Politecnico di Torino

Master's Degree in Data Science and Engineering

Langevin Particle Swarm Optimization with Friction-Based Communication

Supervisors:
Prof. Luigi Preziosi
Prof. Marco Scianna

Candidate:
Ivan Ludvig Tereshko

Academic Year 2024/2025

Abstract

This thesis presents a swarm-based optimization method inspired by the over-damped Langevin dynamics. A swarm consists of particles, each characterized by its position and friction coefficient, and subjected to a random force. The particles communicate by updating their friction coefficients based on relative performance within the swarm.

The friction communication mechanism enables better-performing particles to move slower and remain near optimal regions, while worse-performing particles move faster to explore the search space. This creates a balance between exploration of new areas and careful exploitation of promising regions. The communication mechanism leads to emergent annealing behaviour, where average friction increases over time, similar to temperature reduction in simulated annealing.

Two types of random forces are considered: a Gaussian random force, where the noise variance decreases with the friction coefficient, leading to Brownian motion, and a Lévy-stable random force, producing Lévy flight dynamics.

The method is experimentally evaluated on benchmark function optimization and neural network training, demonstrating competitive performance compared to existing algorithms. A GPU-based implementation enables efficient parallel execution.

Contents

Introduction

Particle swarm optimization is a computational method for optimizing continuous non-linear functions, introduced by Kennedy and Eberhart [1]. The method is inspired by social behaviour in nature, such as the group dynamics observed in bird flocking and fish schooling. Interaction between particles, which implies information sharing, plays a fundamental role in emergent behaviour, allowing particles to capitalize on each other's knowledge and form a collective intelligence.

In the Swarm-Based Gradient Descent (SBGD) method, introduced in [2], the swarm agents are characterized by their position and mass. The agents communicate by transferring a quantity termed "mass", which has the role of slowing down more optimal agents and speeding up the less optimal agents. Thus, the "mass" determines the step size of each agent, allowing lighter agents to take bigger steps to explore, and heavier agents to take smaller steps towards their local minima. The study of SBGD is extended in [3], where the particle's descent direction is chosen randomly, centered on the gradient direction, allowing a more thorough exploration of the search space, increasing the method's performance. The Swarm-based Simulated Annealing method is introduced in [4], which applies an approach similar to simulated annealing to SBGD: particles are subject to Brownian motion with the annealing rate being a decreasing function of their "mass". Thus, the system "cools down" as it approaches the solution by transferring the mass from higher ground agents to more optimal agents at lower ground.

This work considers a swarm optimization method inspired by the overdamped dynamics of a particle swarm with friction transfer. Similarly to "mass" in SBGD, the friction coefficient is transferred from poorer to better-performing particles, allowing better-performing agents to further explore their vicinity. Poorer performing particles are more motile, taking larger steps. The particles are subject to a random force, the choice of which is discussed here. We consider and compare: a Gaussian force, depending on the friction coefficient, leading to Brownian motion, and a Lévy-stable random force with various parameters, leading to Lévy flight dynamics.

Chapter 1

Swarm-Based Optimization Methods

Let $U(\mathbf{x})$ be a continuous potential function, defined on a d-dimensional Euclidean space $\mathbb{R}^d$. We consider the following optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} U(\mathbf{x}) \tag{1.1}$$

The potential is assumed to be confining, meaning $U(\mathbf{x}) \to \infty$ as $|\mathbf{x}| \to \infty$.

1.1 Particle Swarm Optimization (PSO)

The standard Particle swarm optimization (PSO) is an iterative algorithm, where N agents search for the solution with movement influenced by inertia, personal experience and social influence [1]. Each particle i is initialized with a random position $\boldsymbol{x}_i^{(0)}$ in the search space and velocity $\boldsymbol{v}_i^{(0)}$. Each particle keeps track of its personal best solution $\boldsymbol{p}_i$ and the global best solution $\boldsymbol{g}$, reached among all particles. At each iteration, the velocities and positions are updated according to the system

$$\begin{cases} \boldsymbol{v}_i^{(n+1)} = w\boldsymbol{v}_i^{(n)} + c_1\mathbb{A}_1(\boldsymbol{p}_i^{(n)} - \boldsymbol{x}_i^{(n)}) + c_2\mathbb{A}_2(\boldsymbol{g}^{(n)} - \boldsymbol{x}_i^{(n)}) \\ \boldsymbol{x}_i^{(n+1)} = \boldsymbol{x}_i^{(n)} + \boldsymbol{v}_i^{(n+1)} \end{cases} \quad i = 1, \dots, N \tag{1.2}$$

where $w, c_1, c_2 \in \mathbb{R}$ are chosen constants and $\mathbb{A}_1, \mathbb{A}_2 \in \mathbb{R}^{d\times d}$ are diagonal matrices with random samples drawn at each iteration for each particle from the uniform distribution in $[0, 1]$ on the diagonal. The roles of the parameters are as follows:

1. w controls momentum: the contribution of the particle's previous velocity to its current movement;

2. c_1 controls the influence of the best personal solution $\boldsymbol{p}_i$, determining how individualistic are the particles;

3. c_2 controls the impact of the best global solution $\boldsymbol{g}$, determining the extent to which the particles are driven by social influence.

In addition to updates (1.2), the velocity of each particle is clamped up to a maximum velocity $v_{\max}$ on each dimension, preventing excessive movement. The four parameters w, c_1, c_2 and $v_{\max}$ are subject to tuning. Having a significant impact on performance, parameter selection has been the subject of extensive research [5]. A major issue in PSO is premature convergence to local minima without ever discovering the global minimum.

PSO has been successfully applied to a wide range of optimization problems, due to its simplicity and high efficiency, with the most success in the areas of system design, multi-objective optimization, resource allocation, and many others [6].

1.1.1 PSO variants

Bare bones PSO

The Bare bones particle swarm optimization (BPSO) is a simplified version of the PSO with the velocity eliminated [7]. Instead, the particle position $\boldsymbol{x}_i$ is updated by drawing from the Gaussian distribution based on the personal $\boldsymbol{p}_i$ and global $\boldsymbol{g}$ best solutions:

$$\boldsymbol{x}_i \sim \mathcal{N}\left(\frac{\boldsymbol{p}_i + \boldsymbol{g}}{2}, ||\boldsymbol{p}_i - \boldsymbol{g}||\right)$$

The simplification introduced by BPSO makes the algorithm more compact and parameter-free, eliminating the need for parameter tuning. BPSO has seen successful applications, including in feature selection [8], integer programming [9] and vapor-liquid equilibrium modelling [10].

Competitive swarm optimizer

The Competitive swarm optimizer (CSO) is inspired by PSO, but instead of using the best global and personal solutions in updates, it introduces a pairwise competition mechanism, where the winner learns from the loser [11].

At each timestep n, the particles are randomly paired into $N/2$ couples, assuming that the number of particles N is even. Within each pair k, the particles compete by comparing their fitness, resulting in one particle being the winner w and the other the loser l, on the basis of the criterion $U(\boldsymbol{x}_w^{(n)}) \leq U(\boldsymbol{x}_l^{(n)})$. The position and velocity

of the winner remain unchanged, while those of the loser are updated, according to the following system:

$$\begin{cases} \boldsymbol{v}_{k,w}^{(n+1)} = \boldsymbol{v}_{k,w}^{(n)} \\ \boldsymbol{x}_{k,w}^{(n+1)} = \boldsymbol{x}_{k,w}^{(n)} \\ \boldsymbol{v}_{k,l}^{(n+1)} = \mathbb{A}_1 \boldsymbol{v}_{k,l} + \mathbb{A}_2(\boldsymbol{x}_{k,w}^{(n)} - \boldsymbol{x}_{k,l}^{(n)}) + \varphi \mathbb{A}_3(\bar{\boldsymbol{x}}_k^{(n)} - \boldsymbol{x}_{k,l}^{(n)}) \\ \boldsymbol{x}_{k,l}^{(n+1)} = \boldsymbol{x}_{k,l}^{(n)} + \boldsymbol{v}_{k,l}^{(n+1)} \end{cases} \quad k = 1, \dots, N/2 \tag{1.3}$$

where $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3 \in \mathbb{R}^{d \times d}$ are diagonal matrices with random vectors in $[0,1]^d$ on the diagonal, generated for each pair k at each timestep n, $\bar{\boldsymbol{x}}_k^{(n)}$ is the mean position of the swarm and $\varphi \in \mathbb{R}$ is a parameter controlling social influence. The update rule for the loser's velocity $\boldsymbol{v}_{k,l}^{(n+1)}$ is analogous to that of the standard PSO (1.2) with certain differences:

1. The first term $\mathbb{A}_1 \boldsymbol{v}_{k,l}$ accounts for inertia, as in standard PSO, but with an introduced weight $\mathbb{A}_1$
2. The second term $\mathbb{A}_2(\boldsymbol{x}_{k,w}^{(n)} - \boldsymbol{x}_{k,l}^{(n)})$ is the cognitive component: the loser learns from the winner, instead of being guided by the personal best position.
3. The third term $\varphi \mathbb{A}_3(\bar{\boldsymbol{x}}_k^{(n)} - \boldsymbol{x}_{k,l}^{(n)})$ is the social component. However, the particle is influenced by the current mean position of the swarm $\bar{\boldsymbol{x}}_k^{(n)}$, rather than the global best.

As a result, CSO is conceptually similar to the standard PSO, maintaining its simplicity and, furthermore, eliminating the concept of memory in the form of tracking the personal and global best solutions.

CSO shows better performance than the standard PSO and state-of-the-art optimization algorithms in large-scale optimization problems ($\sim 10^3$ dimensions) with successful applications across different domains, such as resource allocation, engineering design, and complex system modelling [12].

1.2 Swarm-Based Gradient Descent (SBGD)

The Swarm-based gradient descent (SBGD) method, introduced in [2], models a swarm of N agents characterized by their position $\boldsymbol{x}_i$ and "mass" m_i. The agents interact by transferring "mass" from the less optimal to the more optimal agents. The step size h_i of each agent i is a decreasing function of its "relative mass" $\widetilde{m}_i = m_i / \max_i m_i$, allowing lighter agents to take longer steps to explore the space, while

heavier agents take shorter steps towards their local minima. The discrete-time update rule for agents with positive "mass" $m_i > 0$ is:

$$\begin{cases} d\boldsymbol{x}_i = -h_i(\boldsymbol{x}_i, \widetilde{m}_i)\nabla U(\boldsymbol{x}_i)\, dt \\ dm_i = -\left(\dfrac{U(\boldsymbol{x}_i) - U_{\min}}{U_{\max} - U_{\min}}\right)^p m_i(t)\, dt, & i \neq i^* \\ m_{i^*} = 1 - \displaystyle\sum_{j \neq i^*} m_j, & i = i^* \end{cases} \tag{1.4}$$

where $i^* = \arg\min_i U(\boldsymbol{x}_i)$ is the index of the current best-performing agent, $U_{\min} = \min_i U(\boldsymbol{x}_i)$, $U_{\max} = \max_i U(\boldsymbol{x}_i)$ and $p > 0$ is a fine-tuning parameter with the default choice $p = 1$. The "mass" update mechanism ensures that at each step the worst performing agent loses all of its "mass" and is eliminated. Thus, the simulation ends after $N-1$ timesteps, when a single particle is left. The update rules also ensure that the total "mass" is conserved. When initialized as $m_i = 1/N$, the "mass" obtains a probabilistic interpretation, indicating the probability of a particular agent reaching the optimal solution.

Numerical simulations on benchmark functions demonstrate the effectiveness of SBGD, but reveal sensitivity to the shift of the target function from the particle initialization positions: when the particles are initialized outside of the vicinity of the solution, the success rate of the algorithm drops significantly.

1.2.1 SBGD variants

SBGD with random descent

Swarm based optimization with random descent (SBRD) is a modification of SBGD, introduced in [3]. The particle's descent direction $\boldsymbol{p}_i$ is chosen randomly, centered around the gradient direction, enabling a more thorough exploration of the search space. The equation for particle position in (1.4) is replaced with

$$d\boldsymbol{x}_i = -h_i(\boldsymbol{x}_i, \widetilde{m}_i)\boldsymbol{p}_i\, dt$$

where the descent direction $\boldsymbol{p}_i = |\nabla U(\boldsymbol{x}_i)|\boldsymbol{\omega}_i$ is sampled from the spherical cap centered around the gradient direction, according to:

$$\boldsymbol{\omega}_i \cdot \boldsymbol{q}_i = r, \quad \boldsymbol{q}_i = \frac{\nabla U(\boldsymbol{x}_i)}{|\nabla U(\boldsymbol{x}_i)|}, \quad r \sim \mathcal{U}\left(\frac{1}{2}(1 + \widetilde{m}_i)\right)$$

where $\mathcal{U}$ denotes the uniform distribution. The maximum deviation from the steepest descent direction is higher for lighter agents, given by $\theta = \arccos(\frac{1}{2}(1 + \widetilde{m}_i))$.

The modification increases the SBGD's performance, being especially crucial for higher-dimensional optimization and in the case of a shifted target function.

Swarm-based simulated annealing

The Swarm-based simulated annealing (SSA) combines SBGD with Simulated annealing (SA) [4]. Particles are subject to Brownian motion with the annealing rate $\sigma(m)$ being a decreasing function of their "mass". The dynamics is described by the system

$$\begin{cases} d\boldsymbol{x}_i = -\nabla F(\boldsymbol{x}_i)\, dt + \sqrt{2\sigma(m_i(t))}\, dW_i \\ dm_i = -(U(\boldsymbol{x}_i) - \overline{U}^{(N)}(t)) m_i(t)\, dt \end{cases} \qquad i = 1, \dots, N \tag{1.5}$$

where dW_i is the Wiener process and $\overline{U}^{(N)}(t)$ is the provisional minimum, calculated as the mass-weighted average of potentials:

$$\overline{U}^{(N)}(t) = \frac{\sum_{i=1}^{N} m_i(t) U(\boldsymbol{x}_i)}{\sum_{i=1}^{N} m_i(t)}$$

As in the original SBGD, the total "mass" remains constant. Contrary to traditional SA, where the annealing rate σ is explicitly controlled, the dynamics "cools down" the better-performing agents implicitly by transferring them "mass" from the worse-performing agents. In contrast, the worse-performing particles are "heated up", enabling them to escape local basins and explore the space for better solutions.

The main result of [4] states that under certain assumptions on $\sigma(m)$ and assuming the uniqueness of the global minimizer of $U(\boldsymbol{x})$, the provisional minimum converges to the global minimum U^* in the large-particle, long-time limit:

$$\lim_{N\to\infty} \overline{U}^{(N)}(t) \xrightarrow{t\to\infty} U^* = U(\boldsymbol{x}^*)$$

Chapter 2

Theoretical Background

2.1 Langevin dynamics

2.1.1 Langevin equation

The Langevin equation, introduced in 1908 [13], describes the dynamics of a macroscopic particle suspended in a viscous fluid. Influenced by random collisions with surrounding microscopic particles, the particle's position fluctuates, resulting in Brownian motion. The original equation in one dimension, for simplicity, can be written as

$$m\ddot{x} = -\mu\dot{x} + F(t) + \omega(t) \tag{2.1}$$

where x is the particle's position, m is its mass, μ is its friction coefficient and $F(t)$ is a force the particle is subjected to. The random force ω is a white noise term (Wiener process), which has zero mean and is uncorrelated at any two different moments in time

$$\langle \omega(t) \rangle = 0, \quad \langle \omega(t)\omega(t') \rangle = 2\mu T \delta(t - t')$$

where T is the temperature. For simplicity, the Boltzmann constant is set to 1 in natural units: $k_B = 1$.

In the overdamped case, the inertia term becomes negligible. If the force $F(t)$ is conservative with potential $U(x)$, we obtain the equation that forms the basis for the method proposed in this work

$$\dot{x} = -\frac{1}{\mu}\frac{\partial U}{\partial x} + \eta(t) \tag{2.2}$$

where $\eta(t) = \omega/\mu$ is the rescaled random force with the autocorrelation function

$$\langle \eta(t)\eta(t') \rangle = 2\frac{T}{\mu}\delta(t-t') = 2D\delta(t-t')$$

Here, the diffusion coefficient D was introduced according to Einstein's relation

$$D = T/\mu \tag{2.3}$$

2.1.2 Fokker-Planck equation

As derived in [14], the corresponding Fokker-Planck Equation (FPE) for the probability density $P(x,t)$, equivalent to the overdamped Langevin equation (2.2), is given by

$$\frac{\partial}{\partial t}P(x,t) = \frac{\partial}{\partial x}\left(\frac{1}{\mu}\frac{\partial U}{\partial x}P(x,t)\right) + D\frac{\partial^2}{\partial x^2}P(x,t) \tag{2.4}$$

This partial differential equation describes the time evolution of the probability density $P(x,t)$, giving the probability that the particle is at position x at time t. The first term on the right-hand side accounts for deterministic gradient descent, and the second term for diffusion.

The continuity equation for probability density $P(x,t)$ is

$$\frac{\partial}{\partial t}P(x,t) + \frac{\partial}{\partial x}J(x,t) = 0 \tag{2.5}$$

where $J(x,t)$ is the probability current

$$J(x,t) = -\frac{1}{\mu}\frac{\partial U}{\partial x}P(x) - D\frac{\partial}{\partial x}P(x)$$

In order to find the steady-state solution $P_0(x)$, which is reached at equilibrium, we assume that $\frac{\partial P(x,t)}{\partial t} = 0$ and rewrite the equation (2.4) as

$$\frac{\partial}{\partial x}\left(\frac{1}{\mu}\frac{\partial U}{\partial x}P_0(x) + D\frac{\partial}{\partial x}P_0(x)\right) = 0$$

This equation expresses that the probability current $J(x)$ is constant in space

$$\frac{\partial}{\partial x}J(x) = 0$$

Given the boundary conditions $P_0(x) \xrightarrow[x\to\infty]{} 0$ and $\frac{\partial}{\partial x}P_0(x) \xrightarrow[x\to\infty]{} 0$, it follows that

$J \equiv 0$. We obtain a separable first order differential equation

$$\frac{1}{\mu}\frac{\partial U}{\partial x}P_0(x) + D\frac{\partial}{\partial x}P_0(x) = 0$$

$$\frac{dP_0}{P_0} = -\frac{1}{D\mu}\frac{\partial U}{\partial x}dx$$

After integrating and using Einstein's relation (2.3), the solution is obtained

$$P_0(x) = \frac{1}{Z}\exp\left(-\frac{U(x)}{T}\right) \tag{2.6}$$

where the constant Z is defined as

$$Z = \int_{-\infty}^{+\infty} \exp\left(-\frac{U(x)}{T}\right) dx$$

to normalize the probability density function $P_0(x)$.

The obtained steady-state solution corresponds to the Boltzmann distribution, which arises in a wide variety of problems. In statistical physics, the Boltzmann distribution describes the energy distribution of a particle system at thermal equilibrium. In machine learning, it is known as the softmax function, commonly used as the activation function in the output layer of a neural network to obtain a probability distribution from its output. An example of the Boltzmann distribution for the non-convex Rastrigin function (5.3) is given in Fig. 2.1. Every local minimum of the potential function leads to a peak in the probability density, indicating that the particle is more likely to be found in these lower potential states.

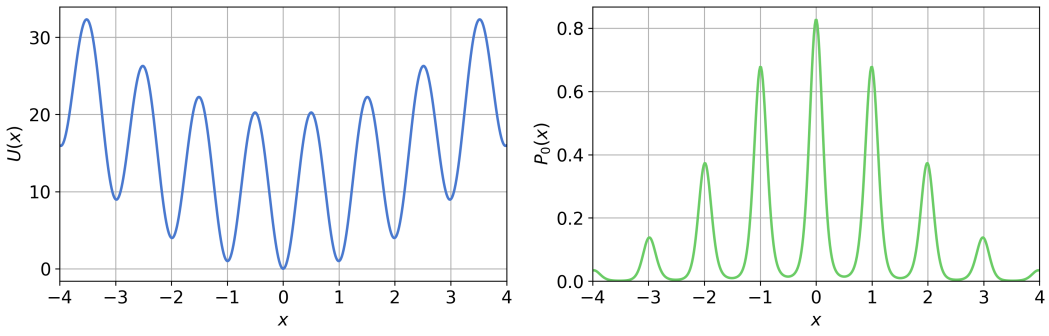

Figure 2.1: Rastrigin function (see (5.3)) and corresponding Boltzmann distribution ($T = 5$)

2.1.3 Convergence to equilibrium

The entropy of the probability distribution P is defined as

$$H(P) = -\mathbb{E}_P[\log P] = -\int_{-\infty}^{+\infty} P(x) \log P(x) dx \tag{2.7}$$

The Helmholtz free energy is given by

$$F = E - TH = \int_{-\infty}^{+\infty} P(x)(U(x) + T \log P(x)) dx = \mathbb{E}_P[U + T \log P] \tag{2.8}$$

To show the convergence of the overdamped Langevin dynamics (2.2) to the unique stationary distribution (2.6), we consider the Kullback-Leibler (KL) divergence D_{KL} of the current probability distribution $P(x,t)$ with respect to the stationary distribution $P_0(x)$:

$$D_{\mathrm{KL}}(P \,\|\, P_0) = \int_{-\infty}^{+\infty} P(x) \log \frac{P(x)}{P_0(x)} dx \tag{2.9}$$

which is a measure of difference between probability distributions, also called the relative entropy. The KL divergence is non-negative and minimized when $P = P_0$, where it is zero. We notice that

$$\begin{aligned} D_{\mathrm{KL}}(P \,\|\, P_0) &= \mathbb{E}_P \left[\log \frac{P}{P_0} \right] \\ &= \mathbb{E}_P \left[\log P - \log P_0 \right] \\ &= \mathbb{E}_P \left[\log P + \frac{U}{T} + \log Z \right] \\ &= \frac{F}{T} + \log Z \end{aligned}$$

The time derivative of free energy F is such that

$$\begin{aligned} \frac{dF}{dt} &= \frac{d}{dt} \int_{-\infty}^{+\infty} P(U + T \log P)\, dx \\ &= \int_{-\infty}^{+\infty} \frac{\partial P}{\partial t} (U + T \log P)\, dx + \int_{-\infty}^{+\infty} P \left(\frac{\partial U}{\partial t} + T \frac{\partial}{\partial t} \log P \right) dx \end{aligned}$$

The second term is null, since $\frac{\partial U}{\partial t} = 0$ and

$$\int_{-\infty}^{+\infty} P \frac{\partial}{\partial t} \log P \, dx = \int_{-\infty}^{+\infty} P \frac{1}{P} \frac{\partial P}{\partial t} \, dx = \int_{-\infty}^{+\infty} \frac{\partial P}{\partial t} \, dx = \frac{\partial}{\partial t} \int_{-\infty}^{+\infty} P \, dx = 0$$

The first term is analysed by expressing $\partial_t P$ from the Fokker-Planck equation (2.4) and using Einstein's relation (2.3):

$$\begin{aligned}\frac{\partial P}{\partial t} &= \frac{\partial}{\partial x}\left(\frac{1}{\mu}P\frac{\partial U}{\partial x}\right) + D\frac{\partial^2 P}{\partial x^2} \\ &= \frac{\partial}{\partial x}\left(\frac{1}{\mu}P\frac{\partial U}{\partial x} + D\frac{\partial P}{\partial x}\right) \\ &= \frac{\partial}{\partial x}\left(\frac{1}{\mu}P\frac{\partial}{\partial x}\left(U + T\log P\right)\right)\end{aligned}$$

Then, the derivative of free energy F becomes

$$\frac{dF}{dt} = \int_{-\infty}^{+\infty} \frac{\partial}{\partial x}\left(\frac{1}{\mu}P\frac{\partial}{\partial x}\left(U + T\log P\right)\right)\left(U + T\log P\right) dx$$

By integration by parts, considering $P(x) \xrightarrow[x\to\pm\infty]{} 0$, we obtain

$$\begin{aligned}\frac{dF}{dt} &= -\frac{1}{\mu}\int_{-\infty}^{+\infty} P\frac{\partial}{\partial x}\left(U + T\log P\right)\frac{\partial}{\partial x}\left(U + T\log P\right)\, dx \\ &= -\frac{1}{\mu}\int_{-\infty}^{+\infty} P\left(\frac{\partial}{\partial x}\left(U + T\log P\right)\right)^2 dx \\ &= -\frac{1}{\mu}\mathbb{E}_P\left[\left(\frac{\partial}{\partial x}\left(U + T\log P\right)\right)^2\right] \leq 0\end{aligned}$$

Therefore, the KL divergence decreases with time:

$$\frac{dD_{\mathrm{KL}}}{dt} = \frac{1}{T}\frac{dF}{dt} \leq 0$$

The KL divergence D_{KL} serves as a Lyapunov functional of the Fokker-Planck equation (2.4), decreasing under the Langevin dynamics until reaching $D_{\mathrm{KL}} = 0$ at equilibrium, so indeed the probability distribution P converges to Boltzmann's distribution P_0.

2.1.4 Simulated Annealing

The overdamped Langevin dynamics (2.2) theoretically converges to the equilibrium Boltzmann distribution (2.6), which suggests setting $T \to 0$ for particles to concentrate at the global minimum. However, the time required to reach equilibrium increases exponentially with $1/T$ [15]. Low temperature diminishes the impact

of random Brownian motion, which prevents particles from leaving local minima, trapping them for a long time.

Simulated annealing is an optimization technique inspired by the annealing process in metallurgy, in which a metal is heated and then slowly cooled, resulting in structural changes that reduce internal energy. The method was proposed by Kirkpatrick et al [16] and Cerny [17], and involves first "melting" the system to a high temperature and then slowly lowering it according to a certain annealing schedule until no changes occur, eventually "freezing" the system at a lower energy state, which approximates the global optimum of the objective function. The original simulated annealing method is an adaptation of the Metropolis-Hastings algorithm [18], which generates a Markov chain with a Boltzmann stationary distribution at a fixed temperature.

Simulated annealing is extended to any algorithm that samples the Boltzmann distribution, including continuous systems, such as the overdamped Langevin dynamics [15]. In this case, the dynamics are described by the following equation

$$\dot{x} = -\frac{1}{\mu}\frac{\partial U}{\partial x} + \sigma(t)\xi(t) \tag{2.10}$$

where ξ is a random Gaussian noise with zero mean and unit variance

$$\langle \xi(t)\xi(t') \rangle = \delta(t - t')$$

Extensive work has been dedicated to analysing the choice of the cooling schedule $\sigma(t)$. In [19] and [20] it is shown that the cooling schedule $\sigma(t) = A/\sqrt{\log t}$ for some $A > 0$ guarantees convergence to the solution under weak conditions on $U(x)$. In [21], the same result is confirmed for the more general case of multiplicative noise $\sigma(x,t) = a(x)b(t)$.

2.2 Lévy processes

2.2.1 Lévy-stable distribution

The symmetric Lévy-stable distribution with stability index α has the characteristic function

$$\varphi(k) = \exp\left(-c^\alpha |k|^\alpha\right) \tag{2.11}$$

where c is a scale parameter and $\alpha \in (0, 2]$ is the stability index. Plots of probability density functions (PDFs) of Lévy-stable distributions with different stability indices α and scale $c = 1$ are presented in Fig. 2.2.

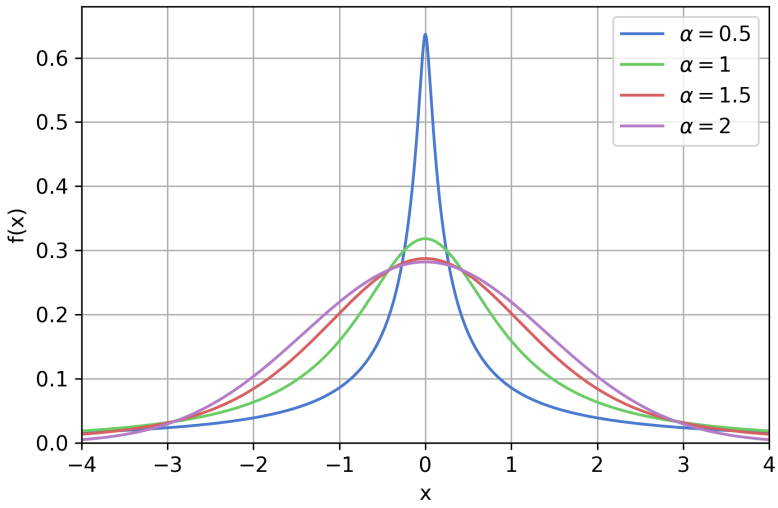

Figure 2.2: Lévy-stable distributions ($c = 1$)

The PDF of a Lévy-stable distribution can be expressed in terms of elementary functions only in two cases:

1. when $\alpha = 2$, the distribution becomes a Gaussian with zero mean and variance $2c^2$:
$$\varphi(k) = \exp\left(-c^2k^2\right) \Rightarrow X \sim N(0, 2c^2)$$

2. when $\alpha = 1$, the distribution is a Cauchy distribution with zero location and scale parameter c:
$$\varphi(k) = \exp\left(-c|k|\right) \Rightarrow X \sim \text{Cauchy}(0, c)$$

2.2.2 Lévy flights

A Lévy flight is a random walk with steps made in random directions with lengths having a stable Lévy distribution. The important features of the distribution, that define the properties of Lévy flights, are the following:

1. Stability. Let X_1 and X_2 be independent random variables drawn from a stable distribution. Then, the linear combination $\alpha X_1 + \beta X_2$ with $\alpha > 0$ and $\beta > 0$ has the distribution as $aX + b$ for some constants $a > 0$ and b. This ensures that a Lévy flight, being a sum of Lévy-stable steps, remains Lévy-stable.

2. For $\alpha < 2$, the distribution is heavy-tailed, decaying as $\sim 1/x^{1+\alpha}$, causing the variance to be infinite. This allows particles to take large steps, exploring other regions.

Examples of 2D Lévy flights for different parameters α are shown in Fig. 2.3. For each stability index α, ten experiments were performed, each starting from $(0, 0)$ and taking 100 steps with the scale factor $c = 1$.

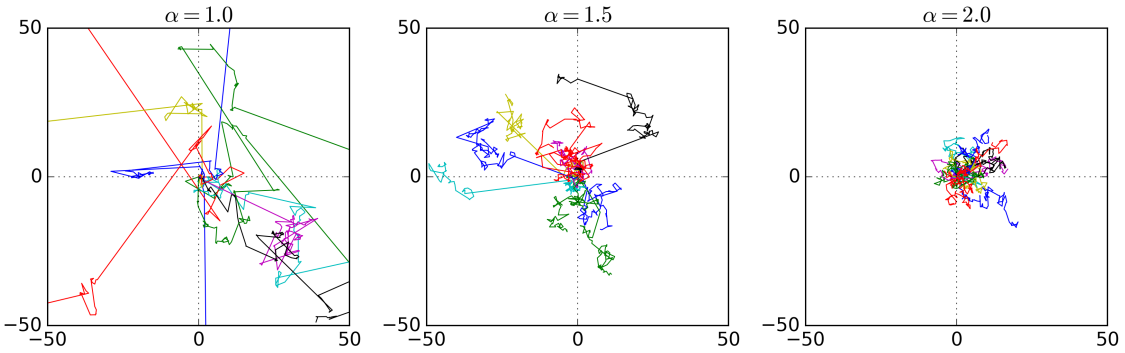

Figure 2.3: Lévy flights for different parameters α ($c = 1$)

Lévy flights exhibit a fractal-like structure composed of series of many short steps interrupted by rare long-distance jumps, reflecting the heavy tails of the underlying distribution. Bigger jumps are more common for lower values of α, as the PDF tail decreases more slowly with lower values of α. The dynamics is scale invariant, meaning that its statistical properties remain consistent across different spatial and temporal scales, allowing efficient exploration of a large search space.

Theoretical results show that Lévy flights are the optimal search strategy in terms of mean search time, given the absence of memory and that the target sites are sparsely and randomly distributed and can be revisited [22]. The Lévy foraging hypothesis predicts that biological systems have adopted Lévy flights as a search strategy, as a result of natural selection [23]. There are extensive studies supporting the hypothesis by confirming Lévy flights in nature, such as the foraging behaviour of the wandering albatross [24], the movement patterns of marine predators while locating prey [25] and the migration of bacteria [26]. However, other studies deny the hypothesis based on other data, such as [27], which analyses high-resolution data of wandering albatross flights and questions the strength of empirical evidence for Lévy flights.

2.2.3 Fractional Fokker-Planck equation

The partial differential equation governing the probability density $P(x, t)$, equivalent to the overdamped Langevin equation (2.2) with symmetric Lévy-stable noise $\eta(t)$,

is the Fractional Fokker-Planck Equation (FFPE). As derived in [28], it is given by

$$\frac{\partial}{\partial t}P(x,t) = \frac{\partial}{\partial x}\left(\frac{1}{\mu}\frac{\partial U}{\partial x}P(x,t)\right) + c^{\alpha}\frac{\partial^{\alpha}}{\partial|x|^{\alpha}}P(x,t) \tag{2.12}$$

where the operator $\frac{\partial^{\alpha}}{\partial|x|^{\alpha}}$ denotes the Riesz fractional derivative, defined via the Fourier transform $\mathcal{F}$ as

$$\mathcal{F}\left\{\frac{\partial^{\alpha}}{\partial|x|^{\alpha}}f(x)\right\}(k) = -|k|^{\alpha}\mathcal{F}\left\{f\right\}(k) \tag{2.13}$$

Since the Fourier transform is a non-local operator, the fractional derivative is likewise spatially non-local, reflecting the long-range jumps of Lévy noise [29].

The FFPE (2.12) generalizes the classical Fokker-Planck equation (2.4) to account for Lévy-stable noise. In the Gaussian limit of $\alpha = 2$, the standard equation is recovered, describing Brownian motion with diffusion coefficient $D = c^2$, leading to thermal equilibrium described by the Boltzmann distribution. In the case of $\alpha \in (0, 2)$, the dynamics becomes anomalous, deviating from thermal equilibrium. The stationary distribution cannot be analytically obtained for an arbitrary potential $U(x)$. For a harmonic potential, the stationary distribution is a non-Boltzmann distribution with no finite variance [30].

2.2.4 Anomalous diffusion

In the absence of the force field of the potential $U(x)$, the FFPE (2.12) describes anomalous diffusion

$$\frac{\partial}{\partial t}P(x,t) = c^{\alpha}\frac{\partial^{\alpha}}{\partial|x|^{\alpha}}P(x,t) \tag{2.14}$$

which is solved in closed form. As shown in [30], the asymptotic behaviour of the solution at $|x| \to \infty$ decays as a power law:

$$P(x,t) \sim \frac{c^{\alpha}t}{|x|^{1+\alpha}} \tag{2.15}$$

This reflects that large displacements, caused by large jumps, dominate the long-term behaviour.

Although the second moment diverges, a scale relation for the mean displacement is obtained by performing scale analysis [31], suggesting that

$$\langle \Delta x^2(t) \rangle \sim t^{2/\alpha} \tag{2.16}$$

which for $\alpha < 2$ describes superdiffusion, spreading faster than Brownian motion.

Chapter 3

Mathematical Model

This chapter describes the mathematical model of the swarm-based optimization algorithm, which is the main object of study of this work. It provides the governing equations for particle motion and friction dynamics, explores different types of random forces, and analyses the key system properties.

3.1 Model Description

A swarm consists of N particles. A particle is identified within the swarm by an index $i \in \mathbb{N}$ and characterized by its position $\boldsymbol{x}_i \in \mathbb{R}^d$ and friction coefficient $\mu_i > 0$. The set $\mathcal{A}$ contains indices of active particles in the swarm. Each active particle obeys the d-dimensional overdamped Langevin equation (2.2):

$$\dot{\boldsymbol{x}}_i = -\frac{1}{\mu_i}\left(\nabla U(\boldsymbol{x}_i) + \boldsymbol{\omega}_i(t)\right), \quad i \in \mathcal{A} \tag{3.1}$$

where $\boldsymbol{\omega}_i$ is a random force. The friction coefficients of active particles evolve in time according to

$$\dot{\mu}_i = -f(\mu_i) \sum_{j \in \mathcal{A}} \frac{U(\boldsymbol{x}_i) - U(\boldsymbol{x}_j)}{U_{\max} - U_{\min}}, \quad i \in \mathcal{A} \tag{3.2}$$

where $U_{\max} = \max_{i \in \mathcal{A}} U(\boldsymbol{x}_i)$, $U_{\min} = \min_{i \in \mathcal{A}} U(\boldsymbol{x}_i)$ and $f(\mu) > 0$ is a positive function, termed friction response rate. In the case of equal fitness of all particles $U_{\max} = U_{\min}$, friction is not updated: $\dot{\mu}_i = 0$.

Friction coefficients serve as a platform for communication between particles, exchanging information about performance. The dynamics (3.2) ensure that when the particle i is in a more optimal position than the particle j (that is, $U(\boldsymbol{x}_i) < U(\boldsymbol{x}_j)$), its friction coefficient μ_i increases, while that of the particle j decreases.

This ensures that the better-performing particles "stick" to their positions, moving slower than the worse-performing particles, which move faster, exploring the area.

The difference in potential values between a pair of particles $U(\boldsymbol{x}_i) - U(\boldsymbol{x}_j)$ is normalized against the range of all particle potentials $U_{\max} - U_{\min}$. This ensures that the scaling of the potential does not affect the dynamics and provides acceleration of friction transfer when $U_{\max}$ is close to $U_{\min}$, which happens when all particles have similar fitness.

When the friction coefficient of a particle falls below a fixed threshold $\delta\mu$, the particle is labelled inactive and is no longer updated. Thus, the set of active particles $\mathcal{A}$ at any time is given by

$$\mathcal{A} = \{i \in \{1, \dots, N\} \mid \mu_i > \delta\mu\} \tag{3.3}$$

The friction response rate function $f(\mu)$ is selected as the following decreasing function:

$$f(\mu) = \frac{\sqrt{\mu}}{1 + \sqrt{\mu}} \tag{3.4}$$

as shown in Fig. 3.1, assuming μ to be positive. This choice ensures that low μ values experience a slower rate of reduction, preventing sudden elimination of poor-performing particles.

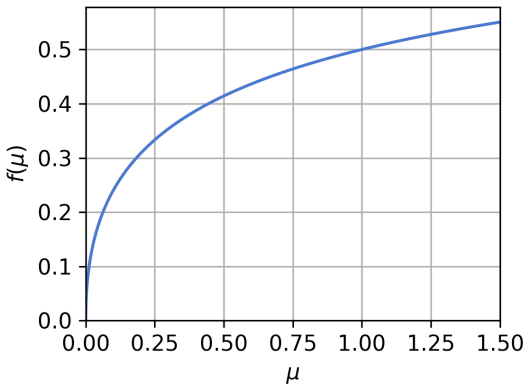

Figure 3.1: Friction response function

To conclude, the mathematical model is described by the system:

$$\begin{cases} \dot{\boldsymbol{x}}_i = \dfrac{1}{\mu_i}\left(-\nabla U(\boldsymbol{x}_i) + \boldsymbol{\omega}_i\right), & i \in \mathcal{A} \\ \dot{\mu}_i = -f(\mu_i) \displaystyle\sum_{j \in \mathcal{A}} \frac{U(\boldsymbol{x}_i) - U(\boldsymbol{x}_j)}{U_{\max} - U_{\min}}, & U_{\max} \neq U_{\min}, \quad i \in \mathcal{A} \\ \dot{\mu}_i = 0, & U_{\max} = U_{\min}, \quad i \in \mathcal{A} \\ U_{\max} = \max\limits_{i \in \mathcal{A}} U(\boldsymbol{x}_i), \quad U_{\min} = \min\limits_{i \in \mathcal{A}} U(\boldsymbol{x}_i) \\ \mathcal{A} = \{i \in \{1, \dots, N\} \mid \mu_i > \delta\mu\} \end{cases} \tag{3.5}$$

3.2 Random Force

3.2.1 No Random Force

In the absence of a random force, the system (3.5) is deterministic, fully governed by the initial positions of the particles $\boldsymbol{x}_i^{(0)}$. Each particle follows the direction of the negative gradient, descending to a local minimum. The equilibrium of (3.1) for active particles is reached at stationary points:

$$|\nabla U(\boldsymbol{x}_i)| = 0, \quad \forall i \in \mathcal{A} \tag{3.6}$$

While the equilibrium of (3.2) is reached when all active particles have the same fitness:

$$U(\boldsymbol{x}_i) = U(\boldsymbol{x}_j), \quad \forall i, j \in \mathcal{A} \tag{3.7}$$

The total energy is chosen as a Lyapunov function:

$$V = \sum_{i \in \mathcal{A}} U(\boldsymbol{x}_i) \tag{3.8}$$

Its derivative along the system trajectories is given by

$$\frac{dV}{dt} = \sum_{i \in \mathcal{A}} \nabla U(\boldsymbol{x}_i) \dot{\boldsymbol{x}}_i = -\sum_{i \in \mathcal{A}} \frac{1}{\mu_i} |\nabla U(\boldsymbol{x}_i)|^2 \leq 0$$

The value is non-positive, vanishing only when all $\boldsymbol{x}_i$ are stationary points of U, indicating that the total energy of the system decreases monotonically and the system converges to the equilibrium (3.6).

As a result, in the equilibrium all particles are either eliminated or reach the same minimum. However, it is not guaranteed that the particles settle in the global minimum, since it may never be discovered. In fact, the global minimum is found if and only if at least one particle is on its slope.

Communication between particles has an insignificant role, making better-performing agents travel slower, and worse-performing agents travel faster, without changing the travel direction. The interaction can be viewed as a selective pressure for worse-performing agents to travel faster to avoid elimination. The case with no random force performs especially poorly with a small number of particles and a large search space since no new regions are explored.

3.2.2 Gaussian Random Force

The Gaussian random force $\boldsymbol{\omega}_i$ in (3.5) is given by

$$\boldsymbol{\omega}_i = \mathbb{C}\boldsymbol{r}\xi_i \tag{3.9}$$

where

- $\mathbb{C} = \text{diag}\{\boldsymbol{c}\} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with elements of a constant vector $\boldsymbol{c} \in \mathbb{R}^d$ on its diagonal, representing the scale in each direction. This allows the method to be invariant under rescaling of variables by modifying the corresponding elements in $\boldsymbol{c}$. For symmetric problems, it is natural to choose $\boldsymbol{c}$ with identical elements. Using identical elements also serves as a way to reduce the number of tuning parameters.
- $\boldsymbol{r} \in \mathbb{R}^d$ is a uniformly sampled random direction vector, lying on the unit d-dimensional sphere.
- $\xi_i \sim \mathcal{N}(0, \sigma^2(\mu_i))$ is a Gaussian random variable with zero mean and variance $\sigma^2(\mu)$, which is a decreasing function of μ.

The function $\sigma^2(\mu)$ is chosen to decrease with a higher friction coefficient μ, so that the better-performing particles are less affected by the random force, staying in the vicinity of their location, while the worse-performing particles are more inclined to leave their local minima and explore the space, as shown in Fig. 3.2.

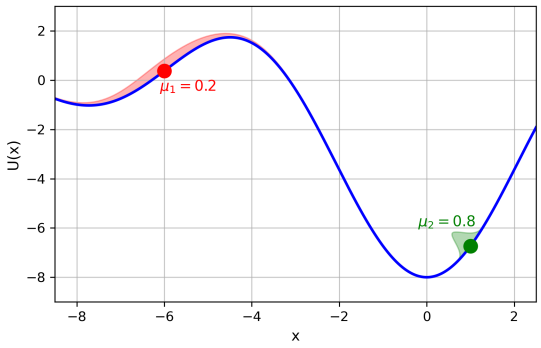

Figure 3.2: Gaussian random force

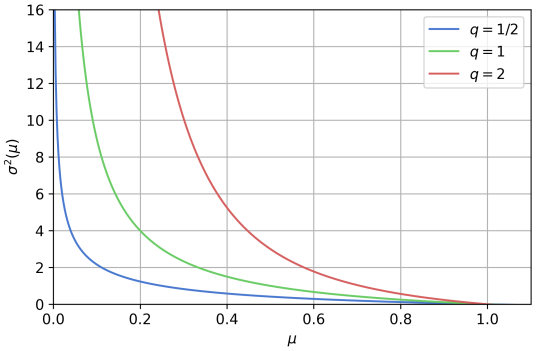

Figure 3.3: Variance functions $\sigma^2(\mu)$

The functions $\sigma^2(\mu)$ that we consider have the following form

$$\sigma^2(\mu) = \begin{cases} \frac{1}{\mu^q} - 1, & \mu \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

where $q > 0$ and we consider the values $q = 0.5, 1, 2$, shown in Fig. 3.3. The parameter q controls the decrease rate, lower q leads to more rapid decrease. Starting from the threshold value of $\mu = 1$, the variance and thus the force becomes null, meaning that these particles proceed solely according to the gradient descent. However, the force may be reapplied to these particles, if their friction coefficient μ falls below the threshold again, as a result of other agents finding a better solution. The variance function $\sigma^2(\mu)$ captures only the shape of the desired dynamics, while the scale is defined by the scale matrix $\mathbb{C}$, taking into account the initial value $\sigma^2(\mu_0)$.

3.2.3 Lévy-stable Random Force

Analogously to the Gaussian random force (3.9), the Lévy-stable random force is defined as

$$\boldsymbol{\omega}_i = \mathbb{C}\boldsymbol{r}\eta_i \tag{3.11}$$

where $\eta_i \sim S_\alpha(1)$ is a random variable drawn from the symmetric Lévy-stable distribution with stability index α and unit scale factor. We do not introduce an explicit relationship between the scale factor and the friction coefficient μ, since large jumps are considerably more common in Lévy flights than in Brownian motion, enabling agents to escape local minima. Moreover, the final step taken by an agent is already inversely proportional to μ, as given in (3.5). Unlike in Brownian motion dynamics, particles are less likely to accumulate in the global minimum, even after reaching it, making it crucial to save the best-found solution.

3.3 Properties

3.3.1 Communication

The importance of communication, in the form of updating friction coefficients based on agents' performance, is demonstrated by running the simulation without any communication. In this case, the dynamics is that of the overdamped Langevin, converging to the equilibrium Boltzmann distribution (2.6). As an example, we run the simulation with $N = 500$ particles optimizing the 1D Rastrigin function with Gaussian noise and $q = 1$. The particle distribution at different timesteps T and the corresponding Boltzmann distribution are presented in Fig. 3.4. It is observed that the particle distribution approximates the theoretical equilibrium distribution, converging to it over time.

The same experiment was run with communication, that is, with friction updates. The particle distribution is presented in Fig. 3.5. In this case, the particles

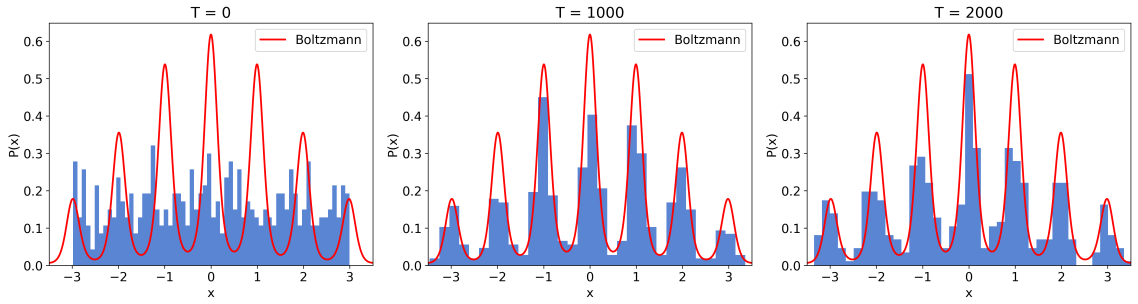

Figure 3.4: Particle position distribution without communication for 1D Rastrigin function ($N = 500$, Gaussian noise with $q = 1$)

accumulate densely at the function minima. As worse-performing particles either move to a better location or get discarded, the swarm becomes more focused on the best basins. Better-performing particles stay in the vicinity of their current locations, since their friction increases. It is observed that the worse basins of attraction are eventually abandoned. In this experiment, running the simulation for a longer time would result in all active particles concentrating in the global minimum. The Boltzmann distribution is shown for reference and does not represent the equilibrium distribution in this case.

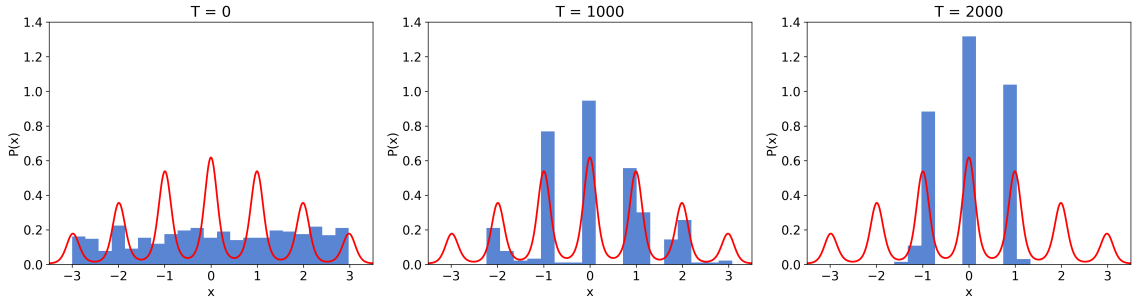

Figure 3.5: Particle position distribution with communication for 1D Rastrigin function ($N = 500$, Gaussian noise with $q = 1$)

As given in (3.7), the system converges to a consensus state, where particles either reach a potential with the same fitness, or become inactive. However, convergence to the global minimum is not guaranteed, since it may never be discovered by any particle, and the system may settle in a local minimum instead. Unlike in the case with no random force, the initialization of particle positions is less crucial: even if no particle is initialized on the slope of the global minimum, it may still be discovered by movement, influenced by the random force.

3.3.2 Emergent Annealing

It is possible to reformulate the evolution of friction coefficients (3.2) in the following way:

$$\begin{aligned}\dot{\mu}_i &= -f(\mu_i)\sum_{j\in\mathcal{A}}\frac{U(\boldsymbol{x}_i)-U(\boldsymbol{x}_j)}{U_{\max}-U_{\min}}\\ &= f(\mu_i)\,\frac{\sum_{j\in\mathcal{A}}U(\boldsymbol{x}_j)-\sum_{j\in\mathcal{A}}U(\boldsymbol{x}_i)}{U_{\max}-U_{\min}}\\ &= f(\mu_i)|\mathcal{A}|\,\frac{\overline{U}-U(\boldsymbol{x}_i)}{U_{\max}-U_{\min}}\end{aligned}$$

where $\overline{U}=\sum_{j\in\mathcal{A}}U(\boldsymbol{x}_j)/|\mathcal{A}|$ is the average fitness of active particles. Since $\sum_{i\in\mathcal{A}}(\overline{U}-U(\boldsymbol{x}_i))=0$, we notice that

$$\sum_{i\in\mathcal{A}}\frac{\dot{\mu}_i}{f(\mu_i)}=0$$

Given that inactive particles are not updated, their contribution is zero. Thus, the dynamics have a conserved quantity:

$$M:=\sum_{i=1}^{N}F(\mu_i)=\text{const},\quad\text{where}\quad F(\mu)=\int\frac{d\mu}{f(\mu)}\tag{3.12}$$

Assuming that $F(\mu)\to 0$ as $\mu\to 0$, the contribution of inactive particles with $\mu<\delta\mu$ into M is negligible. Therefore, the conserved quantity is approximated by restricting the sum only to the active particles:

$$M\approx\sum_{i\in\mathcal{A}}F(\mu_i)$$

For our choice of $f(\mu)$, given by (3.4), the integral is

$$F(\mu)=\int\frac{\sqrt{\mu}+1}{\sqrt{\mu}}\,d\mu=\mu+2\sqrt{\mu}+C$$

Assuming $C=0$, indeed $F(\mu)\to 0$ as $\mu\to 0$. Thus, the conserved quantity is

$$M=\sum_{i\in\mathcal{A}}(\mu_i+2\sqrt{\mu_i})$$

neglecting the maximum possible error of $(N-|\mathcal{A}|)F(\delta\mu)$, corresponding to the contribution of inactive particles.

Since the number of active particles $|\mathcal{A}|$ does not increase with time and M is conserved, the average friction coefficient $\bar{\mu}$ of active particles is a non-decreasing

function. Indeed, when a particle becomes inactive, its contribution to M is already negligible: $F(\mu_i) \leq F(\delta\mu) \approx 0$, so the sum of the contributions of the remaining particles must increase to compensate, keeping M constant.

As $\bar{\mu}$ increases over time, particles take shorter steps and consequently move slower. In the case of Gaussian noise, this also reduces the magnitude of the random force according to (3.10). This behaviour is analogous to the simulated annealing (SA) approach, where the system starts at a high temperature that is gradually lowered to aid convergence. In our model, $1/\bar{\mu}$ plays a similar role as temperature. However, unlike in SA, $\bar{\mu}$ is not explicitly controlled, but its increase emerges naturally from the system dynamics.

The theoretical result of an increasing average $\bar{\mu}$ is supported by experimental results, showing an approximately linear growth. Fig. 3.6 shows the evolution of $\bar{\mu}$ in a simulation with $N = 500$ particles optimizing the 1D Rastrigin function under Gaussian noise with $q = 1$.

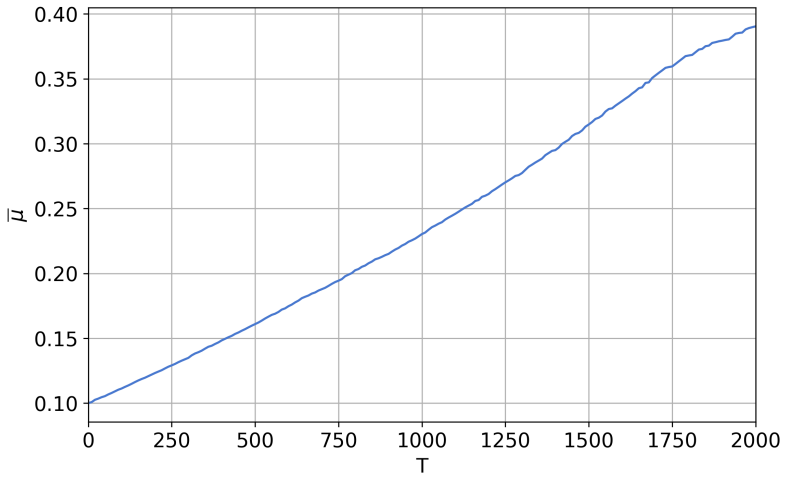

Figure 3.6: Average friction coefficient $\bar{\mu}$ ($N = 500$, Gaussian noise with $q = 1$)

Chapter 4

Algorithm

4.1 Computational Scheme

The system (3.5) is solved using a time discretization scheme with a fixed timestep τ. Stochastic differential equations for $\mathbf{x}_i$ are solved using the Euler-Maruyama method [32], while the ordinary differential equations for μ_i are solved using the explicit Euler method. The update rule for each timestep $n+1$ is given by:

$$\begin{cases} \mathbf{x}_i^{(n+1)} = \mathbf{x}_i^{(n)} + \dfrac{1}{\mu_i^{(n)}} \left(-\nabla U(\mathbf{x}_i^{(n)})\tau + \Delta\boldsymbol{\omega}_i^{(n)} \right) & i \in \mathcal{A}^{(n)} \\ \mu_i^{(n+1)} = \mu_i^{(n)} - \tau |\mathcal{A}^{(n)}| f(\mu_i^{(n)}) \dfrac{U(\mathbf{x}_i^{(n+1)}) - \overline{U}^{(n+1)}}{U_{\max}^{(n+1)} - U_{\min}^{(n+1)}} & |U_{\max} - U_{\min}| > \varepsilon, \quad i \in \mathcal{A}^{(n)} \\ \mu_i^{(n+1)} = 0 & |U_{\max} - U_{\min}| \leq \varepsilon, \quad i \in \mathcal{A}^{(n)} \\ \overline{U}^{(n+1)} = \dfrac{1}{|\mathcal{A}^{(n)}|} \displaystyle\sum_{i \in \mathcal{A}^{(n)}} U(\mathbf{x}_i^{(n+1)}) \\ U_{\min}^{(n+1)} = \min\limits_{i \in \mathcal{A}^{(n)}} U(\mathbf{x}_i^{(n+1)}), \quad U_{\max}^{(n+1)} = \max\limits_{i \in \mathcal{A}^{(n)}} U(\mathbf{x}_i^{(n+1)}) \end{cases} \tag{4.1}$$

where the same notation as in the system (3.5) is used. To avoid numerical instability, a small threshold parameter ε is introduced as a numerical zero. The random force term $\Delta\boldsymbol{\omega}_i^{(n)}$ is sampled at every timestep, according to the Euler-Maruyama scheme as

$$\Delta\boldsymbol{\omega}_i^{(n)} \sim \begin{cases} \sqrt{\tau}\,\mathcal{N}(0, c^2\sigma^2), & \text{if Gaussian} \\ \tau^{1/\alpha}\, S_\alpha(0, c, 0), & \text{if Lévy-stable} \end{cases} \tag{4.2}$$

Here, $S_\alpha(0, c, 0)$ denotes the symmetric Lévy-stable distribution centered at zero with scale c.

After each update of friction coefficients, if the particle i has μ_i lower than a set threshold $\delta\mu$, it is removed, discarding the equations for $\mathbf{x}_i$ and μ_i in (4.1):

$$\mathcal{A}^{(n+1)} = \left\{ i \in \mathcal{A}^{(n)} : \mu_i^{(n+1)} > \delta\mu \right\} \tag{4.3}$$

At each timestep n, the best obtained solution $\boldsymbol{x}^*$ with fitness U^* is tracked:

$$U^* = \min_{i \in \mathcal{A}^{(m)}} U(\mathbf{x}_i^{(m)}), \quad \boldsymbol{x}^* = \arg\min_{\mathbf{x}_i^{(m)}} U(\mathbf{x}_i^{(m)}), \quad m \leq n \tag{4.4}$$

The full state of the system at timestep n is described by the variables:

- $\mathbf{x}_i^{(n)}$: position of particle i
- $\mu_i^{(n)}$: friction coefficient of particle i
- $\mathcal{A}^{(n)}$: set of active particle indices
- $\boldsymbol{x}^*$: best solution obtained so far
- U^*: corresponding best function value

4.2 Function Optimizer

The pseudocode for the function optimizer algorithm is given in Algorithm 1 and described below.

Algorithm 1 Particle swarm optimization with friction transfer

Require: Parameters $N, T, T_{early}, \tau, \delta\mu, \omega$
  $\mathcal{A} \leftarrow \{1, \dots, N\}$
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x} \sim \mathcal{U}(S), \quad \mu_i \leftarrow \mu_0 \quad$ for $i \in \mathcal{A}$
  $\boldsymbol{x}^* \leftarrow \arg\min_{i \in \mathcal{A}} U(\boldsymbol{x}), \quad U^* \leftarrow U(\boldsymbol{x}^*), \quad n^* \leftarrow 0$
  for $n = 0, \dots, T$ **do**
    for $i \in \mathcal{A}$ **do**
      Generate $\Delta\boldsymbol{\omega}_i$
      $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \frac{1}{\mu_i}(-\tau \nabla U(\boldsymbol{x}_i) + \Delta\boldsymbol{\omega}_i)$
      $U_i \leftarrow U(\boldsymbol{x}_i)$
    end for
    $\overline{U} \leftarrow \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} U_i, \quad U_{\min} \leftarrow \min_{i \in \mathcal{A}} U_i, \quad U_{\max} \leftarrow \max_{i \in \mathcal{A}} U_i$
    if $U_{\min} < U^*$ **then**
      $i_n^* \leftarrow \arg\min_{i \in \mathcal{A}} U_i, \quad \boldsymbol{x}^* \leftarrow \boldsymbol{x}_{i_n^*}, \quad U^* \leftarrow U_{\min}, \quad n^* \leftarrow n$
    end if
    if $n - n^* \geq T_{early}$ **then break**
    $\mu_i \leftarrow \mu_i - \tau |\mathcal{A}| f(\mu_i)(U_i - \overline{U})/(U_{\max} - U_{\min}) \quad$ for $i \in \mathcal{A}$
    $\mathcal{A} \leftarrow \{i \in \mathcal{A} : \mu_i > \delta\mu\}$
  end for
  return $\boldsymbol{x}^*$

Initialization

At initialization, each of the swarms N particles is active. The initial particle positions are uniformly sampled from a given search space $\mathcal{S}$. Each particle is assigned the same friction coefficient of μ_0.

$$\begin{cases} \mathbf{x}_i^{(0)} \sim \mathcal{U}(\mathcal{S}) & i \in \mathcal{A}^{(0)} \\ \mu_i^{(0)} = \mu_0 & i \in \mathcal{A}^{(0)} \\ \mathcal{A}^{(0)} = \{1, \dots, N\} \end{cases} \tag{4.5}$$

Update rule

The simulation is run for a given number of steps T with timestep τ. At each timestep, the state is updated according to the scheme (4.1).

Stopping criteria

The simulation is run for a maximum of T steps with an early stopping condition: if the best found solution does not improve after T_{early} steps. In all experiments, the values $T = 1000$ and $T_{early} = 200$ are used unless otherwise stated. The early stopping condition is applied in most cases.

Parameters

The list of algorithm parameters and their typical values are presented in Table 4.1. The random force is defined by two parameters, which require tuning: the scale c and the parameter q connecting friction and variance for Gaussian noise, and the scale c and the stability index α for Lévy noise.

Parameter	**Description**	**Typical values**
N	number of particles	$10 \sim 10^3$
T	number of timesteps	$10^3 \sim 10^4$
τ	timestep	$10^{-4} \sim 10^{-2}$
T_{early}	early stopping threshold	$10^2 \sim 10^3$
μ_0	initial friction coefficient	10^{-1}
$\delta\mu$	minimum friction coefficient threshold	10^{-4}
ω	type of noise	Gauss, Lévy
Gaussian noise parameters		
q	parameter connecting friction and variance	$\{0.5,\ 1.0,\ 2.0\}$
c	noise scale	$\sim \sqrt{\frac{\tau}{d}} \max \lvert\nabla U(x)\rvert$
Lévy noise parameters		
α	stability index	$\{1.0,\ 1.5,\ 2.0\}$
c	noise scale	$\sim \sqrt{\frac{\tau}{d}} \max \lvert\nabla U(x)\rvert$

Table 4.1: Algorithm parameters

4.3 GPU Implementation

Algorithm 1 is parallelized for execution on a graphics processing unit (GPU), executing N concurrent threads, one per particle. The implementation is developed using Rust and the WebGPU Shading Language (WGSL), using the `wgpu` library,

a cross-platform graphics API based on the WebGPU standard [33]. This technical approach enables native execution on Vulkan, as used in this work, and supports browser-based execution on WebGPU after compilation to WebAssembly (WASM). The source code is published in a repository[1].

4.3.1 Storage layout

The structure of Algorithm 1 is well-suited for parallel implementation within each timestep. For a grid search with cardinality $|G|$ and number of samples S, a total of $|G|S$ workgroups are dispatched. A workgroup is associated with a single simulation and consists of N threads, each corresponding to a single particle. Thus, the total number of threads becomes $|G|SN$, significantly speeding up computation.

Each thread independently manages the particle's position $\mathbf{x}_i$ and friction coefficient μ_i, storing these values in private memory with efficient access. If a particle becomes inactive, its corresponding thread skips all computations. The function value $U(\mathbf{x}_i)$ is shared among threads within a workgroup and stored in the workgroup memory. The parallel version requires a workgroup memory synchronization barrier inserted after updating the particle position and the corresponding function value, since these values are used subsequently in friction coefficient updates.

The aggregate values of average function value $\overline{U}$, value $g = |A|/(U_{\max} - U_{\min})$, the best position $\mathbf{x}^*$, the best function value U^* and the best iteration n^* are stored in the workgroup memory and managed by a designated single thread, which requires another workgroup memory synchronization barrier after their calculation. The designated thread also checks the stopping condition. When the simulation is stopped, the thread writes the values U^*, $\mathbf{x}^*$ and n^* to the storage buffer, which is the algorithm's output, accessible by the central processing unit (CPU) after copying. The storage locations of variables are summarized in Table 4.2.

Storage Location	**Variables**
Private memory	$\boldsymbol{x}_i$, μ_i
Workgroup memory	$U(\boldsymbol{x}_i)$, $\overline{U}$, g, $\boldsymbol{x}^*$, U^*, n^*
Storage buffers	$\boldsymbol{x}^*$, U^*, n^*

Table 4.2: Storage locations of variables for GPU algorithm for each simulation

[1]Source code available at `https://github.com/IvanLudvig/swarm_wgpu`

4.3.2 Random variable generation

Due to the deterministic nature of GPU programming, random variables must be generated manually. Uniform random variables are obtained via a hash-based pseudo-random stateless algorithm based on MurmurHash3 [34], which is chosen for its high speed, excellent statistical properties and strong avalanche effect [35]. Including the thread index in the seed is vital to ensure minimal correlation between threads. Other desired distributions are generated by transforming uniform random variables.

Gaussian random variables with mean μ and variance σ^2 are generated using the Box-Muller transform, which maps pairs of independent uniform random variables to standard normal variables (Algorithm 2) [36].

Algorithm 2 Box-Muller transform for Gaussian r.v.

Require: Parameters μ, σ
  Generate $u_1, u_2 \sim \mathcal{U}(0, 1)$
  $R \leftarrow \sqrt{-2 \ln u_1}$
  $\theta \leftarrow 2\pi u_2$
  $x_1 \leftarrow R\cos(\theta), \quad x_1 \leftarrow \sigma x_1 + \mu$
  $x_2 \leftarrow R\sin(\theta), \quad x_2 \leftarrow \sigma x_2 + \mu$
  return x_1, x_2

For computational generation of symmetric Lévy-stable distributed random variables, the algorithm proposed by Chambers, Mallows and Stuck (CMS) [37] is used, which simulates any stable random variable in an efficient way [38]. Algorithm 3 generates a random variable drawn from the Lévy-stable distribution with stability parameter α and scale parameter c. In the case of $\alpha = 1$, the algorithm requires just one uniform random variable. Otherwise, an additional exponentially-distributed random variable is necessary, which is obtained by applying the inverse transform to another independently generated uniform random variable.

Algorithm 3 CMS algorithm for Lévy-stable r.v.

Require: Parameters α, c
  Generate $u \sim \mathcal{U}(-\frac{\pi}{2}, \frac{\pi}{2})$
  if $\alpha \neq 1$ **then**
    Generate $w \sim \mathcal{U}(0, 1)$
    $w \leftarrow -\ln w$
    $x \leftarrow \frac{\sin(\alpha u)}{\cos^{1/\alpha}(u)} \left(\frac{\cos((1-\alpha)u))}{w} \right)^{\frac{1-\alpha}{\alpha}}$
  else
    $x \leftarrow \tan(u)$
  end if
  $x \leftarrow cx$
  return x

4.4 Application to Neural Network Optimization

Algorithm 1 is adapted for use as an optimizer in Neural network (NN) training, where the loss function serves as the potential U. Each particle i maintains an independent set of model weights $\boldsymbol{x}_i$, updated iteratively based on stochastic gradients and injected noise.

Training proceeds over T_E epochs, with each iteration dealing with a single mini-batch of data. During an iteration, the position of each particle is updated according to the scheme (4.1), computing the gradients of the loss function $\nabla U(\boldsymbol{x}_i)$ via back-propagation for each model on the same mini-batch. After weight updates, the friction coefficients are updated based on mini-batch losses, leading to selective-like behaviour that favours better-performing models.

Model weights $\boldsymbol{x}_i$ are initialized using standard initialization (Kaiming uniform). No early stopping mechanism is applied during training. The algorithm is implemented within the PyTorch library.

Chapter 5

Experiments

5.1 Optimization of Benchmark Functions

The performance of the method is assessed on a set of functions selected from the survey [39], which covers benchmark functions widely used for validating and comparing optimization algorithms.

5.1.1 Function Descriptions

Sphere

The sphere function is a simple convex function with a single global minimum at the origin. In this work, it is used solely for algorithm validation.

$$f(\mathbf{x}) = \sum_{i=1}^{d} {x_i}^2 \tag{5.1}$$

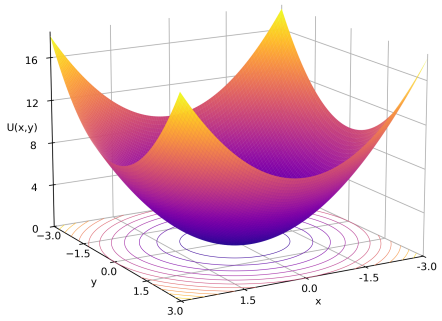

Figure 5.1: 2D Sphere function

Ackley

The Ackley function is a multimodal function with a global minimum at the origin. It is used to evaluate the exploration ability of algorithms.

$$f(\mathbf{x}) = 20 + e - 20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{d} {x_i}^2}{d}}\right) - \exp\left(\frac{\sum_{i=1}^{d}\cos(2\pi x_i)}{d}\right) \tag{5.2}$$

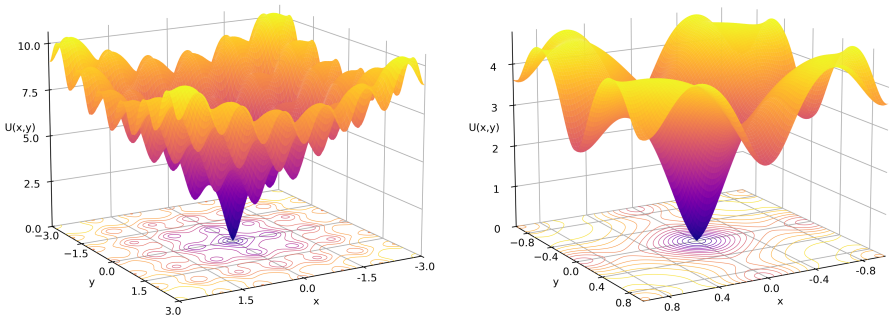

Figure 5.2: 2D Ackley function

Rastrigin

The Rastrigin function is a highly-multimodal function with a global minimum at the origin. Its landscape features frequent and deep local minima, the number of which grows exponentially fast in terms of dimensionality: on $[-3, 3]^d$ the number of local minima is 5^d.

$$f(\mathbf{x}) = \sum_{i=1}^{d} \left({x_i}^2 - 10\cos(2\pi x_i) + 10\right) \tag{5.3}$$

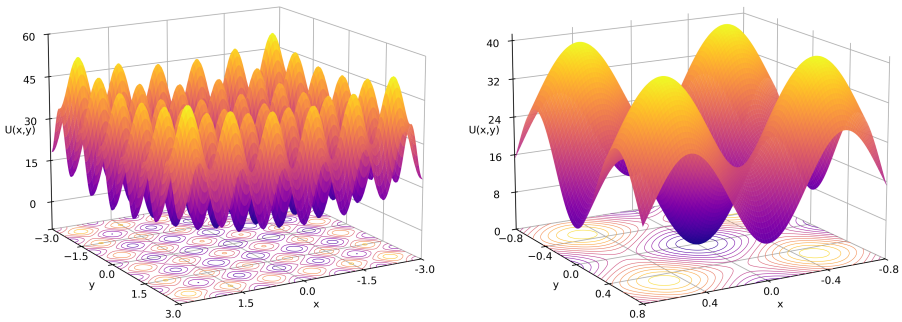

Figure 5.3: 2D Rastrigin function

Rosenbrock

The Rosenbrock function is a non-convex function with a long, narrow, parabolic-shaped valley with a global minimum at $\boldsymbol{x}_o = (1, \ldots, 1)$. It tests the algorithm's ability to navigate an ill-conditioned and deceptive landscape.

$$f(\mathbf{x}) = \sum_{i=1}^{d-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right) \tag{5.4}$$

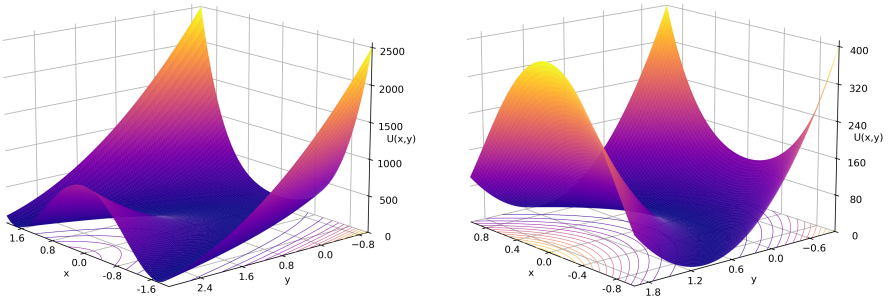

Figure 5.4: 2D Rosenbrock function

5.1.2 Evaluation Methodology

An experiment is deemed successful if the best obtained solution $\boldsymbol{x}^*$ is within the Euclidean distance $\delta r = 0.1$ of the global optimum $\boldsymbol{x}_o$:

$$\|\boldsymbol{x}^* - \boldsymbol{x}_o\| < \delta r$$

The success rate, defined as the fraction of successful experiments, is computed over 1,000 independent simulations. To optimize performance, a two-stage grid-search is applied for the scale parameter c of Gaussian and Lévy noise:

1. Coarse search over $G_1 = \{10^{-4}, \ldots, 10^2\}$ to identify the order of magnitude c_1
2. Refined search over $G_2 = \{0.25c_1, 0.5c_1, \ldots, 8c_1\}$ to determine the optimal c

Only the best success rate across all tested c values is reported. Additionally, the number of completed iterations is recorded in order to evaluate algorithm efficiency.

5.1.3 Performance

Particle positions are uniformly initialized in the hypercube $S = [-3, 3]^d$ with initial friction coefficient $\mu_0 = 0.1$. The algorithm was validated on the Sphere function (5.1), showing a 100% success rate. Success rates on the Ackley, Rastrigin and Rosenbrock functions are reported in Tables 5.1, 5.2 and 5.3 respectively.

d	$q = 0.5$	$q = 1$	$q = 2$
$N = 10$			
6	**91.5%**	**91.4%**	87.4%
12	62.5%	**77.9%**	54.8%
18	**27.1%**	**49.5%**	24.4%
$N = 50$			
6	**100.0%**	**100.0%**	**99.8%**
12	96.0%	**99.0%**	74.2%
18	53.5%	**74.5%**	27.3%

(a) Gaussian noise with q parameter

d	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$
$N = 10$			
6	**95.6%**	**96.1%**	**96.2%**
12	**91.2%**	**91.5%**	89.6%
18	85.3%	**87.3%**	82.0%
$N = 50$			
6	**100.0%**	**100.0%**	**100.0%**
12	95.4%	**99.3%**	**99.0%**
18	86.2%	**90.1%**	88.6%

(b) Lévy noise with α stability index

Table 5.1: Success rate on d-dimensional Ackley function (5.2) with N particles for different noise types ($T = 5000$, $T_{\text{early}} = 500$, $\tau = 10^{-4}$)

d	$q = 0.5$	$q = 1$	$q = 2$
$N = 10$			
2	93.4%	**94.2%**	82.3%
3	**45.2%**	44.5%	41.8%
4	**15.0%**	12.1%	12.4%
$N = 50$			
2	**100.0%**	**100.0%**	**100.0%**
3	**88.8%**	83.0%	87.3%
4	**42.9%**	40.0%	32.5%

(a) Gaussian noise with q parameter

d	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$
$N = 10$			
2	**85.0%**	84.2%	83.6%
3	44.6%	**45.7%**	**46.0%**
4	21.4%	21.2%	**22.4%**
$N = 50$			
2	**97.4%**	96.5%	96.3%
3	**63.2%**	56.3%	52.5%
4	**31.3%**	20.6%	18.2%

(b) Lévy noise with α stability index

Table 5.2: Success rate on d-dimensional Rastrigin function (5.3) with N particles for different noise types ($T = 1000$, $T_{\text{early}} = 200$, $\tau = 10^{-4}$)

The results show several consistent trends. Firstly, increasing the number of particles N leads to a significant improvement in success rates across all test cases. This confirms that a larger swarm is more effective at exploring the search space. Sec-

d	$q = 0.5$	$q = 1$	$q = 2$
$N = 10$			
3	**99.6%**	**99.5%**	**99.1%**
6	38.9%	**39.7%**	37.8%
9	30.6%	31.9%	**33.2%**
$N = 50$			
3	**100.0%**	**100.0%**	**100.0%**
6	60.0%	58.8%	**61.3%**
9	34.8%	**36.9%**	36.1%

(a) Gaussian noise with q parameter

d	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$
$N = 10$			
3	18.8%	**23.5%**	**25.1%**
6	9.1%	**11.5%**	**13.5%**
9	7.5%	**9.2%**	**9.5%**
$N = 50$			
3	53.1%	58.0%	**67.1%**
6	27.6%	36.0%	**43.4%**
9	22.7%	30.1%	**31.2%**

(b) Lévy noise with α stability index

Table 5.3: Success rate on d-dimensional Rosenbrock function (5.4) with N particles for different noise types ($T = 1000$, $T_{\text{early}} = 200$, $\tau = 10^{-5}$)

ondly, the method performs worse as the dimensionality of the problem d increases, which is particularly evident for the highly-multimodal Rastrigin function.

The choice of the noise (Gaussian and Lévy) and its parameters (q and α) has a problem-dependent impact on performance:

- For the Ackley function, Lévy noise yields significantly better results than Gaussian noise, especially in higher dimensions, suggesting the importance of more exploratory behaviour facilitated by longer jumps. Within Lévy noise, the stability index $\alpha = 1$ is the most effective, suggesting a balance in jump lengths. For Gaussian noise, the value $q = 1$ shows the best results.

- For the Rastrigin function, Gaussian noise leads to better performance, particularly for a larger swarm ($N = 50$) and the parameter $q = 0.5$. This suggests that the function's highly-multimodal complex landscape benefits from careful exploration. For Lévy noise, the values $\alpha = 1.0$, 1.5 show better results in the case of fewer particles ($N = 10$) and $\alpha = 0.5$ for more particles ($N = 50$).

- For the Rosenbrock function, Gaussian noise shows significantly better performance than Lévy noise. The success rates are similar across all choices of q. Within Lévy noise, the stability index $\alpha = 1.5$ with shorter average jumps is consistently the most effective. This indicates that Lévy jumps are detrimental for navigating the Rosenbrock's narrow valley.

5.1.4 Shifted Initialization Range

The particle position initialization range is shifted by $B \in \mathbb{R}$ units along each direction: $S_B = [-3+B, 3+B]^d$. We select $B = 6$, 9, so that particles are not initialized on the slope of the minimum. In order for the particles to have enough time to explore the landscape, we increase the number of timesteps T, as well as the early stopping threshold T_{early} to avoid premature convergence.

B	$q = 0.5$	$q = 1$	$q = 2$
$N = 10$			
0	**95.5%**	90.0%	87.1%
6	**60.3%**	58.1%	59.7%
9	12.4%	**14.4%**	12.7%
$N = 50$			
0	100.0%	100.0%	100.0%
6	98.4%	98.9%	98.5%
9	98.4%	98.2%	98.4%

(a) Gaussian noise with q parameter

B	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$
$N = 10$			
0	**90.6%**	88.9%	89.1%
6	55.1%	58.4%	**60.2%**
9	53.1%	**58.6%**	**58.2%**
$N = 50$			
0	**99.2%**	98.7%	98.5%
6	**98.4%**	97.7%	97.3%
9	**97.3%**	95.2%	96.2%

(b) Lévy noise with α stability index

Table 5.4: Success rate on 2D Rastrigin function (5.3) with N particles with B shift for different noise types ($T = 5000$, $T_{\text{early}} = 500$, $\tau = 10^{-4}$)

The negative impact of shifting the initialization range is more pronounced for the smaller swarm ($N = 10$), with success rates dropping considerably for the shift $B = 6$. The additional performance drop for $B = 9$ is less pronounced for Lévy noise, suggesting that longer jumps help particles escape local minima and traverse long distances effectively. In contrast, for Gaussian noise, the drop for $B = 9$ is substantial. With a larger swarm ($N = 50$), both noise types maintain high success rates, demonstrating the effectiveness of a larger swarm. There is no clear preference for the Gaussian noise parameter q or the Lévy stability index α.

5.1.5 Comparison with Other Optimizers

The performance of the proposed method is benchmarked against established optimization algorithms: PSO, SBGD, and Adam. The proposed method is termed Langevin Particle Swarm optimization with Friction-based communication (LPSF). For LPSF, PSO and Adam the same number of iterations is used: $T = 1{,}000$ for Rastrigin and Rosenbrock, and $T = 5{,}000$ for Ackley function, whereas in SBGD, the number of iterations is equal to the number of particles N. The following parameters are used for each optimizer:

- LPSF is reported with the best-performing parameters for each function, selected via grid search described in Subsection 5.1.2.
- PSO uses standard parameters derived from the constriction coefficient approach [40]: inertia weight $w = 0.729$, equal cognitive and social coefficients $c_1 = c_2 = 1.494$ and velocity clamping at $v_{\max} = 1.2$.
- SBGD employs default parameters: $h_0 = 1$ and $p = 1$.
- Adam employs recommended parameters [41]: momentum coefficients $\beta_1 = 0.9$, $\beta_2 = 0.999$ and numerical stability $\varepsilon = 10^{-8}$. Learning rate η is selected via grid search over $\{10^{-2}, 10^{-3}, 10^{-4}\}$.

Table 5.5 summarizes the success rates of the optimizers for various benchmark functions.

Function	$N = 10$			$N = 50$			Adam
	PSO	SBGD	LPSF	PSO	SBGD	LPSF	
Rastrigin, $d = 2$	91.4%	33.5%	94.2%	100%	89.8%	100%	5%
Rastrigin, $d = 3$	62.5%	4.5%	45.2%	99.8%	25.5%	88.8%	1%
Ackley, $d = 6$	78.4%	0.9%	91.4%	100%	100%	100%	0%
Rosenbrock, $d = 3$	38.1%	2.3%	99.5%	100%	35.5%	100%	100%
Rosenbrock, $d = 6$	1.3%	0.0%	37.8%	77.3%	1.4%	61.3%	82%

Table 5.5: Comparison of optimization methods success rates across functions

Adam shows poor performance on the multi-modal Rastrigin and Ackley functions, but performs well on the Rosenbrock function, achieving better results than swarm methods. In most cases, LPSF outperforms the other swarm optimizers. The computational cost of LPSF is comparable to PSO, but it is significantly higher than that of SBGD, which performs considerably worse.

5.2 Neural Network Training

5.2.1 Dataset and Setting

The experiments are conducted on the CIFAR-10 image classification dataset [42], consisting of 60,000 labelled 32×32 images across 10 classes. The dataset is split into 50,000 training images and 10,000 test images. To improve generalization, the training set is augmented with random cropping and horizontal flipping, followed by normalization. The test set is only normalized. Sample images from each class are shown in Fig. 5.5.

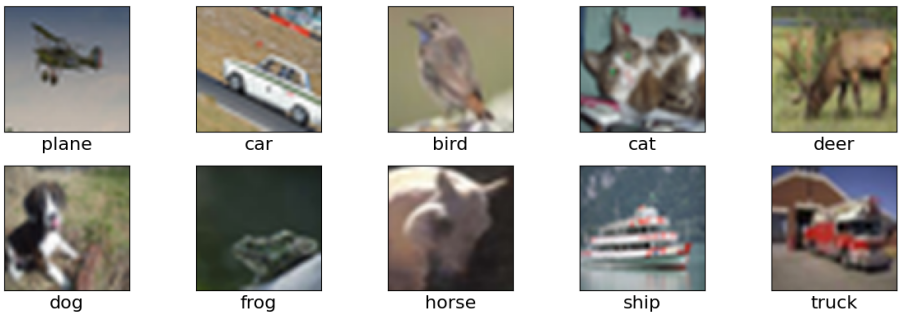

Figure 5.5: Sample images with class labels from CIFAR-10 dataset

The goal is to train a NN that maps images to class labels by minimising the cross-entropy loss on the training data.

5.2.2 Network Architecture

We use a compact convolutional neural network (CNN) with two convolutional layers followed by two fully connected layers, having 545,098 parameters. Its diagram with layers and dimensions is shown in Fig. 5.6. The two convolutional layers use 3×3 kernels with ReLU activations and 2×2 max-pooling, progressively reducing spatial dimensions from 32×32 to 8×8, while increasing feature depth to 64. The flattened output of convolutional layers is passed to a fully-connected layer with 128 output units and ReLU activation, followed by the final output layer projecting to 10 classes.

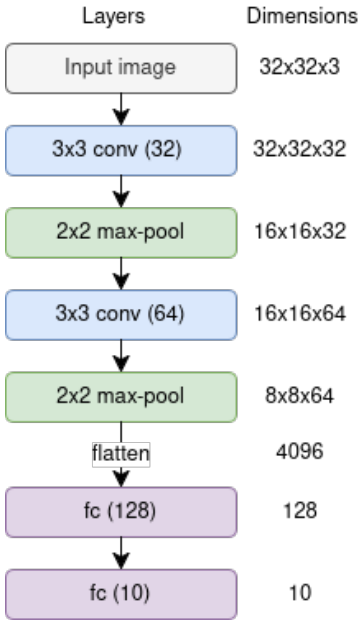

Figure 5.6: Network architecture

Although not state-of-the-art, the chosen CNN is well-suited for benchmarking optimizers due to its simple architecture and low computational complexity. Its shallow depth helps isolate the influence of the optimizer on training.

5.2.3 Comparison with Standard Optimizers

The proposed algorithm is compared with standard optimizers: Adam and stochastic gradient descent (SGD). The following parameters are selected for the optimizers as a result of tuning:

- LPSF: number of particles $N = 10$, timestep $\tau = 10^{-2}$, initial friction $\mu_0 = 10^{-2}$, minimum friction threshold $\delta\mu = 10^{-6}$, Gaussian noise with $q = 1$ and scale $c = 10^{-4}$.
- Adam: learning rate $\eta = 10^{-3}$, default momentum coefficients $\beta_1 = 0.9$, $\beta_2 = 0.999$ and numerical stability $\varepsilon = 10^{-8}$.
- SGD: learning rate $\eta = 10^{-2}$ with no momentum.

Training loss and test accuracy for different optimizers are shown in Fig. 5.7, 5.8. The lowest loss values and highest accuracies obtained during training are presented in Table 5.6.

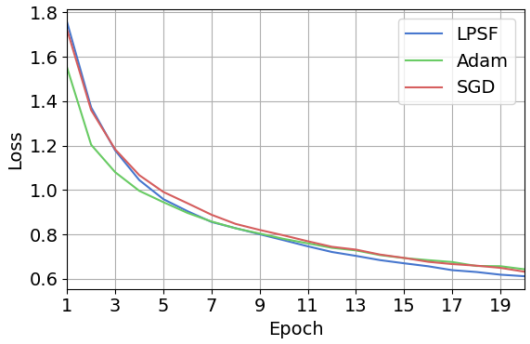

Figure 5.7: Training loss during training with different optimizers

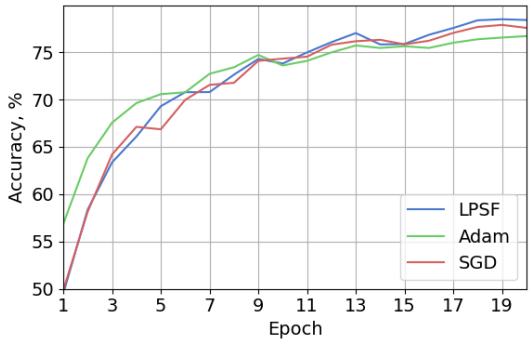

Figure 5.8: Test accuracy during training with different optimizers

Optimizer	Loss	Accuracy
LPSF	0.612	78.47%
Adam	0.643	76.69%
SGD	0.632	77.87%

Table 5.6: Best training loss and test accuracy of optimizers

The proposed algorithm performs slightly better than the established Adam and SGD optimizers in NN training. However, the computational cost of LPSF is significantly higher, taking up to N times more time. Actually, in LPSF, N models, one for each particle, are effectively trained initially. As worse-performing particles are discarded, the computational cost eventually decreases, speeding up training towards the end.

5.2.4 Impact of Particle Swarm Size

Using the same parameters as in the previous section, we compare the performance of the proposed optimizer with different particle sizes N. The loss and accuracy plots during training are presented in Fig. 5.9, 5.10. The best obtained values for loss and accuracy are summarised in Table 5.7.

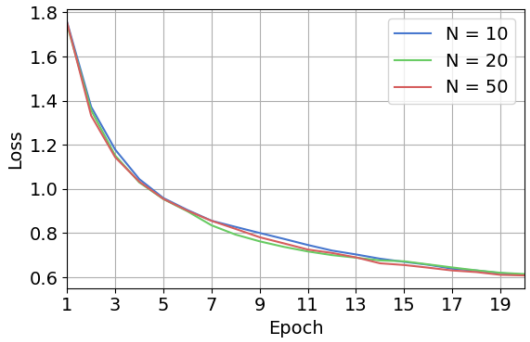

Figure 5.9: Training loss during training with swarm sizes N

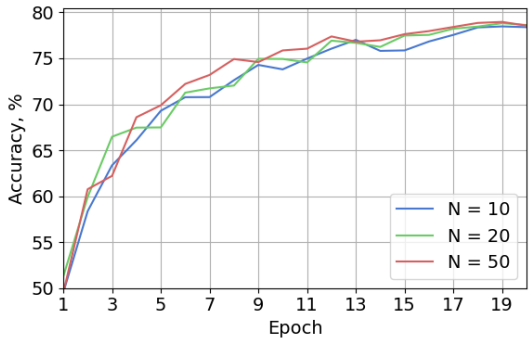

Figure 5.10: Test accuracy during training with different swarm sizes N

N	Loss	Accuracy
10	0.611	78.47%
20	0.612	78.82%
50	0.608	78.96%

Table 5.7: Best training loss and test accuracy of LPSF with N particles

During training, the performance of larger swarms is better. However, towards the end of training, the optimizers converge to configurations with similar performance. There is still a slight increase in the best test accuracy for larger swarms. However, the computational cost of the method increases approximately linearly with N and may not justify the increase in accuracy in practice.

Chapter 6

Conclusions

This work proposed a swarm optimization algorithm where particle motion is inspired by overdamped Langevin dynamics with a friction-based communication mechanism. The communication mechanism, based on agent fitness, plays an essential role by enabling better-performing agents to focus on their current regions, while allowing worse-performing agents to explore the search space. The model exhibits emergent annealing, where the average friction coefficient increases over time, gradually reducing the particles' average step size and eventually settling the swarm without explicit control.

Experiments on multi-modal benchmark functions demonstrate the method's effectiveness compared to other swarm optimizers, such as PSO and SBGD. Performance improves with larger swarm sizes and longer simulation durations, as more of the search space can be covered. No single noise type outperforms others across all considered functions. However, Lévy noise, characterized by a heavy-tailed distribution, enhances exploration, while Gaussian noise facilitates more careful exploitation.

The method successfully trained a CNN on the CIFAR-10 dataset. However, its performance is only marginally better than standard optimizers, at a much higher computational cost. The computational cost scales linearly with the number of particles N, comparable to training N models with SGD. This slight improvement can be attributed to the problem's high dimensionality and the inherent stochasticity of SGD, which makes the additional noise from the proposed method less critical. In contrast with the results from benchmark function minimization, increasing the swarm size did not yield a significant performance improvement in this case.

The algorithm is sensitive to hyperparameter selection. In particular, the timestep τ and the noise scale c are crucial.

Future work could delve deeper into the method's theoretical properties and provide a convergence analysis. Another direction involves modifying the method,

such as:

- Exploring different coupling schemes between noise and friction, such as different functions for $\sigma^2(\mu)$, connecting Gaussian noise with friction, or introducing a connection between Lévy noise parameters and friction.
- Introducing randomness in the gradient direction, as done in [3] for SBGD.
- Modifying the friction update rules. Since communication lies at the heart of emergent behaviour, different choices may lead to different emergent behaviours. Modifications could be made to the friction response function $f(\mu)$ or the update rule itself.

Acronyms

BPSO Bare bones particle swarm optimization.

CMS Chambers, Mallows and Stuck.

CNN convolutional neural network.

CPU central processing unit.

CSO Competitive swarm optimizer.

FFPE Fractional Fokker-Planck Equation.

FPE Fokker-Planck Equation.

GPU graphics processing unit.

KL Kullback-Leibler.

LPSF Langevin Particle Swarm optimization with Friction-based communication.

NN Neural network.

PDF probability density function.

PSO Particle swarm optimization.

SA Simulated annealing.

SBGD Swarm-based gradient descent.

SBRD Swarm based optimization with random descent.

SGD stochastic gradient descent.

WGSL WebGPU Shading Language.

Bibliography

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[2] J. Lu, E. Tadmor, and A. Zenginoglu, *Swarm-based gradient descent method for non-convex optimization*, 2024. arXiv: 2211.17157 [math.NA]. [Online]. Available: https://arxiv.org/abs/2211.17157.

[3] E. Tadmor and A. Zenginoglu, *Swarm-based optimization with random descent*, 2024. arXiv: 2307.12441 [math.OC]. [Online]. Available: https://arxiv.org/abs/2307.12441.

[4] Z. Ding, M. Guerra, Q. Li, and E. Tadmor, *Swarm-based gradient descent meets simulated annealing*, 2024. arXiv: 2404.18015 [math.OC]. [Online]. Available: https://arxiv.org/abs/2404.18015.

[5] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII: 7th International Conference, EP98 San Diego, California, USA, March 25–27, 1998 Proceedings 7*, Springer, 1998, pp. 591–600.

[6] Y. Shi et al., "Particle swarm optimization: Developments, applications and resources," in *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, IEEE, vol. 1, 2001, pp. 81–86.

[7] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*, IEEE, 2003, pp. 80–87.

[8] Y. Zhang, D. Gong, Y. Hu, and W. Zhang, "Feature selection algorithm based on bare bones particle swarm optimization," *Neurocomputing*, vol. 148, pp. 150–157, 2015.

[9] M. G. Omran, A. Engelbrecht, and A. Salman, "Barebones particle swarm for integer programming problems," in *2007 IEEE Swarm Intelligence Symposium*, IEEE, 2007, pp. 170–175.

[10] H. Zhang, D. D. Kennedy, G. P. Rangaiah, and A. Bonilla-Petriciolet, "Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data," *Fluid Phase Equilibria*, vol. 301, no. 1, pp. 33–45, 2011.

[11] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2015. DOI: 10.1109/TCYB.2014.2322602.

[12] D. Chauhan, R. Cheng, et al., "Competitive swarm optimizer: A decade survey," *Swarm and Evolutionary Computation*, vol. 87, p. 101 543, 2024.

[13] P. Langevin et al., "Sur la théorie du mouvement brownien," *CR Acad. Sci. Paris*, vol. 146, no. 530-533, p. 530, 1908.

[14] S. I. Denisov, W. Horsthemke, and P. Hänggi, "Generalized fokker-planck equation: Derivation and exact solutions," *The European Physical Journal B*, vol. 68, no. 4, pp. 567–575, Apr. 2009, ISSN: 1434-6036. DOI: 10.1140/epjb/e2009-00126-3. [Online]. Available: http://dx.doi.org/10.1140/epjb/e2009-00126-3.

[15] S. Geman and C.-R. Hwang, "Diffusions for global optimization," *SIAM Journal on Control and Optimization*, vol. 24, no. 5, pp. 1031–1043, 1986. DOI: 10.1137/0324060. eprint: https://doi.org/10.1137/0324060. [Online]. Available: https://doi.org/10.1137/0324060.

[16] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[17] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985. DOI: 10.1007/BF00940812.

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953, ISSN: 0021-9606. DOI: 10.1063/1.1699114. eprint: https://pubs.aip.org/aip/jcp/article-pdf/21/6/1087/18802390/1087_1_online.pdf. [Online]. Available: https://doi.org/10.1063/1.1699114.

[19] T. Chaing, C. Hwang, and S.-J. Sheu, "Diffusion for global optimization in n r," *Siam Journal on Control and Optimization - SIAM*, Jan. 1987.

[20] G. Royer, "A remark on simulated annealing of diffusion processes," *SIAM Journal on Control and Optimization*, vol. 27, no. 6, pp. 1403–1408, 1989. DOI: 10.1137/0327072. eprint: https://doi.org/10.1137/0327072. [Online]. Available: https://doi.org/10.1137/0327072.

[21] P. Bras and G. Pagès, *Convergence of langevin-simulated annealing algorithms with multiplicative noise*, 2022. arXiv: 2109.11669 [math.PR]. [Online]. Available: https://arxiv.org/abs/2109.11669.

[22] G. M. Viswanathan, S. V. Buldyrev, S. Havlin, M. G. da Luz, E. P. Raposo, and H. E. Stanley, "Optimizing the success of random searches," *nature*, vol. 401, no. 6756, pp. 911–914, 1999.

[23] G. Viswanathan, E. Raposo, and M. da Luz, "Lévy flights and superdiffusion in the context of biological encounters and random searches," *Physics of Life Reviews*, vol. 5, no. 3, pp. 133–150, 2008, ISSN: 1571-0645. DOI: https://doi.org/10.1016/j.plrev.2008.03.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571064508000146.

[24] G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev, E. J. Murphy, P. A. Prince, and H. E. Stanley, "Lévy flight search patterns of wandering albatrosses," *Nature*, vol. 381, no. 6581, pp. 413–415, 1996.

[25] N. E. Humphries et al., "Environmental context explains lévy and brownian movement patterns of marine predators," *Nature*, vol. 465, no. 7301, pp. 1066–1069, 2010.

[26] G. Ariel, A. Rabani, S. Benisty, J. D. Partridge, R. M. Harshey, and A. Be'Er, "Swarming bacteria migrate by lévy walk," *Nature communications*, vol. 6, no. 1, p. 8396, 2015.

[27] A. M. Edwards et al., "Revisiting lévy flight search patterns of wandering albatrosses, bumblebees and deer," *Nature*, vol. 449, no. 7165, pp. 1044–1048, 2007.

[28] D. Schertzer, M. Larchevêque, J. Duan, V. V. Yanovsky, and S. Lovejoy, "Fractional fokker–planck equation for nonlinear stochastic differential equations driven by non-gaussian lévy stable noises," *Journal of Mathematical Physics*, vol. 42, no. 1, pp. 200–212, Jan. 2001, ISSN: 1089-7658. DOI: 10.1063/1.1318734. [Online]. Available: http://dx.doi.org/10.1063/1.1318734.

[29] H. C. Fogedby, "Lévy flights in random environments," *Physical Review Letters*, vol. 73, no. 19, pp. 2517–2520, Nov. 1994, ISSN: 0031-9007. DOI: 10.1103/physrevlett.73.2517. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.73.2517.

[30] S. Jespersen, R. Metzler, and H. C. Fogedby, "Lévy flights in external force fields: Langevin and fractional fokker-planck equations and their solutions," *Phys. Rev. E*, vol. 59, pp. 2736–2745, 3 Mar. 1999. DOI: 10.1103/PhysRevE.59.2736. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.59.2736.

[31] H. C. Fogedby, "Langevin equations for continuous time lévy flights," *Physical Review E*, vol. 50, no. 2, p. 1657, 1994.

[32] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* (Applications of Mathematics). Berlin: Springer, 1992, vol. 23.

[33] gfx-rs, *wgpu: A cross-platform, safe, pure-Rust graphics API*, https://github.com/gfx-rs/wgpu, Accessed: 2025-04-25, 2025.

[34] A. Appleby, *MurmurHash3 on GitHub*, https://github.com/aappleby/smhasher/blob/master/src/MurmurHash3.cpp, 2025. Accessed: May 6, 2025.

[35] S. Dahlgaard, M. Knudsen, and M. Thorup, "Practical hash functions for similarity estimation and dimensionality reduction," in *Advances in Neural Information Processing Systems*, I. Guyon et al., Eds., vol. 30, Curran Associates, Inc., 2017.

[36] G. E. Box and M. E. Muller, "A note on the generation of random normal deviates," *The annals of mathematical statistics*, vol. 29, no. 2, pp. 610–611, 1958.

[37] J. M. Chambers, C. L. Mallows, and B. W. S. and, "A method for simulating stable random variables," *Journal of the American Statistical Association*, vol. 71, no. 354, pp. 340–344, 1976. DOI: 10.1080/01621459.1976.10480344. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1976.10480344. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1976.10480344.

[38] E. Pantaleo, P. Facchi, and S. Pascazio, "Simulations of lévy flights," *Physica Scripta*, vol. 2009, no. T135, p. 014 036, Jul. 2009. DOI: 10.1088/0031-8949/2009/T135/014036. [Online]. Available: https://dx.doi.org/10.1088/0031-8949/2009/T135/014036.

[39] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[40] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] A. Krizhevsky, G. Hinton, et al., "Learning multiple layers of features from tiny images," 2009.