0.681} + A^{0.681}} + \frac{0.449 \cdot C^{0.539}}{0.351^{0.539} + C^{0.539}} - \frac{0.100 \cdot F^{1.055}}{1.439^{1.055} + F^{1.055}} + 0.338 \cdot \nabla^2 B$$

$$\frac{\partial C}{\partial t} = 0.251 \cdot \rho(x,y) - 0.645 \cdot C - \frac{0.257 \cdot E^{0.333}}{0.835^{0.333} + E^{0.333}} - \frac{0.692 \cdot F^{0.587}}{0.237^{0.587} + F^{0.587}} - \frac{0.327 \cdot G^{0.100}}{0.316^{0.100} + G^{0.100}} + 0.359 \cdot \nabla^2 C$$

$$\frac{\partial D}{\partial t} = 0.744 \cdot \rho(x,y) - 0.323 \cdot D + \frac{0.537 \cdot A^{0.432}}{0.474^{0.432} + A^{0.432}} - \frac{0.353 \cdot J^{0.100}}{0.279^{0.100} + J^{0.100}} + 0.320 \cdot \nabla^2 D$$

$$\frac{\partial E}{\partial t} = 0.279 \cdot \rho(x,y) - 0.420 \cdot E + 0.010 \cdot \nabla^2 E$$

$$\frac{\partial F}{\partial t} = 0.532 \cdot \rho(x,y) - 0.163 \cdot F + \frac{0.799 \cdot G^{0.407}}{0.100^{0.407} + G^{0.407}} - \frac{0.963 \cdot H^{0.849}}{0.348^{0.849} + H^{0.849}} + 0.511 \cdot \nabla^2 F$$

$$\frac{\partial G}{\partial t} = 0.858 \cdot \rho(x,y) - 0.010 \cdot G - \frac{0.387 \cdot I^{0.273}}{0.945^{0.273} + I^{0.273}} + 0.010 \cdot \nabla^2 G$$

$$\frac{\partial H}{\partial t} = 0.403 \cdot \rho(x,y) - 0.556 \cdot H + \frac{0.152 \cdot I^{0.894}}{0.563^{0.894} + I^{0.894}} + 0.935 \cdot \nabla^2 H$$

$$\frac{\partial I}{\partial t} = 0.478 \cdot \rho(x,y) - 0.474 \cdot I + \frac{0.749 \cdot K^{0.651}}{0.532^{0.651} + K^{0.651}} + 0.462 \cdot \nabla^2 I$$

$$\frac{\partial J}{\partial t} = 0.094 \cdot \rho(x,y) - 0.758 \cdot J + \frac{0.656 \cdot F^{0.118}}{0.717^{0.118} + F^{0.118}} + \frac{0.592 \cdot I^{0.100}}{0.649^{0.100} + I^{0.100}} - \frac{0.230 \cdot K^{0.590}}{0.536^{0.590} + K^{0.590}} + 0.890 \cdot \nabla^2 J$$

$$\frac{\partial K}{\partial t} = 0.511 \cdot \rho(x,y) - 0.621 \cdot K + 0.747 \cdot \nabla^2 K$$

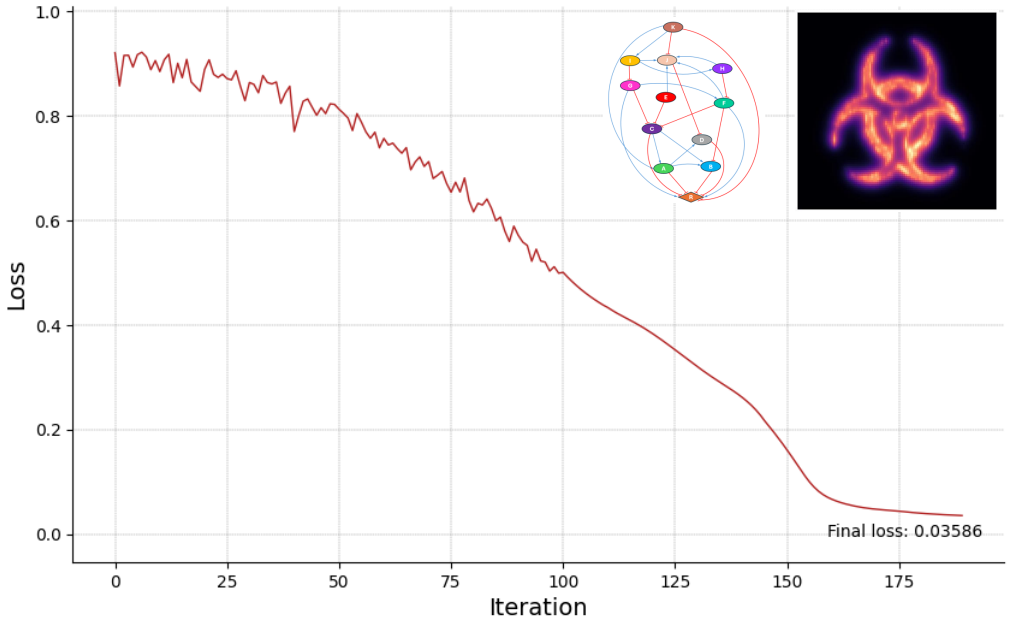


Figure 23: Model 9 (biohazard symbol), loss vs. iteration. Total iterations: 190

0.8.2 Output Data Format and Key Structures

The results generated by each optimization model are stored in an HDF5 ('.h5') file. This file contains the following key data structures:

- `agent`: A three-dimensional matrix containing all the information necessary to construct the GRN graph. For more details about the structure of the agent, refer to Figure 3.
- `run_time`: A one-dimensional array with a single value representing the total optimization time (i.e., the duration required to optimize the model).
- `simulation_results`: A three-dimensional matrix (m, y, x) where:
  - m is the number of saved intermediate results,
  - y and x represent the compartment's shape where the simulation occurs.

  This matrix stores intermediate results of the reporter gene simulation, which can be used to generate a video illustrating the optimization process over iterations.
- `initial_conditions`: A four-dimensional matrix (n, m, y, x) where:

  - n is the number of genes in the GRN, equivalent to the number of initial conditions,
  - m, y, and x have the same meaning as in 'simulation_results'.

  Like 'simulation_results', this matrix can also be used to create a video visualization.

- `target`: A matrix representing the target spatial pattern used as the basis for optimization.
- `cost`: An array containing the cost values from each iteration, which can be visualized to track the progress of the optimization process.

All output data, along with an optional video visualizing the optimization process across iterations, is stored on the GitHub homepage of the GRN-Designer algorithm [57].

0.8.3 Hyperparameters of GRN-Designer

Table 3 summarizes the primary hyperparameters of the GRN-Designer algorithm, along with their descriptions and default values. For additional details on the `zoom` and `zoom_factor` parameters, refer to the documentation of the `scipy.ndimage.zoom` function.

Table 3: Main hyperparameters of the GRN-Designer algorithm.
The third column lists the default values for each hyperparameter. For detailed information on the `zoom` and `zoom_factor` parameters, refer to the documentation of the `scipy.ndimage.zoom` function.

Hyperparameter	Description	Value
`population_size`	Number of agents in the population pool	50
`evolution_one_epochs`	Maximum number of epochs for the first stage of evolutionary optimization	50
`evolution_two_epochs`	Maximum number of epochs for the second stage of evolutionary optimization	0
`optimization_epochs`	Maximum number of epochs for Adam optimization	0

Continued on next page

Hyperparameter	Description	Value
learning_rate	Learning rate for Adam optimization	0.01
sim_mutation_rate	Mutation rate for simulation parameters (e.g., time step and duration)	0.1
initial_condition_mutation_rate	Mutation rate for initial conditions	0.1
parameter_mutation_rate	Mutation rate for gene parameters	0.1
species_insertion_mutation_rate	Mutation rate for adding new genes	0.04
connection_insertion_mutation_rate	Mutation rate for adding new gene connections	0.06
connection_deletion_mutation_rate	Mutation rate for deleting gene connections	0.06
crossover_alpha	Control parameter for blending in crossover operations	0.4
gradient_optimization	Whether Adam optimization should be applied	False
parameter_optimization	Whether gene parameters should be optimized using Adam	False
condition_optimization	Whether initial conditions should be optimized using Adam	False
sim_mutation	Whether simulation parameters should be mutated during the GA phase	True
initial_condition_mutation	Whether initial conditions should be mutated during the GA phase	True
parameter_mutation	Whether gene parameters should be mutated during the GA phase	False
species_insertion_mutation_one	Whether new genes should be added during the first stage of GA	False

Continued on next page

Hyperparameter	Description	Value
connection_insertion_mutation_one	Whether new connections should be added during the first stage of GA	False
connection_deletion_mutation_one	Whether existing connections should be removed during the first stage of GA	False
species_insertion_mutation_two	Whether new genes should be added during the second stage of GA	False
connection_insertion_mutation_two	Whether new connections should be added during the second stage of GA	False
connection_deletion_mutation_two	Whether existing connections should be removed during the second stage of GA	False
initial_condition_crossover	Whether crossover operations should include initial conditions	True
parameter_crossover	Whether crossover operations should include gene parameters	False
simulation_crossover	Whether crossover operations should include simulation parameters	True
cost_alpha	Weight for the primary objective in the fitness function	0.6
cost_beta	Weight for the secondary objective in the fitness function	0.4
zoom_	Whether downscaling should be applied	False
zoom_in_factor	Factor for downscaling operations	0.5
zoom_out_factor	Factor for upscaling operations	2
num_elite_agents	Number of elite agents preserved for crossover	5

Continued on next page

Hyperparameter	**Description**	**Value**
`simulation_min`	Lower bounds for simulation parameters	(5, 0.05)
`simulation_max`	Upper bounds for simulation parameters	(40, 0.3)
`initial_condition_min`	Lower bound for initial conditions	0.0
`initial_condition_max`	Upper bound for initial conditions	2.0
`parameter_min`	Lower bound for gene parameters	(0.010, 0.010)
`parameter_max`	Upper bound for gene parameters	(0.999, 2)
`gradient_optimizer`	Gradient-based optimization algorithm (e.g., Adam or SGD)	Adam
`num_init_genes`	Number of genes to initialize the population	1

Declaration of Authorship

I hereby declare that I, *Loghman Samani (Matrikelnummer: 3585810)*, am the sole author of this thesis. I have completed it independently, using only the resources and tools specified. All passages quoted directly or indirectly from external sources are clearly identified and properly cited.

I confirm that this thesis has not been submitted in the same or similar form to any other examination authority. Furthermore, I certify that the electronic version of this thesis is identical to the printed copies.

L.Samani

Stuttgart, 04/01/2025