A Novel Interactive Slime Mould Algorithm-Based Platform for Generative Art and Optimization

Wen Wen[1], Yuyao Zhang[2], Jiaxuan Xu[3], Shan Wang[4], and Jianghuai Shao[5](✉)

[1] China Architecture Design and Research Group, Beijing, China
[2] Beihang University, Beijing, China
yao777@buaa.edu.cn
[3] Royal College of Art, London SW7 2EU, UK
jiaxuanxu@network.rca.ac.uk
[4] Shanghai Jiao Tong University, Shanghai 200240, China
wang_shan@sjtu.edu.cn
[5] Yonsei University, Seoul, Korea
sjh18283368168@yonsei.ac.kr

Abstract. This study offers a new platform using the Slime Mould Algorithm (SMA) to let people generate complex visual designs via interactive simulations. Designed with WebGL and Three.js, the platform replicates multi-agent chemotaxis motivated by the dynamic adaptability of slime mould *Physarum polycephalum*. Two fundamental processes underlie the platform: a tailored text generating weight system and a tailored image fitting weight system. These mechanisms let users adjust pixel weightings, therefore guiding particles to create emergent, complex designs. Providing both accessibility for non-technical users and great degrees of creative flexibility, the platform combines computer graphics, visual arts, and computational physics. The system lets the development of original, changing artworks by integrating real-time input with dynamic rendering.

Keywords: Slime Mold Algorithm · Biomimetic Computing · Interactive Simulation · Generative Art · Customization Platform

1 Introduction

Natural organisms such as bio-inspired algorithms have drawn increasing attention since they may tackle challenging optimization and design issues. From the behavior of *Physarum polycephalum*, the slime mould algorithm (SMA) has shown to be quite successful among all the methods in simulating biological activities including network optimization, pathfinding, and pattern recognition.

Various uses have seen great exploration of slime mould's capacity to create effective transport networks and react to environmental stimuli. Adamatzky [28] developed the idea of Physarum machines, whereby the foraging behavior of

M. Rauterberg (Ed.): HCII 2025, LNCS 15800, pp. 204–215, 2025.
https://doi.org/10.1007/978-3-031-93160-4_13

the organism was represented to solve computer problems including the shortest path. Jones [29] expanded this study by investigating the emergent computing characteristics of slime mould, in which basic local rules produce intricate, global network forms free from centralized control. The Slime Mould Algorithm is built on this emergent behavior idea.

Further investigation on Physarum-inspired computation has shown how slime mould can solve difficult spatial problems by reacting to local environmental conditions such chemoattractant gradients. Slime mould naturally seeks food sources (analogous to attractants) and avoids harmful substances (repellents), which makes its behavior particularly suitable for optimization problems.

Inspired by the above ideas, this study combines SMA into a dynamic simulation platform in an art design tool to guide particles to reach customized text and image input to form complex graphical patterns in real time by allowing users to interact with a particle system that simulates slime mold behavior. The platform naturally merges scientific research with creative design by aggregating relevant fields such computational physics, computer graphics and visual arts, therefore offering a fresh approach to generative art.

2 Literature Review

Several fundamental theories and ideas from computational biology, unusual computing, and creative design define the evolution of the platform.

2.1 Slime Mould Algorithm (SMA)

The method of slime mould algorithm (SMA) is based on the adaptive behavior of *Physarum polycephalum*, which possesses dynamic response to external variables. SMA can also employ a mathematical model based on adaptive weights to be tuned in the process of feedback observed in the slime moulds [1]. Therefore, when the algorithm explores and exploits the search space by simulating the way slime molds form paths in nature, its efficiency is improved [1–3] (see also [?]), contributing to dealing with static optimization problems such as path-finding and network optimization.

In addition, recent improvements to the SMA incorporate chaotic opposition-based learning and hybrid strategies. These improvements enhance the algorithm's efficiency, expand the scope of problems that SMA can solve to include multi-modal problems, and reduce the likelihood of getting trapped in local optima [4,5].

2.2 The Features of SMA

Chemoattractants and Chemotaxis. Chemotaxis means cells move in a certain direction when there are chemical attractants. These chemicals create gradients. The gradients control the direction of the cells' movement. This kind of movement is important for studying how cells behave and the patterns they form.

The Keller-Segel system was one of the early models to explain the aggregation of slime molds through chemotaxis [10]. In subsequent research, Chalub further explained that the aggregation of slime molds is the molecular movement triggered by changes in chemical concentrations within the medium [11]. These models are usually composed of sets of equations that describe the behavior of cell populations and concentrations of different forms of chemoattractants [12]. In the slime mold experiments by Dolak et al., it was found that cells respond to specific concentrations of cyclic AMP, which is a substance with stimulating properties. Its concentration gradients may vary with time and space, thus determining the way cells spread and aggregate [13]. In particular, temporally dependent gradients are significant [14] because they enhance the concentration of the stimulating substance, which is essential for chemotaxis, without a significant delay in response [15]. In response to growth factors, cells undergo a range of intracellular signalling processes which, in turn, result in directed movement towards source of the stimulus. This involves alteration of intracellular pH for movement regulation [16]. Moreover, the persistence of components of chemotactic signaling enables cells' response to these markers and prevent constant movement toward stimuli. Exploiting this "short term memory" allows more efficient movement towards stimuli [17].

Probabilistic Decision-Making. Probabilistic decision-making refers to the process of making choices based on probability and statistical information. Many improved slime mould algorithms, such as the Improved Slime Mould Algorithm (ISMA) and the Enhanced Slime Mould Algorithm (ESMA), have focused on how to make better decisions. They use dynamic probability thresholds, as well as strategies such as Gompertz curves and adaptive learning to make optimal decisions, prevent the algorithm from getting trapped in local optima, and enhance its global search capabilities [18–20,22]. Algorithms like the Evolutionary Multimode Slime Mould Optimization (EMSMO) use probabilistic selection functions to determine the sequence of heuristics, allowing for a more flexible and adaptive search process [21]. This probabilistic approach is crucial for simulating the exploratory behavior of slime moulds.

Emergent Behavior and Self-organization. Slime molds exhibit emergent behavior and self-organization. They show emergent intelligence for growth and adaptation through the transport of nutrients and chemical signals as they construct complex environments. Their behavior is studied using neural cellular automata in the context of distributed dynamical systems [23]. The Physarum Experiments showcase the organism's remarkable capacity to form collective behaviors and navigate its surroundings despite lacking a central nervous system [24]. Slime molds are also adept at solving complex problems, such as graph-solving and decision-making, by creating self-organization networks. These networks are optimized through decentralized models that mimic behaviors like chemotaxis [25]. The emergence of behavior in slime molds is closely tied to self-organization patterns of rhythmic contractions [26]. These contractions enable

the molds to make dynamic transitions between various behaviors. Moreover, this complexity can be mapped to biomechanical dynamics, which offers valuable insights into the origin of their behavior [27].

2.3 Application and Challenge of SMA

Currently, the Slime Mould Algorithm is mainly applied in the field of graph optimization, including problems such as the Shortest Path Tree Problem, Supply Chain Network Design, Maze Problem, and Multi-source Multi-sink Minimum Cost Flow Problem [8]. For example, it can be used to design efficient transportation networks, find the shortest paths, and optimize supply chains. In the engineering field, since the Slime Mould Algorithm (SMA) can handle non-linear and multi-modal features, it can be used to efficiently address complex non-linear tasks, such as structural design optimization and economic load dispatch [5]. In addition, after combining SMA with technologies like Particle Swarm Optimization or Genetic Algorithms, it has achieved success in multi-objective optimization. These combinations improve diversity and convergence speed. For instance, the Multi-Objective Slime Mould Algorithm (MOSMA) uses elitist non-dominated sorting and crowding distance operators to handle multi-objective problems [6]. Other hybrid algorithms, such as those combined with Differential Evolution, enhance performance by improving local search and population diversity [7], making them suitable for global and combinatorial optimization.

However, the potential of slime mold algorithm (SMA) in artistic applications has not been tapped. Unlike static optimization, art and design work requires flexibility, adaptability and real-time response. SMA's Emergent Behavior and Self-Organization capabilities can be used to generate dynamic patterns in real time for generative art, interactive design, and immersive simulations. However, at present, the research on these applications is still blank.

2.4 The Development Trend of SMA

In the field of creative applications, because SMA has the characteristics of Emergent Behavior and Self-Organization, it can be used to generate complex and changing patterns and respond to user input or different changing environmental factors in real time. In addition, the hybrid algorithm that combines SMA with machine learning can enhance its ability to adapt to dynamic input and produce more complex real-time output [9]. With the continuous development of SMA computing framework, its scalability and integration with modern technologies such as WebGL and real-time rendering engine are very important to expand its application. The combination of SMA and creative exploration opens up new possibilities in the field of design and art.

3 Methods

3.1 Technical Framework: Computational Environment and Tools

Combining new web-based technologies with libraries helps the platform be designed to provide real-time interactivity and great speed. The study apply the following instruments and frameworks (Fig. 1 and Table 1)

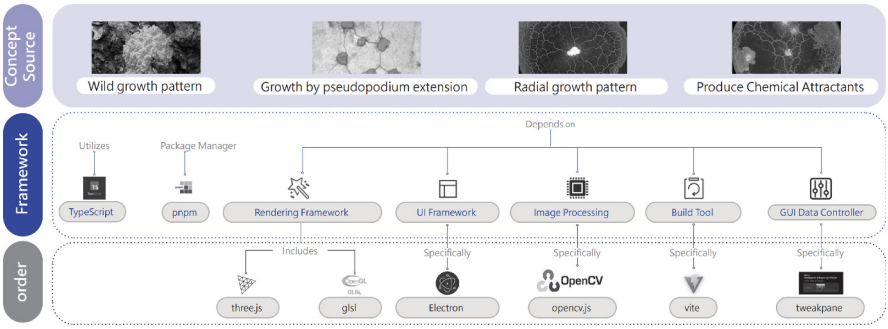

Fig. 1. Technical Framework

Table 1. Tools and frameworks used in the platform development

Tool	Purpose	Details
TypeScript	Core programming language	Provides strong typing and object-oriented features for maintainability and scalability
Three.js	3D rendering and particle system simulation	Simplifies WebGL usage and enables the creation of dynamic visual scenes
WebGL	Graphics rendering	JavaScript API for rendering 2D and 3D graphics in real-time
opencv.js	Image processing	Handles edge detection and image segmentation for image input functionality
Electron	Application framework	Supports cross-platform desktop application development
Vite	Build tool	Ensures fast compilation and efficient development workflow
tweakpane	GUI tool	Provides a simple interface for users to control simulation parameters dynamically

This technical stack allows for high-performance rendering and real-time interaction, forming a modern, scalable development environment.

3.2 Agent-Based Simulation Process

Inspired by the chemotactic movement of *Physarum polycephalum* towards attractants, this simulation depicts agent behaviour based on the slime mould algorithm (SMA). The agents interact in simple ways to create complex and emergent patterns by copying actual chemotactic responses. The approach uses the stages shown below:

Sense: Every agent begins by assessing pheromone levels three times around its present course. This stage resembles the chemoattractant reactivity of the slime mold. Using provided offsets (input parameter: θ), the agents "sample" their surrounds to assess the gradient of the chemical attractant. Later decisions are based on directional knowledge gained by agents during the detecting phase.
Rotate: Agents use the found pheromone concentrations to guide their next path. The agent guarantees its movement towards areas with a high likelihood of reward or resource (input parameter: chemical attractant) by selecting the path with the highest density of attractants.
Move: Every agent moves forward under a recently defined directive. Inspired by the computed rotation, this movement step transforms the actors' spatial data inside the environment. Real-time modification of movement's course and speed responds to always changing pheromone concentrations.
Deposit: Arriving at a new location, every agent deposits a designated pheromone amount. For other agents, the release of attractant at the agent's location serves as a breadcrumb defining their paths of interest.
Diffuse: At every place, the pheromones implanted start to spread outward. This spreading process distributes the chemical signal from every deposit, therefore influencing the surrounding area and drawing other agents (input parameter: decay).
Decay: With time, the pheromone concentrations at any location progressively decrease. This stage of degradation encourages exploration and the development of efficient paths by preventing agents from always clustering in the same location. It replicates the transient character of the chemical signals produced by slime mould, which vanish with time to avoid depending on outdated paths (input parameter: decay).

3.3 Algorithm Development

Particle System and Reaction-Decay Model. Driven by the reaction-decay model, which models chemotactic (attractant-seeking) and anti-chemotactic (obstacle-avoidant) behavior in a dynamic environment, the particle system forms the central focus of the simulation. Inspired by the behavior of Physarum polycephalum, the system is built according on the reaction-decay model. This function also shows how flexible the platform is for several uses, including tasks involving optimization in limited surroundings (Table 2).

Table 2. Key parameters in the reaction-attenuation model

Parameter	Description	Value Range
Initial Particle Count	Number of particles at simulation start	100–500
Attractant Strength	Influence of chemical attractants on particles	0.1–1.0
Repellent Strength	Influence of obstacles on particle behavior	0.1–1.0
Reaction-Decay Iterations	Number of iterations to refine particle paths	1–10

Probabilistic Threshold Decision Algorithm. A probabilistic threshold decision approach is applied to replicate the exploration and exploitation activities observed in *Physarum*, hence improving the realism of particle movement. The method determines, depending on the local chemical gradient, the likelihood that a particle will choose a given path:

$$\text{Probability}(path_i) \;=\; \frac{C_i}{\sum_j C_j},$$

where C_i is the local concentration across a given path, and the denominator is the sum of concentrations for all potential paths.

A random number r is generated within the interval $[0, 1]$, mimicking the stochastic nature of particle movement in natural systems. The particle will follow $path_i$ if

$$\sum_{k=0}^{i-1} \text{Probability}(path_k) < r \;\leq\; \sum_{k=0}^{i} \text{Probability}(path_k).$$

This means the path isn't deterministic and random selection allow the particles to go on varied possible routes. As r can change, it is possible for the algorithm to explore by taking random deviations from the attractant gradient (exploration) or follow along the direction with the greatest concentration of chemoattractant (exploitation). Path Selection Process: It's a diverse and repeated process for all particles at every simulation iteration. This way stochastically determines whether a potential equipment path of a unique particle can be chosen or not in any round, which allows the paths of those particles to change its character in every round based on their environment, mimicking the behaviour of slime mould in nature (Table 3).

Table 3. Main steps of the probabilistic threshold decision algorithm

Step	Description
Random Number Generation	Generates a random number r to simulate non-deterministic decision-making.
Threshold Comparison	Compares r with the calculated probabilities for different paths.
Path Selection	Chooses a path based on the probability distribution of chemoattractants and repellents.

Iterative Optimization. iterations improve the current solution of the network of paths as particles tend to be attracted towards attractants and keep away from obstacles. With its introduction of random variation, the algorithm reduces the risk of particles becoming stuck in local optima, allowing for more thorough global exploration of the simulation space instead. This random decision based on threshold is inspired by the probabilistic response of slime mold, thus helping the particles mimic the behavior of Physarum polycephalum to adapt to real-world conditions. Particles navigate their environment, taking advantage of what they find, and this leads to complex, emergent patterns of behavior.

3.4 Customizing Text Generation Weight Process

The process of weight generation takes the text provided by the user and transforms it into a pixel-based weight format that affects the behavior of the particles:

Font Rendering and Configuration: The text is rendered with a chosen font—each letter is transformed into a pixel grid. It also employs a dynamically fitting centering algorithm.

Edge Detection: The Sobel edge detection algorithm detects the edges of every character. The gradient magnitudes are computed by:

$$G = \sqrt{(G_x)^2 + (G_y)^2}.$$

Threshold Segmentation: The text is segmented based on pixel intensity, isolating those features that will be useful in guiding particles.

Normalization: Normalizing the pixel values so that the text does not take up a significant portion of the weight.

3.5 Customizing Image Fitting Weight Process

Users can upload images to be transformed into chemoattractant maps based on pixel value, determining the relative strength of chemoattractant:

Pixelate the Image: The image is translated into a pixel grid, and each pixel becomes a node in the chemoattractant field.

Edge Detection: The Canny edge detection algorithm accentuates the significant contours and edges in the image.

Threshold Segmentation: The pixel intensities are segmented; therefore, particles tend to learn the larger features of the image.

Normalization and Gradient Generation: Pixel values are normalized and turned into a chemoattractant gradient, attracting particles towards regions of high intensity.

Users can change the text features that affect particle behavior, resulting in emergent visual patterns.

3.6 Dynamic Rendering and User Interaction

The platform integrates dynamic rendering with an intuitive user interface to enhance the interactivity of the simulation. WebGL and Three.js promote real-time rendering, enabling users to modify parameters like particle count, speed, and chemoattractant strength using the GUI supplied by tweakpane.

As seen in Fig. 2, The platform provides three rendering modes: *Default Rendering Mode*, *Sobel Rendering Mode*, and *Customized Rendering Mode*. Using ordinary edge detection, the default mode creates grayscale outputs showing particle pathways. Sobel mode may also render in color gradients, which offers flexibility, scientific analysis, and artistic creativity. Customized mode enables increased aesthetic control, allowing users to overlay gradients, modify color mixing, and manipulate texture settings for personalized visualizations.

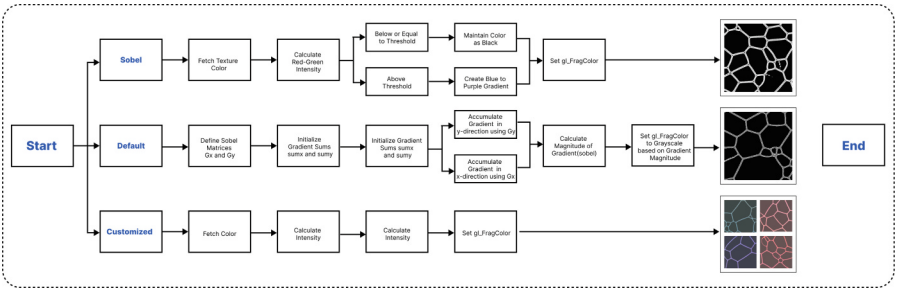

Fig. 2. Dynamic Rendering Mode Diagram

4 Results

The platform was tested by artists and beta users to assess its usability and effectiveness in generating customizable designs. The results demonstrate the system's ability to create dynamic, visually complex outputs.

Users reported that the text generation weight process allowed for significant control over text appearance. The platform enabled users to create both bold, structured typography and more abstract, flowing text-based designs (Table 4 and Fig. 3).

Table 4. User feedback of the text & image generation design

Metric	Result
Pattern Recognition Accuracy	92%
Customization Complexity	Moderate to High
User Satisfaction	38% increase

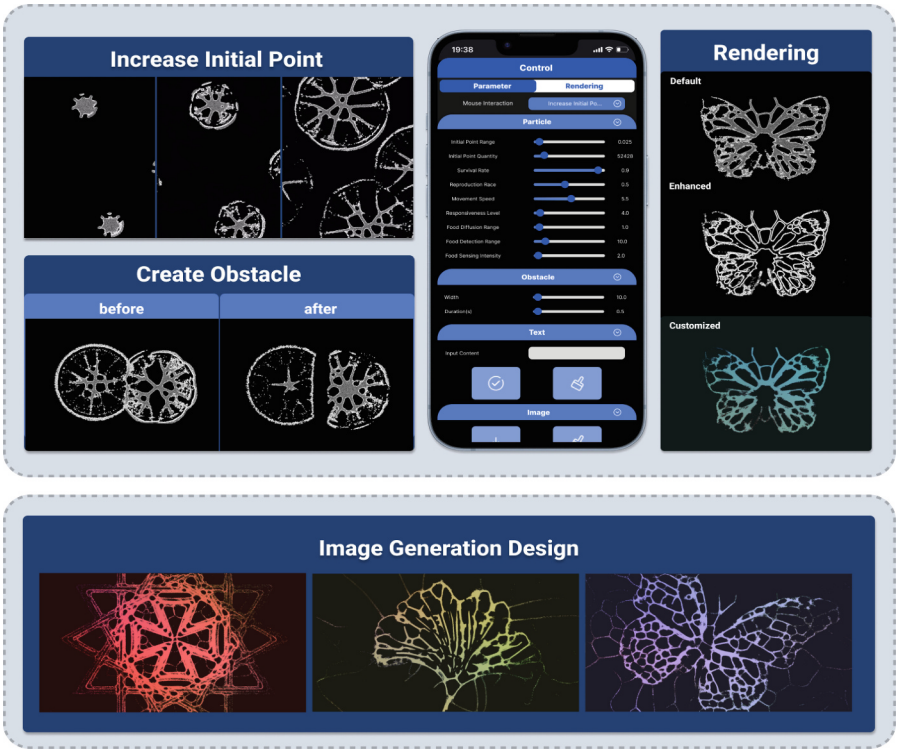

Fig. 3. Fitting Text and Image Processing Flowchart

5 Discussion and Conclusion

This platform generates dynamic, emergent designs by efficiently combining bio-inspired algorithms with artistic tools. Chemical Anti-Attractants (CAA) improve the simulation by allowing realistic obstacle avoidance and adaptive navigation, hence increasing its relevance to optimization problems and artistic inquiry. The probabilistic decision-making algorithm includes randomness into particle behavior, so reducing stagnation in local optima. Sobel edge detection enhances the capacity of the system to analyze complex text and image data, so supporting accurate particle interactions and producing visually pleasing results.

The platform has numerous rendering styles, including default, customizable, and Sobel-based options, allowing both analytical and creative results. Experimental findings demonstrated the platform's capacity to generate complex patterns and accommodate diverse user inputs. This multidisciplinary tool efficiently integrates computational biology, generative art, and computer graphics, offering significant potential for applications in optimization, visual design, and educational resources.

References

1. Li, S., Chen, H., Wang, M., Heidari, A., Mirjalili, S.: Slime mould algorithm: a new method for stochastic optimization. Future Gener. Comput. Syst. **111**, 300–323 (2020)
2. Kamboj, V., et al.: A cost-effective solution for non-convex economic load dispatch problems in power systems using slime mould algorithm. Sustainability **14**(5), 2586 (2022)
3. Tang, A., Tang, S., Han, T., Zhou, H., Xie, L.: A modified slime mould algorithm for global optimization. Comput. Intell. Neurosci. 1–14 (2021)
4. Chen, H., Li, X., Li, S., Zhao, Y., Dong, J.: Improved slime mould algorithm hybridizing chaotic maps and differential evolution strategy for global optimization. IEEE Access **10**, 66811–66830 (2022)
5. Mostafa, M., Rezk, H., Aly, M., Ahmed, E.: A new strategy based on slime mould algorithm to extract the optimal model parameters of solar PV panel. Sustain. Energy Technol. Assess. **42**, 100849 (2020)
6. Premkumar, M., Jangir, P., Sowmya, R., Alhelou, H., Heidari, A., Chen, H.: MOSMA: multi-objective slime mould algorithm based on elitist non-dominated sorting. IEEE Access **9**, 3229–3248 (2021)
7. Houssein, E., Mahdy, M., Blondin, M., Shebl, D., Mohamed, W.: Hybrid slime mould algorithm with adaptive guided differential evolution algorithm for combinatorial and global optimization problems. Expert Syst. Appl. **174**, 114689 (2021)
8. Zhang, X., Gao, C., Deng, Y., Zhang, Z.: Slime mould inspired applications on graph-optimization problems. In: Unconventional Computation and Natural Computation, pp. 519–562 (2016)
9. Wei, Y., Othman, Z., Daud, K., Luo, Q., Zhou, Y.: Advances in slime mould algorithm: a comprehensive survey. Biomimetics **9**(1), 31 (2024)
10. Greenberg, J., Alt, W.: Stability results for a diffusion equation with functional drift approximating a chemotaxis model. Trans. Am. Math. Soc. **300**, 235–258 (1987)
11. Chalub, F., Dolak-Struss, Y., Markowich, P., Oelz, D., Schmeiser, C., Soreff, A.: Model hierarchies for cell aggregation by chemotaxis. Math. Models Methods Appl. Sci. **16**, 1173–1197 (2006)
12. Bärwolff, G., Walentiny, D.: Numerical and analytical investigation of chemotaxis models. In: Simulation and Mathematical Methods in Bioeconomics, pp. 3–18 (2018)
13. Dolak, Y., Schmeiser, C.: Kinetic models for chemotaxis: hydrodynamic limits and spatio-temporal mechanisms. J. Math. Biol. **51**, 595–615 (2005)
14. Vicker, M., Schill, W., Drescher, K.: Chemoattraction and chemotaxis in Dictyostelium discoideum: myxamoeba cannot read spatial gradients of cyclic adenosine monophosphate. J. Cell Biol. **98**, 2204–2214 (1984)
15. Höfer, T., Maini, P., Sherratt, J., Chaplain, M., Murray, J.: Resolving the chemotactic wave paradox: a mathematical model for chemotaxis of Dictyostelium amoebae. J. Biol. Syst. **3**, 967–973 (1995)
16. McRobbie, S.: Chemotaxis and cell motility in the cellular slime molds. Crit. Rev. Microbiol. **13**(4), 335–375 (1986)
17. Van Duijn, B., Inouye, K.: Regulation of movement speed by intracellular pH during Dictyostelium discoideum chemotaxis. Proc. Natl. Acad. Sci. U.S.A. **88**(11), 4951–4955 (1991)

18. Li, D., Gao, F.: Improved slime mould algorithm based on Gompertz dynamic probability and Cauchy mutation with application in FJSP. J. Intell. Fuzzy Syst. **44**, 10397–10415 (2023)
19. Qiu, Y., Li, R., Zhang, X.: Simultaneous SVM parameters and feature selection optimization based on improved slime mould algorithm. IEEE Access **12**, 18215–18236 (2024)
20. Xiong, W., Li, D., Zhu, D., Li, R., Lin, Z.: An enhanced slime mould algorithm combines multiple strategies. Axioms **12**, 907 (2023)
21. Zhong, R., Zhang, E., Munetomo, M.: Evolutionary multi-mode slime mould optimization: a hyper-heuristic algorithm inspired by slime mould foraging behaviors. In: 2023 Congress in Computer Science, Computer Engineering Applied Computing (CSCE), pp. 2153–2160 (2023)
22. Naik, M.K., Panda, R., Abraham, A.: Adaptive opposition slime mould algorithm. Soft. Comput. **25**(22), 14297–14313 (2021). https://doi.org/10.1007/s00500-021-06140-2
23. Barbieux, A., Canaan, R.: EINCASM: emergent intelligence in neural cellular automaton slime molds. arXiv preprint (2023). arXiv:2305.13425
24. Barnett, H.: The Physarum Experiments (and Being Slime Mould) (2016)
25. Cameron, A.: Modelling Self-organising Networks with Slime Mould Physarum Polycephalum (2017)
26. Fleig, P., Kramar, M., Wilczek, M., Alim, K.: Emergence of behaviour in a self-organized living matter network. eLife, **11**, e62863 (2022)
27. Fleig, P., Kramar, M., Wilczek, M., Alim, K.: Emergence of behavior in a self-organized living matter network. bioRxiv 2020.09.06.285080 (2020)
28. Adamatzky, A.: Physarum Machines: Computers from Slime Mould. World Scientific(2010)
29. Jones, J.: Emergent Computing Characteristics in Slime Mould (2011). (Unpublished or internal reference)