Adaptive Autoscaling in Serverless Platforms via Non-Stationary Gradient Descent

J. Anselmi
Inria – Ghost

10ème journée COSMOS

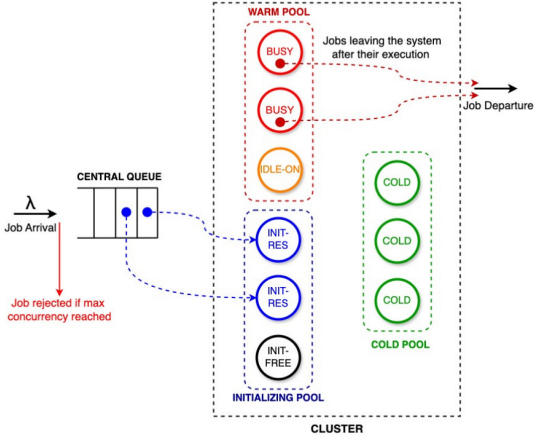
Paris, December 2025

Joint work with
D. Ardagna, B. Gaujal, A.W. Kambale, L.-S. Rebuffi

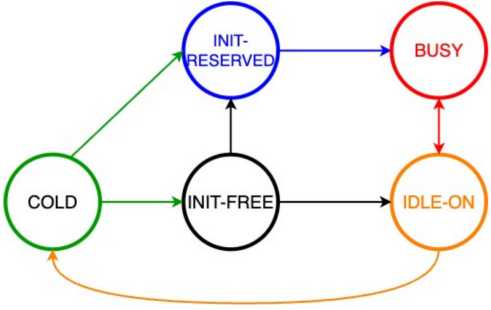
Based on
Non-Stationary Gradient Descent for Optimal Autoscaling, IEEE/ACM Transactions on Networking, 2025
Autoscaling in Serverless Platforms via Online Learning with Convergence Guarantees, submitted.

Autoscaling Architecture – Server/Function States

Goal: adjust the current service capacity automatically in response to the current load



(a) Snapshot of system at a given time



(b) State transitions for each function.

Autoscaling Algorithm

Scale-up principle: *"Activate a function (server) upon a job arrival if no active idle server exists"*

Problem: Occurrence of coldstarts

Idea: (from inventory ctl)

- Keep a pool of idle servers
  - "# idle instances" $\in [\theta_{idle}, \theta_{stock}]$
  - *container prewarming*
- Deactivate an idle instance after $\text{Exp}(\theta_{\text{exp}})$ secs

Control parameters:

- θ_{idle}, θ_{stock}: scale-up rule.
- θ_{exp}: scale-down rule.

Autoscaling Algorithm

Scale-up principle: *"Activate a function (server) upon a job arrival if no active idle server exists"*

Problem: Occurrence of coldstarts

Idea: (from inventory ctl)

- Keep a pool of idle servers
  - "# idle instances" $\in [\theta_{idle}, \theta_{stock}]$
  - *container prewarming*
- Deactivate an idle instance after Exp(θ_{exp}) secs

Control parameters:

- θ_{idle}, θ_{stock}: scale-up rule.
- θ_{exp}: scale-down rule.

```
1  On each arrival (with rate λ) do
2      if #idle-on_functions = 0 and #cold_functions > 0
         then
3          #init-reserved_functions := 1;
4          #init-free_functions_to_spawn := π_{θ_stock};
5          Initialize #init-reserved_functions;
6          Initialize #init-free_functions_to_spawn;
7      else if #idle-on_functions > 0 then
8          #idle-on_functions := #idle-on_functions - 1;
9          if #idle-on_functions < π_{θ_idle} and
             #cold_functions > 0 then
10             #init-free_functions_to_spawn := π_{θ_stock} −
                 (#idle-on_functions + #init-free_functions);
11         Serve the request with an idle-on function;
12         Initialize #init-free_functions_to_spawn;
13     else
14         Reject job
15 end
16 On each expiration of an idle-on function (with rate
     θ_exp) do
17     transition to cold the expired idle-on function;
18 end
```

Optimal Autoscaling

Objective (informal): Learn the three control parameters

$$\theta = (\theta_{stock}, \theta_{idle}, \theta_{exp}) = (\theta_1, \theta_2, \theta_3)$$

to minimize a cost that balances **job delay** (or blocking) and **energy consumption**.

We will compute the optimal trade-off using a stochastic gradient descent algorithm.

Markov Model

State: $x \in \mathcal{X} := \left\{ x = (x_i \in \mathbb{N} : i = 1, \ldots, 4) \;\middle|\; \sum_{i=1}^{3} x_i \leq N,\ x_4 \leq x_3 \right\}$

- x_1: idle-on functions
- x_2: busy functions
- x_3: initializing functions
- x_4: reserved (waiting) jobs

Capacity (N servers max):

$$x_1 + x_2 + x_3 \leq N, \qquad x_4 \leq x_3$$

Dynamics: job arrivals and completions, server expirations and initializations drive a continuous-time Markov chain.

Markov Chain: Transitions and Rates

The Markov chain $(X_t)_t$ has the following transition matrix:

Recall $x = (\text{idle-on, busy, init, init-res})$

$$Q_\theta(x, x') = \begin{cases} \lambda \mathbb{I}_{\{x_1=0, N-\sum_{i=1}^3 x_i>0\}} & \text{if } x' = x + e_4 + (1 + \pi_{\theta_{stock}}(x))\, e_3 \\ \lambda \mathbb{I}_{\{0<x_1<\pi_{\theta_{idle}}, N-\sum_{i=1}^3 x_i>0\}} & \text{if } x' = x - e_1 + e_2 + e_3\,(\pi_{\theta_{stock}}(x) - x_1 - x_3) \\ \lambda \mathbb{I}_{\{x_1 \geq \pi_{\theta_{idle}}\}} & \text{if } x' = x - e_1 + e_2 \\ \mu x_2 & \text{if } x' = x + (e_1 - e_2)\mathbb{I}_{\{x_4=0\}} - e_4\, \mathbb{I}_{\{x_4>0\}} \\ \theta_{exp}\, x_1 & \text{if } x' = x - e_1 \\ \beta x_3 \mathbb{I}_{\{x_4>0\}} & \text{if } x' = x + e_2 - e_3 - e_4 \\ \beta x_3 \mathbb{I}_{\{x_4=0\}} & \text{if } x' = x + e_1 - e_3 \end{cases}$$

for all states $x, x' \in \mathcal{X}$, with $x' \neq x$ and

$$\pi_{\theta_{stock}}(x) = \text{“scale-up policy”}$$

i.e., the number of servers to initialize upon job arrival.

Stochastic Optimization for Autoscaling

Instantaneous cost:

$$C(x) = \sum_{i=1}^{4} w_i x_i + w_{\text{rej}} \mathbf{1}_{\{x_2+x_4=N\}}.$$

- $w_1, \ldots, w_4$: energy cost per function state.
- w_{rej}: penalty for dropped jobs.
- Note: θ is not 'directly inside' the instantaneous cost C

Objective:

$$f(\theta) = \mathbb{E}\big[C(X_\infty(\theta))\big].$$

Technical difficulty: The stationary state $X_\infty(\theta)$ is not accessible!
$\rightarrow$ (even when transition rates are not known)

General Stochastic Optimization Framework

Setup.

- The system evolves as a continuous-time Markov chain $(X_t^\theta)_{t\geq 0}$.
- The controller knows the cost function $C(\cdot)$ and the current state X_t^θ, and nothing else!

Goal. Tune the parameter vector $\theta \in \mathbb{R}^d$ to minimize the stationary cost: $f(\theta) = \mathbb{E}[\, C(X_\infty^\theta)\,]$.

General Stochastic Optimization Framework

Setup.

- The system evolves as a continuous-time Markov chain $(X_t^\theta)_{t\geq 0}$.
- The controller knows the cost function $C(\cdot)$ and the current state X_t^θ, and nothing else!

Goal. Tune the parameter vector $\theta \in \mathbb{R}^d$ to minimize the stationary cost: $f(\theta) = \mathbb{E}[\, C(X_\infty^\theta)\,]$.
Key Difficulties.

- The cost C does *not* depend directly on θ (only through X_∞^θ), and ∇C is unavailable.
- We *cannot* sample from X_∞^θ: only transient trajectories of (X_t^θ) are observable.

$\Rightarrow$ This prevents the use of standard SGD or classical Kiefer-Wolfowitz methods.

Approach. To develop a multi-dimensional, non-stationary stochastic gradient method that converges using only transient samples of the Markov process.

Related Work

(Pflug, 1994) introduced the first Kiefer-Wolfowitz–type scheme in our setting, but without convergence rates.

A Robbins–Monro SA approach with known gradients, used in (Chandak, Borkar, Dodhia, 2022), provides anytime convergence rates that depend on the Lipschitz constant of the cost

Results for non-convex costs and non-reversible Markov chains (as in our case) were developed in (Sun, Sun, Yin, 2018) and othe rpapers, but they require access to the gradient.

Non-stationary ergodic noise / full knowledge of the model parameters has been addressed in other papers; e.g.,(Duchi, Agarwal, Johansson, Jordan, 2012), (Tournaire, Castel-Taleb, Hyon, 2023).

Non-Stationary Gradient Descent (Kiefer-Wolfowitz) – Scalar Version

NSGD **Algorithm.**

- Initialize the system in state x_{start} with parameter θ_0.
- At iteration n (time T_n):
  - Run K simulated trajectories of length $\tau_n = \tau \log(n+1)$ with $\theta_n + \delta_n$ and compute

$$\hat{f}_n(\theta_n + \delta_n) = \frac{1}{K} \sum_{i=0}^{K-1} C\left(X^{\theta_n+\delta_n}_{T_n+i\tau_n,\ T_n+(i+1)\tau_n} \right)$$

    (or in practice, observe the system $K\tau_n$ steps).
  - Run another K simulations under $\theta_n - \delta_n$ and compute

$$\hat{f}_n(\theta_n - \delta_n) = \frac{1}{K} \sum_{i=0}^{K-1} C\left(X^{\theta_n-\delta_n}_{T_n+(K+i)\tau_n,\ T_n+(K+i+1)\tau_n} \right).$$

  - Update the parameter via the symmetric finite-difference estimate:

$$\theta_{n+1} \;=\; \theta_n \;-\; \gamma_n \frac{\hat{f}_n(\theta_n + \delta_n) \;-\; \hat{f}_n(\theta_n - \delta_n)}{2\,\delta_n}.$$

Assumptions for Convergence

AS1 The function $f(\theta)$ has a unique minimizer θ^*, is $\mathcal{C}^3$, and f' is Lipschitz and bounded.

AS2 The step-size sequences $(\gamma_n)_n$, $(\delta_n)_n$, and $(\tau_n)_n$ satisfy:

$$\delta_n \to 0, \qquad \tau_n \to +\infty, \qquad \sum_n \gamma_n = \infty, \qquad \sum_n \gamma_n^2 \delta_n^{-2} < \infty,$$

with $(\gamma_n)_n$, $(\delta_n)_n$ and $(\gamma_n/\delta_n)_n$ decreasing.

AS3 Finite variance:

$$\sup_\theta \mathrm{Var}_\theta\big(F(X_\infty^\theta)\big) < \infty.$$

AS4 Uniform geometric mixing: for all θ, x, and t

$$\left\| P_\theta^t(x, \cdot) - m_\theta \right\|_1 \leq C_1 \rho^t.$$

AS5 The function f is strongly convex – $f'(\theta)(\theta - \theta') \geq \kappa(\theta - \theta')^2$, $\kappa > 0$, $\forall \theta, \theta' \in \mathbb{R}$.

Convergence Theorems

Let $(\theta_n)_n$ be the sequence generated by *NSGD*.

Theorem ($\theta \in \mathbb{R}^d$)

Under Assumptions AS1→AS4, $\theta_n \xrightarrow{a.s.} \theta^*$.

Theorem ($\theta \in \mathbb{R}$)

Let $\delta_n = n^{-\frac{2}{3}}$, $\gamma_n = \frac{1}{n}$, $T_n = \frac{\alpha \log n}{\log(1/\rho)}$ *with* $\alpha > 1$, *Under Assumptions AS1→AS5,*

$$\limsup_{n\to\infty} \mathbb{E}\left[(\theta_n - \theta^*)^2\right] n^{2/3} \leq O(1)$$

Sketch of the Proof – θ Scalar

We rewrite the parameter update rule as

$$\theta_{n+1} = \theta_n - \gamma_n\big(f'(\theta_n) + \Delta_{\mathrm{diff},n} + \Delta_{\mathrm{mart},n} + \Delta_{\mathrm{mix},n}\big)$$

where we define the decomposition:

$$\Delta_{\mathrm{mart},n} := \frac{F(\theta_n+\delta_n, X_\infty^{\theta_n+\delta_n}) - F(\theta_n-\delta_n, X_\infty^{\theta_n-\delta_n})}{2\delta_n} - \frac{f(\theta_n+\delta_n) - f(\theta_n-\delta_n)}{2\delta_n},$$

$$\Delta_{\mathrm{diff},n} := \frac{f(\theta_n+\delta_n) - f(\theta_n-\delta_n)}{2\delta_n} - f'(\theta_n),$$

$$\Delta_{\mathrm{mix},n} := \frac{\hat{f}_n(\theta_n+\delta_n) - \hat{f}_n(\theta_n-\delta_n)}{2\delta_n} - \frac{F(\theta_n+\delta_n, X_\infty^{\theta_n+\delta_n}) - F(\theta_n-\delta_n, X_\infty^{\theta_n-\delta_n})}{2\delta_n}.$$

$\Rightarrow$ better control of certain difference terms.

Sketch of the Proof of Th. 1: Almost Sure Convergence

- Show that $(\theta_n)_{n\in\mathbb{N}}$ is an APT (asymptotic pseudotrajectory) of the o.d.e. $\theta = f'(\theta)$.
  - $\lim_{t\to\infty} \sup_{0\le h\le T} d\big(X(t+h), \Phi_h(X(t))\big) = 0$

  $\Rightarrow$ Long-run behavior = deterministic semiflow of an ODE
- Use the fact that this o.d.e has a single fixed point (θ^*).
- Construct of a Lyapunov function to show convergence of the flow of the o.d.e. to the unique fixed point.
  - For the (negative) gradient flow f strictly decreases along nonstationary trajectories
- All this implies a.s. convergence of the sequence θ_n to θ^*.

Sketch of the Proof of Th. 2: Convergence Rate in the Convex Case

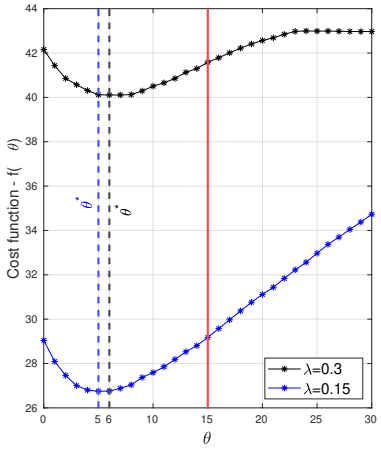
As for the computation of the convergence rate, we decompose the gap between $|\theta_n - \theta^*|$ into the terms $\Delta_{\mathrm{mart},n}$, $\Delta_{\mathrm{diff},n}$ and $\Delta_{\mathrm{mix},n}$:

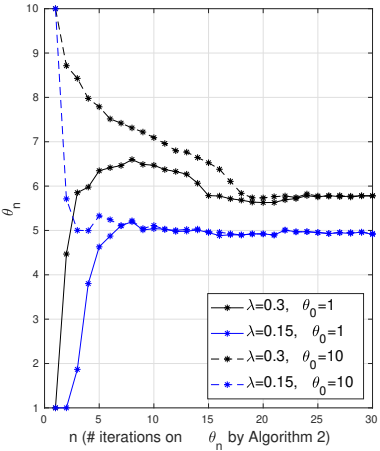
- $\Delta_{\mathrm{mart},n}$ is a difference of martingales whose expectation is 0.
- $\Delta_{\mathrm{diff},n}$ is bounded using classical stochastic gradient arguments for convex functions (AS5).
- $\Delta_{\mathrm{mix},n}$ is the hardest term to bound because it involves non-stationary terms. Here, uniform mixing (AS4) plays a major part.

Back to Autoscaling: It fits the Framework

- All assumptions required in Theorems 1–2 can be ensured in the autoscaling problem by applying a suitable truncation/extension procedure.
- The most delicate part is proving that f is $\mathcal{C}^3$ and establishing uniform mixing. Both follow from regularity properties of the Drazin inverse of $(I - P_\theta)$.
- The convergence rate of *NSGD* depends on the strong convexity of the cost function. This has been verified only numerically in our examples (see next slide).

Cost Function $f(\theta)$ and Behavior by *NSGD* ($\theta = \theta_{stock}$ scalar)

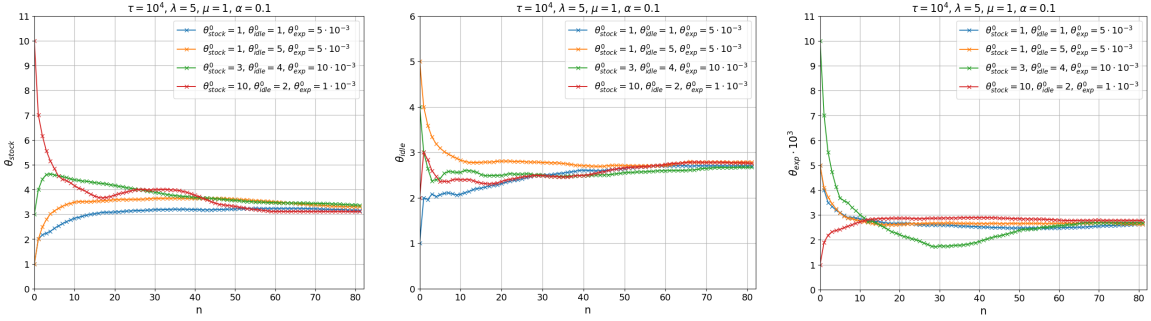




The red line corresponds to the truncation to the interval $[0, M]$ over which the function appears to be convex

Experimental Evaluation on SimFaaS $\theta = (\theta_{stock}, \theta_{idle}, \theta_{exp})$

- Simulator developed on top of SimFaaS [Mahmoudi and Khazaei, 2021] for realistic emulation of public serverless platforms [`https://github.com/Wamuhindo/NSGD`]

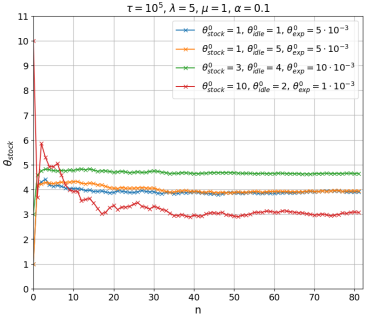


(a) θ_{stock} **(b)** θ_{idle} **(c)** θ_{exp}

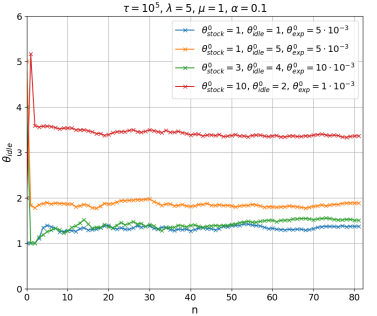
Sequence of θ's produced by *NSGD* under exponentially distributed service times.

Experimental Evaluation on SimFaaS $\theta = (\theta_{stock}, \theta_{idle}, \theta_{exp})$

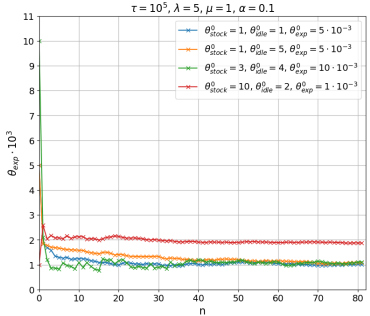
NSGD over Pareto service times:



(a) θ_{stock}



(b) θ_{idle}



(c) θ_{exp}

Evolution of control parameters $\boldsymbol{\theta}_{\text{init}}$, $\boldsymbol{\theta}_{\text{idle}}$, and $\boldsymbol{\theta}_{\text{exp}}$ over 80 iterations under Pareto-distributed service times.

LSTM–PPO Benchmark [Agarwal et al. 2024]

Core Idea. A recurrent RL agent learns an autoscaling policy from observed system metrics.

It integrates Long Short-term Memory (LSTM) with Proximal Policy Optimization (PPO).

LSTM Component:

- Captures temporal patterns and stochasticity of the workload.
- Processes a window of observations:
  - avg. CPU & memory utilization, avg. execution time, number of replicas, proportion of served requests, total arrivals.

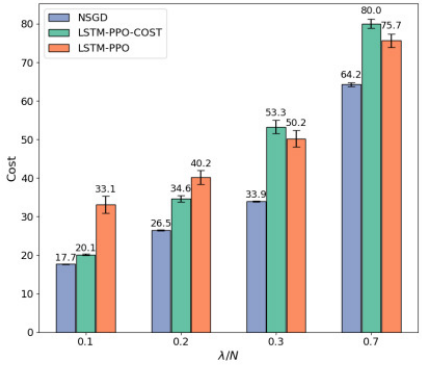
PPO Component:

- Learns the scaling policy using policy-gradient updates.
- Action space: $\{-2, -1, 0, +1, +2\}$ (scale down / do nothing / scale up).
- Reward combines performance (served requests) and resource use.

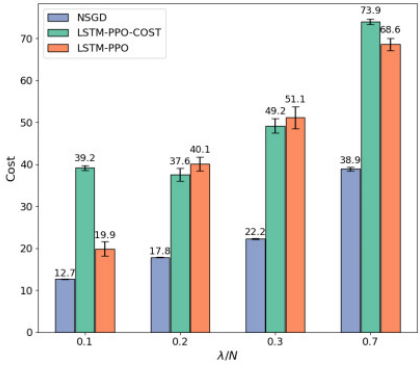
Implementation [`https://github.com/Wamuhindo/NSGD`]:

- Simulator built on RLlib, extended to expose CPU and memory metrics.

Comparison with LSTM-PPO (I)



(a) Exponentially distributed service times.

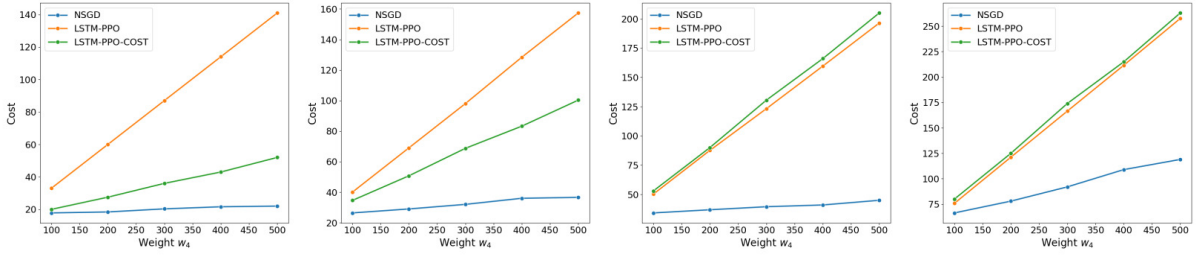


(b) Pareto distributed service times.

Figure 4: Cost comparison of $NSGD$ with LSTM-PPO [16] under 10 different runs and three arrival rates. The service times follow an exponential distribution in subfigure (a) and a Pareto distribution in subfigure (b). LSTM-PPO considers two scenarios, one with its original reward (subfigure (a) orange) and another (LSTM-PPO-COST) with our cost as the negative of the reward (subfigure (b) green).

Comparison with LSTM-PPO (II)

Cost sensitivity analysis when changing weights



(a) $\lambda/N = 0.1$ (b) $\lambda/N = 0.2$ (c) $\lambda/N = 0.3$ (d) $\lambda/N = 0.7$

Figure 6: Cost sensitivity analysis when changing weights associated with *init-reserved* (w_4) and rejection (w_{rej}) for different values of λ/N.

Main Takeaways

- **Modeling**: autoscaling framed as a multi-parameter stochastic optimization problem balancing energy and performance.
- **Algorithmic**: *NSGD* performs online SGDbased tuning of three key control parameters.
- **Theory**: convergence guarantees established in Markovian settings.
- **Simulation**: extensive, realistic experiments (SimFaaS) show robustness even in non-Markovian workloads.
- **Insight**: principled multi-parameter control can significantly reduce costs compared to heuristic or state-of-the-art RL algorithms.

La fin

A. W. Kambale, J. Anselmi, D. Ardagna, B. Gaujal *Autoscaling in Serverless Platforms via Online Learning with Convergence Guarantees*, submitted

J. Anselmi, B. Gaujal, L.S. Rebuffi *Non-Stationary Gradient Descent for Optimal Auto-Scaling in Serverless Platforms*, IEEE/ACM Transactions on Networking, vol. 33, no. 4, pp. 1574-1587, Aug. 2025

Simulator: `https://github.com/Wamuhindo/NSGD`