Title

Appropriate noise addition to metaheuristic algorithms can enhance their performance.

Journal

Scientific reports, 13(1)

ISSN

2045-2322

Authors

Choi, Kwok Pui
Kam, Enzio Hai Hong
Tong, Xin T
et al.

Publication Date

2023-03-01

DOI

10.1038/s41598-023-29618-5

scientific reports

Check for updates

OPEN

Appropriate noise addition to metaheuristic algorithms can enhance their performance

Kwok Pui Choi[1], Enzio Hai Hong Kam[2], Xin T. Tong[3] & Weng Kee Wong[4]✉

Nature-inspired swarm-based algorithms are increasingly applied to tackle high-dimensional and complex optimization problems across disciplines. They are general purpose optimization algorithms, easy to implement and assumption-free. Some common drawbacks of these algorithms are their premature convergence and the solution found may not be a global optimum. We propose a general, simple and effective strategy, called heterogeneous Perturbation–Projection (HPP), to enhance an algorithm's exploration capability so that our sufficient convergence conditions are guaranteed to hold and the algorithm converges almost surely to a global optimum. In summary, HPP applies stochastic perturbation on half of the swarm agents and then project all agents onto the set of feasible solutions. We illustrate this approach using three widely used nature-inspired swarm-based optimization algorithms: particle swarm optimization (PSO), bat algorithm (BAT) and Ant Colony Optimization for continuous domains (ACO). Extensive numerical experiments show that the three algorithms with the HPP strategy outperform the original versions with 60–80% the times with significant margins.

Over the past couple of decades, Swarm Intelligence has and continues to inspire a steadily rising numbers of nature-inspired swarm-based algorithms for optimizing high-dimensional complex cost functions, including those that do not have analytic forms. Examples of swarm-based algorithms include particle swarm optimization (PSO), bat algorithm (BAT), ant colony optimization for continuous optimization (ACO). Swarm-based optimization algorithms are motivated by nature or animal behavior and then thoughtfully formulated into an algorithm that iterates to the optimum based on a couple of equations. Generally, these algorithms are easy to code and implement, and do not need gradient information or technical assumptions for them to generally work well. Computer codes are widely and freely available, which have undoubtedly fueled numerous and various applications of these algorithms to tackle many different types of complex real-world optimization problems. Documentation of their effectiveness is widespread and their meteoric rise in applications and interest in both industry and academia is well documented, see[1,2], for example. Applications of these algorithms are diverse and include estimating parameters in mixed nonlinear pharmacokinetic and pharmacodynamic models[3] and tackling various optimization problems in energy conservation[4], medical sciences[5] and agriculture[6]. A most recent review of PSO applications in different areas is[7].

There are known challeneges of nature-inspired swarm-based optimization algorithms, and we highlight two: (1) They require effective tuning of their parameters to achieve optimal performance[8]; and (2) they often suffer from premature convergence to a local optimum of the cost function, especially when the cost function is high-dimensional and multi-modal[9,10]. In recent years, Yang et al. and Choi et al.[11–13] proposed general strategies to tackle the first challenege using different tools. We opine that tuning parameters is less of an issue now with the introduction of the racing algorithm[14,15].

In this work, we propose an innovative and simple Perturbation–Projection (PP) strategy to address the latter challenge by adding noise to a nature-inspired swarm-based algorithm. An interesting part of the strategy is to have only half of the swarm more actively engaged in the exploration for the optimum. The proposed methodology is general, and as long as certain unrestrictive technical conditions are met, our techniques ensure almost sure convergence to a global optimum. We apply the methodology to three popular swarm-based algorithms and results from an extensive simulation not only support they tend to converge to the global optimum but additionally, show that they tend to outperform the original algorithms. In the literature, there are frequently different

[1]Department of Statistics and Data Science, National University of Singapore, Singapore 117546, Singapore. [2]Department of Computer Science, National University of Singapore, Singapore 117417, Singapore. [3]Department of Mathematics, National University of Singapore, Singapore 119076, Singapore. [4]Department of Biostatistics, University of California at Los Angeles, Los Angeles 90095, USA. ✉email: wkwong@ucla.edu

modifications to enhance performance of the original metaheuristic algorithm. Some seek to tackle specialized problems better and others aim to speed up the algorithm. A common aim is to avoid premature convergence and there are numerous proposals to address this issue for swarm-based algorithms; see, for example[16–19]. These methods tend to be valid either for only one specific algorithm and not for a class of algorithms. Another commonality is that they do not generally have a rigorous mathematical theory to support their improved convergence to the global optimum. The distinctive features of our proposed modifications are that they are supported by mathematical theory, simple to implement and applicable to a broad class of swarm-based algorithms.

"Stochastic enhancement of an algorithm's exploration" section describes how we add noise to a nature-inspired metaheuristic algorithm and modify the algorithm using a Perturbation-Projection (PP) strategy, In "Applications" section, inspired by the fact that agents in many swarms are often divided to specialize in different aspects of optimization (exploration and exploitation), we introduce an additional feature, heterogeneity, in the PP strategy. We then demonstrate how to modify the original PSO, BAT and ACO algorithms, by applying the PP strategy on half of the swarm agents, to meet the required conditions for almost sure convergence to the global optimum. We denote these modified algorithms by "hm" (heterogeneously modified) in front of their names, i.e., hmPSO, hmBAT and hmACO. In "Numerical experiments" section, we conduct extensive numerical experiments using many commonly used test functions to evaluate the usefulness of the modified algorithms. Results show that the modified algorithms are either competitive or they outperform the original algorithms. We also conduct tuning parameter analysis for the Heterogeneous Perturbation-Projection (HPP) strategies. "Conclusions" section concludes with some directions for future work.

The paper has two sets of Supplementary Material: Supplementary Material A lists the 28 test functions of various types in tabular form we used to evaluate and compare performances of the various algorithms. Supplementary Material B lists five additional test functions from CEC2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization in Supplementary Table 2 for our numerical experiment 2, including Supplementary Figures 1 and 2 to show the relative performance of the various algorithms in different ways. Supplementary Tables 3–5 compare, respectively, the performance of PSO, BAT, ACO relative to their hm versions using test functions in Supplementary Table 2.

Stochastic enhancement of an algorithm's exploration

In this section, we introduce a simple and effective strategy to enhance a nature-inspired swarm-based algorithm so that the enhanced algorithm is guaranteed to converge almost surely to a global optimum. We assume the cost function can be high-dimensional, non-differentiable, non-separable or non-convex with multiple local or global optima. The key idea in the proposed algorithms is simple: we incorporate an additional stochastic component with an appropriate noise level to a swarm-based algorithm to enable it to escape from a local minimum and iterate itself to a global optiumum.

The idea of adding noise to an algorithm to improve the search process is not entirely new. Several optimization subfields have incorporated such a strategy in the search of a global optimum and they have mixed results. For example, Ding and Tan, Sun et al., Neelakantan et al., Selman et al., Chen et al., Zhou et al., Jin et al.[20–26] considered different types of optimization problems and explored adding different types of noise and its amount to different types of search procedures to find an optimum. They include adding noise to extricate a search from a saddle point or improve local searches to applications in training neural networks or adding gradient noise to improve learning for very deep networks. Interestingly, the results are mixed, implying that introducing noise to a search process is not always advantageous.

There is also the perennial problem of searching for the optimum within the user-specified region or/and the optimum has to satisfy various constraints. There are various techniques specially designed for meeting such requirements and sample papers that address this issue in different ways are[27–31]. For example, Maurice[30] advocated the usage of a reflecting wall to bounce back particles that went beyond the search space, and[31] took advantage of the fact that D-optimal designs frequently have support points at the boundary of the search space, and suggested bringing back out-of-region search particles to the boundary. Our proposed algorithm described below has the distinguishing features that (i) the search is always confined to the user-specified region and the solution satisfies all the constraints, (ii) the strategy is quite general and applies to a wide class of swarm-based metaheuristic algorithms, and (iii) the algorithm is guaranteed to converge almost surely to a global optimum. Additionally, our work appears to be the first in the literature to add noise to a nature-inspired metaheuristic algorithm and unlike the above cited work, our results suggest that such a strategy is generally helpful for a swarm-based algorithm for finding the global optimum.

Without loss of generality, we may assume that we have a minimization problem since $\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) = -\max_{\mathbf{x}\in\mathcal{D}}\{-f(\mathbf{x})\}$. Here $f(\mathbf{x})$ is a user-specified real-valued cost function defined on a given compact subset $\mathcal{D}$ of $\mathbb{R}^d$. For many real-world applications, there are physical and budgetary constraints and we assume that these constraints have been appropriately incorporated and formulated as a general minization problem. For many other problems, like those in machine learning, regularization terms, sometimes denoted by $\rho\|\mathbf{x}\|_1$ (or $\rho\|\mathbf{x}\|_2^2$) are often added to the nonnegative loss/cost function f to prevent over-fitting. The regularization parameter ρ can often be selected through various cross validation methods[32,33]. For such situations, the goal is then to equivalently minimize the function $F(\mathbf{x}) = f(\mathbf{x}) + \rho\|\mathbf{x}\|_1$ (or $f(\mathbf{x}) + \rho\|\mathbf{x}\|_2^2$) over $\mathcal{D}$, which is the closed L_1-ball (or L_2-ball) centered at the origin with radius $f(\mathbf{0})/\rho$. This is because for $\mathbf{x} \notin \mathcal{D}$, $F(\mathbf{x}) > F(\mathbf{0})$ so the global minima are in $\mathcal{D}$.

To solve the optimization problem, we resort to nature-inspired swarm-based algorithms, which are increasingly used to solve complex and high-dimensional optimization problems. They generally employ an evolutionary-like or swarm-based strategy to search for an optimum by first randomly generating a set of candidate solutions of given size n. Depending on the particular swarm-based algorithm, candidate solutions are called

agents or particles. These algorithms have tuning parameters, embrace common search principles and have good exploration and exploitation abilities. They also include stochastic components and tuning parameters and a swarm-based type of algorithm may be generally described as follows:

At time or iteration $t = 1, 2, \ldots$, let $\mathbf{x}_i(t)$ denote the ith particle's position or candidate solution of the optimization problem. Let $\{\mathbf{x}_i(t) : 1 \le i \le n\}$ denote the collection of candidate solutions of the swarm of size n. In addition, there is auxiliary information from each member of the swarm and we denote them collectively by $\{\mathbf{v}_i(t) : 1 \le i \le n\}$. For example, in PSO and BAT, the auxiliary information of each particle is its velocity with which it flies to the current position. We denote them by

$$\mathbf{x}(t) = (\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)) \text{ and } \mathbf{v}(t) = (\mathbf{v}_1(t), \ldots, \mathbf{v}_n(t)). \tag{1}$$

ACO does not have velocity in its formulation. Suppose the algorithm has a stochastic update rule of the following form: for agent i,

$$\mathbf{x}_i(t+1) = \Phi_i(\mathbf{x}(t), \mathbf{v}(t), f), \quad \mathbf{v}_i(t+1) = \Psi_i(\mathbf{x}(t), \mathbf{v}(t), f), \tag{2}$$

for $1 \le i \le n$, where Φ_i, Ψ_i are some functions of which the outputs can be random. At either a pre-selected deterministic time or a stopping time T, the algorithm is terminated and the best solution $\min_{1\le i\le n, 0\le t\le T} f(\mathbf{x}_i(t))$ is output.

A key desirable property of a global optimization algorithm is its ability to identify a global optimum eventually. We formulate the following conditions on the algorithm's agents, $\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)$.

(C1) At any iteration, all agents stay in $\mathcal{D}$, i.e., $\mathbf{x}_i(t) \in \mathcal{D}$ for $1 \le i \le n$ and $t = 0, 1, \ldots, T$.

(C2) Some of the agents are explorative, so they can find any regions of improvement with positive probability. Specifically, there exists an $\alpha > 0$, independent of t, such that for any d-dimensional ball B in $\mathcal{D}$ satisfying $f(\mathbf{y}) \le \min_{1\le i\le n} f(\mathbf{x}_i(t))$, for all $\mathbf{y} \in B$, the following holds

$$\max_{1\le i\le n} \mathbb{P}_t(\mathbf{x}_i(t+1) \in B) \ge \alpha |B|, \tag{3}$$

with $|B|$ denoting the volume of B and $\mathbb{P}_t$ the conditional probability with information at the t-th iteration. This condition can often be achieved by giving a lower bound of the conidtional density. See Proposition 2 as an example.

(C3) The algorithm is time improving, i.e., if we let $m(t) = \min_{1\le i\le n} f(\mathbf{x}_i(t))$, $m(t+1) \le m(t)$ almost surely. Note that (C3) itself does not guarantee the algorithm converge, since an algorithm in stalemate also satisfies it.

Now we are ready to state our first result.

Theorem 1 *Suppose f is a continuous function defined on a given compact subset $\mathcal{D}$ of $\mathbb{R}^d$ with at least one minimizer in the interior of $\mathcal{D}$. Any swarm-based algorithm that satisfies (C1)–(C3) will converge to the global minimum almost surely (a.s). In other words, if the swarm-based algorithm has n agents, $\lim_{t\to\infty} m(t) = \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ almost surely*

Moreover, for any error threshold $\epsilon > 0$, there is a constant $1 > \alpha(\epsilon) > 0$ which depends on the behavior of f so that for any $t \ge 1$, $\mathbb{P}(m(t) > \epsilon + \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})) < (1 - \alpha(\epsilon))^{t-1}$.

Proof Let $\mathbf{x}_*$ denote a minimizer of f, which is an interior point of $\mathcal{D}$. Without loss of generality, we assume $f(\mathbf{x}_*) = 0$. By (C3), $m(t) := \min_{1\le i\le n} f(\mathbf{x}_i(t))$ is a non-increasing sequence, thus it suffices to show that $m(t) \xrightarrow{P} 0$ as $t \to \infty$. Given a fixed $\epsilon > 0$, by continuity of f, there exists $\delta > 0$ such that $0 \le f(\mathbf{x}) \le \epsilon$ for $\mathbf{x} \in B(\mathbf{x}_*, \delta) := \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_*\| \le \delta\}$. Since $\mathbf{x}_*$ is an interior point, $B(\mathbf{x}_*, \delta) \subset \mathcal{D}$ for a sufficiently small δ. By (C2), at each iteration, there is an agent $\mathbf{x}_i$ such that either $f(\mathbf{x}_i(t)) \le \epsilon$ or $\mathbb{P}_t\big(\mathbf{x}_{i(t+1)}(t+1) \in B(\mathbf{x}_*, \delta)\big) \ge \alpha |B(\mathbf{x}_*, \delta)| =: \alpha(\epsilon)$. Note that $\alpha(\epsilon) = \alpha |B(\mathbf{x}_*, \delta)| \le 1$ when (C2) holds. If $f(\mathbf{x}_i(t)) \le \epsilon$, then $m(t) \le \epsilon$. If $\mathbf{x}_{i(t+1)}(t+1) \in B(\mathbf{x}^*, \delta)$, then $m(t+1) \le f(\mathbf{x}_{i(t+1)}(t+1)) \le \epsilon$. By the tower property of conditional expectation,

$$\mathbb{P}(m(t) > \epsilon) \le \mathbb{P}(\mathbf{x}_{i(s+1)}(s+1) \notin B(\mathbf{x}^*, \delta), \text{ for } s = 1, \ldots, t-1) \tag{4}$$

$$= \mathbb{E} \prod_{s=1}^{t-1} \mathbb{P}_s(\mathbf{x}_{i(s+1)}(s+1) \notin B(\mathbf{x}^*, \delta)) \le \mathbb{E} \prod_{s=1}^{t-1} (1 - \alpha(\epsilon)) = (1 - \alpha(\epsilon))^{t-1}. \tag{5}$$

Therefore, $m(t) \xrightarrow{P} 0$ as $t \to \infty$. □

Theorem 1 shows that any swarm-based algorithm satisfying (C1)–(C3) is guaranteed to converge to a neighborhood of global optimum after finitely many iterations. Many swarm-based algorithms satisfy (C3), but not all swarm-based algorithms stay within the search space $\mathcal{D}$ in their iterations thus violating condition (C1) or they are exploratory in the sense of condition (C2). For example, if an agent of the PSO swarm is initialized near the boundary of $\mathcal{D}$ and the momentum points beyond $\mathcal{D}$, this agent will likely leave $\mathcal{D}$ in the next iteration. Furthermore, if the swarm agents arrive at the same location, say a suboptimal local optimum, with zero momentum, the algorithm will stay at this location in all subsequent iterations[9,10]. However, "Perturbation-projection strategy"

section below shows that, with slight modifications of an algorithm, we can ensure the modified algorithm satisfies (C1)–(C2).

Our result actually provides a finite iteration analysis. In particular, given an error tolerance, $\epsilon > 0$, and probability tolerance, $\delta > 0$, Theorem 1 says if $t > t_0 = \left\lceil \frac{\log \delta}{\log(1-\alpha(\epsilon))} \right\rceil$ (which is a finite number), the algorithm at iteration t_0 can find the global optimum with ϵ accuracy with probability at least $1 - \delta$. Such form of finite iteration analyses can also be found for other stochastic algorithms in machine learning, for example the stochastic gradient desecent[33].

Perturbation-projection strategy. A first step is to ensure all agents stay in $\mathcal{D}$. For this purpose, we introduce the projection onto $\mathcal{D}$ ensuring the agents stay inside the search space, $\mathcal{D}$, a compact subset in $\mathbb{R}^d$:

$$\chi_{\mathcal{D}}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{D}} \|\mathbf{y} - \mathbf{x}\|_2,$$

where $\|\mathbf{y} - \mathbf{x}\|_2$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. In other words, $\chi_{\mathcal{D}}(\mathbf{x})$ is a point in $\mathcal{D}$ that is closest to $\mathbf{x}$. When ties occur, they can be broken arbitrarily. By definition, $\chi_{\mathcal{D}}(\mathbf{x}) = \mathbf{x}$ if $\mathbf{x} \in \mathcal{D}$. In general, adding the projection to an algorithm's iterations reduces the agents' distances to the optimal solution[33]. In particular, if $\mathbf{x}_* \in \mathcal{D}$ and $\mathcal{D}$ is convex, it can be shown that $\|\chi_{\mathcal{D}}(\mathbf{x}) - \mathbf{x}_*\| \leq \|\mathbf{x} - \mathbf{x}_*\|_2$. In many optimization problems, the search space, $\mathcal{D}$, is usually a d-dimensional hyper-rectangle and it follows that $\mathcal{D}$ is convex.

A second step is to enhance the algorithm's exploration ability by injecting noise into its dynamics. In stochastic optimization algorithms, such as the perturbed gradient descent and the Langevin algorithm, Gaussian noise is added to the classical gradient descent algorithm. This injection of noise effectively helps the algorithm to get out of saddle points or local minimums efficiently[34,35]. We adopt here the same strategy due to its simplicity.

By combining these two steps, the modified algorithm can be formulated as

$$\mathbf{x}_i(t+1) = \chi_{\mathcal{D}}(\chi_{\mathcal{D}}(\mathbf{x}'_i(t+1)) + \mathbf{w}_i(t)), \tag{6}$$

where $\mathbf{x}'_i(t+1)$ is the position of the ith agent after applying the algorithm's update rule at $t+1$, and $\mathbf{w}_i(t)$ is an independent draw from a density p_t such that there is a constant $\alpha > 0$

$$p_t(\mathbf{y} - \mathbf{x}) > \alpha, \quad \text{for } \mathbf{x}, \mathbf{y} \in \mathcal{D}. \tag{7}$$

The noise density p_t in general can be adaptive, depending on the evolution of the algorithm up to time t. Here we only need it to be strictly positive and bounded away from zero. We call this method of modification in (6) a perturbation-projection (PP) modification or strategy. The proposition below shows that with PP strategy, the modified algorithm satisfies (C1) and (C2).

Proposition 2 *Suppose A is a swarm-based algorithm and at each iteration, every agent is projected onto $\mathcal{D}$. If the perturbation-projection modification (6) is applied to at least one of the agents, then (C1) and (C2) hold.*

Proof Since $\chi_{\mathcal{D}}$ is applied to all agents, (C1) clearly holds. At the $(t+1)$th iteration, denote the agent to which the modification is applied by $i = i(t+1)$. Let $\mathbf{y} = \chi_{\mathcal{D}}(\mathbf{x}'_i(t+1))$.

Then, for any d-dimensional ball B in $\mathcal{D}$,

$$\max_{1\leq k\leq n} \mathbb{P}_t \quad (\mathbf{x}_k(t+1) \in B) \geq \mathbb{P}_t(\mathbf{x}_{i(t+1)}(t+1) \in B) \tag{8}$$

$$\geq \mathbb{P}_t(\mathbf{y} + \mathbf{w}_i(t) \in B) = \int_B \mathbb{P}_t(\mathbf{y} + \mathbf{w}_i(t) \in d\mathbf{x}) = \int_B \mathbb{P}_t(\mathbf{w}_i(t) \in d(\mathbf{x} - \mathbf{y})) = \int_B p_t(\mathbf{y} - \mathbf{x})d\mathbf{x} \geq \alpha|B|, \tag{9}$$

showing (C2) holds. □

Applications

Before we proceed to illustrate how to implement our PP strategy in Theorem 1 to three nature-inspired swarm-based optimization algorithms, namely, particle swarm optimization (PSO), bat algorithm (BAT) and ant colony optimization for continuous domains (ACO), we incoproate an observation from nature. It is common to observe that agents in a swarm in nature are often divided to specialize in different tasks and they collaborate to improve collective effectiveness. Inspired by this, we set some agents in the swarm to specialize in exploration while others in exploitation. In the context of PP strategy, we only perturb the exploration agents so they help exploring the search space while we do not perturb exploitation agents so as not to hamper their convergence capability. We call this heterogeneous perturbation-projection, abbreviated to HPP, strategy. Proposition 2, which only requires the perturbation strategy applied to at least one agent, continues to hold for algorithms with HPP strategy. Consequnetly, Theorem 1 guarantees HPP modified algorithms converge a.s to a global optimal solution. We denote the modified version of an algorithm A using HPP strategy by hmA. We also summarize its formulation as Pseudo code 1. For the present work, the proportions of exploration agents and exploitation agents are taken to be 1/2. One can treat the proportion of exploration agents as an additional parameter in an algorithm to be tuned, and this will be left for future work.

Our HPP strategy can be considered as a special case of cooperative or collaborative learning, which suggests that mixing agents of different responsibilities can lead to improved overall performance. Existing works

on cooperative swarm-based algorithms can be found in[36–38]. In comparison, our HPP strategy focuses more on the cooperation between exploration agents and exploitation agents. Recently,[39] also applies this idea to local optimization algorithms such as gradient descent and Langevin algorithm.

Pseudo code 1 HPP modification for swarm-based algorithms

Input: t-th iteration particle position $\mathbf{x}(t)$, auxillary information $\mathbf{v}(t)$, and their swarm-based algorithm updates Φ, Ψ.
Update: each particle with swarm-based algorithm $\mathbf{x}_i(t+1) = \Phi_i(\mathbf{x}(t), \mathbf{v}(t), f), \mathbf{v}_i(t+1) = \Psi_i(\mathbf{x}(t), \mathbf{v}(t), f), i = 1, \ldots, n$.
Update: first half particles with $\mathbf{x}_i(t+1) = \chi_{\mathscr{D}}(\chi_{\mathscr{D}}(\mathbf{x}_i(t+1)) + \mathbf{w}_i(t)), i = 1, \ldots, n/2$.
Output: $t+1$-th iteration particle position $\mathbf{x}(t+1)$, auxillary information $\mathbf{v}(t+1)$

PSO and hmPSO. In[40], Kennedy and Eberhart proposed the particle swarm optimization (PSO) algorithm. The algorithm has stimulated many refinements and inspired many variants, and these algorithms find wide applications in many fields. A recent search of "particle swarm optimization" in the Web of Science generated more than to 32,000 articles and almost 600 review articles. A small sampling of early examples of how PSO has been studied and modified over the years are[36,41–45,45–48]. PSO probably has the most modified versions among nature-inspired metaheuristic algorithms and they continue to this day.

PSO algorithm models after the movement of a flock of birds looking for food collectively. To align with our terminology in this article, we think of birds in PSO as agents, and food as a global minimizer of an objective function. At the next iteration, $t+1$, each agent veers towards the best location it has found (cognitive/memory component) and the best location known to the flock up to iteration t (social component). To describe the memory effect, we allocate a personal "memory" agent $\mathbf{x}_i^*(t)$ to record the best location agent i has found up to iteration t, and a global "memory" agent $\mathbf{x}^*(t)$ to record the best location found by all agents collectively so far. Recall the basic PSO algorithm runs the following steps iteratively:

1. Generate two independent random vector $\mathbf{U}_1$ and $\mathbf{U}_2$ from the uniform distribution on $[0,1]^d$ and let
$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{U}_1 \circ (\mathbf{x}_i^*(t) - \mathbf{x}_i(t)) + c_2\mathbf{U}_2 \circ (\mathbf{x}^*(t) - \mathbf{x}_i(t)), \tag{10}$$
where w, c_1, c_2 are given constants, and $\circ$ denotes the Hadamard product.
2. Update $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$.
3. Update personal best
$$\mathbf{x}_i^*(t+1) = \begin{cases} \mathbf{x}_i(t+1), & \text{if } f(\mathbf{x}_i^*(t)) > f(\mathbf{x}_i(t+1)); \\ \mathbf{x}_i^*(t), & \text{otherwise}. \end{cases} \tag{11}$$
4. Repeat steps 1–3 for all agents, then update the global best agent
$$\mathbf{x}^*(t+1) = \begin{cases} \mathbf{x}_j^*(t+1), & \text{if condition H holds}, \\ \mathbf{x}^*(t), & \text{otherwise}, \end{cases} \tag{12}$$
where $j = \text{argmin}_{1\le i\le n} f(\mathbf{x}_i^*(t+1))$, $m(t) = \min_{1\le i\le n} f(\mathbf{x}_i^*(t))$ and condition H: $m(t+1) < m(t)$.

Our modified PSO, denoted by hPSO, is to apply the HPP strategy in step 2 only to the exploration agents (labelled as $1, \ldots, n/2$):

2') Update, for $1 \le i \le n/2$,
$$\mathbf{x}_i(t+1) = \chi_{\mathcal{D}}(\chi_{\mathcal{D}}(\mathbf{x}_i(t) + \mathbf{v}_i(t+1)) + \mathbf{w}_i(t)), \tag{13}$$
where $\mathbf{w}_i$'s are independent multivariate normal distributed with mean vector $\mathbf{0}$ and covariance matrix $\sigma\mathbf{I}$.

Since its introduction in 1995, there have been numerous attempts to analyze the basic PSO convergence behavior. In our opinion, Yuan and Yin provided in[49] the first rigorous proof of the weak convergence of PSO to a global optimum, without overly restrictive assumptions. Moreover, using stochastic approximation technique, they provide the rate of convergence. In another direction,[50] introduced two smoothed versions of PSO and prove that they converge almost surely to a cost function's global optimum.

The next proposition shows that if we modify the basic PSO algorithm by replacing the update step 2 by the new noise enhanced update step 2', the modified mPSO algorithm converges to a global minimum of the cost function almost surely

Proposition 3 *Let f be a continuous function defined on a compact subset $\mathcal{D}$ of $\mathbb{R}^d$ and one of its minimizers is in the interior of $\mathcal{D}$. Then the algorithm hmPSO converges to $\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ almost surely.*

Proof By Theorem 1, it suffices to verify that hmPSO satisfies (C1)–(C3). Note that $\mathbf{x}_i(t+1) \in \mathcal{D}$ by projection. As $\mathbf{x}_i^*(t+1)$ takes either value $\mathbf{x}_i^*(t)$ or $\mathbf{x}_i(t+1)$, therefore $\mathbf{x}_i^*(t+1) \in \mathcal{D}$ if $\mathbf{x}_i^*(t) \in \mathcal{D}$. Similar argument applies to $\mathbf{x}^*(t+1)$, so (C1) holds by induction. Since we add noise to the exploration agents $\mathbf{x}_i$ $(1 \le i \le n/2)$. Proposition 2 implies (C2) holds. Consider the global memory agent, by step 4, $f(\mathbf{x}^*(t+1)) \le f(\mathbf{x}^*(t))$, so (C3) is verified. This completes the proof. □

BAT and hmBAT. The success of bats, dolphins, shrews and other animals in applying echolocation technique to hunt and navigate inspired[51] to propose the Bat Algorithm (BAT) in 2010. BAT meets with rising popularity in applications. According to a recent search of the Web of Science, over 1,500 articles, nearly 800 proceedings papers and 48 review papers have been written on this algorithm and its variants in just ten years. We refer interested readers in BAT and its wide ranging applications to the review articles in[52,53] and the references therein. Recent applications of the BAT algorithm include[54,55]. As with other metaheuristic algorithms, BAT has many modified or hybridized versions for improved performance in various ways; some examples of the modified versions can be found in[56–60] and some examples of BAT algorithm hybridized with another metaheuristic algorithm are[61–63].

Let $\mathbf{x}_i(t), 1 \le i \le n$, denote the position of the i-th bat at t-th iteration, and $\mathbf{x}^*(t)$ tracking the best position found by the cauldron of n bats. Recall BAT runs the following iterations iteratively:

1. Generate n independent U_i from the uniform distribution on $[f_{\min}, f_{\max}]$, $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + U_i(\mathbf{x}_i(t) - \mathbf{x}^*(t))$.
2. Let r_0 (pulse rate) denote a user-specified threshold. Generate n independent random numbers r_i from the uniform distribution on $[0, 1], 1 \le i \le n$. Update according to (a) $r_i < r_0$ or (b) $r_i \ge r_0$:
   (a) If $r_i < r_0$, $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$.
   (b) If $r_i \ge r_0$, $\mathbf{x}_i(t+1) = \mathbf{x}^*(t) + \epsilon_i(t)$, where the components of $\epsilon_i(t)$ are independent mean 0 normal distributions with standard deviation 0.001.
3. Generate n independent random numbers r_i from the uniform distribution on $[0, 1]$. If r_i is less than a threshold r_A (loudness), or $f(\chi_{\mathcal{D}}(\mathbf{x}_i(t))) < f(\mathbf{x}_i(t+1))$ update $\mathbf{x}_i(t+1) = \chi_{\mathcal{D}}(\mathbf{x}_i(t))$.
4. Update global best $\mathbf{x}^*(t+1) = \mathbf{x}_{i^*}(t+1)$, where $i^* = \arg\min_i f(\mathbf{x}_i(t+1))$.

In the procedure above, $f_{\min}, f_{\max}$ are given constants which describe the minimal and maximal frequencies. The threshold probabilities r_0 and r_A describe the pulse and emission rates. The random noise $\epsilon_i(t)$ describes the loudness of each bat.[51] also suggests using time varying $\epsilon_i(t)$ and r_0 for improved performance. Here we fix them as in the standard values suggested by the MATLAB package provided by the same author.

Our modified BAT, denoted by hmBAT, is to apply the HPP strategy by an additional step after step 3:

3') Update, for $1 \le i \le n/2$,

$$\mathbf{x}_i(t+1) = \chi_{\mathcal{D}}(\chi_{\mathcal{D}}(\mathbf{x}_i(t) + \mathbf{v}_i(t+1)) + \mathbf{w}_i(t+1)). \tag{14}$$

Proposition 4 *Let f be a continuous function defined on $\mathcal{D}$, a compact subset of $\mathbb{R}^d$. Suppose further that a minimizer of f lies in the interior of $\mathcal{D}$, then the algorithm hmBAT converges to $\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ almost surely.*

Proof By Theorem 1, it suffices to verify hmBAT satisfies (C1)–(C3). Note that $\mathbf{x}_i(t+1) \in \mathcal{D}$ by projection. As $\mathbf{x}^*(t+1)$ takes either value $\mathbf{x}^*(t)$ or $\mathbf{x}_i(t+1)$ for some i, therefore it is in $\mathcal{D}$ if $\mathbf{x}^*(t) \in \mathcal{D}$. So (C1) holds by induction.

To verify (C2), we denote the value of $\mathbf{x}_i(t+1)$ after step 3') as $\mathbf{y}_i(t)$, which is given by equation (14). Following the proof of Proposition 2, we can show that for any d-ball $B \subset \mathcal{D}$, $\mathbb{P}(\mathbf{y}_i(t) \in B) \ge \alpha|B|$. If all $\mathbf{y} \in B$ satisfies $f(\mathbf{y}) < f(\mathbf{x}_i(t))$, we have

$$\mathbb{P}(\mathbf{x}_i(t+1) \in B) \ge (1 - r_A)\mathbb{P}(\mathbf{y}_i(t) \in B) \ge \alpha(1 - r_A)|B|. \tag{15}$$

So (C2) holds.

By step 4, $f(\mathbf{x}^*(t+1)) \le f(\mathbf{x}^*(t))$, so (C3) is verified. This completes the proof of Proposition 4. □

ACO and hmACO. Ant colony Optimization (ACO) was initially proposed to solve discrete optimization such as the traveling salesman problem[64,65]. Later, it is adapted to solve continuous optimization problems in[66]. We focused our discussion of ACO on this adaptation, since our discussion so far was mainly about continuous optimization.

We shall call the particles in ACO ants. At each iteration, ACO maintains an archive of n best ants $\mathbf{x}_1(t), \ldots, \mathbf{x}_n(t)$. Each ant is a d-dimensional vector with components $\mathbf{x}_i(t) = (x_{i,1}(t), \ldots, x_{i,d}(t))$. Without loss of generality, we assume the ants are ranked so $f(\mathbf{x}_1(t)) \le f(\mathbf{x}_2(t)) \le \cdots \le f(\mathbf{x}_n(t))$. In the beginning of a new iteration, ACO generates m new ants by sampling Gaussian mixture distribution at each dimension. The Gaussian mixtures consist of Gaussian distributions centered at each archived ant with weights according to the ant's performance. After the new ants are generated, they are inserted to the archive, and the archive will then remove m ants with worst performances. Specifically, ACO repeats steps 1 to 4 until termination criterion is met.

1. For each dimension j, construct a 1-dimensional Gaussian mixture p_j density centered at $\{x_{1,j}(t), \ldots, x_{n,j}(t)\}$

$$p_j(x) = \sum_{i=1}^{n} \frac{w_i}{\sigma_{i,j}\sqrt{2\pi}} \exp\left(-\frac{(x - x_{i,j}(t))^2}{2\sigma_{i,j}^2}\right). \tag{16}$$

The weight w_i is a decreasing sequence so better performing ants are favored.[66] suggests using

$$w_i = \frac{\exp(-(i-1)^2/(2q^2n^2))}{\sum_{i=1}^{n}\exp(-(i-1)^2/(2q^2n^2))}, \tag{17}$$

with certain tuning parameter q. Smaller q will give more preference to the best performing ant $\mathbf{x}_1$. The standard deviation is suggested to be chosen as $\sigma_{i,j} = \sigma\sum_{k=1}^{n}|x_{i,j}(t) - x_{k,j}(t)|/(n-1)$, with σ being a tuning parameter.

2. Generate m new ants $\mathbf{y}_1(t),\ldots,\mathbf{y}_m(t)$ independently. Each follows the product distribution $\mathbf{y}_i(t) \sim \prod_{j=1}^{d} p_j(x), i = 1,\ldots,m$.
3. Combine them with existing archived ants and reorder them. That is, we let

$$\{\mathbf{x}'_1,\ldots,\mathbf{x}'_{n+m}\} = \{\mathbf{x}_1(t),\ldots,\mathbf{x}_n(t),\mathbf{y}_1(t),\ldots,\mathbf{y}_m(t)\} \tag{18}$$

where $f(\mathbf{x}'_1) \le f(\mathbf{x}'_2) \le \cdots \le f(\mathbf{x}'_{n+m})$.

4. Update archive with the best ants $\mathbf{x}_i(t+1) = \mathbf{x}'_i$ for $i = 1,\ldots,n$.

Our modified ACO, denoted by hmACO, is to apply the HPP strategy in step 2):

2') Generate m new ants $\mathbf{y}_1,\ldots,\mathbf{y}_m$ independently by first sampling from the product distribution $\mathbf{y}'_i(t) \sim \prod_{j=1}^{d} p_j(x), i = 1,\ldots,m$. Then, let

$$\mathbf{y}_i(t) = \chi_{\mathcal{D}}\big(\chi_{\mathcal{D}}\big(\mathbf{y}'_i(t)\big) + \mathbf{w}_i(t)\big)\big) \tag{19}$$

and assume that the first batch of ants $\mathbf{x}_i(0)$ are in the feasible set $\mathcal{D}$. Otherwise, replace them with $\chi_{\mathcal{D}}(\mathbf{x}_i(0))$.

We show that hmACO converges almost surely as long as $m < n/2$. This setting is relevant in practice, since the computational cost comes mainly from the new ants generation and evaluations. Intuitively, having many new ants without updating the archive will make the algorithm less efficient. For example, Socha and Dorigo[66] suggests using $m = 2$ and $n = 50$.

Proposition 5 *Let f be a continuous function over $\mathcal{D}$, a compact subset of $\mathbb{R}^d$. Suppose further that a minimizer of f lies in the interior of $\mathcal{D}$, then the algorithm hmACOconverges to $\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x})$ almost surely if $m < n/2$.*

Proof By Theorem 1, it suffices to prove that mACO satisfies (C1)–(C3). Note that $\mathbf{x}_i(t+1)$ is either $\mathbf{y}_j(t)$ or $\mathbf{x}_j(t)$ for some j, the former is in $\mathcal{D}$ by projection, so (C1) holds by induction. Since $m < n$ and only m ants will be removed from $\{\mathbf{y}_1(t),\ldots,\mathbf{y}_m(t),\mathbf{x}_1(t),\ldots,\mathbf{x}_n(t)\}$, $\mathbf{x}_1(t)$ remains in the archive at iteration $t+1$. This implies (C3) holds.

For any d-dimensional ball B in $\mathcal{D}$, let $\mathbb{P}'_t$ be the conditional probability given $\mathbf{y}'_1(t),\ldots,\mathbf{y}'_m(t)$, then for any j

$$\mathbb{P}'_t(\mathbf{y}_j(t) \in B) \ge \mathbb{P}'_t(\chi_{\mathcal{D}}(\mathbf{y}'_j(t)) + \mathbf{w}_j(t) \in B) = \int_B p_t(\chi_{\mathcal{D}}(\mathbf{y}'_j(t)) - \mathbf{x})d\mathbf{x} \ge \alpha|B|. \tag{20}$$

Note that if $f(\mathbf{y}) \le f(\mathbf{x}_1(t))$ for all $\mathbf{y} \in B$ and $\mathbf{y}_j(t) \in B$, we have $f(\mathbf{y}_j(t)) \le f(\mathbf{x}_1(t))$ and $\mathbf{y}_j(t)$ will be kept in the archive at iteration $t+1$. Consequently, $\mathbb{P}'_t(\mathbf{x}_j(t+1) \in B \text{ for some } j) \ge \alpha|B|$ and (C2) holds. □

Numerical experiments

We conduct two sets of numerical experiments, Experiment 1 and Experiment 2 to compare the perormance of A and hmA for A = PSO, BAT or ACO. Details are given in "Experiment 1" and Experiment 2" sections. In "Tuning parameter analysis for HPP strategy" section, we report the analysis of tuning parameters used in HPP strategy. The main objective of our experiments is to compare the performance of A and hmA when A is PSO, BAT or ACO; it is not our intention to compare PSO, BAT and ACO.

The swarm size used is 32 in all the comparison studies below. At each run, we start the algorithms A and hmA with the same initialization of the particles in the swarm.

Experiment 1. In this experiment, we conduct extensive numerical experiments to compare the performance of A and hmA, where A = PSO, BAT or ACO, for a computing budget on a large class of very different types of cost functions. We use two measures to evaluate how likely and by how much the modified algorithm outperforms the original algorithm.

Test functions and details of Experiment 1. A total of 28 test functions are used for this experiment. The functions are compiled by Wahab et al.[67], which are of varying dimensions and their exact expressions can be found in Supplementary Table 1 in Supplementary Material: A. The list of functions includes commonly used test functions such as Ackley, Griewank, Powell, Rastrigin, Sphere with different properties. Some are unimodal, multi-modal, separable or inseparable. The majority of the functions are defined for arbitrary dimension d. For such functions, we investigate the effect of dimension on the optimization algorithms for selected dimensions, i.e., $d = 5, 10, 20$ and 40. The other test functions have dimensions $d = 2$, except for one with $d = 4$, which we exlcude. In total, we have a set $\mathcal{C}$ of 70 test functions and we summarize the numerical results according to their dimensions, i.e., $d = 2, 5, 10, 20$ and 40.

For each cost function $f \in \mathcal{C}$, we conduct 100 runs of the algorithms A and hmA, each for 10,000 iterations, where A = PSO, BAT, or ACO. To gauge the progress of each run of an algorithm, we output the algorithm's best functional values found during the search at the tth iteration, where $t = 50, 100, 200, 400, 1000, 3000$ and $10,000$.

For all experiments, we use $n = 32$ agents/particles and for hmA (A = PSO, BAT or ACO). We employ commonly used parameter settings for PSO, BAT and ACO, namely,

- PSO: $w = 0.729$ (inertia weight), $c_1 = c_2 = 1.5$ (acceleration constants);
- BAT: $f_{\min} = 0$ (minimum frequency), $f_{\max} = 100$ (maximum frequency), $r_0 = 0.5$ (pulse rate), $r_A = 0.5$ (loudness);
- ACO: $q = 10^{-4}, \sigma = 0.85, n = 32, m = 2$.

For hmA, we use the same parameters as in A with the stochastic perturbations being independent normal variates, each with mean 0 and standard deviation 0.005.

Comparison and results. This subsection compares the performance of an algorithm A versus one of its modification B using several test functions from the set $\mathcal{C}$, functions in the set are commonly used to compare performance of an algorithm in the engineering literature. We are particularly interested to ascertain how their performance depends on the dimension of the problem. Here, $\mathcal{F}$ is the subset of $\mathcal{C}$ that contains functions of the same dimension $d = 2, 5, 10, 20$ or 40. Our two key questions are:(a) How likely does B outperform A and (b) On average, how much does B outform A at termination, in terms of the criterion value and $B = hmA$.

(a) *How likely will B outperform A?*

Let $A_f(t)$ and $B_f(t)$ denote the (theoretical) best functional values from algorithms A and B, respectively, for the test function f, up to the tth iteration. Similarly, let $A_{r,f}(t)$ and $B_{r,f}(t)$ denote the best functional values from the algorithms A and B, respectively, up to the tth iteration at the rth run. The probability that algorithm B outperforms algorithm A for f at the tth iteration is $\mathbb{P}(B_f(t) < A_f(t))$, and it can be estimated by $\sum_{r=1}^{100} I(B_{r,f}(t) < A_{r,f}(t))/100$. The average of these estimates over $\mathcal{F}$, denoted by $P_{B\succ A}(t)$, can be thought of as an average winning proportion of B over A (or simply, winning proportion). Specifically,

$$P_{B\succ A}(t) := \frac{1}{100|\mathcal{F}|} \sum_{f\in\mathcal{F}} \sum_{r=1}^{100} I(B_{r,f}(t) < A_{r,f}(t)). \tag{21}$$

Similarly, interchanging A and B above, we define the winning proportion of A over B, $P_{A\succ B}(t)$. Since $P_{A\succ B}(t) + P_{B\succ A}(t) = 1$, it suffices to track $P_{B\succ A}(t)$. Loosely speaking, if $P_{B\succ A}(t) > 1/2$, B is more likely to outperform A for $f \in \mathcal{F}$. Indeed, the larger the value of $P_{B\succ A}(t)$, the more likely it is that B outperforms A.

Supplementary Figure 1 in the Supplementary Material displays the comparison results and we observe the following. When t is fixed, the winning proportion of HPP modified algorithm outperforming its counterpart increases as the dimension of the test function increases. Both mBAT and hmBAT are comparable in their performance against BAT, and the winning proportion increases steadily from about 0.5 to 0.7 for dimension at least 5. Interestingly, the HPP strategy has more significant enhancement effect on PSO and ACO.

(b) *How much does B (or A) outperform A (or B)?*

For simplicity, we suppress the dependence of a notation on other parameters/notations when the context is clear. To this end, let m_* and m^* denote respectively, the minimum (i.e., the best) and maximum (i.e., the worst) of the outputs from 100 runs of A and 100 runs of B for f at the tth iteration. We view m_* as a proxy of the best possible output by A and B; and let $R = m^* - m_*$ denote the range of the 200 outputs of A and B. Define the relative error of A with respect to A and B, $RE_{A,A\cdot B,f}(t)$, to be the average of $(A_{r,f}(t) - m_*)/R$ over 100 runs; and the relative error of B relative to A and B, $RE_{B,A\cdot B,f}(t)$ is similarly defined. We consider the relative instead of absolute error in order to cancel out the effect due to a scale change of f (i.e., if we consider cf instead of f). For brevity, we simply call this quantity the relative error. Note that the relative errors lie between 0 and 1. Small relative error of A (i.e., $RE_{A,A\cdot B,f}(t)$) indicates the results from algorithm A, as a whole, are closer to the best possible value m_*. Further, if the relative error of A is less than that of B, then generally results from algorithm A are closer to m_* than those from B's. A similar interpretation can be extended over a class of functions $f \in \mathcal{C}$ if we define $RE(A, A\cdot B)(t) := \frac{1}{|\mathcal{F}|}\sum_{f\in\mathcal{F}} RE_{A,A\cdot B,f}(t)$ and call this the overshoot of A relative to A and B.

Supplementary Figure 2 in the Supplementary Material displays various performances of the PSO, BAT and ACO algorithms listed across columns relative to their modified algorithms in their first 10,000 iterations. Each row displays summarized results from test functions grouped by one of their dimensions with $d = 2, 5, 10, 20$ and 40 resulting in 5 subfigures per column. Each subfigure displays the relative error of (i) A relative to A and hmA combined (red curve), and (ii) hmA relative to A and hmA combined (blue curve). If the blue (respectively, red) curve is lower, it indicates the hmA (respectively, A) is more superior than A. The subfigures are informative. For example, from subfigures in the rows for $d = 10$ and $d = 40$, we observe that: (i) hmBAT outperforms BAT by a significant margin for all dimensions d; (ii) When $d = 40$, relative errors of hmPSO are nearly 0 from 1000 iterations onwards. This implies that hmPSO significantly outperforms the basic PSO from 1000 iterations; (iii) At $d = 10$, hmA outperforms A for A = PSO, BAT and ACO from 400 iterations.

Combining the observations in (a) and (b), Supplementary Figures 1 and 2 convincingly demonstrate that the HPP strategy significantly improves BAT, ACO and PSO, particularly when the dimension of the test function is large. Improvement is measured in two complementary aspects, "how likely" and "how much" one algorithm outperforms the other.

Experiment 2. Upon the suggestion of a reviewer, we conduct numerical experiment 2 using selected benchmark functions in[68,69]. These functions are intended for the CEC2017 competition in a special session on constrained single objective real-parameter optimization. Since the focus of our paper is on unconstrained optimization, we select optimization problems in[69] with inequality constraints for experiment 2 and transform them to unconstrained optimizations by modifying the original objective function $f(\mathbf{x})$ in the problem

$$\min\{f(\mathbf{x}) : \mathbf{x} \in [-a, a]^d\} \text{ such that } g_k(\mathbf{x}) \leq 0, \text{ for } 1 \leq k \leq K;$$

to one with the objective function $f_+(\mathbf{x})$ given by

$$f_+(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{if } g_k(\mathbf{x}) \leq 0, 1 \leq k \leq K, \\ +\infty, & \text{otherwise}. \end{cases} \tag{22}$$

Benchmark functions and details of Experiment 2. Benchmark constrained optimization problems, C01, C02, C04, C05 and C20 in[69] are chosen for comparing the performance of A and hmA where $A =$ PSO, BAT or ACO using the device in (22), who also introduced random translation, $\mathbf{x} - \mathbf{o}$, followed by random rotations of the translated variables by randomly generated orthogonal matrices, in addition to the constrained optimization.

Throughout the comparisons, swarm size used is 32. At each run of A and hmA, we start them with the same initialization. Parameters used for PSO, hmPSO, BAT, hmBAT, ACOand hmACOfollow that in "Test functions and details of Experiment 1" section. Dimension, d, of the objective functions for comparison are chosen to be 10, 30, 50 and 100.

Results. Following[69], we tabulate the algorithm's best, median, worst, average, standard deviations function values at the 5000th and 10,000th iterations out of 100 runs of the algorithms. The results, rounded to three decimal places, are presented in Supplementary Tables 3, 4 and 5 in the Supplementary Material, respectively, for comparing PSO veruse hmPSO, BAT versus hmBAT and ACO versus hmACO. After rounding the outputs from the algorithms in each run at either 5000th or 10,000th iteration, we compare which algorithm, A or hmA, gives better result (ties are ignored) and give the winning algorithm one vote. The rows, labeled as 'Wins', report the total winning votes of an algorithm. Entries that are in boldface denote the better results. Some of the entries are reported as 'Inf' due to one or more of the constraints are not satisfied and hence the corresponding f_+ takeing infinity as the function value.

We have the following general observations from the results. (i) From Supplementary Table 3, hmPSO outperforms PSO substantially, particularly so when the dimension of the benchmark function increases. Notably, the median and the mean of hmPSO best function values, even at halfway point, are substantially better, suggesting hmPSO give more consistent results across runs. The consistency of hmPSO results across runs of the algorithm can also be seen from the standard deviations, which are markedly smaller than those of PSO. (ii) Above observations also apply to hmACO as compared with ACO from Supplementary Table 5 though not as striking as in the hmPSO case. For benchmark function C20, the two algorithms, hmACOand ACO, are close in terms of their performance. There are also runs in which both ACO and hmACO do not satisfy some of the constraints in the optimization of benchmark functions C01 and C02. (iii) From Supplementary Table 4, hmBAT marginally outperforms BAT.

Tuning parameter analysis for HPP strategy. As demonstrated in "Comparison and results" section, having an additional stochastic component in a modified swarm-based algorithm can enhance the exploration ability of the algorithm. The stochastic component chosen is Gaussian with mean 0 and standard deviation $\sigma = 0.005$. We expect the noise level, as indicated by the value of σ or a heavier-tailed distribution, affects the efficiency of the modified algorithm since strong level of noise interferes the algorithm's exploitation ability. We conduct further numerical experiments with the same set-up as above to investigate the effects of σ in the Gaussian noise or having t-distributions that have heavier tails than the Gaussian on the performance of mA and hmA.

(i) *On the standard deviation of the Gaussian noise* We tried $\sigma = 0.005, 0.01, 0.02$ and 0.05. The plots of $P_{B \prec A}(t)$ (analogous to Supplementary Figure 1 and $RE(A, A \cdot B)(t)$ (analogous to Supplementary Figure 2) are very similar for hmA ($A =$ PSO, BAT and ACO) for all the σs; and $\sigma = 0.005$ and 0.01. Hence, we recommend using σ in [0.005, 0.01]. Due to space consideration, these plots are given in Supplementary Figure 3 of the Supplementary Material.

(ii) *On the choice of distribution* We examine the effect on the performance of the modified algorithms if we change the Gaussian distribution with a heavier-tailed distribution. We choose distributions $0.01\sqrt{(m-2)/m}\, t_m$ for $m = 5, 10, 30$ and 60 where t_m denotes the t-distribution with m degrees of freedom. The scaling factor $0.01\sqrt{(m-2)/m}$ is to ensure the scaled t-distribution has the same standard deviation as that of the Gaussian noise in (i) above. Note also that t_m converges in distribution to a standard normal and the scaling factor to 1 as m approached to infinity. For m large, we do not expect essential difference between the t-distribution and the Gaussian noise.

The analogous plots of $P_{B \prec A}(t)$ and $RE(A, A \cdot B)(t)$ using t-distributions for the choices of degrees of freedom look almost the same as the corresponding plots using Gaussian noise. These plots are not included in this article due to space constraints. Interested readers can view these plots in Supplementary Figure 4 of the Supplementary Material.

Conclusions

Metaheuristic algorithms have found wide ranging applications in optimization problems. However, majority of these algorithms have no guarantee of converging to a global optimum, which is a desirable feature to have. In this paper, we propose an intuitive modification of an algorithm to ensure this global convergence capability. Moreover, we provide a rigorous mathematical proof of this guarantee. We first delineate in Theorem 1 sufficient conditions, namely (C1)–(C3), for a swarm-based algorithm to converge almost surely to a global optimum of an objective function. The class of objective functions for which Theorem 1 holds is very large. A heterogeneous perturbation-projection (HPP) strategy is then proposed to modify a given swarm-based algorithm to ensure the sufficient conditions (C1)–(C3) hold, and hence Theorem 1 follows, guaranteeing the modified algorithm converges almost surely to a global optimum. The proposed strategy is natural and simple to implement yet not algorithm-specific. We also demonstrated how this strategy is applied to PSO, Bat and ACO algorithms. Extensive numerical experiments were conducted and the results show that the modified algorithm either outperforms or performs on par with the original algorithm over a finite computational budget. Indeed, our method of proof leads to a form of finite iteration analyses (see the paragraph before "Perturbation-projection strategy" section). The tuning parameter analyses in "Tuning parameter analysis for HPP strategy" section suggest that applying the HPP strategy with mean 0 and standard deviation 0.005 Gaussian distribution for the additional stochastic component helps to improve swarm-based algorithms. We contend that even though we have proved the proposed algorithms theoretically converge to the global optimum, the numerics may not, in practice, always converge to the global minimum for all problems. Our theoretical results only discuss unconstrained optimization problems, which already include a wide range of practical applications. Additionally, we show in the second numerical experiment that the proposed algorithms can also be applied to inequality constrained optimization problems through a simple conversion.

We conclude this paper with an interesting observation from the results of our two experiments: When A outperforms hmA, A frequently only produces a marginally better result; however, when hmA outperforms A, the modified version quite often outperforms A by a relatively large margin. We offer a heuristic argument for such plausibility. When both A and hmA are exploring in the same neighbourhood of a local minimum, the stochastic element hinders the exploitation by a small amount and hence A is more likely to produce marginally better solution. However, when the occasion arises for hmA to make a big jump, thus leaving a local minimum's neighbourhood, hmA has the potential to explore a better region and hence produces noticeably better solution than the original algorithm does. This observation reinforces the idea that adding noise appropriately to metaheuristic algorithms can enhance their performance.

Consent for participate. All authors have given consent to submit to this journal for review.

Data availibility

The datasets used and/or analysed during the current study available from the corresponding author on reasonable request.

Code availability

Will be provided whent he paper is accepted.

Received: 24 July 2022; Accepted: 7 February 2023
Published online: 31 March 2023

References

1. Whitacre, J. M. Recent trends indicate rapid growth of nature-inspired optimization in academia and industry. *Computing* **93**, 121–133 (2011).
2. Whitacre, J. M. Survival of the flexible: Explaining the recent popularity of nature-inspired optimization within a rapidly evolving world. *Computing* **93**, 135–146 (2011).
3. Kim, S. & Li, L. A novel global search algorithm for nonlinear mixed-effects models using particle swarm optimization. *J. Pharmacokinet. Pharmacodyn.* **38**, 471–495 (2011).
4. Dev, K. *et al.* Energy optimization for green communication in IOT using Harris hawks optimization. *IEEE Trans. Green Commun. Netw.* **6**, 685–694 (2022).
5. Gundluru, N. *et al.* Enhancement of detection of diabetic retinopathy using Harris hawks optimization with deep learning model. *Comput. Intell. Neurosci.* (2022).
6. Mendes, J. M., Oliveira, P. M., Filipe Neves, F. N. & dos Santos, R. M. Nature inspired metaheuristics and their applications in agriculture: A short review. In *EPIA Conference on Artificial Intelligence EPIA 2019: Progress in Artificial Intelligence* (2020).
7. Gad, A. G. Particle swarm optimization algorithm and its applications: A systematic review. *Arch. Comput. Methods Eng.* **29**, 2531–2561 (2022).
8. Huang, C., Li, Y. & Yao, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans. Evol. Comput.* **24**, 201–216 (2019).
9. Merkle, D. & Middendorf, M. Swarm intelligence. In *Search Methodologies*, 401–435 (Springer, 2005).
10. Kaur, A. & Kaur, M. A review of parameters for improving the performance of particle swarm optimization. *Int. J. Hybrid Inf. Technol.* **8**, 7–14 (2015).
11. Yang, X.-S., Deb, S., Loomes, M. & Karamanoglu, M. A framework for self-tuning optimization algorithm. *Neural Comput. Appl.* **23**, 2051–2057 (2013).
12. Yang, X.-S. & He, X. Swarm intelligence and evolutionary computation: Overview and analysis. In *Recent Advances in Swarm Intelligence and Evolutionary Computation*, 1–23 (2015).
13. Choi, K. P., Lai, T. L., Tong, X. T. & Wong, W. K. A statistical approach to adaptive parameter tuning in nature-inspired optimization and optimal sequential design of dose-finding trials. *Stat. Sinica* **31**, 1 (2021).
14. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M. & Stützle, T. The Irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **3**, 43–58 (2016).

15. Birattari, M., Balaprakash, P. & Dorigo, M. The aco/f-race algorithm for combinatorial optimization under uncertainty. http://link.springer.com/article/10.1007 (2007).
16. Parsopoulos, K. E., Plagianakos, V., Magoulas, G. & Vrahatis, M. Objective function "stretching" to alleviate convergence to local minima. *Nonlinear Anal. Theory Methods Appl.* **47**, 3419–3424 (2001).
17. Stacey, A., Jancic, M. & Grundy, I. Particle swarm optimization with mutation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, Vol. 2, 1425–1430 (IEEE, 2003).
18. Krohling, R. A. Gaussian particle swarm with jumps. In *2005 IEEE Congress on Evolutionary Computation*, Vol. 2, 1226–1231 (IEEE, 2005).
19. Elshamy, W., Emara, H. M. & Bahgat, A. Clubs-based particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, 289–296 (IEEE, 2007).
20. Ding, K. & Tan, Y. Comparison of random number generators in particle swarm optimization algorithm. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC)* 2664–2671 (2014).
21. Sun, J., Wu, X., Palade, V., Fang, W. & Shi, Y. Random drift particle swarm optimization algorithm: Convergence analysis and parameter selection. *Mach. Learn.* **101**, 345–376 (2015).
22. Neelakantan, A. *et al.* Adding gradient noise improves learning for very deep networks. *ICLR* (2015).
23. Selman, B., Kautz, H. A. & Cohen, B. Noise strategies for improving local search. In *AAAI-94 Proceedings* 337–343 (1994).
24. Chen, X., Du, S. S. & Tong, X. T. On stationary-point hitting time and ergodicity of stochastic gradient Langevin dynamics. *J. Mach. Learn. Res.* **21**, 1–40 (2020).
25. Zhou, M. *et al.* Towards understanding the importance of noise in training neural networks. In *Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97* (2019).
26. Jin, C., Ge, R., Netrapalli, P., Kakade, S. M. & Jordan, M. J. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70* (2017).
27. Zhang, W., Xie, X. & Bi, D. Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Vol. 2, 2307–2311 (2004).
28. Padhye, N., Deb, K. & Mittal, P. Boundary handling approaches in particle swarm optimization. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), Advances in Intelligent Systems and Computing*, Vol. 201 (eds. J. C. Bansal et al.) 287–298 (2013).
29. Chu, W., Gao, X. & Sorooshian, S. Handling boundary constraints for particle swarm optimization in high-dimensional search space. *Inf. Sci.* **181**, 4569–4581 (2011).
30. Maurice, C. *Standard Particle Swarm Optimisation* (Technical Report, HAL Achives Ouvertes, 2012).
31. Qiu, J., Chen, R.-B., Wang, W. & Wong, W. K. Using animal instincts to design efficient biomedical studies via particle swarm optimization. *Swarm Evol. Comput.* **18**, 1–10 (2014).
32. James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning*, Vol. 112 (Springer, 2013).
33. Shalev-Shwartz, S. & Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press, 2014).
34. Ge, R., Huang, F., Jin, C. & Yuan, Y. Escaping from saddle points: Online stochastic gradient for tensor decomposition. In *Proceedings of the Conference on Learning Theory* (2015).
35. Chen, X., Du, S. S. & Tong, X. T. On stationary-point hitting time and ergodicity of stochastic gradient Langevin dynamics. *J. Mach. Learn. Res.* **21**, 1–41 (2020).
36. Van den Bergh, F. & Engelbrecht, A. P. Cooperative learning in neural networks using particle swarm optimizers. *S. Afr. Comput. J.* **2000**, 84–90 (2000).
37. Van den Bergh, F. & Engelbrecht, A. P. A cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput.* **8**, 225–239 (2004).
38. Zhang, X. *et al.* Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties. *IEEE Trans. Cybern.* **50**, 4454–4468 (2019).
39. Dong, J. & Tong, X. T. Replica exchange for non-convex optimization. *J. Mach. Learn. Res.* **22**, 1–59 (2021).
40. Kennedy, J. & Eberhart, R. C. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks IV*, 1942–1948 (IEEE, 1995).
41. Shi, Y. & Eberhart, R. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*. https://doi.org/10.1109/ICEC.1998.699146 (1998).
42. Bratton, D. & Kennedy, J. Defining a standard for particle swarm optimization. In *2007 IEEE swarm intelligence symposium*, 120–127 (IEEE, 2007).
43. Poli, R. Analysis of the publications on the applications of particle swarm optimization. *J. Artif. Evol. Appl.* (2008).
44. Eberhart, R. C. & Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 84–88 (IEEE, 2000).
45. Clerc, M. & Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **6**, 58–72 (2002).
46. Trelea, I. C. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Inf. Process. Lett.* **85**, 317–325 (2003).
47. Chen, X. & Li, Y. A modified PSO structure resulting in high exploration ability with convergence guaranteed. *IEEE Trans. Syst. Man Cybern. B* **37**, 1271–1289 (2007).
48. Pedersen, M. & Chipperfield, A. J. Simplifying particle swarm optimization. *Appl. Soft Comput.* **10**, 618–628 (2010).
49. Yuan, Q. & Yin, G. Analyzing convergence and rates of convergence of particle swarm optimization algorithms using stochastic approximation methods. *IEEE Trans. Autom. Control* **60**, 1760–1773 (2015).
50. Tong, X., Choi, K. P., Lai, T. L. & Wong, W. K. Stability bounds and almost sure convergence of improved particle swarm optimization methods. In *Research in Mathematical Sciences* (2021).
51. Yang, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, 65–74 (Springer, 2010).
52. Akhtar, S., Ahmad, A. & Abdel-Rahman, E. M. A metaheuristic bat-inspired algorithm for full body human pose estimation. In *2012 Ninth Conference on Computer and Robot Vision*, 369–375 (IEEE, 2012).
53. Yang, X.-S. & He, X. Bat algorithm: Literature review and applications. *Int. J. Bioinspired Comput.* **5**, 141–149 (2013).
54. Cai, X. *et al.* Bat algorithm with gaussian walk for directing orbits of chaotic systems. *Int. J. Comput. Sci. Math.*https://doi.org/10.1504/ijcsm.2014.064070 *(2014).*
55. Xue, F., Cai, Y., Cao, Y., Cui, Z. & Li, F. Optimal parameter settings for bat algorithm. *Int. J. Bioinspired Comput.* **7**, 125. https://doi.org/10.1504/IJBIC.2015.069304 (2015).
56. Osaba, E. *et al.* A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection. In *Swarm and Evolutionary Computation: BASE DATA*. https://doi.org/10.1016/j.swevo.2018.04.001 (2018).
57. Binu, D. & Selvi, M. Bfc: Bat algorithm based fuzzy classifier for medical data classification. *J. Med. Imaging Health Inform.* **5**, 599–606 (2015).
58. Wang, G., Chu, H. & Mirjalili, S. Three-dimensional path planning for UCAV using an improved bat algorithm. *Aerosp. Sci. Technol.*https://doi.org/10.1016/j.ast.2015.11.040 *(2016).*

59. Khooban, M. & Niknam, T. A new intelligent online fuzzy tuning approach for multi-area load frequency control: Self adaptive modified bat algorithm. *Int. J. Electr. Power Energy Syst.* **71**, 254–261 (2015).
60. Lu, S., Xia, K. & Wang, S. Diagnosis of cerebral microbleed via VGG and extreme learning machine trained by gaussian map bat algorithm. *J. Ambient Intell. Hum. Comput.*https://doi.org/10.1007/s12652-020-01789-3 *(2020).*
61. He, X., Ding, W. & Yang, X. Bat algorithm based on simulated annealing and Gaussian perturbations. *Neural Comput. Appl.* **25**, 459–468 (2014).
62. Shrichandran, G., Sathiyamoorthy, S., Malarchelvi, P. & Kezia, S. A hybrid glow-worm swarm optimization with bat algorithm based retinal blood vessel segmentation. *J. Comput. Theor. Nanosci.* **14**, 2601–2611 (2017).
63. Kishore, P., Kishore, S., Kumar, E., Kumar, K. & Aparna, P. Medical image watermarking with DWT-bat algorithm. In *2015 International Conference on Signal Processing and Communication Engineering System* 270–275 (2015).
64. Dorigo, M. *Optimization, Learning and Natural Algorithms*. Ph.D Thesis, Politecnico di Milano (1992).
65. Dorigo, M., Birattari, M. & Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **1**, 28–39 (2006).
66. Socha, K. & Dorigo, M. Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **185**, 1155–1173 (2008).
67. Ab Wahab, M. N., Nefti-Meziani, S. & Atyabi, A. A comprehensive review of swarm optimization algorithms. *PLoS ONE* **10**, e0122827 (2015).
68. LaTorre, A. *et al.* A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. *Swarm Evol. Comput.* **67**, 100973 (2021).
69. Wu, G., Mallipeddi, R. & Suganthan, P. N. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. In *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report* (2017).

Acknowledgements

The authors would like to thank the Institute for Mathematical Sciences at the National University of Singapore for the generous support to host a cross-disciplinary workshop on Particle Swarm Optimization and Evolutionary Computation in 2018, where the authors met and started the work. We are also grateful to Professor T. L. Lai from Stanford University who constantly advised us and collaborated with us on earlier work. We thank the reviewers for their many helpful comments which improve the paper.

Author contributions

K.P.C. and X.T. came up with the ideas for the research work, including the theoretical foundations. E.H.H.K. wrote codes, implemented them and did all the graphical and computation work. W.K.W. provided the overall supervision, did the literature search and wrote the bulk of the manuscript. All authors reviewed and edited parts of the manuscript repeatedly. All authors have given consent.

Funding

Choi's research was supported by the Singapore MOE Academic Research Funds R-155-000-222-114. Tong's research was supported by MOE Academic Research Funds R-146-000-292-114. Dr. Wong received partial support from the National Institute of General Medical Sciences of the National Institutes of Health under Award Number R01GM107639.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-29618-5.

Correspondence and requests for materials should be addressed to W.K.W.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.