Tomas Bata University in Zlín
Faculty of Applied Informatics

Doctoral Thesis

Development and Modification of Modern Bio-Inspired Swarm Algorithms

Vývoj a modifikace moderních bio-inspirovaných hejnových algoritmů

Author:	**Ing. Anezka Kazikova**
Branch of study:	Engineering Informatics
Supervisor:	Assoc. prof. Ing. Roman Senkerik, Ph.D.
Advisor:	Ing. Michal Pluhacek, Ph.D.

Zlín, 2022

I would like to thank with all my heart all that have helped me with my study. Thanks to my husband for all his patience. Thanks to my children. You keep popping up, and I love you. Thanks to my mom-in-law Anna, who helped me with all things great and small. Thank you for letting me pursue this passion of mine. I am thankful to Heather, my VIP reader. And finally, thanks to my mom Eva. I love you, and I miss you.

I would also like to thank my supervisor and advisor for your help and advice, and for keeping me on track whenever I was about to set off on a crusade to save the world. A big thanks goes to the whole A.I. Lab team from Tomas Bata University. You made science great fun.

"Algorithms are conceived in analytic purity in the high citadels of academic research, heuristics are midwifed by expediency in the dark corners of the practitioner's lair...

... and are accorded lower status."

Glover, 1997

"Be the change you want to see in the world."

Unknown

ABSTRAKT

Hejnové algoritmy se staly standardním nástrojem novodobé optimalizace. Příliv nových metaheuristik však přinesl kritiku vůči kvalitě, kvantitě a diskutabilní novosti těchto optimalizačních technik. Tato práce se zabývá momentálními trendy hejnových algoritmů v oblasti vývoje a modifikace, ale i nástrahami, které skýtají.

Už přes 30 let se metaheuristické algoritmy potýkají se stále stejnými problémy. Otázka stagnace, předčasné konvergence či nízké rozlišnosti řešení je výzvou, která je důležitá dnes stejně jako v počátcích oboru. To nemění ani vývoj nových algoritmů, protože ty mnohdy spíše odkrývají limity stávajících metodologických postupů v benchmarkingu, než aby přispívaly ke skutečnému posunu v optimalizaci. Nové metaheuristiky tak čelí předsudkům a všeobecné nedůvěře. Přestože otázka správných postupů je velmi aktuální, většina současných doporučení zůstává zpravidla v teoretické rovině bez praktické aplikace. To si tato práce klade za cíl začít měnit.

Autorka navrhuje sadu doporučení pro vývoj nových metaheuristik, které pak implementuje ve vlastním návrhu hejnového algoritmu s únikovým mechanismem z lokálního optima. Bizoní algoritmus představuje ukázku vývoje orientovaného na konkrétní optimalizační problém a zároveň funguje jako model vybraných aktuálních trendů a modifikací. Spojením teorie s praxí tato práce otevírá cestu k řešení nové generace výzev.

SUMMARY

Swarm algorithms have become standard tools of modern optimization. However, the advent of new metaheuristics brought a wave of criticism against the quantity, quality, and novelty of these optimization techniques. This dissertation describes the current trends in development and modification of swarm algorithms, as well as the challenges it includes.

For several decades metaheuristic algorithms have fought the very same optimization problems. The issues of stagnation, premature convergence, or low diversity of the solutions are dealt with today as well as in the beginning. The development of new algorithms does not state a change. Rather than genuinely advancing the field, new algorithms raise malpractice awareness in benchmarking. Due to the common low standard of their proposal studies, novel metaheuristics face a significant stigma of general distrust and disrespect. Although the good practice in benchmarking is a very recent topic, most current guidelines stay strictly in theory, i.e., are not applied. This work aims to start a change in this regard.

The Author proposes a set of recommendations for new metaheuristic development and implements them in a new swarm algorithm, which was developed with an escape mechanism out of the local optimum containment challenge. The Bison Algorithm showcases problem-oriented development and models current trends and modifications. The connection between theory and practice opens a way toward a new generation of challenges.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
BAT	Bat Algorithm
BFOA	Bacterial Foraging Optimization
BIA	Bison Algorithm
BSA	Bison Seeker Algorithm
CEC	Congress on Evolutionary Computation
CS	Cuckoo Search
D	Dimension
DE	Differential Evolution
EC	Evolutionary Computation
EG	Elite group
ES	Evolutionary Strategies
FFA	Firefly Algorithm
GA	Genetic Algorithm
GWO	Grey Wolf Optimizer
IEEE	Institute of Electrical and Electronics Engineers
NP	Population size
PSO	Particle Swarm Optimization
RG	Running group
SA	Simulated Annealing
SA BIA	Self-Adaptive Bison Algorithm
SG	Swarming Group
SHADE	Success History Based Adaptive Differential Evolution
SI	Swarm intelligence
SDP	Self Driven Particles
SPP	Self Propelled Particles
SR	Symbolic Regression
WOA	Whale Optimization Algorithm
XAI	Explainable Artificial Intelligence

1 Introduction

Artificial Intelligence has become an essential part of human lives today. Sometimes explicitly, sometimes undercover, AI guides us on the road, assists our phones, drives autonomous vehicles, controls our calendar, Google, Netflix, even Facebook have an entire department focused on advancement and more profound research on Artificial Intelligence. As a technology, it helps almost everywhere and it also proves to be a brilliant optimization tool.

The current trend in optimization is to find inspiration in nature. Simulations of various bio-inspired phenomena solve nontrivial optimization tasks. Complex optimization problems may find solutions by simulating diverse natural phenomena like foraging, hunting, courtship behavioral patterns, the Darwinian theory of evolution, and Mendelian genetic processes. The foundation lies in the multi-agent system. Each agent represents one particular solution to the solved problem, whose quality is determined by an objective function.

The bio-inspired optimizers are called metaheuristics. They include a wide range of algorithms like Differential Evolution [371], the Genetic Algorithm [166], Particle Swarm Optimization [219], the Cuckoo Search [439], the Grey Wolf Optimizer [269], the Self-Organizing Migrating Algorithm [443], the Passing Vehicle Search Algorithm [354], and many others.

Unlike classical mathematical optimization methods (e.g., linear, dynamic, or integer programming, [199, 318]), metaheuristics cannot guarantee the actual discovery of optimal solutions. However, as compensation, they offer a viable solution in a reasonable time. Thus, the metaheuristic approach can be constructive, especially when exact mathematical methods struggle to solve a problem in a tolerable period of time.

But even metaheuristics sometimes sail against the wind. There are currently more than 300 algorithms [275], which often face criticism questioning their actual novelty, contribution, and their quality. Many troubles come from fallacious parameter settings, the inapplicability of the proposed algorithms, or embedding the same principles under a new alias [275, 368]. General disrespect for the population-based algorithms is not the only problem they face. Other struggles include standard optimization setbacks such as stagnation, premature convergence, low diversification of the population, or local optimum containment.

This dissertation analyses the current trends in metaheuristic algorithms, including the optimization and development struggles. It investigates specific methods employable to tackle these problems and suggests recommendations for novel metaheuristic development. As proof of concept, adopting the suggested guidelines, the Author introduces a new metaheuristic swarm algorithm with a mechanism against local optimum containment.

The thesis is structured as follows:

- Section 2 reviews metaheuristic algorithms. It describes the popular algorithms' selection, classification systems, current optimization struggles, and types of metaheuristic modifications.

- Section 3 describes a particular category of metaheuristics: swarm algorithms. It presents a selection of popular optimizers and analyses the numerous reservations against the development of novel swarm algorithms. As a reaction, it formulates a set of recommendations for new metaheuristic development.

- Section 4 designates the goals and methods of this dissertation.

- Section 5 proposes a new swarm algorithm based on bison herds' behavioral patterns while adopting previous sections' principles and recommendations.

- Section 6 reviews available modifications of the Bison Algorithm and outlines the development process. The later part of the section analyses the assets which result from the modifications when facing the local optimum containment challenge.

- Section 7 investigates the performance of the proposed algorithm and its modifications. The algorithm is compared to four popular swarm algorithms and two competition winners on the set of problems of IEEE CEC 2015 and 2017 benchmarking test suites. It examines the solution errors, mean convergence, population diversity, and computational complexity of the algorithms, and the benefits of the modifications.
- Section 8 evaluates the benefit of the thesis, considering both the applications and critical points of view.
- Finally, the conclusion contemplates the work and considers its meaning for science and practice.

2 Metaheuristic Optimization

Modern optimization methods fall into two groups: *exact* and *heuristic* methods. Exact mathematical methods guarantee finding the optimal solution to the solved problem. The run-time, though, often increases with higher complexity and dimensionality of the problem. Hence, complicated or large-scale tasks may require a compromise between optimality and computational time. That is when heuristics and metaheuristics come into use [318].

Although metaheuristic solutions lack the guarantee of optimality, they offer a reasonable computation time. Population-based metaheuristics use a multi-agent approach. Every agent represents a solution to the solved problem and is evaluated by the objective function value. This way, metaheuristics operate with a set of (often random) solutions to the problem, right from initialization, and improve them in the process.

The difference between heuristics and metaheuristics lies in problem dependency. While heuristics solve a specific task, metaheuristics apply to a broad range of problems, treating them as black boxes [382].

Metaheuristics typically simulate bio-inspired phenomena. The inspiration comes from both natural (and supernatural) sources, including a wide range of scientific fields such as Physics [41], Chemistry [180], Biology [37], behavioral patterns of animal groups [58, 440], the Darwinian theory of evolution, and Mendelian principles of genetics [166, 371].

Examples of exact optimization methods include linear and integer programming, dynamic programming, branch-and-bound, or Lagrangean relaxation methods [318]. On the other hand, metaheuristics are identified mainly by the inspiration source. Typical instances are Evolutionary Strategies [53], the Genetic Algorithm [166], Particle Swarm Optimization [219], Differential Evolution [371], Ant Colony Optimization[106], the Artificial Bee Colony algorithm [200] or the Cuckoo Search [439]. The exact and heuristic methods do not necessarily have to be segregated. Since both approaches provide advantages, suitable combinations may benefit from synergy [318].

The following chapters review metaheuristics from four points of view:

- Section 2.1 describes the selection of popular metaheuristics, their basic methodology and principles,
- Section 2.2 depicts metaheuristics classification systems,
- Section 2.3 identifies contemporary optimization struggles and corresponding countermeasures that tackle them,
- and Section 2.4 names current modification trends.

2.1 Popular Metaheuristic Algorithms

This section describes the main principles of the most popular metaheuristics. However, it is not easy to select a few algorithms out of the ocean of metaheuristics, let alone measure the popularity of an algorithm. The number of metaheuristic algorithms rises every year. In the *Comprehensive Taxonomies of Nature- and Bio-inspired Optimization*, the authors (Molina et al., 2020) list 324 metaheuristics [275].

The algorithm's popularity is affected by diverse factors including the publication date, the form of the original proposal, naming conventions, and the published language. A very recent algorithm may be disadvantaged compared with well-established algorithms known for decades. The original proposal's form also has an important role – first and foremost if the popularity measure relates to the original publication citation score. Some algorithms do not have one original publication of a new metaheuristic proposal, but the idea is dispersed into several publications. The Evolutionary Strategies may serve as an example, as they were formed in several publications across the 1960s [339, 340, 356].

Tab. 2.1 Top 10 metaheuristics most cited in Scopus database in 30/10/2020 – 1/11/2020 (swarm-based algorithms excluded).

	Algorithm	Year	Reference	Scopus Citations ▼	Inspiration Class
1	**Genetic Algorithm**	1975	[166]	34,159	Breeding
2	**Simulated Annealing**	1983	[220]	26,959	Chemistry
3	**Differential Evolution**	1997	[370]	14,769	Breeding
4	**Self-Driven Particles**	1995	[401]	4,195	Physics
5	**Gravitational Search Algorithm**	2009	[335]	3,237	Physics
6	**Evolution Strategies**	1973	[339]	2,093	Breeding
7	**Biogeography Based Optimization**	2008	[365]	2,082	Breeding
8	**Teaching-Learning Based Optimization Algorithm**	2011	[334]	1,688	Human
9	**Imperialist Competitive Algorithm**	2007	[24]	1,458	Human
10	**Harmony Search**	2005	[233]	1,281	Physics

Another popularity factor is the naming convention: an inappropriate name may discourage researchers from using the algorithm. In some cases, using an unconventional algorithm may even jeopardize the work's reputation. For instance, imagine a proposal for a Covid-19 vaccination designed with the help of the Zombie Survival Optimization algorithm [292]. Finally, the reputation of the inspiration source is essential. For example, when somebody disputes Darwin's theory of evolution [26], it may reflect on the Differential Evolution's popularity.

To the best of the Author's knowledge, there is no universal measure of an algorithm's popularity. Therefore, the most popular metaheuristics were based on the citation score of the algorithms proposal publications. The complete list of metaheuristics (from [275]), with the addition of the citation scores from the Scopus and Google Scholar databases, is in Appendix A.

Table 2.1 presents the top 10 most-cited metaheuristics in the Scopus database. Since the main focus of this work was on swarm algorithms, these were excluded from Table 2.1, and listed separately in Section 3. The following section provides a brief description of the six top-cited algorithms.

2.1.1 Genetic Algorithm

The Genetic Algorithm (GA) was introduced by John Holland in 1975 [166]. The algorithm used Darwin's natural selection principles and was the first to implement mutation, selection, and cross-over techniques in optimization. Figure 2.1 shows the flowchart of the Genetic Algorithm. The individuals – also called chromosomes – were initially represented by binary vectors. The cross-over operation splits two chromosomes in half and swaps the remaining parts; the mutation randomly changes one chromosome gene. The Genetic Algorithm principles have become the starting point of many other metaheuristics [275].

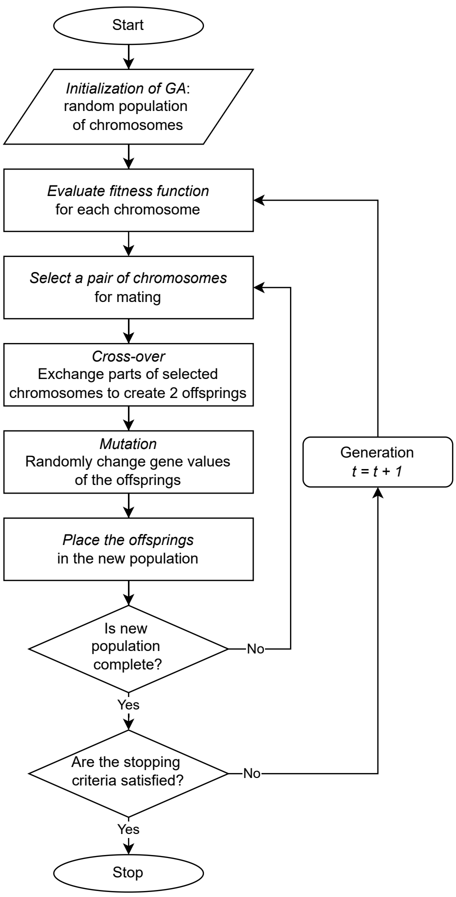


Fig. 2.1 Flowchart of the Genetic Algorithm.

2.1.2 Simulated Annealing

Simulated Annealing (SA) was proposed by Kirkpatrick et al. in 1983 [220] and was inspired by the physical annealing of materials. It provides a specific way to face the local optimum containment problem.

Simulated Annealing allows the degradation of solution quality with a probability affected by the potential quality decrease and the current cooling temperature (Eq. 2.1). Degradation probability tends to be initially high and lowers with more iterations [101]. Figure 2.2 shows a flowchart of Simulated Annealing.

$$P = exp(\frac{-(f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j))}{T}) \tag{2.1}$$

Where:

- $f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)$ is the decrease in the objective function values of the current and potential solutions,
- and T is the cooling temperature parameter.

2.1.3 Differential Evolution

Differential Evolution (DE) was developed by Storn and Price in 1995 [371] and implemented the Genetic Algorithm's basic principles. DE works with individuals of randomly generated vectors. Each generation creates a new population of offsprings with the recombination technique and adopts a better solution from the parent-offspring couple. Differential Evolution employs both mutation and crossover and is considered the pillar of modern optimization [148].

The algorithm operates with two major parameters: the scaling factor F and the cross-over rate CR. The algorithm employs various mutation strategies. The basic one is called the **Rand/1/Bin** and follows three steps: mutation, cross-over, and selection.

In the mutation step, the algorithm randomly selects three solutions from

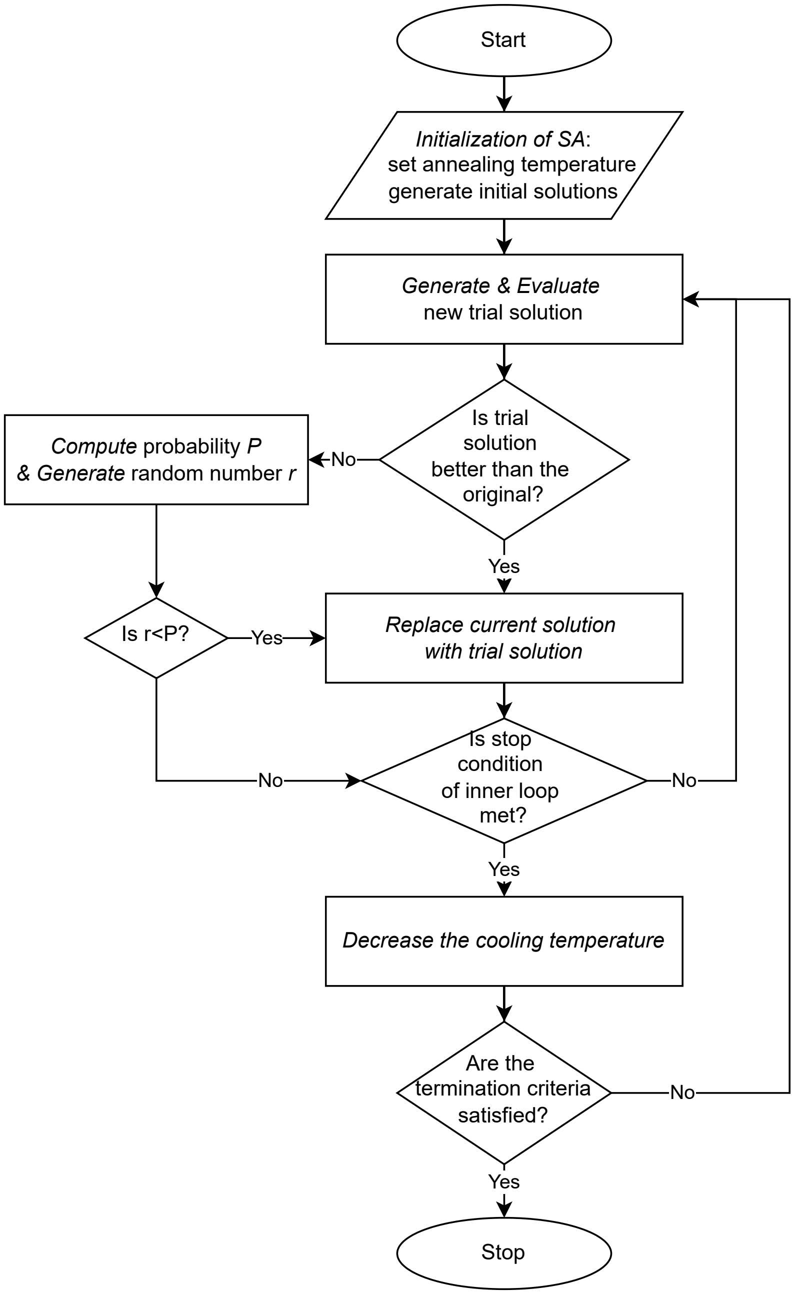


Fig. 2.2 Flowchart of Simulated Annealing.

the population and compiles them to create a **mutation vector** v (Eq. 2.2). **The cross-over step** creates the **trial vector** u, selecting the attributes either from the mutation vector v or the original solution based on the cross-over rate CR (Eq. 2.3). The j_{rand} variable prevents the trial vector from copying the existing solutions. Finally, the selection step selects the better solution from the offspring–parent pair for the next generation.

$$v_{j,i} = x_{r_1,i} + F_i(x_{r_2,i} - x_{r_3,i}) \tag{2.2}$$

$$u_{j,i} = \begin{cases} v_{j,i} & if\ U\,[0,1] \leq CR\ or\ j = j_{rand} \\ x_{j,i} & otherwise \end{cases} \tag{2.3}$$

Where:

- $v_{j,i}$ is the mutation vector,
- j is the iterator through dimensions $j = 1...D$,
- i is the index of the individual,
- $x_{r_1,i}$, $x_{r_2,i}$, and $x_{r_3,i}$ are the randomly selected solutions from the population,
- F_i is the scaling factor parameter,
- $u_{j,i}$ is the trial vector,
- $U\ [0,1]$ is a random number of the uniform distribution within given bounds,
- CR is the cross-over rate parameter,
- j_{rand} is the index that the trial vector automatically adopts from the mutation vector,
- and $x_{j,i}$ is the original parent solution.

Differential Evolution has many modifications. Among the most advanced are JDE, or SHADE [403]. The current trend is to use the memory of successful solutions or parameter configurations and adaptive parameters instead of static ones. The algorithm's flowchart in its canonical form is depicted in Figure 2.3.

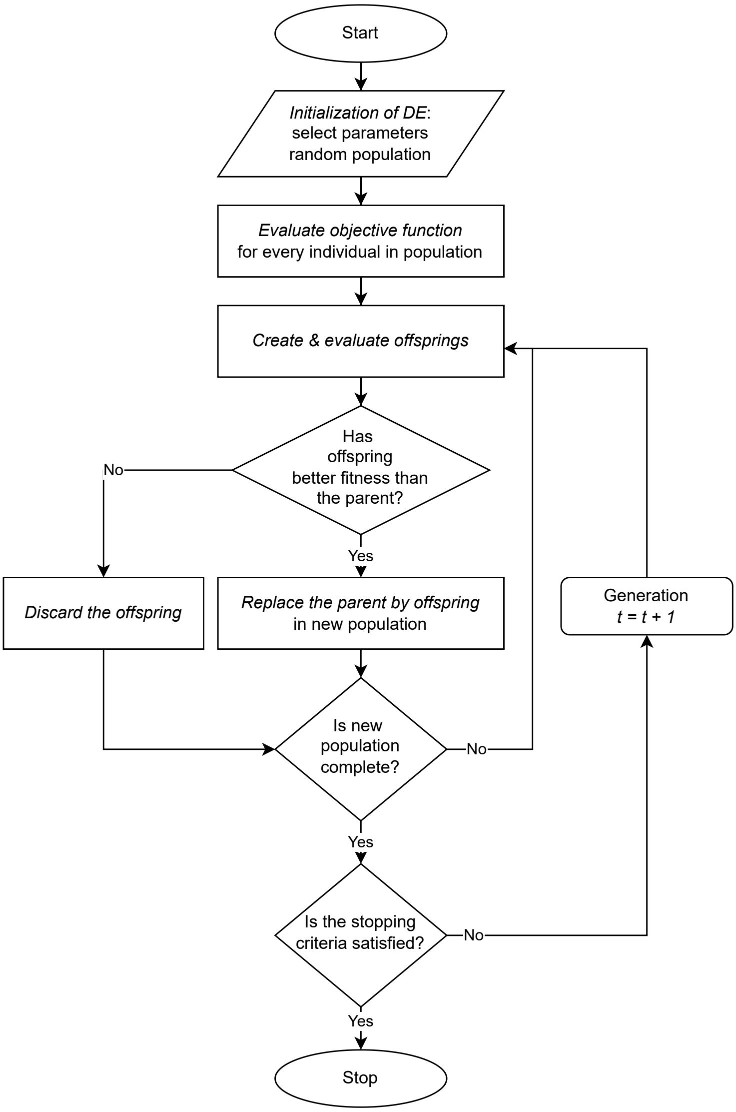


Fig. 2.3 Flowchart of Differential Evolution.

2.1.4 Self–Driven Particles

Self-Driven Particles (also called Self-Propelled Particles, the Vicsek model or SPP) was proposed by Vicsek et al. in 1995 [401]. This originally extended the Bird-Like Object Model (BOID) by Reynolds (1987) [182, 342].

The model describes the collective motion of fish schools and bird flocks. With several mathematical equations, Vicsek managed to specify a collective motion model that strongly resembles the motion of birds. The movement is described by Eq. 2.4, the velocity of a particle $v_i(t+1)$ was constructed to have an absolute value v and the direction θ in Eq. 2.5. This model is frequently used in swarm robotics, biomedical applications such as drug delivery or cargo transportation [304].

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + v_i(t)\Delta t \tag{2.4}$$

$$\theta_i(t+1) = \langle\theta(t)\rangle_r + \Delta\theta \tag{2.5}$$

Where:

- $\boldsymbol{x}_i$ is the particle agent,
- v_i presents the velocity of the particle,
- Δt is the time interval between two position and direction updates,
- $\theta(t+1)$ is the angle of the movement direction,
- $\langle\theta(t)\rangle_r$ denotes the average direction of the particles' velocities,
- and $\Delta\theta$ represents noise.

The Self-Propelled Particles model was listed as a metaheuristic in [275] and [437]. However, the model does not fit entirely the scope of other metaheuristics described in this thesis. Foremost, the SPP is not an optimization algorithm.

Figure 2.4 implies two possible meanings of the word 'metaheuristic'. According to Sörensen and Glover (2013), the definition of metaheuristics identifies the *metaheuristic framework* as a set of rules or strategies for the design of heuristic algorithms and a *metaheuristic algorithm* that implements these rules [369]. Based on this definition, the SPP model is a metaheuristic framework that does not offer an optimization procedure but a valuable functional motion model.

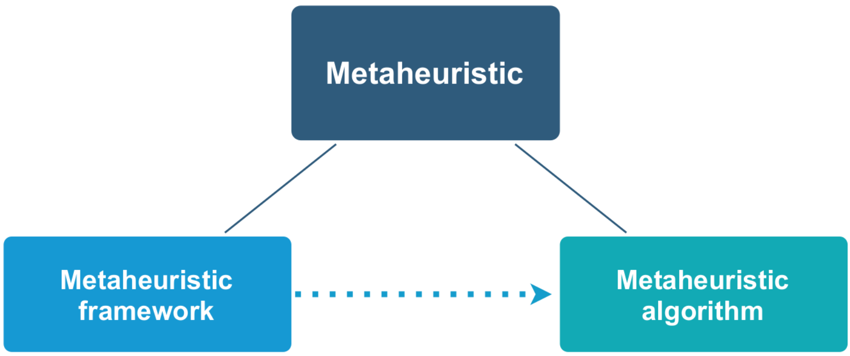


Fig. 2.4 Definition of metaheuristics by Sörensen and Glover (2013).

2.1.5 Gravitational Search Algorithm

The Gravitational Search Algorithm (GSA) was developed by Rashedi et al. in 2009 [335]. The algorithm simulates Newton's laws of gravity and motion. The applied metaphor regards individual solutions as objects whose performance is evaluated by their masses. All objects attract each other and move towards heavier masses, in correspondence with the inspiration source. At the same time, heavy masses move slower than light ones. The algorithm is described in a sequence of equations (Eq. 2.6 – 2.10). The next position of the solution is calculated in Eq. 2.10.

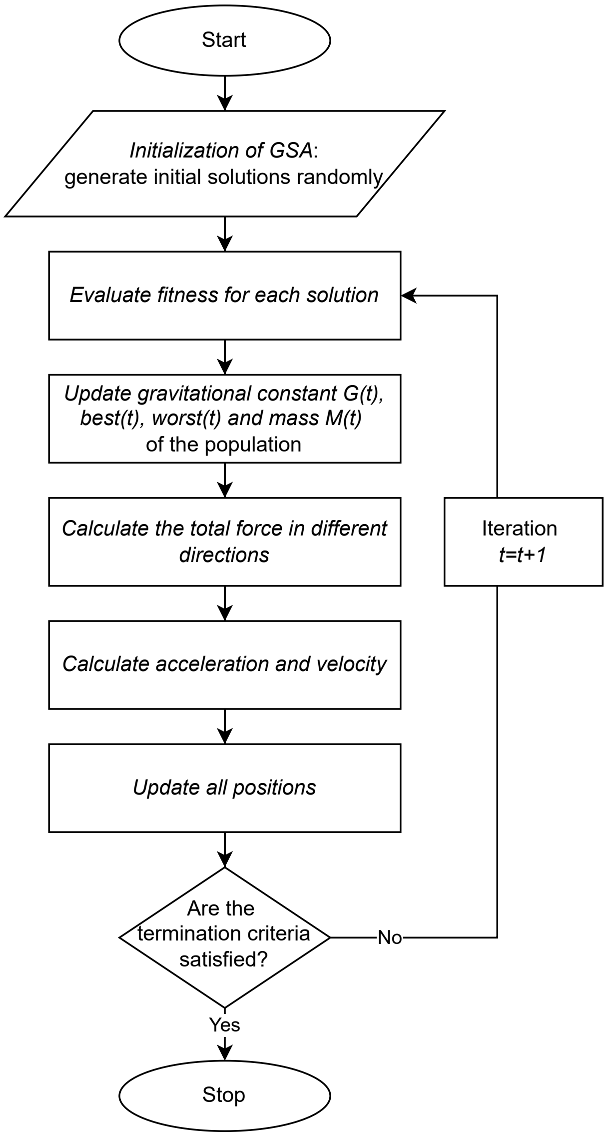


Fig. 2.5 Flowchart of the Gravitational Search Algorithm.

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \tag{2.6}$$

$$F_i^d(t) = \sum_{j \in k_{best}, j \neq i} rand_j F_{ij}^d(t) \tag{2.7}$$

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{2.8}$$

$$v_i^d(t+1) = rand \times v_i^d(t) + a_i^d(t) \tag{2.9}$$

$$x_i^d(t+1) = x_i^t + v_i^d(t+1) \tag{2.10}$$

Where:

- $F_{ij}^d(t)$ is the force acting on mass i from mass j,
- $G(t)$ is the gravitational constant as a function of time,
- M_{aj}, M_{pi}, M_{ii} are the active, passive, and inertia mass respectively,
- R_{ij} is the distance between the i and j solutions,
- ε is a small constant,
- x_i^d is the position of the solution i at dimension d,
- $F_i^d(t)$ is the total force on solution i,
- $rand_i$, $rand_j$ are uniform random values in range (0-1),
- k_{best} is the set of first K agents with the vest objective values and the biggest masses,
- v_i^d is the velocity of the solution i,
- and a_i^d is the acceleration of the solution i.

The principle of the Gravitational Search Algorithm very much resembles Particle Swarm Optimization. However, according to [335], there are several considerable differences in the movement strategy. First, GSA calculates the movement from all the available solutions, while PSO employs only the personal and global best solutions. Unlike PSO, GSA includes the distance between the solutions in the calculation procedure. Finally, GSA is a memory-less algorithm, whereas PSO utilizes personal and global best solutions. A flowchart of the algorithm is presented in Figure 2.5.

2.1.6 Evolutionary Strategies

Evolutionary Strategies (ES) were developed by Bienert, Rechenberg, and Schwefel in the 1960s [339]. At first, Evolutionary Strategies resembled the Genetic Algorithm: but the original ES did not use cross-over and worked with real numbers instead of Binary ones. Over time, several ES variants were proposed: including two-membered, multi-membered, recombinant, and self-adaptive strategies.

ES operate with several parameters: μ to define the number of parent solutions, λ for the offspring, and $+$ or , for the selection strategies. The first selection strategy $(+)$ denotes that the new population adopts the best solutions of both parents and offsprings, while the latter (,) promotes only offsprings. The syntax is therefore $(\mu+\lambda)$-ES or (μ,λ)-ES [38]. Figure 2.6 shows a flowchart of ES.

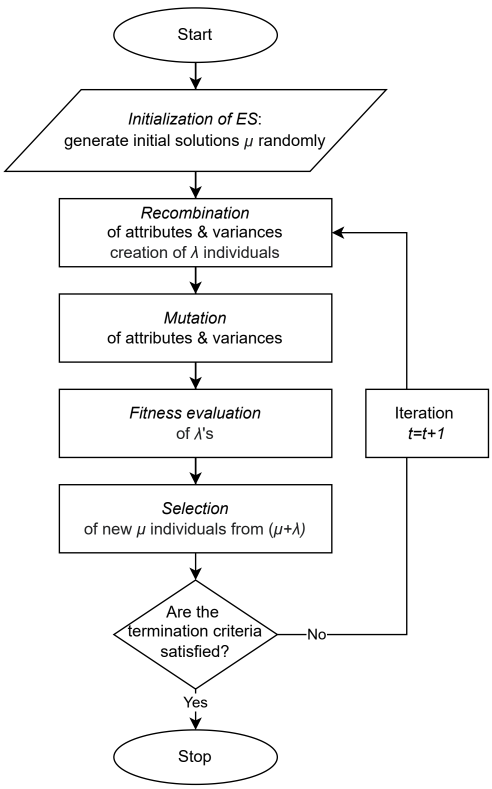


Fig. 2.6 Flowchart of the $\mu+\lambda$ Evolutionary Strategy.

2.2 Classification of Metaheuristic Algorithms

This section describes three classification systems: by the inspiration source, by behavior, and by application.

2.2.1 By Source of Inspiration

The selected inspiration source designates the majority of metaheuristic classification [139, 275, 331]. Comprehensive taxonomy from Molina et al. specifies the main classes based on Swarm Intelligence, Breeding, Physics, Chemistry, Human Behavior, Plants, and Miscellaneous [275]. Swarm Intelligence algorithms are further divided into subcategories based either on the simulated behavior (like foraging or movement) or closer identification of the inspiration source to aquatic, terrestrial, micro, flying, or others. This classification and the enumeration of the individual classes are shown in Figure 2.7. Table 2.2 shows the number of metaheuristics in each category and the Swarm Intelligence–based subclasses.

Tab. 2.2 Enumeration of metaheuristics by class and subclasses of swarm intelligence-based class. Data were extracted from [275].

Inspiration class	Algorithms	SI-subclasses I		SI-subclasses II	
Swarm intelligence	154	Micro	15	Foraging	92
Physics	51	Flying	57	Movement	62
Human behavior	37	Other	23		
Misc	33	Terrestrial	40		
Breeding	26	Aquatic	19		
Chemistry	13	Sum	154	Sum	154
Plants	10				
Sum	324				

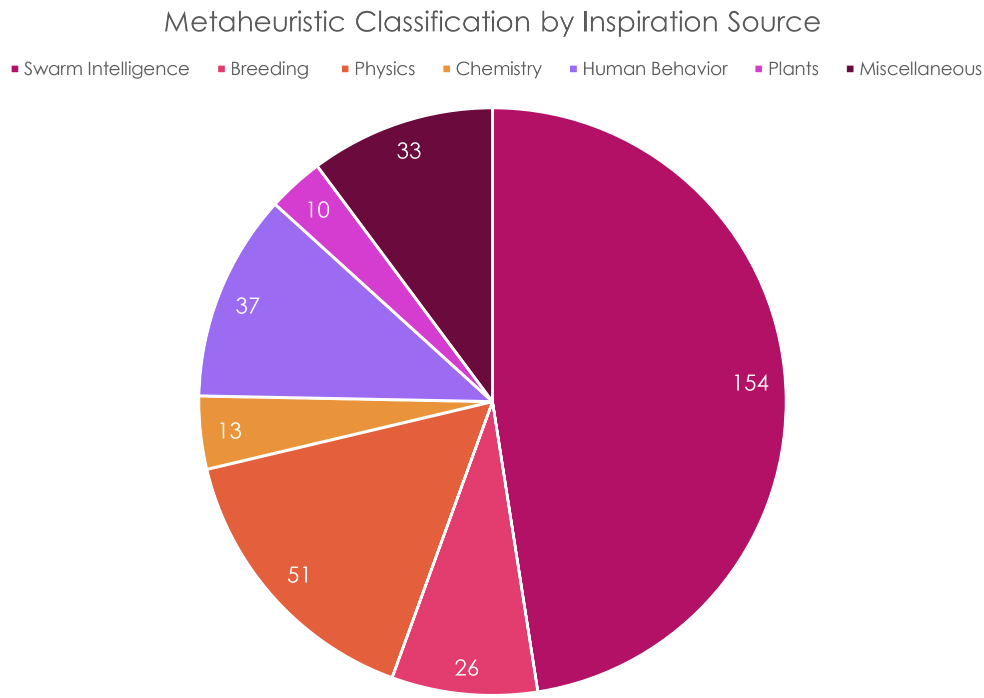


Fig. 2.7 Number of metaheuristics in classes defined by inspiration source.

2.2.2 By Algorithms' Behavior

Recently, there has been an effort to classify metaheuristics based on the algorithms' behavior [72, 275]. In [72], the author (Chu et al., 2020) distinguishes three types of behavior classifiers by learning, interaction, and diversification patterns. Learning behavior traces how solutions learn from their predecessors: whether globally, individually (e.g., inside of a neighborhood), or not at all. Interaction distinguishes if the individuals cooperate or compete. Finally, diversification describes the general tendency of the population to converge or diffuse.

In addition, Molina et al. (2020) proposed a behavior taxonomy based on the methodology of creating new solutions [275]. The classification distinguishes two categories: differential vector movement, which uses one reference solution, and creates a new solution by a shift or mutation. The second category uses more than a single reference, but a combination of solutions by a crossover, combination, or indirect coordination between the solutions. Figure 2.8 summarizes the classes mentioned.

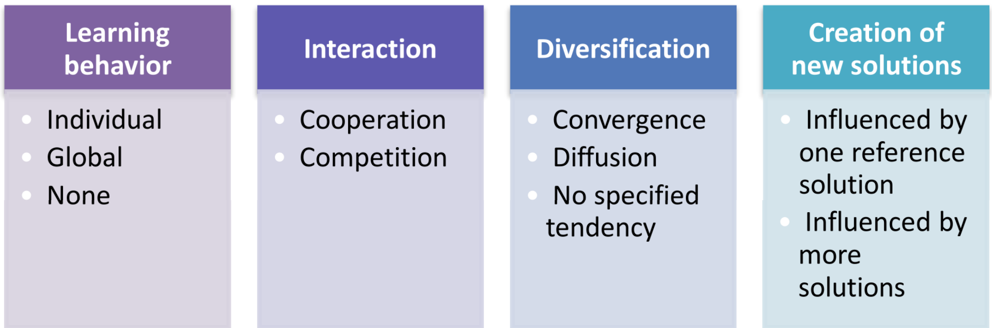


Fig. 2.8 Classification of metaheuristics based on algorithms' behavioral patterns.

Unlike inspiration source-based classification, which merely provides interesting facts about the modeled metaphor, behavior-based categories offer more information about the algorithms. Such an investigation may lead to uncovering potential similarities between the algorithms. For example, using the second creation-based taxonomy in [275] revealed that some algorithms have very similar behavioral patterns, despite their different inspiration sources (like GSA and PSO). On the other hand, algorithms coming from identical bio-inspiration, like the Dolphin Echolocation and Dolphin Partner Optimization, belong to entirely diverse behavior categories.

Albeit, the behavior-based classification of metaheuristics is in its infancy. Besides the cases mentioned, other categories could take into account other aspects, such as stochasticity, determinism, number of parameters, memory use, or to distinguish population-based and single point-based solutions.

2.2.3 By Application Field

Finally, metaheuristic classification may be named by the field of application, how the optimization algorithm is used, and what kind of problems it solves. The optimization areas include, for instance:

- Discrete / Continuous optimization
- Constrained / Hybrid / Unconstrained optimization
- Large-scale optimization
- Single objective / Multi-objective / Many-objective optimization
- Unimodal / Multimodal optimization
- Dynamic / Static optimization
- Sequential or Parallel optimization
- Applications

The application field's selection is closely related to the possible metaheuristics modifications [98, 100, 452]. Changing the metaheuristic algorithm's orientation from one type of problem to another (e.g., from discrete to continuous problems or extending single-objective algorithms to multi-objective optimization) is a frequent modification step. The modification trends are further analyzed in Section 2.4.

2.3 Optimization Struggles of Current Metaheuristics and Corresponding Countermeasures

Current metaheuristic algorithms face two key adversities: general optimization problems and existential problems. The former are basic phenomena connected to the optimization process, such as stagnation, premature convergence, or low population diversity. They are common points of the struggle of all the optimization techniques. The latter result from criticism directed at novel metaheuristics and are further analyzed in Section 3.2. This section introduces the common optimization challenges of current metaheuristics and corresponding methods to avoid them.

2.3.1 Stagnation and Premature Convergence

Premature convergence and *stagnation* are key optimization struggles of metaheuristics Both describe a similar setback of solutions that stop proceeding towards the global optimum. However, while premature convergence often relates to the low diversity of the population or local optimum confinement, according to Zelinka et al., stagnation might happen with no apparent reason [231]. Neri and Tirronen define stagnation as an undesirable situation in which the algorithms do not converge yet maintain a high population diversity [290]. Stagnation may be caused by various factors, including inappropriate parameter configuration or the problem's dimensionality [108].

Nicoara links premature convergence with dominating solutions, leading the population to a local optimum. That is why the primary tool to avoid premature convergence is enhancing the solutions' diversity [294].

Perils closely relate to exploitation and exploration practice. Too much exploitation causes premature convergence, and too much exploration slows down the optimization process [254]. Therefore, it is essential to set an appropriate exploitation-exploration balance; in fact, the main difference between metaheuristics is defined by the way in which they try to achieve this balance [39].

Since premature convergence and stagnation are affected by the parameter settings, the allocation of the current population, and the nature of the objective function, methods to tackle the predicaments of the optimization include:

- Dynamic adaptation of parameters
- Randomization of the parameters
- Diversification of the population
- Population restart
- Population subgroups

2.3.2 Local Optimum Containment

Local optimum containment describes a state in which the whole population of solutions merges into one local optimum. Figure 2.9 illustrates the problem. Many metaheuristics do not have a mechanism to escape local optimum. Moreover, some optimizers prioritize the exploitation of found solutions at the expense of exploration in later iterations. However, acquired quick convergence may lead to an unintended improvement of a local optimum instead of finding the global one [460].

The methods of tackling local containment problems differ based on elitism characteristics: whether the next population adopts only better quality solutions. Elitist algorithms rely on variation operators. They use mutation, crossover, diversity enforcement, parallel populations, and other mechanisms to create a solution out of the incriminated area.

The second approach to dealing with the local containment problem allows for even worse quality solutions. According to Oliveto (2018), both elitist and non-elitist approaches provide benefits, and it is still unclear when one should be preferred to the other [301].

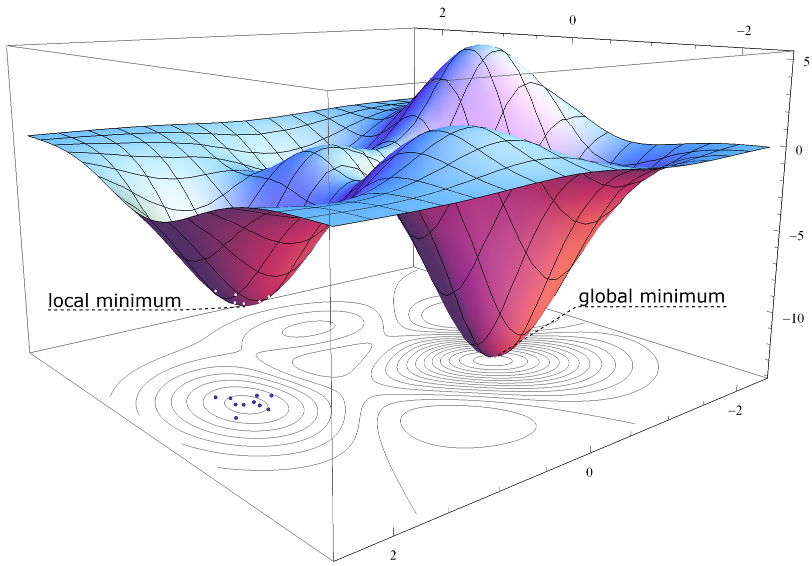


Fig. 2.9 Example of local optimum containment.

2.3.3 Low Diversity of the Population

A population with low diversity refers to a state where all solutions are very similar. The problem may often be closely connected with the local optimum containment. The solutions' insufficient diversity poses a severe problem, mainly if the next generation is created from the existing solutions.

There are multiple diversity measures (see, e.g., [68, 293, 316]). This thesis works with the diversity measure formulated in Equations 2.11, 2.12 proposed by Polakova et al. [316].

$$\text{Diversity} = \frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}(x_{i,j} - \overline{\mathbf{x}}_{\mathbf{j}})^2 \tag{2.11}$$

$$\overline{\mathbf{x}}_{\mathbf{j}} = \frac{1}{NP}\sum_{i=1}^{NP} x_{i,j} \tag{2.12}$$

Where:

- NP is the population size,
- D is the dimensionality of the problem,
- i and j are the population and dimension iterators respectively,
- $x_{i,j}$ is the vector value of the solution at the given dimension,
- and (x_j) is the corresponding mean of the solutions.

There is a connection between the found solutions' diversity and the desirable exploration–exploitation balance. The exploitation practice concentrates new solutions around a promising area and hence lowers population diversity. However, in the case of Particle Swarm Optimization, a fast clustering of solutions may compromise the algorithm's search potential and deteriorate the exploration ability [68].

Diversity-enhancing methods include [426]:

- Reinitialization of some individuals
- Adaptive parameters
- Adjustment of control parameters that directly affect exploration
- Multiple subpopulations
- Repulsion mechanism

2.3.4 Parameter Configuration

Most algorithms use a wide range of parameters, which often require expert knowledge a priori. Therefore, users usually follow the initial parameter recommendations. However, there is no guarantee of how and for what type of problems these parameters were set; many initial parameter configurations were set empirically on a limited selection of problems.

The No Free Lunch theorem states that there is no universal algorithm to solve every problem optimally. However, this theory may be extended even for parameter selection: while one set of parameters may work on a specific problem, it may be entirely inappropriate for others [51].

A proper choice of parameters has a considerable effect on optimization algorithms' performance [217]. For example, providing a small population of Particle Swarm Optimization leads to premature stagnation [123]. Also, parameter requirements may change during the optimization process, implying the benefit of an adaptive parameter approach.

There are two main approaches, *offline* and *online*, that deal with the parameter configuration problem. The offline approach is called *Parameter Tuning* and addresses the selection of proper parameters before applying the algorithm. These parameters do not change during the optimization process [172]. The Parameter Tuning strategies are mostly based on the generate–evaluate principle. First, they generate different parameter settings and evaluate them on a training set of problems by selected performance metrics. The Parameter Tuning practices are, for example, F-Race, iRace, REBAC, ParamILS, SPO, or SMAC [172, 366, 243].

The second, *online*, approach is called *Parameter Control*. Unlike the offline approach, Parameter Control adjusts the parameters during the actual run. The main classification of Parameter Control is based on what is changed and how [119]. The major types of online control involve deterministic (e.g., time-dependent), adaptive (with feedback from the search process), and self-adaptive strategies [119]. The self-adaptive approach encodes parameter selection as an instance of the optimization problem and runs the optimization algorithm on itself. Dragoi et al. [108] also add a hybrid strategy that combines the algorithms with fuzzy logic or chaotic systems [80, 239]. The possibilities of how to apply adaptive control parameters are multiple. There are several strategies based on the multi-populational principle ([152, 311, 394]). For instance, Pham (1995) used a strategy in which subpopulations compete for the computing time [311]. Pellerin et al. (2004) proposed a self-adaptive Genetic Algorithm in which one solution represents the current parameter settings [309].

In Ph.D. Thesis [366], Smit (2012) describes the Parameter Control Framework as a Self-Adaptivity manual. Based on his guidelines, the development process of the self-adaptive algorithm consists of three steps:

1. The choice of controlled parameter
2. The choice of condition to launch the control mechanism
3. The technique for adjusting the parameter value

2.3.5 Countermeasures Against Optimization Struggles

Many optimization struggles are connected. Premature convergence is often induced by both local optimum containment and low diversity measures. Stagnation resembles premature convergence. A wrong choice of control parameters may inflict all of the problems mentioned.

Therefore, the suggested counter methods also often overlap. What solves one problem may work on the other. Table 2.3 names several fundamental methods used in the fight against the optimization threats mentioned with some examples of use.

Tab. 2.3 Methods to tackle optimization struggles with examples of publications that use them.

Method	Examples of use
Dynamic adaptation of the parameters	[50, 298]
Parameter Tuning	[366]
Parameter Control	[108]
Randomization of the parameters	[327]
Diversification of the population	[428]
Population restart	[196]
Population subgroups	[3, 254]
Allow non-improving steps	[220]
Diversity enhancement	[426]

The Bison Algorithm proposed in Section 5 adopts a variation of the subgroup mechanism performing the exploitation and exploration separately. Exploration happens at the same rate throughout the whole optimization process and thus provides an escape mechanism from traps of local optima.

2.4 Modifications of Metaheuristics

After the initial proposal of a novel methodology, algorithms typically shift their focus to another area of optimization, e.g., from continuous to discrete, from single objective to multi-objective. Thorough testing and more applications also lead to new ideas on improving the developed algorithm [98, 100].

Modifications aim to expand the optimization capabilities of established metaheuristics. Some propose adjustments that try to patch the identified gaps or cover the corresponding algorithm's current optimization struggles. Typical examples are hybridization and dynamic adaptivity of parameters (Sections 2.3.4 and 2.4.1).

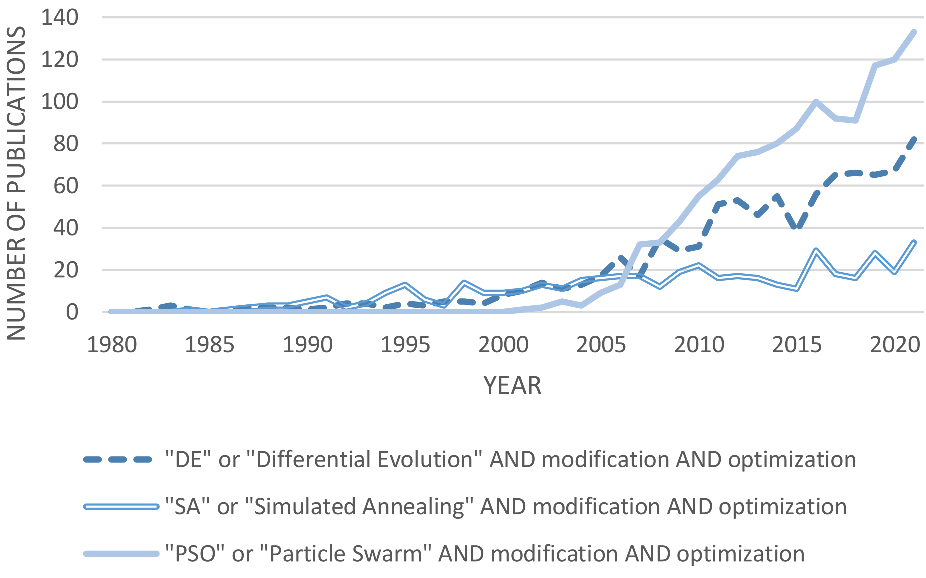


Fig. 2.10 The number of publications in the Scopus database with title/abstract/keywords including selected metaheuristics' names, modification, and optimization in years 1980–2021 (accessed 28/03/2022).

Modifications enable a way for the evolution of algorithms more than 25 years old. Figure 2.10 represents the number of publications belonging to the modification topic of selected metaheuristics in the Scopus database in 1980–2021[1). To provide a comparison with the novel metaheuristic proposal avalanche (in Section 3.2), Figure 2.11 compares the number of publications with novel metaheuristic proposals and the number of publications about the modifications of Particle Swarm Optimization[2). This comparison illustrates the popularity (and frequency) of metaheuristic modifications. Particle Swarm Optimization had hundreds of modifications already in 2015 [452]. In 2020, the sum of PSO modifications publications was 1,000.

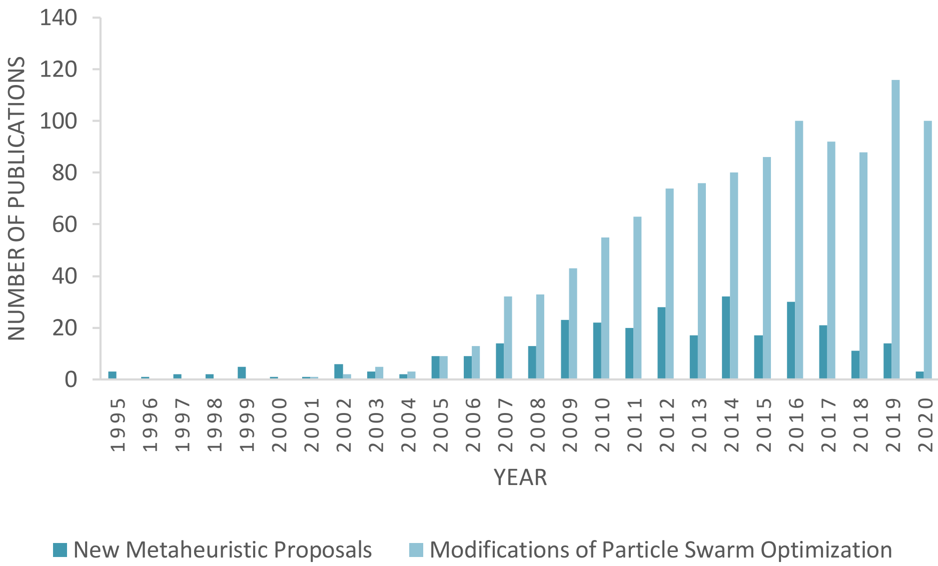


Fig. 2.11 The number of publications in the Scopus database addressing novel metaheuristic proposals compared to the number of publications addressing Particle Swarm Optimization modifications.

The modification possibilities are virtually unlimited. Modifications may affect parameter behavior, boost stochasticity, add multiple swarms or populations, or respond to a poignant problem that needs to be solved [187]. The classification of

1) The database query (presented in the legend of Figure 2.10) searched all the titles, abstract and keywords, including: the acronym of the metaheuristic or the algorithm's name and "modification" and "optimization". The optimization keyword was necessary, as both differential evolution and simulated annealing apply to a wide range of non-metaheuristic scientific fields. The results were processed on 18/11/2020.

2) Figure 2.11 shows data from 1995–2020, since PSO was first introduced in 1995.

modifications distinguishes classes like external/internal, major or slight modifications, hybridization, parallelism, or extension to other optimization fields (identified in Section 2.2.3) [187, 452].

The naming convention usually reflects the modified algorithm. Table 2.4 depicts the names and acronyms of selected Particle Swarm Optimization modifications.

Tab. 2.4 Selected examples of Particle Swarm Optimization modifications.

Acronym		**Modification Name**
APSO	[419]	Adaptive Particle Swarm Optimization
BBPSO	[444]	Bare-bones Particle Swarm Optimization
Center PSO	[241]	Center Particle Swarm Optimization
CPSO	[76]	Chaotic Catfish Particle Swarm Optimization
FPSO	[193]	Fuzzy Particle Swarm Optimization
MPSO	[242]	Modified Particle Swarm Optimization
OPSO	[102]	Opposition-Based Particle Swarm Optimization
PSO-GA	[228]	Hybrid Particle Swarm Optimization Genetic Algorithm
PSO-DT	[161]	Particle Swarm Optimization with Disturbance Term
PSOPC	[162]	Particle Swarm Optimization with Passive Congregation
PSOTVAC	[54]	Particle Swarm Optimization with Time-Varying Accelerator Coefficients
QPSO	[188]	Quantum-Behaved Particle Swarm Optimization
SPSO	[61]	Simplified Particle Swarm Optimization

2.4.1 Hybridization

Hybridization is a technique that combines two or more algorithms to improve the optimizer's methodology. The aim is to tackle the most common optimization problems collectively and cooperatively. There are many ways to hybridize an algorithm. The algorithms involved may solve the same problem, or one algorithm might tune another's parameters. A multi-stage collaborative hybrid employs different algorithms on the exploitation and exploration practice; the algorithms might swap sequentially, run in parallel, or share a population. Integrative hybrid algorithms implement characteristic methodologies across metaheuristics (e.g., GA's mutation into PSO).

Even though many examples and applications show that hybridization may significantly improve optimization performance [34, 45], it also increases the complexity of the algorithms and adds more parameters [392]. Hybridization also comes with specific non-uniform (occasionally unpronounceable) jargon. Table 2.5 represents some examples of hybrid algorithms naming conventions [382, 392].

Tab. 2.5 Examples of hybrid metaheuristic names.

Acronym		Full Name of the Hybrid Algorithm
PSO–SQP	[402]	Particle Swarm Optimization with Sequential Quadratic Programming
FFPSO	[223]	Hybrid Firefly and Particle Swarm Optimization Algorithm
mFFPSO	[197]	Hybrid MultiSwarm Firefly and Particle Swarm Optimization
HABCDE	[184]	Hybrid Artificial Bee Colony Differential Evolution

3 Swarm Algorithms

Swarm algorithms are metaheuristics based on the collective intelligence phenomenon as a characteristic feature of animal swarms, flocks, or herds. Animal groups often make smart decisions only with local information and simple rules but no centralized control. Some believe that collective intelligence's origin lies in adapting to the environment and community [440].

While swarm intelligence itself remains a mystery, its simulations produced a significant number of swarm algorithms. In fact, with more than 150 specimens, swarm-based algorithms account for the majority of known metaheuristic algorithms (see Section 2.2.1). Table 3.1 represents the list of swarm-based metaheuristics. The data were derived from the *Comprehensive Taxonomies of Nature- and Bio-Inspired Optimization* by Molina et al. [275]. The following exceptions were made to the origin:

- Some publications were switched for an earlier paper than referenced in [275], e.g., the Artificial Bee Colony [200], the Great Salmon Run Algorithm [281], African Buffalo Optimization [299], and Glowworm Swarm Optimization [224].
- Similar algorithms proposed by the same authors were merged into one algorithm (e.g., the Bacterial Foraging Algorithm [91, 240, 307], or the (Cultural) Coyote Optimization Algorithm [312, 313]).
- The Improved Raven Roosting Algorithm [395] was replaced by original Raven Roosting Optimization [49].
- The table adds information about the original proposals' popularity measured by the Scopus database's citation score of the original proposal publication (data were collected 27/01/2021).

This section studies the current trends in swarm algorithms. It starts by introducing the most popular swarm optimizers. The algorithms' selection was

Tab. 3.1 Swarm-inspired metaheuristics sorted by the number of citations of the original proposal paper (from 27/01/2021).

	Swarm Algorithm	**Acronym**	**Year**	**Original Paper**	**▼ Scopus ▼ Citations**
1	Particle Swarm Optimization	PSO	1995	[115]	11198
2	Ant Colony Optimization	ACO	1996	[107]	8199
3	Artificial Bee Colony	ABC	2005	[200]	4096
4	Cuckoo Search	CS	2009	[439]	3856
5	Grey Wolf Optimizer	GWO	2014	[269]	3842
6	Bat Inspired Algorithm	BAT	2010	[436]	2597
7	Whale Optimization Algorithm	WOA	2016	[265]	2222
8	Bacterial Foraging Optimization	BFOA	2002	[307]	2197
9	Firefly Algorithm	FA	2009	[430]	2143
10	Moth Flame Optimization Algorithm	MFO	2015	[262]	1105
11	Ant Lion Optimizer	ALO	2015	[268]	1048
12	Sine Cosine Algorithm	SCA.2	2016	[263]	1011
13	Krill Herd	KH	2012	[143]	1004
14	Salp Swarm Algorithm	SSA.2	2017	[267]	924
15	Fruit Fly Optimization Algorithm	FOA	2012	[305]	907
16	Dragonfly Algorithm	DA	2016	[264]	777
17	Shuffled Frog-Leaping Algorithm	SFLA	2006	[126]	707
18	Bees Algorithm	BA	2006	[310]	695
19	Grasshopper Optimisation Algorithm	GOA	2017	[352]	682
20	Crow Search Algorithm	CSA	2016	[23]	632
21	Symbiosis Organisms Search	SOS	2014	[67]	630
22	Cuckoo Optimization Algorithm	COA	2011	[328]	612
23	Group Search Optimizer	GSO.1	2009	[163]	526
24	Modified Cuckoo Search	MCS	2011	[404]	424
25	Harry's Hawk Optimization Algorithm	HHO	2019	[165]	388
26	Cat Swarm Optimization	CSO	2006	[71]	356
27	Social Spider Optimization	SSO.2	2013	[86]	296
28	Bacterial Chemotaxis Optimization	BCO.2	2002	[287]	284
29	Bee Colony Optimization	BCO	2005	[389]	283
30	Chicken Swarm Optimization	CSO.1	2014	[255]	269
31	Glowworm Swarm Optimization	GSO	2005	[224]	258
32	Dolphin Echolocation	DE.1	2013	[204]	250
33	Pigeon Inspired Optimization	PIO	2014	[111]	244
34	Virtual Bees Algorithm	VBA	2005	[429]	232
35	Spider Monkey Optimization	SMO	2014	[26]	208
36	Lion Optimization Algorithm	LOA	2014	[330]	200
37	Regular Butterfly Optimization Algorithm	RBOA	2019	[18]	186
38	Elephant Herding Optimization	EHO	2016	[405]	182
39	Eagle Strategy	ES.1	2010	[432]	181
40	Social Spider Algorithm	SSA	2015	[441]	176
41	Spotted Hyena Optimizer	SHO	2017	[103]	168
42	Monarch Butterfly Optimization	MBO.1	2019	[407]	164
43	Hunting Search	HuS	2010	[300]	161

	Swarm Algorithm	Acronym	Year	Original Paper	▼ Scopus ▼ Citations
44	BeeHive Algorithm	BHA	2004	[411]	156
45	Squirrel Search Algorithm	SSA.1	2019	[186]	155
46	Bird Swarm Algorithm	BSA	2016	[256]	152
47	Monkey Search	MS	2007	[282]	148
48	Migrating Birds Optimization	MBO.2	2012	[113]	141
49	Wolf Search Algorithm	WSA.1	2012	[386]	131
50	Shark Search Algorithm	SA	1998	[181]	128
51	Fish School Search	FSS	2008	[136]	119
52	Virus Colony Search	VCS	2016	[237]	116
53	Wasp Colonies Algorithm	WCA	1991	[259]	109
54	Bee Swarm Optimization	BSO	2010	[8]	107
55	Wolf Pack Search	WPS	2007	[424]	100
56	Bees Swarm Optimization Algorithm	BSOA	2005	[109]	96
57	Catfish Optimization Algorithm	CAO	2011	[75]	96
58	Artificial Algae Algorithm	AAA	2015	[400]	90
59	Shark Smell Optimization	SSO	2016	[4]	89
60	Bee System	BS.1	2002	[245]	88
61	Fish Swarm Algorithm	FSA	2011	[397]	88
62	Coyote Optimization Algorithm	CCOA	2018	[313]	85
63	Flocking Base Algorithms	FBA	2006	[88]	80
64	Seeker Optimization Algorithm	SOA	2006	[90]	77
65	Lion Algorithm	LA	2012	[329]	71
66	Bees Life Algorithm	BLA	2018	[42]	64
67	Fast Bacterial Swarming Algorithm	FBSA	2008	[73]	63
68	Satin Bowerbird Optimizer	SBO	2017	[351]	60
69	Snap-Drift Cuckoo Search	SDCS	2017	[332]	59
70	Roach Infestation Problem	RIO	2008	[160]	58
71	Bee System	BS	1998	[353]	51
72	Cuttlefish Algorithm	CFA	2013	[117]	49
73	Dolphin Partner Optimization	DPO	2009	[427]	49
74	Wolf Colony Algorithm	WCA.1	2011	[238]	47
75	Chaotic Dragonfly Algorithm	CDA	2018	[355]	46
76	African Buffalo Optimization	ABO	2015	[299]	42
77	Collective Animal Behavior	CAB	2012	[85]	41
78	Swallow Swarm Optimization	SSO.1	2013	[291]	40
79	Sperm Whale Algorithm	SWA	2016	[116]	40
80	Termite Hill Algorithm	TA	2012	[459]	40
81	Elephant Search Algorithm	ESA	2015	[97]	38
82	Penguins Search Optimization Algorithm	PSOA	2013	[146]	36
83	Prey Predator Algorithm	PPA	2014	[155]	35
84	Egyptian Vulture Optimization Algorithm	EV	2013	[378]	34
85	Red Deer Algorithm	RDA	2016	[128]	33
86	Termite Colony Optimization	TCO	2010	[164]	31
87	The Great Salmon Run Algorithm	TGSR	2012	[281]	30
88	Viral Systems Optimization	VSO	2008	[82]	29
89	Natural Aggregation Algorithm	NAA	2016	[244]	28

	Swarm Algorithm	Acronym	Year	Original Paper	▼ Scopus ▼ Citations
90	Pity Beetle Algorithm	PBA	2018	[198]	27
91	Simulated Bee Colony	SBC	2009	[251]	25
92	Bacterial-GA Foraging	BGAF	2007	[64]	24
93	Goose Team Optimization	GTO	2008	[409]	24
94	Raven Roosting Optimisation Algorithm	RRO	2016	[49]	24
95	Binary Whale Optimization Algorithm	BWOA	2019	[341]	22
96	Honeybee Social Foraging	HSF	2007	[323]	22
97	Killer Whale Algorithm	KWA	2017	[44]	22
98	Mouth Breeding Fish Algorithm	MBF	2018	[185]	22
99	Hierarchical Swarm Model	HSM	2010	[62]	21
100	Cricket Behavior-Based Algorithm	CBBE	2015	[56]	20
101	Naked Mole Rat	NMR	2019	[348]	20
102	Swarm Inspired Projection Algorithm	SIP	2009	[373]	20
103	Slime Mould Algorithm	SMA	2008	[276]	20
104	Invasive Tumor Optimization Algorithm	ITGO	2015	[385]	19
105	Optimal Foraging Algorithm	OFA	2017	[458]	19
106	Bumblebees	BB	2009	[81]	18
107	Magnetotactic Bacteria Optimization Algorithm	MBO	2013	[270]	17
108	Bee Colony-Inspired Algorithm	BCIA	2009	[176]	16
109	Cheetah Based Algorithm	CBA	2018	[222]	16
110	Laying Chicken Algorithm	LCA	2017	[168]	16
111	Weightless Swarm Algorithm	WSA	2012	[391]	16
112	Wasp Swarm Optimization	WSO	2005	[314]	16
113	Meerkats Inspired Algorithm	MIA	2018	[221]	13
114	Blind, Naked Mole-Rats Algorithm	BNMR	2013	[380]	13
115	Modified Cockroach Swarm Optimization	MCSO	2014	[297]	13
116	Bat Intelligence	BI	2012	[249]	12
117	Consultant Guide Search	CGS	2010	[417]	12
118	Virtual Ants Algorithm	VAA	2006	[433]	12
119	Frog Call Inspired Algorithm	FCA	2009	[284]	11
120	Locust Swarms Optimization	LSO	2009	[63]	11
121	Butterfly Optimizer	BO	2015	[225]	10
122	Artificial Beehive Algorithm	ABA	2009	[286]	10
123	Animal Behavior Hunting	ABH	2014	[288]	10
124	Flock by Leader	FL	2012	[32]	10
125	Good Lattice Swarm Optimization	GLSO	2007	[374]	10
126	Rhino Herd Behavior	RHB	2018	[408]	10
127	Biology Migration Algorithm	BMA	2019	[448]	9
128	Seven-Spot Labybird Optimization	LBO	2013	[410]	9
129	Nomadic People Optimizer	NPO	2020	[349]	9
130	Bioluminiscent Swarm Optimization Algorithm	BSO.1	2011	[96]	8
131	Group Escape Behavior	GEB	2011	[260]	8
132	Camel Travelling Behavior	COA.1	2016	[177]	7
133	Mox Optimization Algorithm	MOX	2011	[261]	7
134	Population Migration Algorithm	PMA	2009	[449]	7
135	Surface-Simplex Swarm Evolution Algorithm	SSSE	2017	[322]	7
136	Virus Optimization Algorithm	VOA.1	2009	[194]	7
137	Artificial Tribe Algorithm	ATA	2012	[66]	6
138	Bison Behavior Algorithm	BIA	2017	[214]	6
139	Andean Condor Algorithm	ACA	2019	[11]	5

	Swarm Algorithm	Acronym	Year	Original Paper	▼ Scopus ▼ Citations
140	African Wild Dog Algorithm	AWDA	2013	[376]	5
141	Jaguar Algorithm	JA	2015	[60]	5
142	Mosquito Flying Optimization	MFO.1	2016	[10]	5
143	Reincarnation Concept Optimization Algorithm	ROA	2010	[361]	5
144	Artificial Searching Swarm Algorithm	ASSA	2009	[65]	4
145	Bacterial Colony Optimization	BCO.1	2012	[296]	4
146	Worm Optimization	WO	2014	[17]	4
147	Zombie Survival Optimization	ZSO	2012	[292]	4
148	Hoopoe Heuristic Optimization	HHO.1	2012	[122]	3
149	OptBees	OB	2013	[248]	3
150	Bald Eagle Search	BES	2019	[12]	2
151	See-See Partridge Chicks Optimization	SSPCO	2015	[302]	2
152	Hypercube Natural Aggregation Algorithm	HYNAA	2020	[247]	0

similar to the metaheuristics selection in Section 2.1. The algorithms were compared based on the citation score of the original proposals. Section 3.1 describes the top 9 swarm algorithms.

Section 3.2 analyzes criticism that swarm algorithms face. It presents all kinds of reservations concerning these algorithms' quantity and quality and the corresponding existing (and non-existing) implications. In response, the Author proposes a set of recommendations for the future development of new swarm algorithms and metaheuristics in Section 3.3.

3.1 Popular Swarm Algorithms

This section presents the fundamentals and basic principles of the top 9 most popular swarm algorithms. The popularity rate was based on the number of references citing proposal publications in the Scopus database. The selection represents only a tiny sample of the total mass of swarm algorithms; however, an attentive reader might notice several peculiarities in the terminology and reccurring methods.

It should be mentioned that three of the presented algorithms: the Grey Wolf Optimizer, the Bat Algorithm, and the Firefly Algorithm, were analyzed for their novelty [55]. The authors (Camacho Villalón et al.) concluded that the examined optimizers were merely modifications of former Particle Swarm Optimization variants. However, since there is still no defined metric to distinguish the novelty between the algorithms, these algorithms are presented as an example of popular swarm optimizers in Sections 3.1.5, 3.1.6, and 3.1.9.

3.1.1 Particle Swarm Optimization

Kennedy and Eberhart developed Particle Swarm Optimization (PSO) in 1995 [219]. The inspiration came from the emerging behavior of fish swarms and bird flocks. It employs a population of particles flying at various speeds through the search space. The speed varies for every particle, based on the local and global optimal positions. The basic PSO motion is described in Equations 3.1 and 3.2. Figure 3.1 shows a flowchart of the optimization process.

The PSO algorithm ignited interest in swarm algorithms and the collective intelligence phenomenon and thus sowed the seeds of the development of other bio-inspired algorithms [440].

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{3.1}$$

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha r_1[\boldsymbol{g}^* - \boldsymbol{x}_i^t] + \beta r_2[\boldsymbol{x}_i^* - \boldsymbol{x}_i^t] \tag{3.2}$$

Where:

- $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_i^{t+1}$ are the current and next solutions at iteration t,
- $\boldsymbol{v}_i$ is velocity of the i^{th} solution in range $(0, v_{max})$,
- r_1, r_2 are random vectors in range (0-1),
- α and β are constant acceleration parameters,
- $\boldsymbol{x}_i^*$ and $\boldsymbol{g}^*$ are the local and global best solutions respectively.

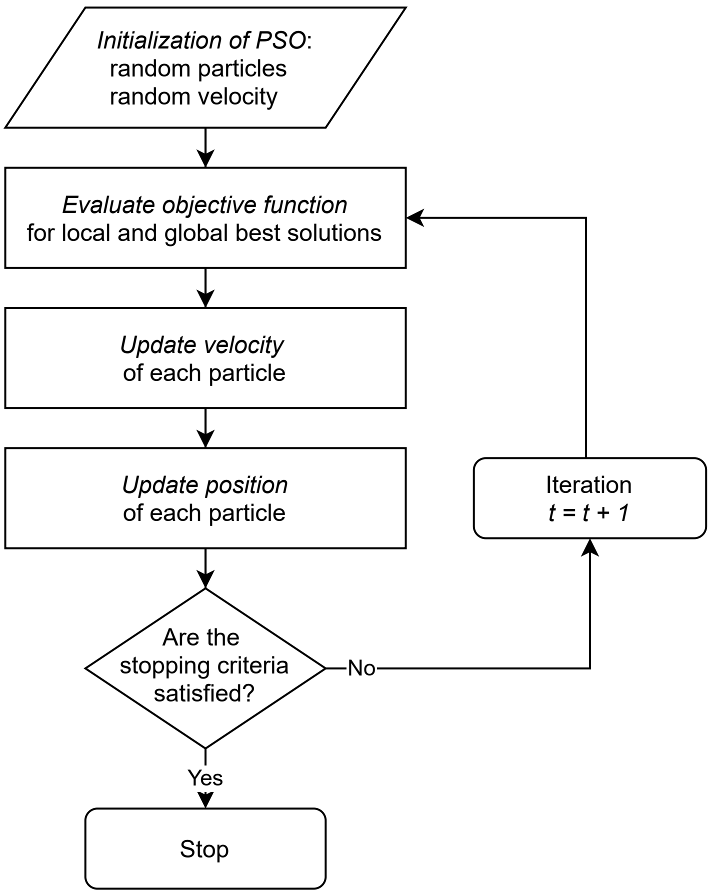


Fig. 3.1 Flowchart of Particle Swarm Optimization.

3.1.2 Ant Colony Optimization

Ant Colony Optimization (ACO) was introduced by Dorigo in his dissertation thesis in 1992 [105]. It simulates ant colonies' behavior when seeking and collecting food as a problem of optimal path searching. As an example of collective intelligence, ant colonies, despite being blind, accumulate enormous food supplies. The algorithm is based on pheromone paths that indicate their quality. Since the pheromones are designed to evaporate, shorter routes are advantaged.

Ants move from node i to node j with probability $p_{i,j}$ (Eq. 3.3). Pheromone level is updated with Eq. 3.4.

$$p_{i,j} = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})} \tag{3.3}$$

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \Delta\tau_{i,j} \tag{3.4}$$

Where:

- $\tau_{i,j}$ is the amount of pheromone on edge i, j,
- α is a parameter to control the influence of $\tau_{i,j}$,
- $\eta_{i,j}$ is the desirability of edge i, j (typically $1/d_{i,j}$),
- β is a parameter to control the influence of $\eta_{i,j}$,
- ρ is the rate of pheromone evaporation,
- and $\Delta\tau_{i,j}$ is the amount of pheromone deposited, typically given by:

$$\Delta\tau_{i,j} = \begin{cases} 1/L_k & \text{if ant } k \text{ travels on edge } i,j \\ 0 & otherwise \end{cases} \tag{3.5}$$

Where:

- L_k is the cost of the k^{th} ant's tour (typically length).

Initially, the ACO algorithm solved combinatorial problems in discrete space. However, nowadays, there are multiple modifications for other application fields, the continuous domain included [153]. A flowchart of the ACO algorithm is depicted in Figure 3.2.

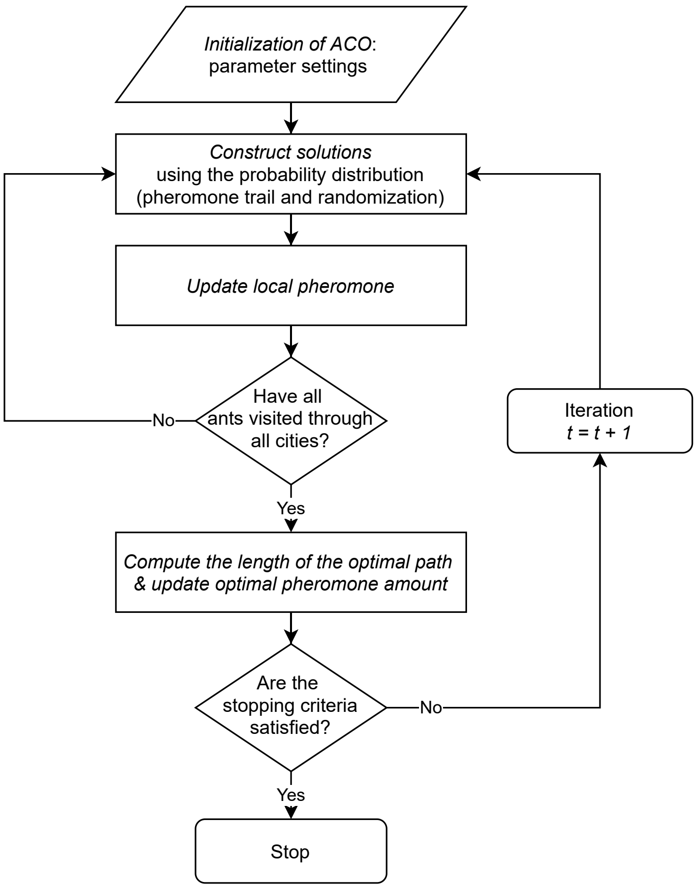


Fig. 3.2 Flowchart of Ant Colony Optimization.

3.1.3 Artificial Bee Colony

In 2005, Karaboga proposed another member of the sizable bee-inspired metaheuristic family: the Artificial Bee Colony (ABC) [200]. Table A.1 of all the metaheuristic algorithms in Appendix A count up to 17 bee-based algorithms and one Bumblebee. However, the ABC is the most popular one.

The algorithm simulates the specialization characteristic of bee colonies, dividing the solutions into employed bees, onlookers, and scouts. At first, one half of the colony consists of employed bees, the other half of onlookers. The employed bees correspond to good solutions. The onlookers simulate the bees waiting in the hive. They can choose the source to collect food from and move towards better solutions dependent on the probability given by Eq. 3.6. The candidate solutions are made by Eq. 3.7.

$$p_i = \frac{f(\boldsymbol{x}_i)}{\sum_{n=1}^{SN} f(\boldsymbol{x}_n)} \tag{3.6}$$

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}) \tag{3.7}$$

Where:

- $f(\boldsymbol{x}_i)$ is the objective function value of the solution $\boldsymbol{x}_i$,
- and SN is the number of food sources, equal to the number of employed bees,
- $k \in 1, 2, ..., SN$ and $j \in 1, 2, ..., D$ are randomly chosen indexes $k \neq i$,
- and $\phi_{i,j}$ is a random number between [-1, 1].

If the solution does not improve for a defined limit of iterations, the food source is abandoned, and replaced by a new scout solution that improves its quality (Eq. 3.8).

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j) \tag{3.8}$$

Where:

- x_i is the abandoned solution,
- and j is the index $j \in 1, 2, ..., D$.

All the bees move towards better solutions only. The algorithm parameters include the number of food sources SN, which also defines the number of employed or onlooker bees, the limit of unimproved solutions, and the maximum number of iterations MCN.

Similarly to ACO, the ABC algorithm is quite an effective optimizer for discrete optimization, but there are also continuous problem variations. A flowchart of the ABC algorithm is depicted in Figure 3.3.

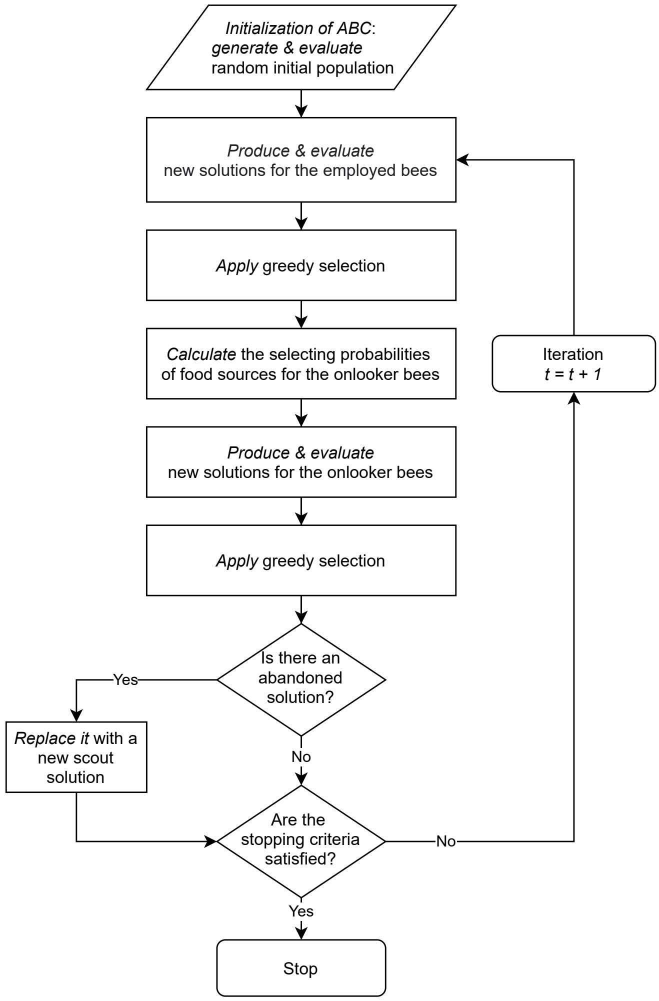


Fig. 3.3 Flowchart of the Artificial Bee Colony Algorithm.

3.1.4 Cuckoo Search

Cuckoo Search Optimization (CS) was developed by Yang and Deb in 2009 [439]. It simulates the aggressive reproduction strategy of cuckoos. Cuckoos are known for their brood parasitism: laying eggs in other birds' nests. The algorithm considers each solution as an egg of a quality defined by the objective function value.

In each iteration, the algorithm creates a new cuckoo solution by Eq. 3.9 and replaces the original solution if it is of better quality. The algorithm applies the probability that the cuckoo egg will be discovered. A fraction of the worst nests are abandoned and replaced by new random solutions.

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \alpha \oplus \mathrm{Levy}(\lambda) \tag{3.9}$$

Where:
- $\mathbf{x}_i$ is the i^{th} solution from the population,
- t presents the current iteration,
- $\alpha > 0$ is the step size related to the scales of the problem of interest (for most cases $\alpha = 1$),
- and Levy(λ) presents a random number of the Levy distribution:

$$\mathrm{Levy} \sim u = t^{-\lambda} \tag{3.10}$$

The main advantage of the algorithm is that there are only two parameters: the population size NP and probability p_a controlling the randomization balance. The Cuckoo Search thus represents a robust algorithm with extraordinary optimization capabilities. The main loop of the algorithm is described in a flowchart (Figure 3.4).

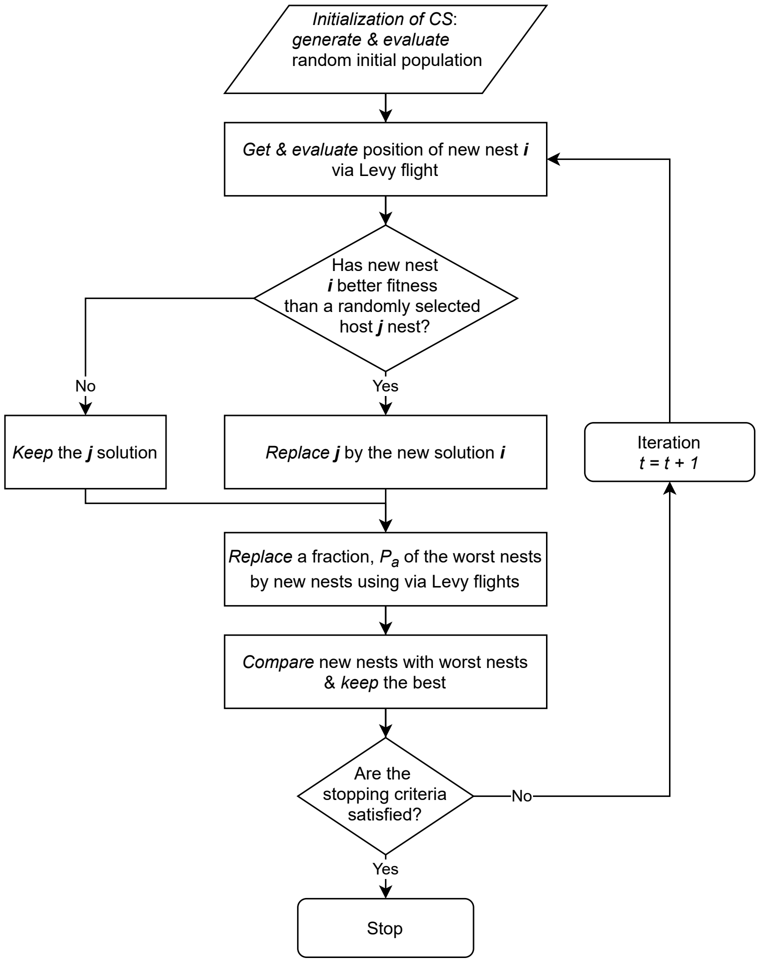


Fig. 3.4 Flowchart of the Cuckoo Search Algorithm.

3.1.5 Grey Wolf Optimizer

The Grey Wolf Optimizer was developed by Mirjalili in 2014 [269]. The algorithm mimics the hunting mechanism and hierarchy of a wolf pack. The social hierarchy model consists of alpha, beta, delta, sorted by the objective function value, and the rest of the solutions are omegas. According to the rank defined, wolves command lower ranks and obey higher ones. Wolves hunt by encircling their prey. The motion is described in Eqs. 3.11–3.14.

$$\boldsymbol{D} = |\mathbf{C} \cdot \mathbf{X}_p(t) - \mathbf{X}(t)| \tag{3.11}$$

$$\mathbf{X}(t+1) = \mathbf{X}_p(t) - \mathbf{A} \cdot \mathbf{D} \tag{3.12}$$

Where:

- t is the current iteration,
- $\mathbf{X}_p$ is the position vector of the prey,
- and $\mathbf{X}$ is the position vector of a grey wolf,
- and $\mathbf{A}$ and $\mathbf{C}$ are coefficient vectors calculated as follows:

$$\boldsymbol{A} = 2\mathbf{a} \cdot \mathbf{r}_1 - \mathbf{a} \tag{3.13}$$

$$\boldsymbol{C} = 2 \cdot \mathbf{r}_2 \tag{3.14}$$

Where:

- $\mathbf{a}$ linearly decreases from 2 to 0,
- and $\mathbf{r}_1$, $\mathbf{r}_2$ are random vectors in [0,1] range.

The hunting movement updates the solutions according to the positions of the best agents as represented in Eqs. 3.15–3.17. A flowchart of the GWO is depicted in Figure 3.5.

$$\boldsymbol{D}_\alpha = |\mathbf{C}_1 \cdot \mathbf{X}_\alpha - \mathbf{X}|, \boldsymbol{D}_\beta = |\mathbf{C}_2 \cdot \mathbf{X}_\beta - \mathbf{X}|, \boldsymbol{D}_\delta = |\mathbf{C}_3 \cdot \mathbf{X}_\delta - \mathbf{X}| \tag{3.15}$$

$$\boldsymbol{X}_1 = \boldsymbol{X}_\alpha - \boldsymbol{A}_1 \cdot (\boldsymbol{D}_\alpha), \boldsymbol{X}_2 = \boldsymbol{X}_\beta - \boldsymbol{A}_2 \cdot (\boldsymbol{D}_\beta), \boldsymbol{X}_3 = \boldsymbol{X}_\delta - \boldsymbol{A}_3 \cdot (\boldsymbol{D}_\delta) \tag{3.16}$$

$$\boldsymbol{X}(t+1) = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3} \tag{3.17}$$

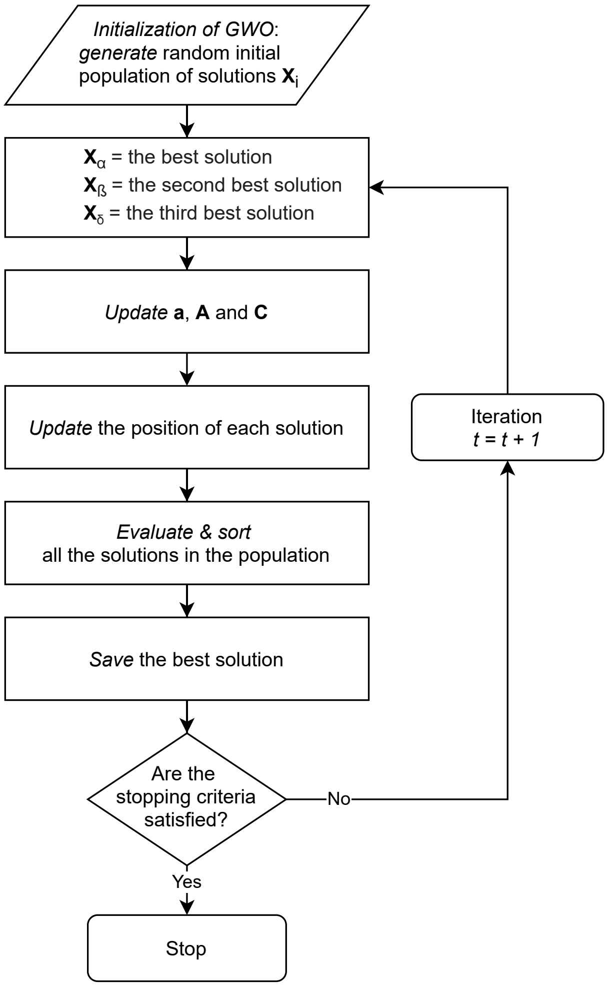


Fig. 3.5 Flowchart of the Grey Wolf Optimizer.

3.1.6 Bat Algorithm

The Bat Algorithm (BAT) was developed by Yang in 2010 [436]. It simulates the echolocation ability of microbats and operates with a model of wavelengths and frequencies. The algorithm is defined by Eqs. 3.18–3.20, determining frequency, velocity, and new solution computation.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{3.18}$$
$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_i^t - \mathbf{x}_*)f_i \tag{3.19}$$
$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t \tag{3.20}$$

Where:

- f_i is a frequency, that is initially drawn uniformly from $[f_{min}, f_{max}]$, and essentially controls the pace and the movement range of the solutions,
- $\beta \in [0, 1]$ is a random vector from a uniform distribution,
- $\mathbf{x}_*$ is the current global best solution,
- t presents the current iteration,
- $\mathbf{v}_i$ is the velocity,
- and $\mathbf{x}_i$ is the new solution,

Each bat generates a new solution locally with Eq. 3.21. In addition, the algorithm operates with loudness and pulse rates defined by Eq. 3.22 and 3.23.

$$\mathbf{x}_{new} = \mathbf{x}_{old} + \epsilon A^t \tag{3.21}$$
$$A_i^{t+1} = \alpha A_i^t \tag{3.22}$$
$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \tag{3.23}$$

Where:

- $\epsilon \in [-1, 1]$ is a random number,
- $A^t =< A_i^t >$ is the average loudness of all the bats at the iteration t,
- A_i^t is the loudness of the current solution x_i at iteration t, decreasing once the solution is improved,
- α and γ are constants,
- and r_i is the pulse rate, that increases once the solution x_i is improved.

The increase or decrease of the pulse rate and loudness variables are defined for any $0 < \alpha < 1$ and $\gamma > 0$ in Eq. 3.24.

$$A_i^t \to 0, r_i^t \to r_i^0, \text{as} t \to \infty \tag{3.24}$$

In the initial proposal [436], the author (Yang, 2010) admits that with specific parameter settings, the algorithm becomes the standard PSO (or eventually the Harmony Search). The main loop of the algorithm is shown in Figure 3.6.

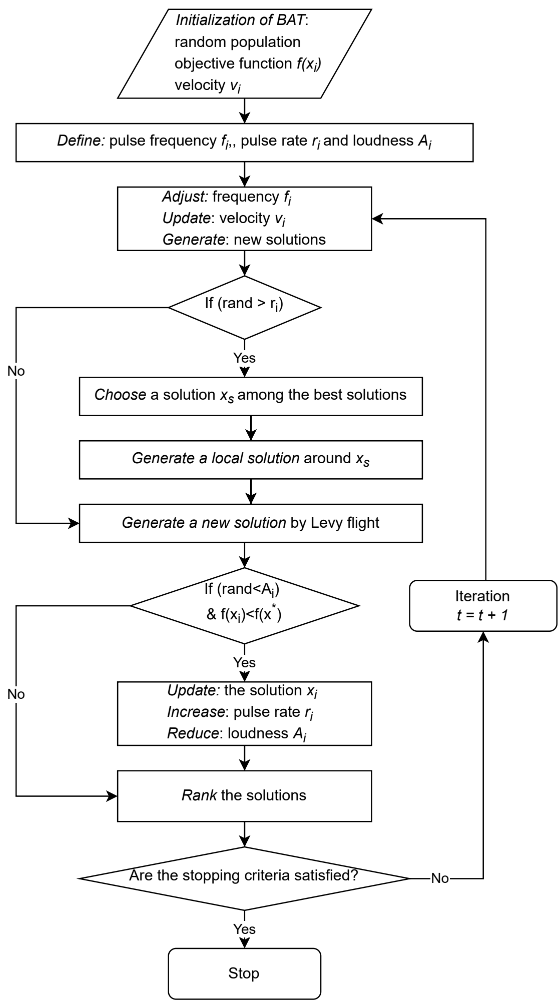


Fig. 3.6 Flowchart of the Bat Algorithm.

3.1.7 Whale Optimization Algorithm

Mirjalili and Lewis developed the Whale Optimization Algorithm (WOA) in 2016 [265]. The algorithm models the hunting behavior of whales and the spiral bubble-net maneuver.

Like the Grey Wolf Optimizer, the Whale Optimization Algorithm assumes that the best objective function value solution is the one closest to the target prey. Other solutions update their positions accordingly (Eqs. 3.25 – 3.28).

$$\boldsymbol{D} = |\mathbf{C}.\boldsymbol{X}^*(t) - \boldsymbol{X}(t)| \tag{3.25}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}^*(t) - \boldsymbol{A} \cdot \boldsymbol{D} \tag{3.26}$$

$$\boldsymbol{A} = 2\boldsymbol{a} \cdot \boldsymbol{r} - \boldsymbol{a} \tag{3.27}$$

$$\boldsymbol{C} = 2 \cdot \boldsymbol{r} \tag{3.28}$$

Where:

- t is the current iteration,
- $\boldsymbol{A}$ and $\boldsymbol{C}$ are the coefficient vectors calculated by Eqs. 3.27 and 3.28,
- $\boldsymbol{X}^*$ presents the position of the best solution,
- $\boldsymbol{X}$ is the current solution,
- . indicates an element-by-element multiplication,
- $\boldsymbol{a}$ linearly decreases from 2 to 0,
- and $\boldsymbol{r}$ is a random vector in range of $[0, 1]$.

The bubble-net attacking behavior represents the exploitation phase of the algorithm. The movement consists of two mechanisms: shrinking encircling, or spiral updating. The former is managed by decreasing the value of $\boldsymbol{a}$ in Eq. 3.27, while the latter simulates a helix-shaped movement by Eq. 3.29.

$$\boldsymbol{X}(t+1) = \boldsymbol{D'} \cdot \exp^{bl} \cdot \cos(2\pi l) + \boldsymbol{X}^*(t) \tag{3.29}$$

Where:

- $\boldsymbol{D'} = |\boldsymbol{X}^*(t) - \boldsymbol{X}(t)|$ is the distance of the i^{th} and the current best solu-

tion,
- b is a constant defining the spiral shape,
- and l is a random number in between [-1,1].

The selection of the bubble-net mechanism employed is random (Eq. 3.30).

$$\boldsymbol{X}(t+1) = \begin{cases} \boldsymbol{X}^*(t) - \boldsymbol{A} \cdot \boldsymbol{D} & \text{if } p < 0.5 \\ \boldsymbol{D'} \cdot \exp^{bl} \cdot \cos(2\pi l) + \boldsymbol{X}^*(t) & \text{if } p \geq 0.5 \end{cases} \tag{3.30}$$

Where:
- p is a random number in range [0,1].

The exploration phase of the Whale Optimization Algorithm simulates the search for prey behavior, moving away from a randomly chosen reference solution (Eqs. 3.31 – 3.32).

$$\boldsymbol{D} = |\boldsymbol{C} \cdot \boldsymbol{X}_{rand} - \boldsymbol{X}| \tag{3.31}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}_{rand} - \boldsymbol{A}.\boldsymbol{D} \tag{3.32}$$

Where:
- $\boldsymbol{X}_{rand}$ is a random solution from the current population.

Despite the relatively complex structure of the algorithm, the WOA has only two adjustable parameters: A and C. A flowchart of the algorithm is represented in Figure 3.7.

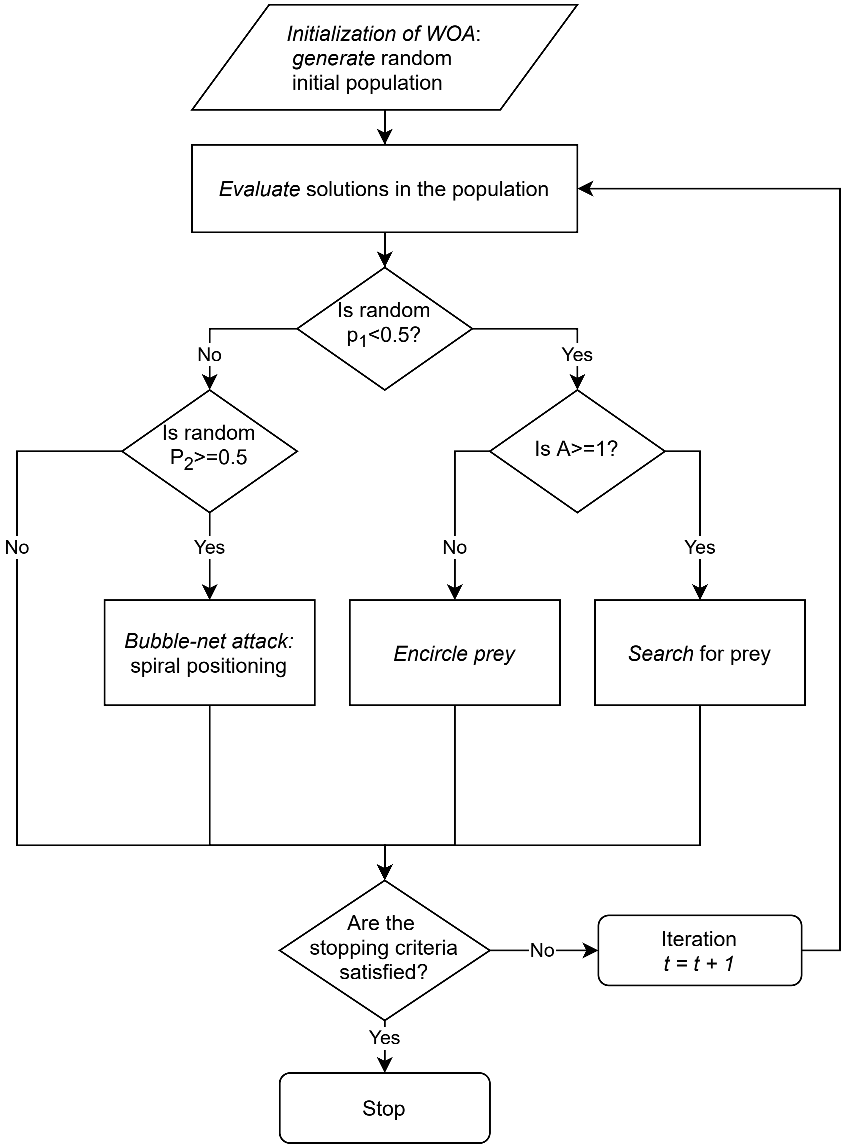


Fig. 3.7 Flowchart of the Whale Optimizer Algorithm.

3.1.8 Bacterial Foraging Optimization

Passino developed the Bacterial Foraging Optimization Algorithm in 2002 [307]. The algorithm models the foraging behavior of bacteria E.Coli living in human intestines, mainly four mechanisms observed in a natural bacterial system: chemotaxis, swarming, reproduction, and elimination-dispersal.

The **chemotaxis** procedure involves the swimming and tumbling of the E.coli cell via flagella. The movement is described by Eq. 3.33.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{3.33}$$

Where:

- θ represents the i^{th} solution,
- j, k, l are the indexes for the chemotactic, reproduction and elimination-dispersal events respectively,
- $C(i)$ is the size of the step taken in the random direction specified by the tumble (run length unit),
- and Δ indicates a vector in the random direction in between $[-1, 1]$.

The **swarming** movement simulates the aggregation into groups, moving in characteristic patterns with high bacterial density. The E. coli bacteria have a control guidance system repelled by alkaline and acidic environments and attracted towards neutral ones. The swarming behavior is presented in Eq. 3.34.

$$\begin{aligned} J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l)) = \\ \sum_{i=1}^{S} [-d_{attractant}\exp(-w_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] + \\ \sum_{i=1}^{S}[h_{repellant}\exp(-w_{repellant}\sum_{m=1}^{p}\theta_m - \theta_m^i)^2)] \end{aligned} \tag{3.34}$$

Where:

- $J_{cc}(\theta, P(j,k,l))$ is the objective function value to be added to the actual objective function to a present time varying objective function,
- S is the total number of solutions in the population,
- p is the number of variables to be optimized,
- $\theta = [\theta_1, \theta_2, ..., \theta_p]^T$ is a point in the p-dimensional search domain,
- and $d_{attractant}$, $w_{attractant}$, $h_{repellant}$, $w_{repellant}$ are different coefficients.

The **reproduction** step simulates the bacterium's ability to divide itself. When E. coli find a good food source, they get longer and break in the middle to form a replica of themselves. Analogously, the model abandons the solutions of worse quality while the best solutions are duplicated.

Finally, **elimination and dispersal** liquidate random bacteria at a small probability and replace them with randomly initialized solutions. This mimics the event in the natural bacterial population, during which a whole region of bacteria is killed or dispersed into a new environment. The complete flowchart of the algorithm is shown in Figure 3.8.

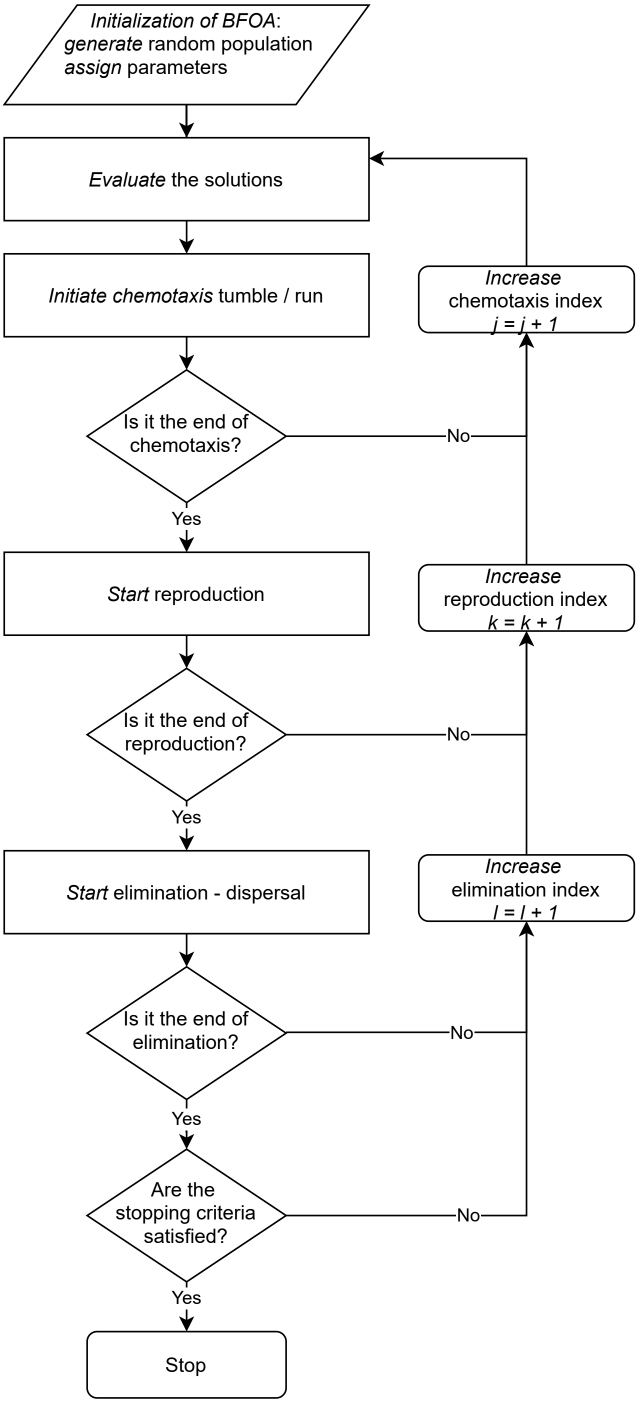


Fig. 3.8 Flowchart of the Bacterial Foraging Optimization Algorithm.

3.1.9 Firefly Algorithm

The Firefly Algorithm (FFA) was developed by Yang in 2009 [430]. The algorithm mimics the courtship behavior of fireflies. In the proposed model, fireflies' attractivity is defined by their ability to shine. Brightness decreases with distance. One firefly represents one solution of an optimized task, and the objective function computes its light-emitting abilities. In each iteration, every firefly looks around and moves towards the brighter fireflies by Eq. 3.35. The original proposal operates with the Cartesian distance (Eq. 3.36). A flowchart of the algorithm is in Figure 3.9.

$$\mathbf{x}_i = \mathbf{x}_i + \beta_0 \exp^{-\gamma r_{ij}^2}(\mathbf{x}_j - \mathbf{x}_i) + \alpha(rand - \frac{1}{2}) \tag{3.35}$$

Where:

- $\mathbf{x}_i$, $\mathbf{x}_j$ present the current and the more attractive solutions respectively,
- β_0 is the attractiveness at distance $r = 0$,
- r is the distance between the current and more attractive solution,
- γ is the variation of the attractiveness,
- α is the randomization parameter,
- and $rand$ is a random number drawn from a uniform distribution $[0, 1]$.

$$r_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| = \sqrt{\sum_{k=1}^{D}(x_{i,k} - x_{j,k})^2} \tag{3.36}$$

Where:

- i, j are the index keys of the compared solutions,
- $x_{i,k}$ is the k^{th} component if the i^{th} solution in the population,
- and D presents the dimensionality of the problem.

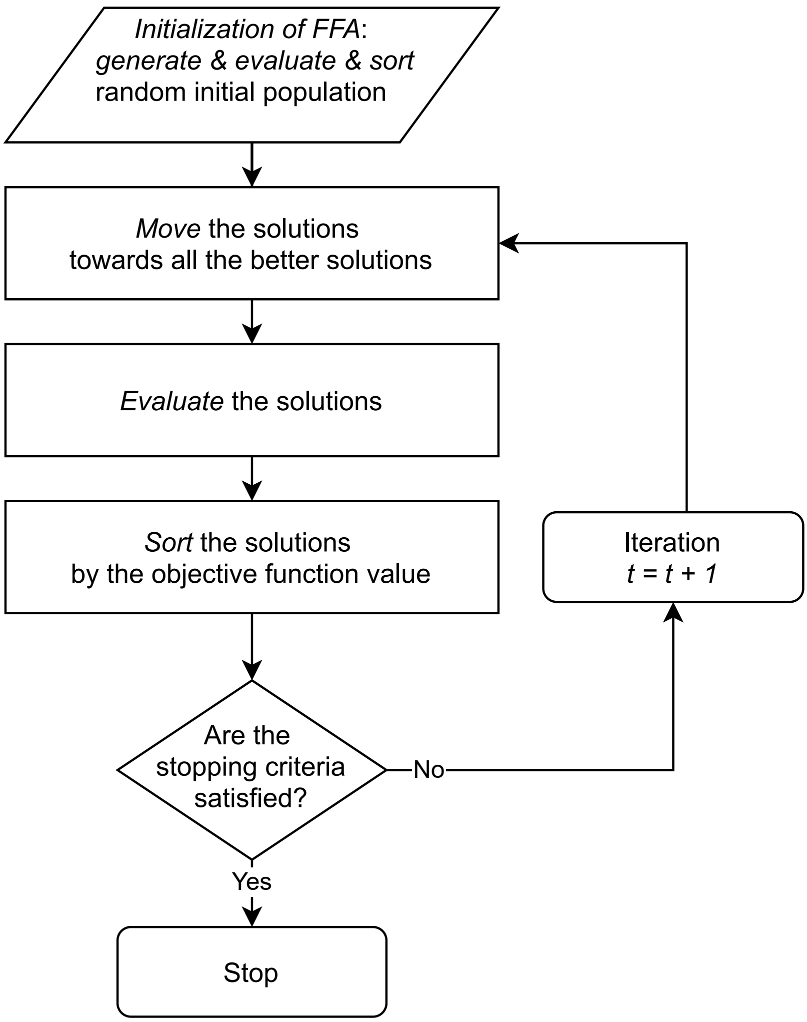


Fig. 3.9 Flowchart of the Firefly Algorithm.

3.2 Swarm Algorithms Criticism

The increasing emergence of swarm algorithms during the last few decades evoked a "novel algorithms dilemma." The introduction of metaphor-based development was followed by a massive wave of new swarm algorithms with gold rush resemblance [100]. To clarify the trend, Figure 3.10 shows the proportion of swarm-based algorithms compared to the total number of new metaheuristics created in the years 1973–2018 based on data from [275].

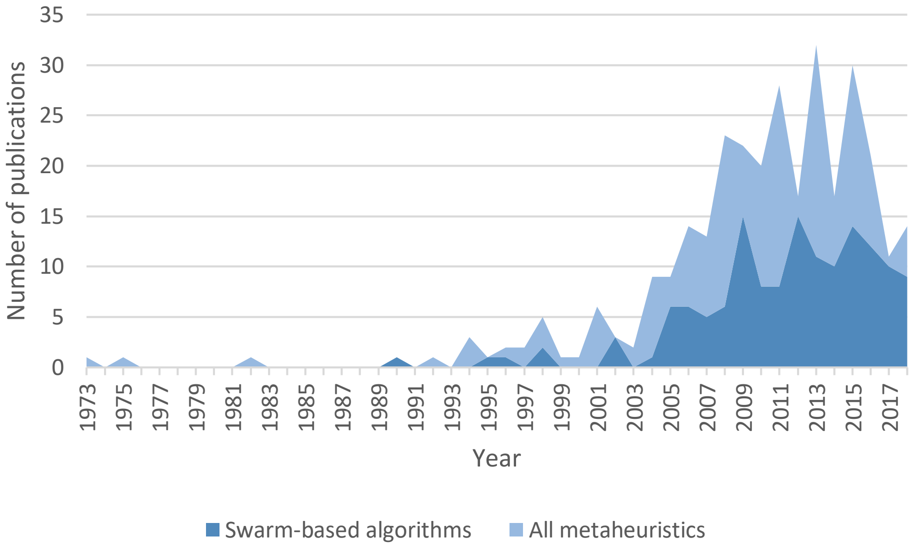


Fig. 3.10 The number of metaheuristic proposals in the years 1973-2018.

As a reaction to this "*metaheuristic avalanche*," an eye-catching project, the Evolutionary Computation Bestiary, started in 2018. The EC Bestiary[1)] catalogs the metaphor-based metaheuristics. The primary purpose of this catalog is to highlight the number of metaheuristics and to point out (and make fun of[2)]) the pitfalls of metaheuristic development. However, it also offers valuable sources that bring to light valid recurring mistakes linked to many metaheuristic proposals.

[1)] Available at `https://github.com/fcampelo/EC-Bestiary`, accessed 01/11/2021

[2)] See, e.g., the Twitter account Daily Bio-heuristics of metaheuristic inspiration for every day (`https://twitter.com/BioHeuristics`) or the Ghost Detection Algorithm Parody (`http://oneweirdkerneltrick.com/spectral.pdf`), both accessed 01/11/2021

The plethora of new metaphor-based optimizations provoked a tide of criticism, questioning mainly the asset and novelty of such algorithms [344, 367, 368]. The authors point out these reservations:

- Bio-inspired lingo
- Duality of the algorithms
- Too simplified models of bio-inspiration
- Excessive focus on competition and novelty
- Experiments of poor quality
- Undefined relation between academic and real-world problems

3.2.1 Bio-Inspired Lingo

Many metaheuristics use specific unorthodox language adapted to the metaphor that inspired them. Standard terms from evolutionary computation like the solution, population, or fitness are replaced by a range of names like flies, bats, bees, harmony, melody, and sounds. However, according to [367], the standard vocabulary would help a better understanding.

The lingo paradigm does not concern only recent swarm algorithms but the original contributions as well. Figures 2.1, 2.3, and 3.1 show that even the Genetic Algorithm, Differential Evolution, and Particle Swarm Optimization do not use the same terminology. However, with a proper definition of the special terms, different terminology mostly does not stand in the way of general understanding.

3.2.2 Duality of Algorithms

One of the crucial weak points of novel metaheuristic development is that many *novel* algorithms are, in fact, not novel at all. They merely repackage old ideas with new aliases. The problem becomes more pressing with the ever-rising number of metaheuristics.

Several algorithms, including the Harmony Search, the Grey Wolf Optimizer, the Firefly Algorithm, and the Bat Algorithm, were accused of reiterating existing principles [55, 414, 415]. Although to the Author's knowledge, there is no known metric to estimate the novelty of a proposed technique yet, several steps have been made in this direction.

In the *Comprehensive Taxonomy of Metaheuristics*, Molina et al. suggest a possible way to expose the duality of newly proposed algorithms through a behavior classification system (see Section 2.2.2) [275]. Using their original metrics, the authors concluded that 37% of the algorithms reviewed could be regarded as incremental variants of the existing algorithms rather than a novel algorithm.

In 2021, Armas et al. made exciting progress in this regard in the *Similarity in Metaheuristics: a Gentle Step towards a Comparison Methodology* [93]. Many algorithms reuse known techniques such as following the best solution, randomizing them, sorting them by quality, or mutation. Therefore, the author proposed a methodology for the algorithms' description based on module composition. The formal description of these algorithms enabled a new similarity metric applied to 15 algorithms.

3.2.3 Bio-Inspiration Stress

Metaheuristic development conceals a paradox regarding the inspiration source. On the one hand, there is an excessive focus on the inspiration source, admiring natural optimization principles and advocating why it should be modeled. On the other hand, the models are often oversimplified, hardly resembling the simulated phenomena [229, 367]. Regarding the aforementioned questionable novelty of some proposed optimizers, a different inspirational source should not justify recreating an already existing algorithm [275, 367].

Despite the overemphasis, the inspirational source is not the main characteristic of metaheuristic algorithms. A similar bio-inspiration does not necessarily lead to a similar algorithm. The Dolphin Partner Search and the Dolphin Echolocation algorithms may serve as an example since they fall into different behavior classes, despite the identical mammal model [275]. Then again, various inspirational sources do not guarantee a different algorithm.

3.2.4 Excessive Focus on Competition

The following criticism points to too much competitiveness. The current setup is over-focused on the algorithms' performance and how many algorithms were outperformed. This so-called *up-the-wall game* falls short for multiple reasons [52]. First, the results of such comparisons are oversensitive to numerous conditions: the benchmark and algorithms selection, parameter configuration, termination criteria, coding skills, or used programming language [217, 218, 232, 367]. Thus, the transferability of these results is questionable.

The No Free Lunch theorem claims that there is no ultimate function to solve every possible problem optimally [259]. Therefore, not only that one algorithm cannot outperform all others, but beating a bunch of others does not make the heuristic insuperable. It merely means that the results favored the promoted algorithm on the testbed being examined [51].

Competitive testing might imply which algorithm is faster or came with a better solution, but it does not reveal why [167]. However, understanding the inner mechanisms of metaheuristic optimizers is more important than performance, as performance is affected by too many factors. Also, the quality of the solution is not the only criterion for algorithmic evaluation. Finally, the focus on competition might indirectly lead to cheating, often seconded by private source codes [344].

3.2.5 Experiments of Poor Quality

In *Heuristic Scheduling: Running Away from the Bio-inspired Tsunami* [344], the author (Ruiz, 2002) highlights the frequent poor quality of studies. The black marks of such practice are inappropriate comparisons, lack of statistical testing to check significance, insufficiently large samples of objective functions, and lack of care in selecting competitors. Inappropriate comparisons rise from diverse starting points, different processor employment, compilers, and uneven stopping criteria.

The most crucial point, however, concerns the fair comparison condition. Many algorithms are compared to the basic versions of the algorithms, rather than the state-of-the-art methods. Parameters are tuned for the promoted optimizer only, algorithms examined on biased problems, results presented in tables only without additional context or proper interpretation [232].

3.2.6 Gap between Academic and Real-World Problems

The frequent practice of testing and validating optimization algorithms is to analyze the optimizer on a showcase minimization problem with a known location of the global optimum. Such problems are called benchmarks and consist of well-known problems like the De Jong's, Schwefel's, Rosenbrock's, Michalewicz's, Easom's Function, and many others.

While some algorithms define their own test sets, others use the advantage of already defined testbeds, e.g., from the IEEE CEC benchmark competitions. The CEC test suites provide a wide range of optimization tasks covering various characteristics such as the simple unimodal, multimodal, and compositions problems. The testbed has already defined evaluation criteria as a bonus, and the proposed algorithm may be easily compared to other optimizers.

However, there is still an unresolved question: How well do standard benchmark problems reflect dynamics of real-world optimization tasks [399]? There often may be little relevance of academic problems to the real world [343]. Defining the relation of benchmark problems to real-world optimization tasks remains one of the open issues of heuristic optimization.

One of the exposed benchmark handicaps concerns a missing noise [232]. Although noise is a characteristic component of real problems, most academic problems suppress it. In this matter, the BBOB 2009 benchmark creates an exception, as it includes the noise into the problem testbed [137].

3.2.7 General Disrespect for Novel Metaheuristic Proposals

The reservations above result in preconceptions harming novel metaheuristic proposals, though mainly during the invisible process of reviews and rejections that are part of publication practice. The author of Running Away from the Bio-inspired Tsunami (Ruiz, 2002) [344] addresses these algorithms as spam and hyperbolically suggests a possible solution to the problem with his own metaheuristic-like acronym *RTHOTP* – meaning: "*Reject The Hell Out of These Papers.*" Of course, an actual breakthrough paper would probably never be rejected, despite its potential natural inspiration source. Still, the statement may reflect general scientific public opinion on the novel algorithms avalanche.

The publication instructions for the Swarm Intelligence Journal employed since the 11$^{\text{th}}$ Volume[3)] may serve as an example of such a mindset. The guidelines directly address the metaheuristic issue, stating:

> *"There is a relatively recent trend that consists in taking a natural system/process and use it as a metaphor to generate an algorithm whose components have names taken from the natural system/process used as metaphor. This algorithm is often advertised as a "new natural metaphor algorithm" and used to solve a specific problem (most of the time an optimization problem).*
>
> *Unfortunately, this approach has become so common that there are now hundreds of so-called "new" algorithms that are submitted (and unfortunately often also published) to journals and conferences every*

3) https://media.springer.com/full/springer-instructions-for-authors-assets/pdf/1593723_Additional_submission_instructions.pdf, accessed 01/11/2021

year. The problem is that it often takes a lot of work and effort for editors, and sometime referees, to understand why the authors are using the proposed metaphor, what is really new and what is the same as the old with just a new name, and whether the proposed algorithm is just a small incremental improvement of a known algorithm or a radically new idea.

The number of such manuscripts submitted to Swarm Intelligence has greatly increased in the last few years. I have therefore asked the associate editors to pay particular attention to these "natural metaphor" inspired manuscripts and to send them to referees only if the manuscript seems to be of very high quality. In other words, I have asked the associate editors to increase the number of manuscripts that they reject directly so as to decrease the work load on referees, who are a precious resource that we need to protect."

(Instructions for publications in the Swarm Intelligence Journal,

Volume 11, p.1)

The instructions further direct some additional rules for metaphor-based algorithms: advising that the inspirational source should be thoroughly scientifically understood and must match the simulation model in a formal mathematical way. It should avoid self-made bio-linguistic terms, and the authors are expected to advocate the novelty of their approach. Without meeting these conditions, further rejections are to be referred to the guidelines document. Also, the instructions highlight the importance of fair comparison and good practice in experiments. The Swarm Intelligence Journal's full instructions are quoted in Appendix B: *Instructions for the Swarm Intelligence Journal Submissions Regarding Novel Natural Metaphor Articles.* Similar guidelines face mischievous practice in the Journal of Heuristics (Appendix C).

In November 2021, more than one hundred of scientists signed a letter named *Metaphor-based metaheuristics, a call for action: the elephant in the room* [15].

The letter warned from the mischievous practice and recommended optimization oriented journals to accept only articles that:

i use standard terminology

ii provide novel and useful concepts

iii name the motivation of the research based on scientific base

iv present a fair comparison with state-of-the-art methods and practices

3.2.8 Lack of Insight

Metaheuristics often deal with problems in a black-box manner. They serve as a general optimization tool to solve any problem. However, though it neatly illustrates the versatility of these optimizers, it also raises concerns about the credibility of the results. How did the algorithms discover the proposed outcome? Is the examined system biased?

A sole solution often may not represent a satisfactory outcome. The end-user might benefit from additional information that would undertake various what-if scenarios and outline which hyper parameters strongly affect the outcome and which barely.

These questions ignited the interest in a new trend: the Explainable Artificial Intelligence (XAI), which forms this field towards better understandability, interpretability, and transparency of AI outcomes. In Evolutionary Computation Techniques, it means, among others, advancing optimizers toward a better understanding of the inner dynamics of the algorithms and their parameters [19].

3.2.9 Implications of Criticism

There are three general responses to the abovementioned reservations:

1. Complete ignorance of critical points
2. Rejection of novel metaheuristics
3. Reflection of valid critical points in future metaheuristic development

Unfortunately, the first two approaches are adopted in most scenarios: the users of metaheuristics, completely ignoring the suggestions for good practice on one hand, versus the reviewers, who are getting fed up with metaheuristic malpractice and are inclined to reject the novel metaheuristic once they have read the title.

In 2005, Lee and Geem proposed the Harmony Search algorithm inspired by musicians' improvisation [233]. Five years later, the algorithm was proved to be a particular case of Evolutionary Strategies [414]. Its novelty and contribution were impeached. However, the effect was minimal. Figure 3.11 presents the number of publications citing the Harmony Search proposal versus the exposing publication. The ratio illustrates that the algorithm's popularity was not caused by, nor despite, the duality revelation. It was rather unnoticed.

Most recently, in 2020/2021, a group of researchers stood up against the practice of this corrupt science and worked towards solving the named issues. Researchers called attention to current metaheuristic problems [132, 344, 368]. IEEE established a benchmarking taskforce[4)] and networks[5)]. Molina et al. (2020) fought the metaphor threat by proposing behavior-based classification of metaheuristics [275]. LaTorre et al. (2020) proposed a set of guidelines for fair methodology in metaheuristic development and comparison [232]. The first steps were even made towards the detection of similarity between algorithms [93]. In 2020, more

[4)] `https://cmte.ieee.org/cis-benchmarking/`, accessed 01/11/2021
[5)] `https://sites.google.com/view/benchmarking-network/home`, accessed 01/11/2021

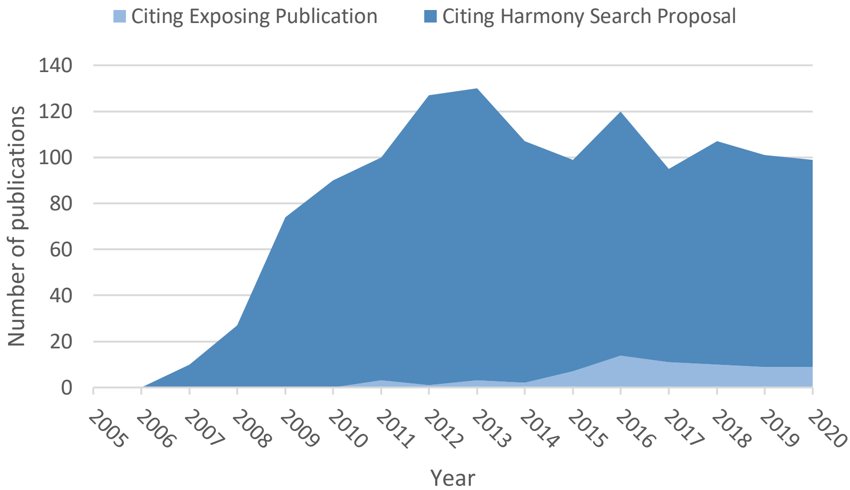


Fig. 3.11 Number of publications citing the Harmony Search Algorithm versus the exposure publication in 2005-2020.

than a dozen researchers created guidelines for benchmarking, summarizing their ideas on the best practice and declaring open issues [27]. To raise the standard, some journals made avoidance of the malpractice presented mandatory (see Appendices B and C). The mood is supported of the evolution of the metaheuristic optimization field.

Still, when compared to the number of contrasting literature, the effort to improve the metaheuristic situation concerns just a drop in the ocean of academic publications. In addition, critical publications mainly state the problem and mark the territory of metaheuristic badlands. The reservations point to the corrupt practice and wrongs in metaheuristic optimization but rarely offer solutions, and the applications are scarce.

3.3 Recommendations for New Metaheuristics Development

The criticism mentioned can be formulated on a positive note into guidelines for metaheuristic design. The Author presents a set of rules based on recommendations from the following sources: [27, 178, 232, 275, 344, 367]. The guidelines may be sorted into the algorithm's design, experimental and comparison practice in Sections (Section 3.3.1 - 3.3.4). Finally, Section 3.3.5 summarizes all the proposed recommendations in a convenient check list.

3.3.1 Guidelines for Algorithm Design

The first set of recommendations concerns the development of a new metaheuristic or its variant. According to LaTorre [232], researchers should name their motivation for the development. New algorithms should not root from finding a new metaphor but from an original, clearly expressed idea. Algorithms are advised to use standard vocabulary rather than metaphor-based terms [368] and adopt flowchart descriptions for a better understanding.

New algorithms should honor the *keep it simple* principle. The contribution of each component should be analyzed and carefully considered, if the individual contribution is small. Beautiful examples of such practice are in [232, 315], where simplifying advanced, even state-of-the-art, algorithms.

The algorithm's design should always be transparent. It is advisable to keep in mind the possibilities of being configured by automatic techniques right from the start. New algorithms should be handled as a configurable framework with each module, function, and parameter clearly defined, seeking the way toward explainable AI [372].

Development recommendations culminate in the absolute necessity of sharing the source codes of novel algorithms [178, 275]. Public source codes benefit both the developer and the user. They improve the usability of the algorithm, enable future advancement, and allow for the detection of potential glitches in the proposal.

3.3.2 Guidelines for Selection of Benchmark and Algorithms to Be Compared

After developing a new algorithm, standard practice compares the proposal with other metaheuristics to highlight its contribution and performance. The fairness of such comparisons is crucial and concerns the whole design of the experiment.

The first step is the selection of algorithms to be compared. Many algorithms compare with original well-established algorithms only (see, e.g., [17, 49, 164, 194, 238, 314, 386, 391]). However, general advice is to employ also the reference version of the algorithm also (i.e., the one that is modified), other algorithms based on a similar principle, and the best algorithms so far for the selected benchmark [232].

The last point may be debatable since it resembles a parallel with the match of David and Goliath: How could newborn ideas be expected to outperform algorithms fine-tuned for decades? In this regard, it is essential to note that superior performance is not the only vital contribution made by an algorithm if the algorithm offers a relevant methodology advancement or a completely new technological procedure. At the same time, many algorithms, like the Firefly Algorithm, Cuckoo Search, and others, are widely used, despite not being state-of-the-art. Still, it is unacceptable to cherry-pick algorithms with worse performance, to claim the effectiveness of an algorithm proposal. Thus, sole comparison with a basic, randomly chosen algorithm should be avoided [232].

The appropriate selection of algorithms to be compared relates to the goal of the experiment. Where the goal is to create an algorithm of superb performance on the testbed being examined, the state-of-the-art comparison is in place. Is the goal to present new methodology? It should be compared to similar algorithms to show the difference. If the goal is to advance a piece of standard optimization technology, the experiment should prove the transferability of the knowledge and its usefulness for other optimization techniques.

The selection of the benchmark problem test set is also vital. The benchmarks should be biased neither towards the coordination system nor towards the origin. Researchers are advised to use the standard competition benchmark test suites, e.g., the IEEE CEC benchmarks [232], which employ problems of broad characteristics, and potential pitfalls are more likely to be exposed.

3.3.3 Guidelines for Experimental Setup

The actual experiment should provide conditions as close as can be for all the algorithms being examined. One's own implementations with an open source code are preferable to literature-based results. Each algorithm should employ parameter settings tuned for the problems being examined. All results should be tested for statistical significance.

When following the CPU computation time as a termination condition, all algorithms should be examined on the same computer, in the same programming language, programmed by the same programmer, and ideally, share most functions.

3.3.4 Guidelines for Results' Analysis

Experiments should allow negative results. Aside from performance-oriented tests, they should focus on understanding and deepening the knowledge of algorithms. The presentation should provide tables, statistical data, figures, ranks, and charts. The discussion should not just declare a winner but analyze and interpret the context of the results. Problem characteristics might lead the way. Discovering that one algorithm excels on separable/multimodal/unimodal/noisy problems provides more information than that one algorithm outperformed some others on a randomly composed benchmark test set.

At the same time, practitioners should be cautious about generalization of results [51]. Each experiment should be described precisely. It is essential to distinguish between algorithm and algorithm instances (e.g., Particle Swarm Optimization vs. Particle Swarm Optimization with particular parameter settings) and problems vs. problem instances (e.g., the Sphere Function vs. the Sphere Function in 5 dimensions with boundary limitations) [51].

Finally, the analysis of results should fulfill and evaluate the initial hypotheses declared during the motivation stage. The need for justification of new algorithm development was expressed in multiple publications [93, 232, 344, 368]. According to LaTorre [232], the arguments in favor of the usefulness of novel methodology are: undeniable novelty, results surpassing state-of-the-art optimizers, and contribution to methodology, the last of which has to be precisely described and argued.

In this regard, the Author would like to add another motivation for the justification of novel metaheuristic development: aiming the development of novel algorithms at the known problems of current metaheuristic practice. Tackling the fundamental puzzles of metaheuristic optimization may lead to the evolution of metaheuristics.

3.3.5 Guidelines Summary

Guidelines for Algorithm Design

- Name motivation (not metaphor-based)
- Use standard vocabulary
- Share the source code of novel algorithms
- Describe algorithms with flowcharts for a better understanding
- Analyze components of the proposed algorithm individually
- Keep it simple

Guidelines for Selection of Algorithms to be Compared and Benchmark

- Select algorithms to be compared with respect to the goal of the experiment

For performance-oriented comparison, compare algorithms with:

- Original version of the algorithm (first proposal)
- Reference version of the algorithm (the one that is modified)
- Best algorithms so far on the benchmark being examined (competition winner)
- Other algorithms operating on a similar principle

Select benchmark problems:

- Of broad characteristics without bias
- Prefer standard benchmark test sets

Guidelines for Experimental Setup

- Prefer own implementation over literature-based results
- Provide the same conditions for all the experiments
- Share the source codes of all the algorithms
- Tune the parameters of all the algorithms for the problem at hand with

statistical tests
- Combine multiple performance measures

When examining the CPU execution time, all algorithms should:

- Be coded by the same programmer
- Be coded in the same programming language
- Share most functions
- Be examined on the same computer

Guidelines for Results' Analysis

- Use statistical tests for significance
- Allow negative results
- Show results in context, provide interpretation
- Be cautious with generalization
- Depict the results in both graphs and tables
- Advocate assets and contribution of the algorithm (novelty/performance/ methodology/challenge particular problem)

4 Goals and Methods of the Dissertation

1. **Map the current scene** of modern swarm algorithms, its trends, and challenges.
2. **Investigate** the methods addressing the weaknesses of swarm algorithms.
3. **Propose** a set of recommendations for new metaheuristics creation.
4. **Proof of concept testing**: Implement the proposed recommendations and methods in a new swarm algorithm.
5. **Evaluate the benefits** of the proposed algorithm for applied sciences.

Methods of fulfillment of goals of the dissertation include:

Critical analysis:

- Of novel metaheuristics creation process and its challenges.
- Of modifications, trends, and weaknesses of swarm algorithms.

Experiments:

- The proposed algorithm is compared to other state-of-the-art algorithms on various benchmarking testbeds IEEE CEC 2015, and 2017.
- The experiments focus on the investigation of the dynamics and inner processes of the proposed algorithm.

Evaluation:

- Evaluation of the algorithms comply with the evaluation criteria [25].
- The results are examined for statistical significance.
- Identification of the types of optimization problems suitable for the proposed techniques through an in-depth analysis of results.

Programming:

- Algorithms are coded in Python or MATLAB.
- Results are examined in Wolfram Mathematica.

5 Bison Algorithm

Even though many current algorithms try to balance the exploration and exploitation rate to prevent premature stagnation, many metaheuristics prioritize exploiting the found solutions at the expense of the exploration in later iterations, which may lead to the unintended improvement of local optimum solutions instead of finding the global one.

To avoid this pattern, the Author proposed a new swarm optimization algorithm that emphasizes the exploration process. The algorithm finds inspiration in the protection mechanism and the running advancements of bison herds.

5.1 Motivation

The proposed algorithm was designed to prove two concepts: applying good practice recommendations and advocating a novel development justification argument. Manifold publications (like [27, 84, 167, 344, 367, 368]) point to the common substandard practice of metaheuristic proposals and result in general conclusions. The actual applications of these studies are, however, scarce (see, e.g., [232]). The proposed algorithm was designed to follow the presented guidelines, demonstrating the significance of the recommendations from Section 3.3.

The second motivation was to advocate an additional justification argument for metaheuristic development. So far, three arguments have justified the usefulness of novel algorithm proposals [232], namely:

- Superb performance surpassing state-of-the-art optimizers
- Absolute novelty
- Contribution to methodology, e.g., improving a commonly used technique

In addition, the Author would like to add a fourth argument,

- Aiming development at tackling the current optimization problem

To support this argument, an algorithm aimed at fighting the local optimum containment problem was developed. It embeds a unique mechanism to avoid local optimum and ensures the same rate of exploration throughout the whole optimization process.

5.2 Inspiration

The Bison Algorithm simulates the typical behavior of bison herds: swarming and running. When predators attack bison, they form a circle with strong cattle at the outer edge of the circle. The weaker ones (like calves) hide inside the circle in a safer position. When running, bison can reach a velocity of 56 km per hour and keep it up for as much as thirty minutes [36, 333]. These behavior patterns serve as model exploitation and exploration techniques for a new, swarm-oriented methodology called the Bison Algorithm.

5.3 Definition

The main characteristic of the Bison Algorithm lies in the division of the population into two groups. The first group, called the swarming group, takes care of exploitation, approaching closer to the center of several fittest solutions. In contrast, the second group steadily examines the search space for new, promising solutions.

The algorithm is outlined in Algorithm 1 and a simplified flowchart in Figure 5.1. The UTB A.I. Lab Github repository[1)] hosts the source code of the Bison Algorithm, which is free to use.

[1)] `https://github.com/TBU-AILab/Bison-Algorithm`, accessed 01/11/2021

Algorithm 1 Pseudocode of the Bison Algorithm.

1. Initialization:
Objective function: $f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1,\ x_2,\ ...,\ x_D)$
Generate swarming group $\boldsymbol{SG}$ randomly
Generate running group $\boldsymbol{RG}$ around $\boldsymbol{x}_{best}, f(\boldsymbol{x}_{best}) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \ \epsilon \ \boldsymbol{SG}$
Select elite bison group $\boldsymbol{EG}$ based on obj. function value
Sort the population and redefine $\boldsymbol{SG}$ based on obj. function
value $\boldsymbol{SG}_j = (sort(\boldsymbol{SG} \cup \boldsymbol{EG}))_j, j = 1, ..., |\boldsymbol{SG}|$
Generate the *run direction* vector (Eq. 5.4)
2. For every iteration i do
3. Compute the *center* of $\boldsymbol{EG}$ (Eqs. 5.1, 5.2)
4. For every bison $\boldsymbol{x}$ in $\boldsymbol{SG}$ do
5. Compute a new candidate solution $\boldsymbol{x}_{new}$ (Eq. 5.3)
6. If $f(\boldsymbol{x}_{new}) < f(\boldsymbol{x})$ then $\boldsymbol{x} = \boldsymbol{x}_{new}$
7. End for
8. Adjust *run direction* vector (Eq. 5.5)
9. For every bison $\boldsymbol{x}$ in $\boldsymbol{RG}$ do
10. $\boldsymbol{x} = \boldsymbol{x} +$ *run direction* (Eq. 5.6)
11. End for
12. Redefine $\boldsymbol{SG}$ for the next iteration $i+1$:
$\boldsymbol{SG}_{i+1,j} = (sort(\boldsymbol{SG}_i \cup \boldsymbol{RG}_i))_j, j = 1, ..., |\boldsymbol{SG}|$
13. End for

The control parameters of the Bison Algorithm include:

- **Population** *NP* defines the number of individuals in the population,
- **Elite group size** *EG* determines the number of fittest solutions for center computation,
- **Swarm group size** *SG* represents the number of solutions in the swarming group,
- **Overstep parameter** defines the maximum length of the swarming movement to the center (0 meaning no movement, 1 meaning a maximum movement to the center)

The parameters were tuned on the IEEE CEC 2017 benchmark for 10 and 30 dimensions, recommending the following configuration: population of 50, elite group size of 20, swarm group size of 40 and overstep of 3.5 [210].

5.4 Swarming Group

The swarming group shifts its members in the direction of the center of several fittest solutions. Preliminary experiments promoted ranked center computation (Eqs. 5.1, 5.2), considering the order of the best solutions during the calculation. Every solution in the swarming group computes a new solution candidate that replaces the current swarmer if it improves its quality (Eq. 5.3).

The flowchart Figure 5.2 represents the principle of the swarming movement. Figure 5.3 a) shows the cumulative locations of the swarming group in 50 iterations during the actual run on the 2-dimensional Rastrigin's Function.

$$\boldsymbol{weight} = (10, 20, 30, ..., 10 \cdot EG) \tag{5.1}$$

$$\boldsymbol{ranked\ center} = \sum_{i=1}^{EG} \frac{weight_i \cdot \boldsymbol{x}_i}{\sum_{j=1}^{EG} weight_j} \tag{5.2}$$

$$\boldsymbol{x}_{new} = \boldsymbol{x} + (\boldsymbol{ranked\ center} - \boldsymbol{x}) \cdot \boldsymbol{random}(0, overstep) \tag{5.3}$$

Where:

- EG is the elite group size parameter,
- i, j are indexes for center computation $i, j = 1, ..., |EG|$,
- $\boldsymbol{x}$ and $\boldsymbol{x}_{new}$ represent a swarming group solution and a new position candidate,
- $\boldsymbol{random}$ yields a random vector of unified distribution within the arguments,
- and $overstep$ defines the maximum length of the swarming movement.

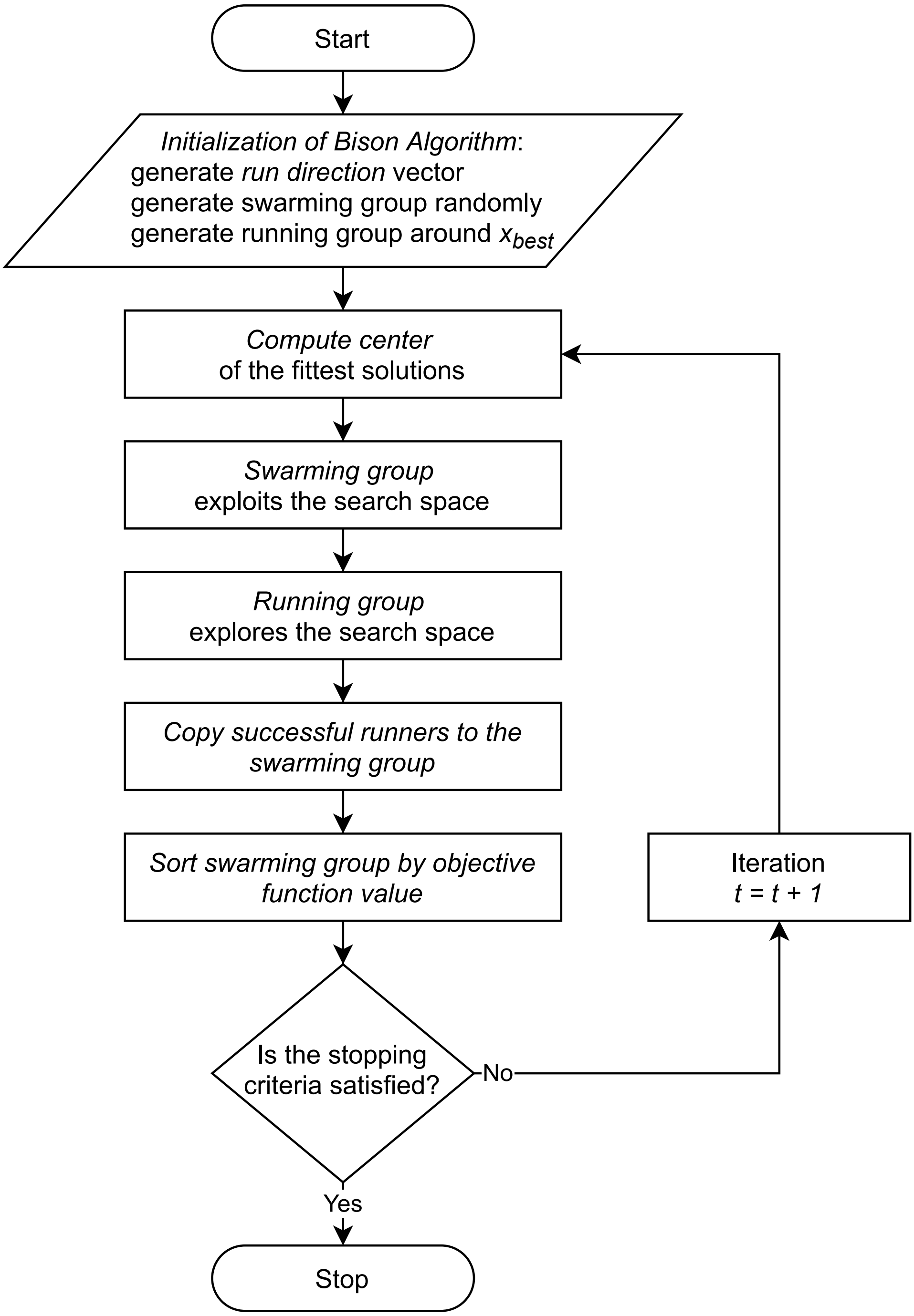


Fig. 5.1 Simplified flowchart of the Bison Algorithm.

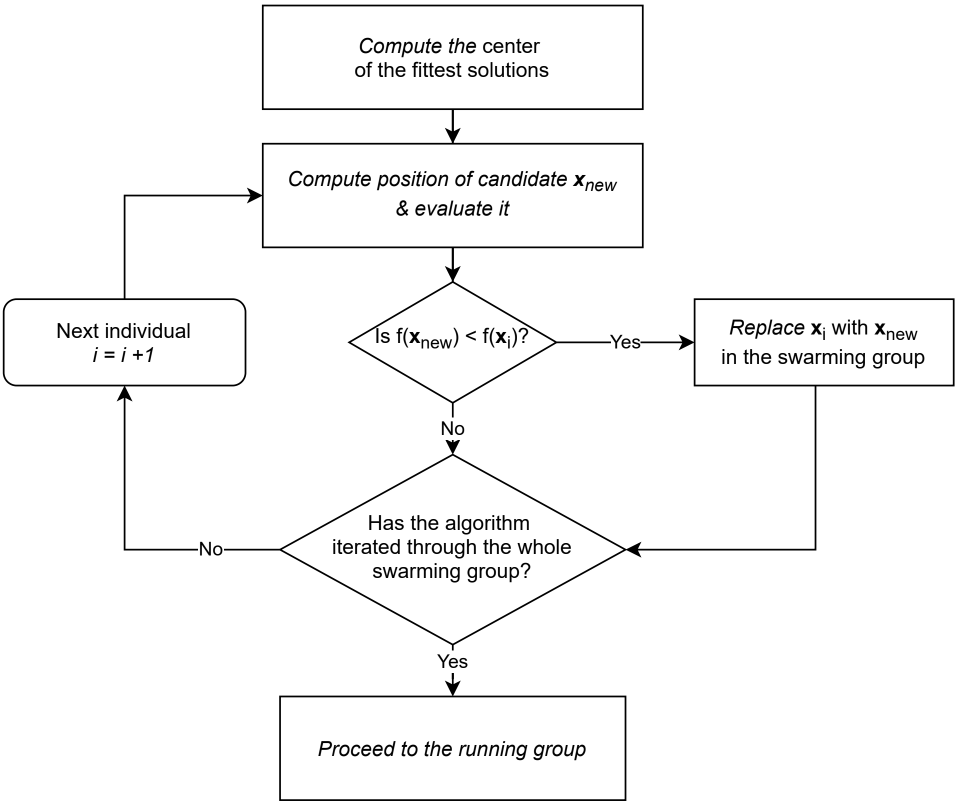


Fig. 5.2 Flowchart of swarming movement.

5.5 Running Group

The running group systematically shifts the solutions in the run direction vector (Eqs. 5.4, 5.5, 5.6). The algorithm generates the run direction vector randomly during the initialization (Eq. 5.4), and then it slightly changes in every iteration (Eq. 5.5). The running movement happens even if it lowers the quality of the solutions. Figure 5.3 b) presents the cumulative movement of the running group of 50 iterations.

$$run\ direction_D = random(\frac{ub - lb}{45}, \frac{ub - lb}{15}) \tag{5.4}$$

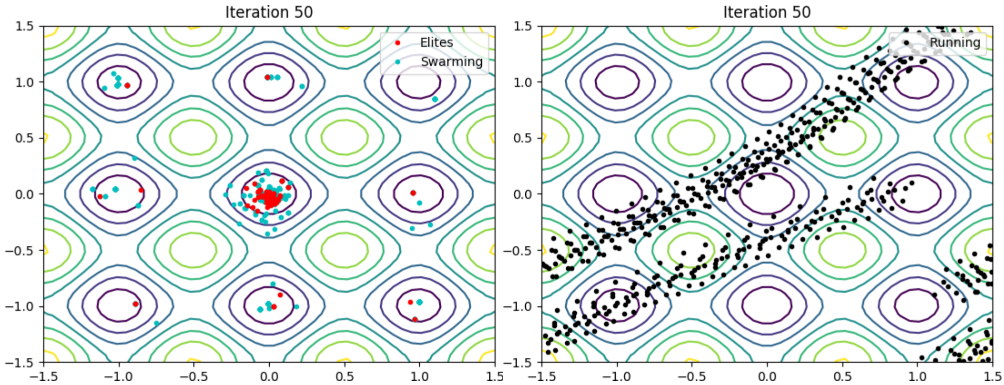
$$\boldsymbol{run\ direction}_{i+1} = \boldsymbol{run\ direction}_i \cdot random(0.9, 1.1) \tag{5.5}$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{run\ direction}_{i+1} \tag{5.6}$$

Where:

- $\boldsymbol{run\ direction}$ is the run direction vector,
- D is a dimension,
- i is current iteration,
- ub and lb are the upper and lower boundaries of the search space,
- and $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ represent a running group member and its updated position.

This exploration implementation crosses the borders quite often. The study *Border Strategies of the Bison Algorithm* [213] examined the most feasible border strategy and recommended using the hypersphere strategy, connecting the dimensional upper and lower boundaries. Figure 5.4 shows the movement of the whole population to validate the model of the proposed algorithm on a 2-dimensional Schwefel's Function.



(a) Swarming group (b) Running group

Fig. 5.3 Cumulative movement of the swarming group (a) and the running group (b) on the 2-dimensional Rastrigin's Function.

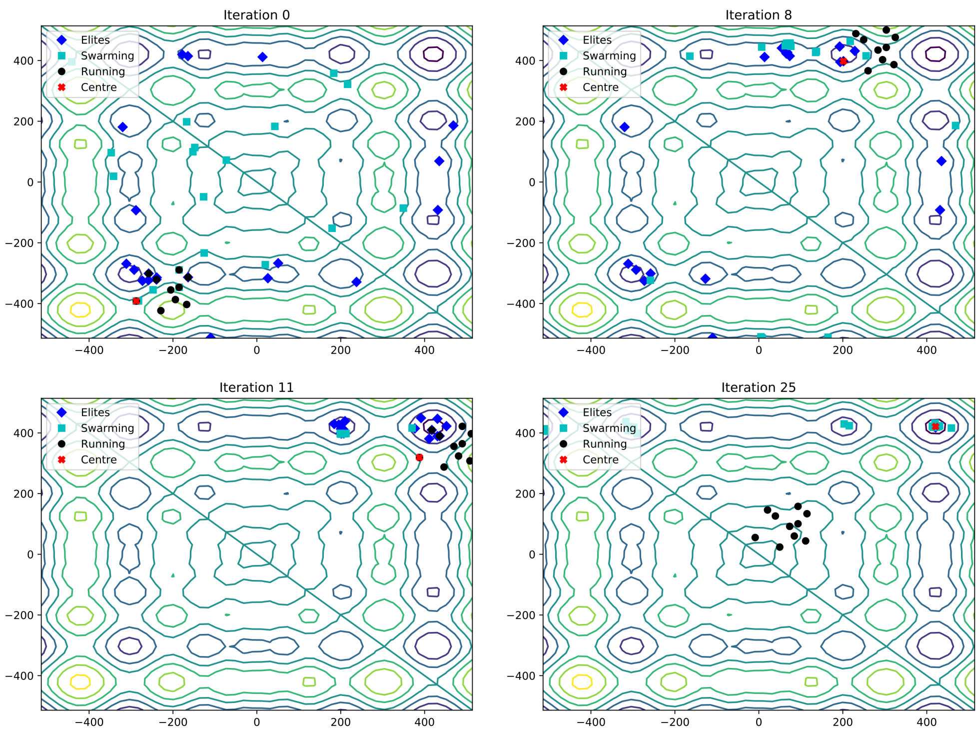


Fig. 5.4 Positions of solutions on the 2-dimensional Schwefel's Function.

6 Modifications of the Bison Algorithm

The development of the Bison Algorithm was a gradual process that started with a conundrum. Imagine a search space with a global optimum surrounded by a narrow, monotonous neighborhood. When adding the local optimum containment problem, trapping all solutions in one area may considerably hinder the capability of finding the true optimum.

The Bison Algorithm was developed with an escape mechanism from local containment. The key idea was to employ the unique exploration and exploitation potential of bison herds. Simulating these behavior patterns in two groups based on their objective function value separately founded the first version of the algorithm [214].

Further variations investigated the options for how to advance the explorative capacity of the algorithm. The first modification tested the benefits of coherent exploration, which was ultimately employed in all the modifications that followed [212]. The Bison Seeker modification investigated the behavior change of the explorative solutions when discovering an attractive solution [211]. The Run Support Strategy added a new parameter to support the utilization of the discovered solutions [216].

Figure 6.1 maps the development process, application, and corresponding studies. The following sections present the modifications of the Bison Algorithm.

- Section 6.1 describes the original proposal of the Bison Algorithm,
- Section 6.2 presents the Run Support Strategy,
- Section 6.3 outlines the Bison Seeker modification,
- and Section 6.4 explains the self-adaptive variation of the Bison Algorithm.

Finally, Section 6.5 presents how each modification handles the local optimum containment challenge.

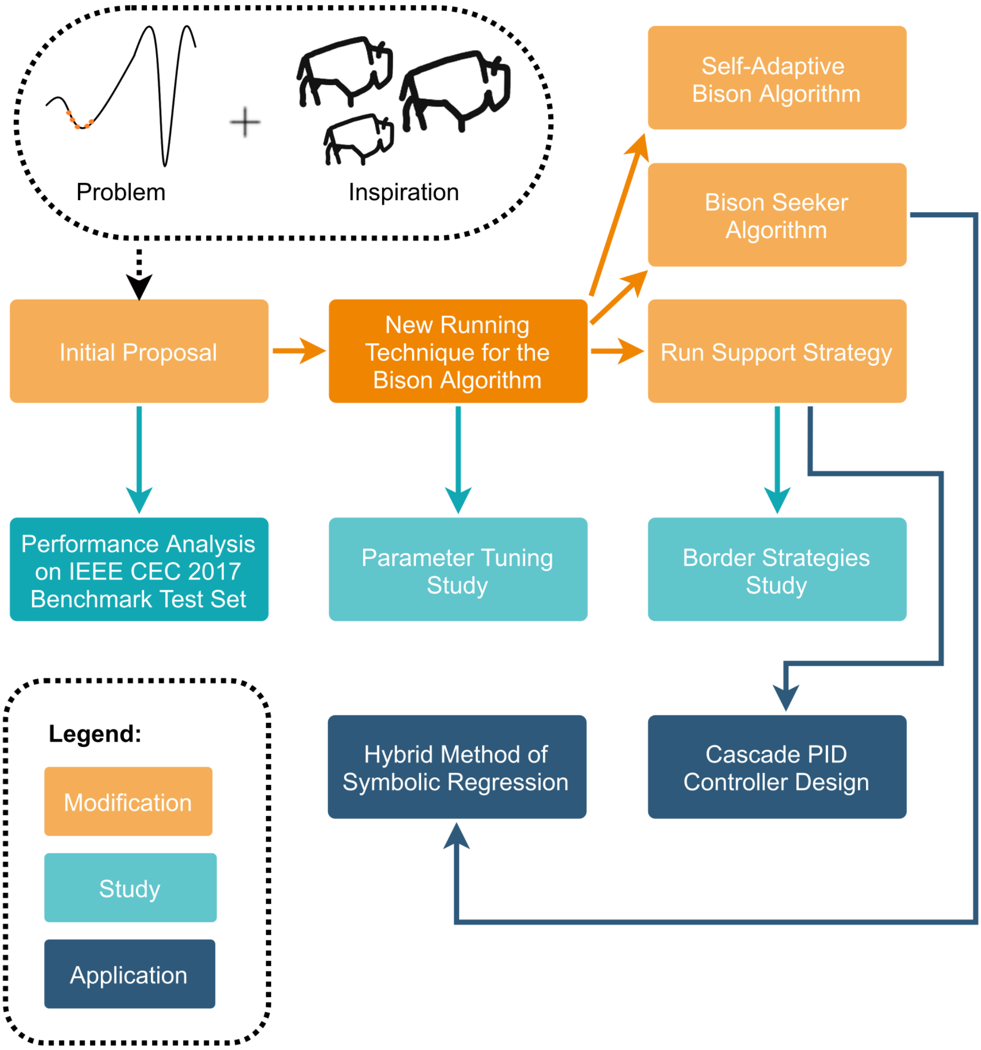


Fig. 6.1 Pipeline of the Bison Algorithm development process.

6.1 Original Bison Algorithm Proposal

The initial proposal of the algorithm divided the population into two groups solely by the objective function value criterion. Solutions of better quality were in the swarming group, while the rest were in the running group. Therefore, the groups could swap their members easily. Figure 6.2 shows a movement of the original algorithm, namely the scattering of the running group throughout the search area since iteration 1.

This feature suppressed the potential asset of "*collective group search*". Therefore, the redefinition of the Bison Algorithm, as presented in Section 5.3 and Algorithm 1, suppressed this trait [212]. This modification kept the exploring solutions together, and successful solutions were copied to the swarming group.

The original algorithm is presented in pseudocode Algorithm 2 and Figure 6.3. Besides the exploration-exploitation solution classification, the algorithm works with the same mechanism for swarming and running movement.

Algorithm 2 Pseudocode of the Original Bison Algorithm.

1. Initialization:
   Objective function: $f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1,\ x_2,\ ...,\ x_D)$
   Generate swarming group SG randomly
   Generate running group RG around $\boldsymbol{x}_{best}$
2. For every iteration i do
3. swarming group swarm
4. running group run
5. sort the whole population by the objective function value
   -> swarming group SG = $\{x_1,\ x_2,\ \ldots,\ x_{|SG-1|}\}$
   -> running group RG = $\{x_{|SG|},\ x_{|SG+1|},\ \ldots,\ x_{|NP|}\}$
6. End for

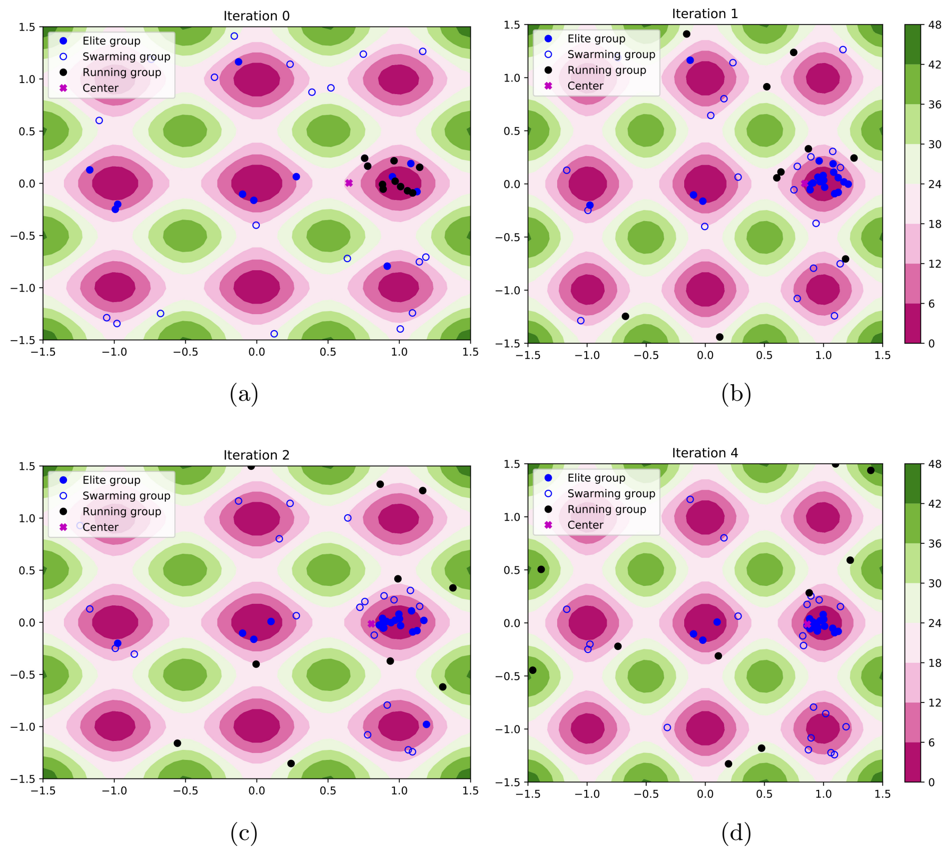


Fig. 6.2 Movement of the swarming group and running group solutions in the original Bison Algorithm.

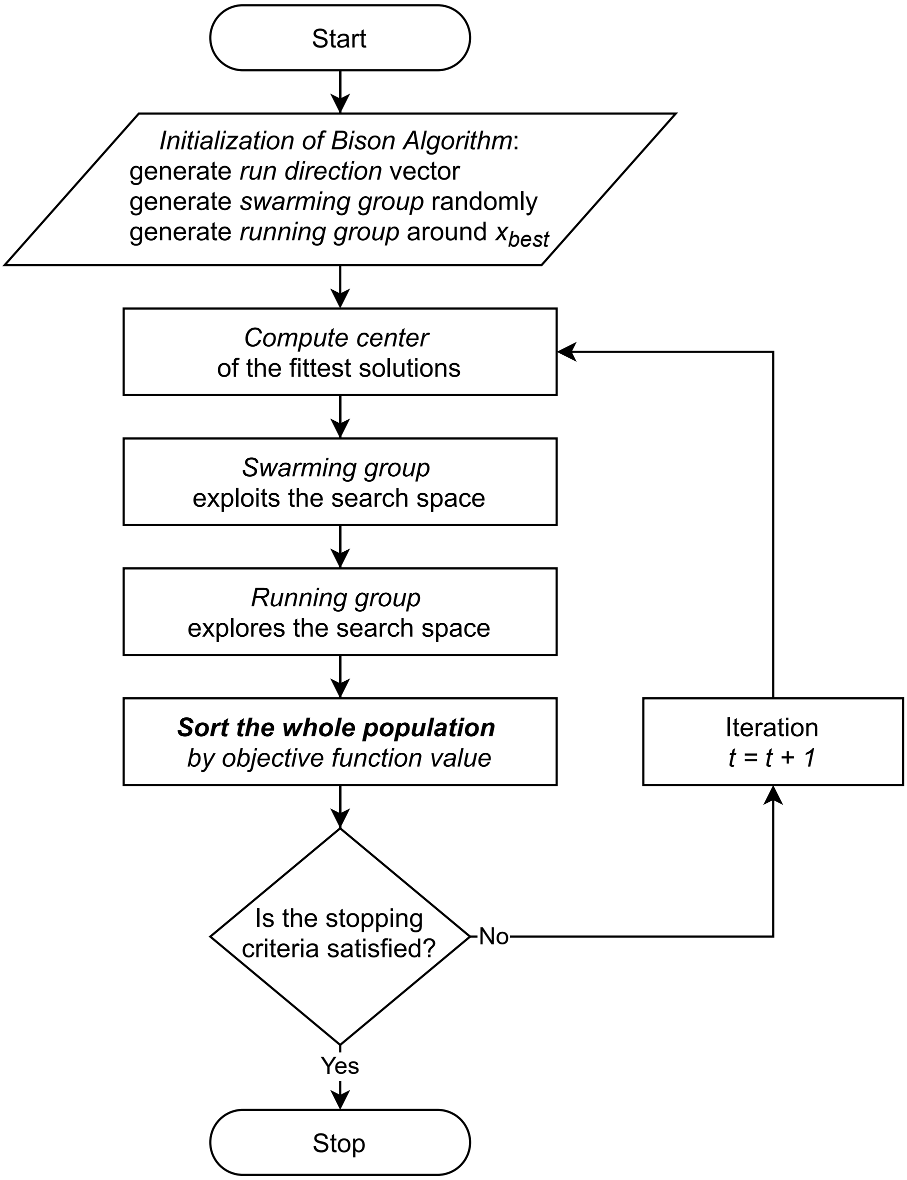


Fig. 6.3 Flowchart of the original Bison Algorithm proposal.

6.2 Run Support Strategy

The Run Support Strategy modification was developed to improve the utilization of newly discovered solutions [216]. In this modification, a promising solution found by the running group replaces the target of the swarming movement for several iterations with a much smaller overstep parameter. If the running solutions do not find a better solution, the target of the swarming movement stays with the center of several fittest solutions like in the standard Bison Algorithm. The main principle is shown in Algorithm 3 and Figure 6.4.

Algorithm 3 Center Computation of the Run Support Strategy.

```
If f(x_runner) < f(x_swarmer) do:
    For next run support iterations do
        target = x_runner
        overstep = rand(0.95, 1.05)
    End for
End if
```

Where:

- $\boldsymbol{x}_{runner}$ and $\boldsymbol{x}_{swarmer}$ are the current running and the worst swarming solutions, respectively,
- $f(\boldsymbol{x})$ is the objective function value,
- *run support* is an additional parameter defining the number of iterations for the planned exploitation of the promising solution,
- $rand(from,\ to)$ is a random number in the range of the two given arguments,
- ***target*** is the target of the swarming movement, called *"**center**"* in the original Bison Algorithm,
- and *overstep* is the overstep parameter.

To illustrate the asset of the Run Support Strategy, Figure 6.5 shows the movement of the Bison Algorithm with the Run Support Strategy solving 2-dimensional Schwefel's Function with the global optimum placed at [418.982887, 418.982887]. In this case, the swarming group solutions were trapped in local optimum until

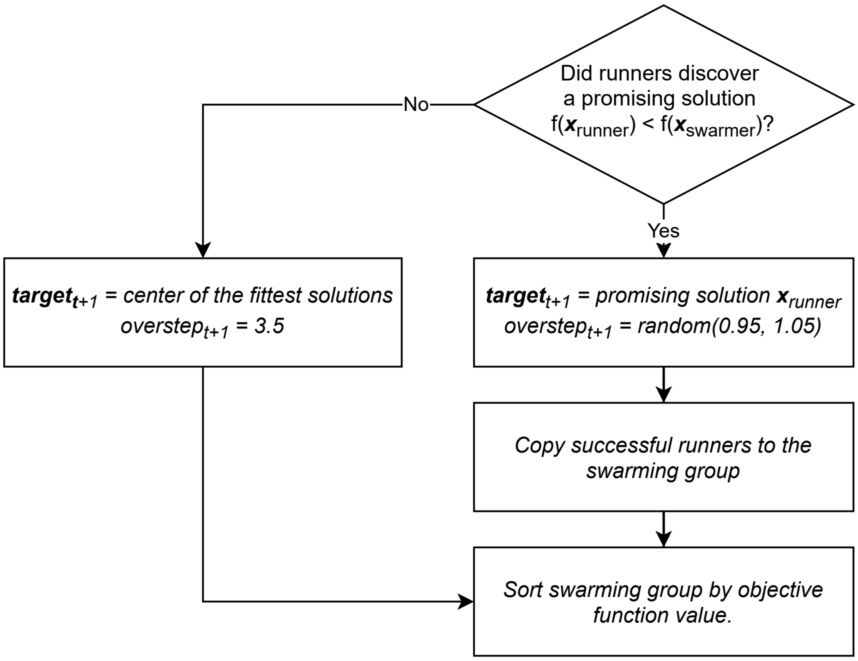


Fig. 6.4 Flowchart of the Run Support Strategy.

iteration 40 (Figure 6.5 b). However, in the next iteration, the exploring running group discovered better solutions. Replacing the target of the swarming movement (Figure 6.5 c) helped the swarming group escape the local optimum and find the global one (Figure 6.5 d).

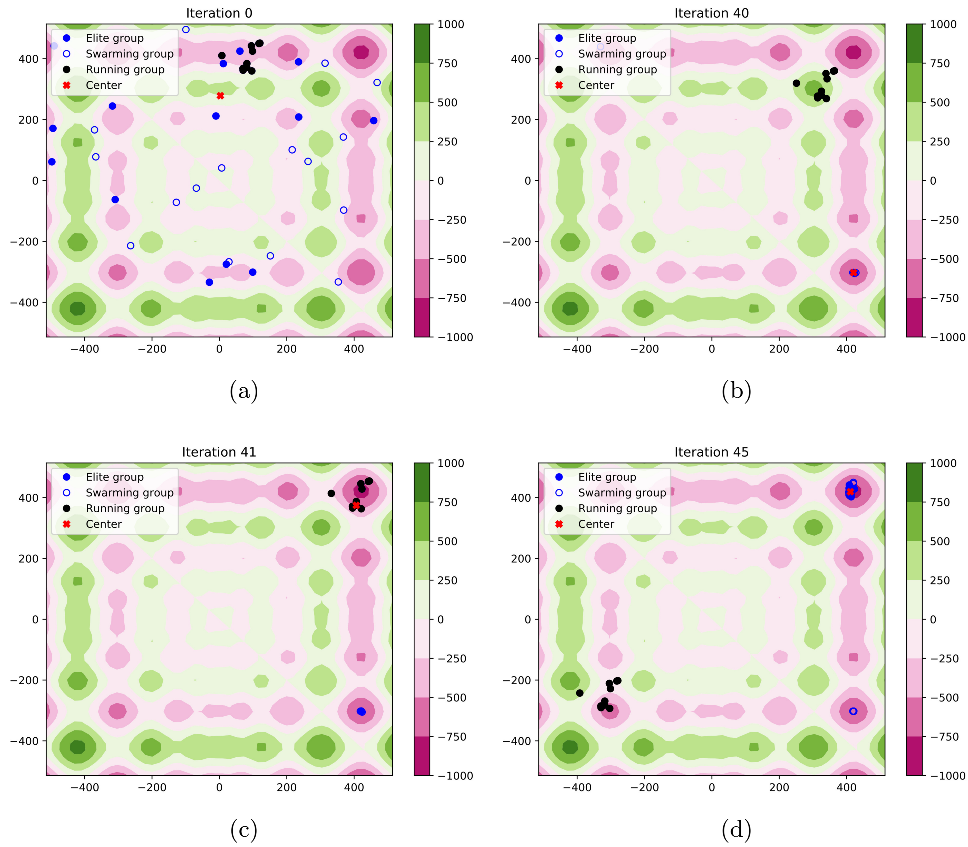


Fig. 6.5 Performance of the Bison Algorithm with the Run Support Strategy on the 2-dimensional Schwefel's Function.

6.3 Bison Seeker Algorithm

The Bison Seeker modification offers a different approach [211]. When the running group discovers a promising solution, it changes its behavior from running to seeking and exploits the promising area on its own. After one iteration, the running group continues the running procedure, in the original formation, from where it ended. The main principle is represented in Figure 6.6. The $\mathbf{x}_{runner}$ and $\mathbf{x}_{swarmer}$ represent the current running solution (explored iteratively) and the worst swarming solution.

The movement of the Bison Seeker Algorithm is represented in Figure 6.7. The running group clusters toward the promising area in Figure 6.7 b) and d) and then returns to the standard formation in Figure 6.7 a), c).

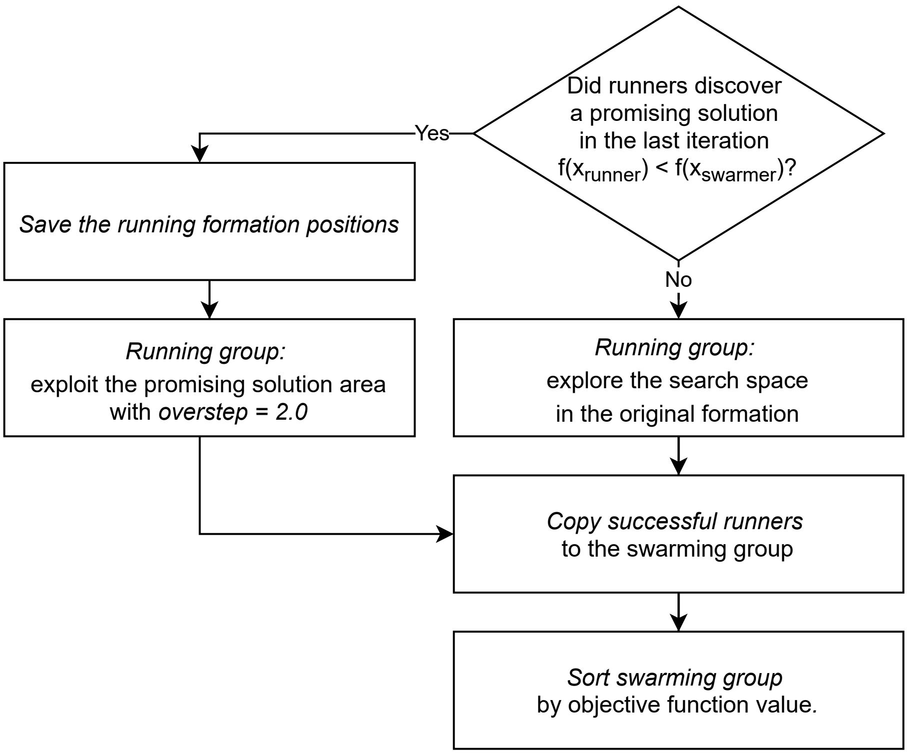


Fig. 6.6 Flowchart of the Bison Seeker modification.

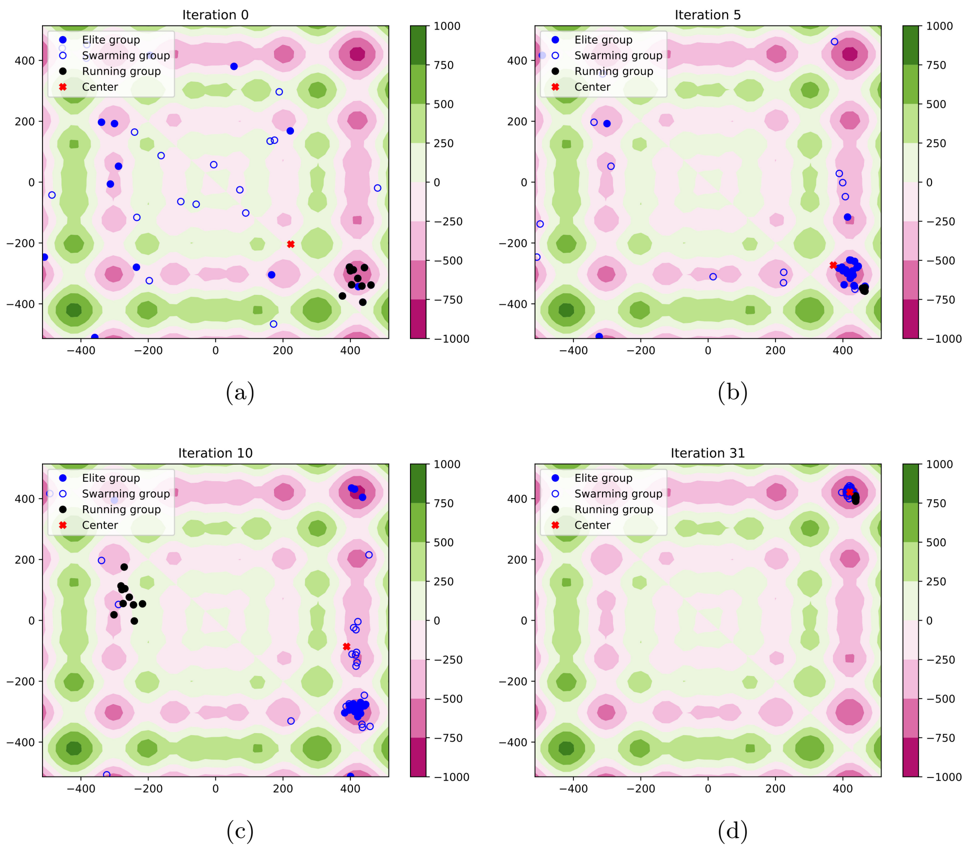


Fig. 6.7 Performance of the Bison Seeker Algorithm on the 2-dimensional Schwefel's Function.

6.4 Self-Adaptive Bison Algorithm

The self-adaptive modification employs multiple parallel populations of various parameter settings. The "core population" represents a standard population, with the initial configuration based on the parameter tuning experiment [210]. Additional populations modify a selected parameter: one after another, the populations raise or lower the swarming group SG parameter, the elite group EG parameter, and the $overstep$ parameter.

After each iteration, the configuration with the most successful solution (further referenced as the population S) shares the best result with the core population, which adopts its corresponding characteristic parameter. The population S then raises (or lowers, if it is the lower population type) its characteristic parameter to avoid having two populations of the same configuration. The adjustment step is 1 for SG and EG and 0.01 for the $overstep$ parameter.

The main idea of the Self-Adaptive Bison Algorithm is presented in Figure 6.9 and Figure 6.10. Figure 6.8 shows the parameter setting configurations of the initial populations. The self-adaptive modification opens a new way of understanding the effect of the parameter configuration on the examined landscape. The code of the Self-Adaptive Bison Algorithm is available at Tomas Bata University Artificial Intelligence Laboratory's GitHub repository[1].

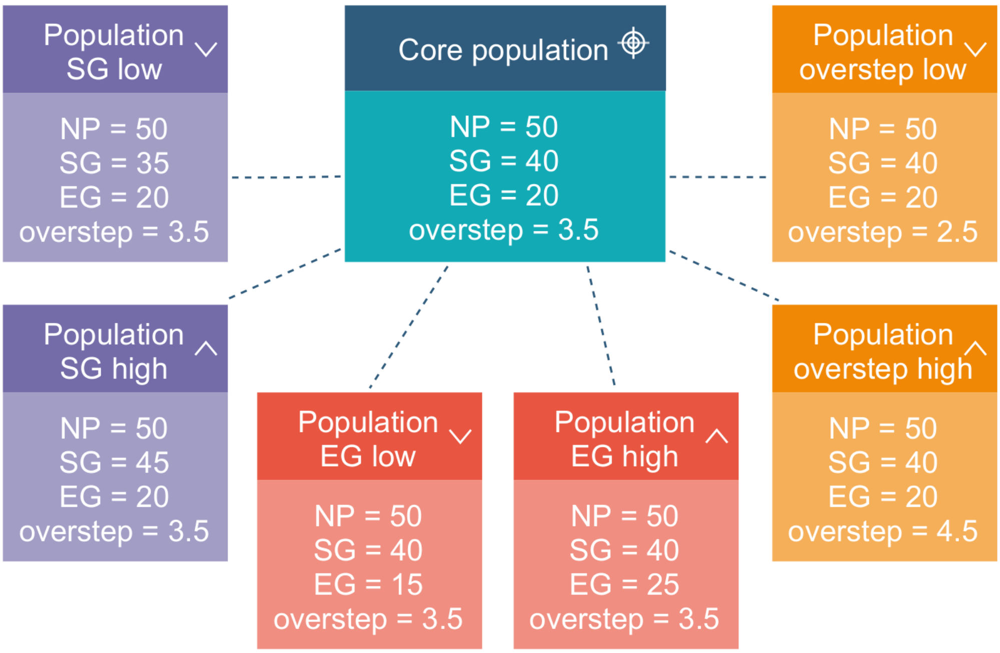


Fig. 6.8 Initial parameter settings of all the sub-populations in the Self-Adaptive Bison Algorithm.

[1] `https://github.com/TBU-AILab/Bison-Algorithm-OOP`, accessed 02/06/2022

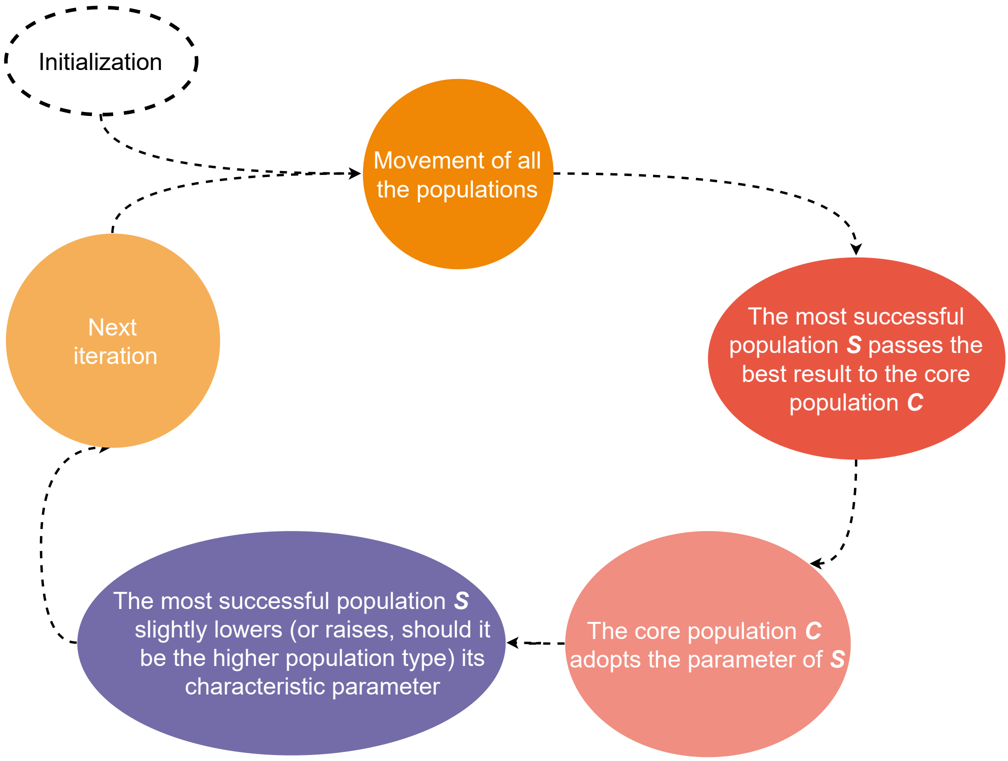


Fig. 6.9 Simplified principle of the Self-Adaptive Bison Algorithm.

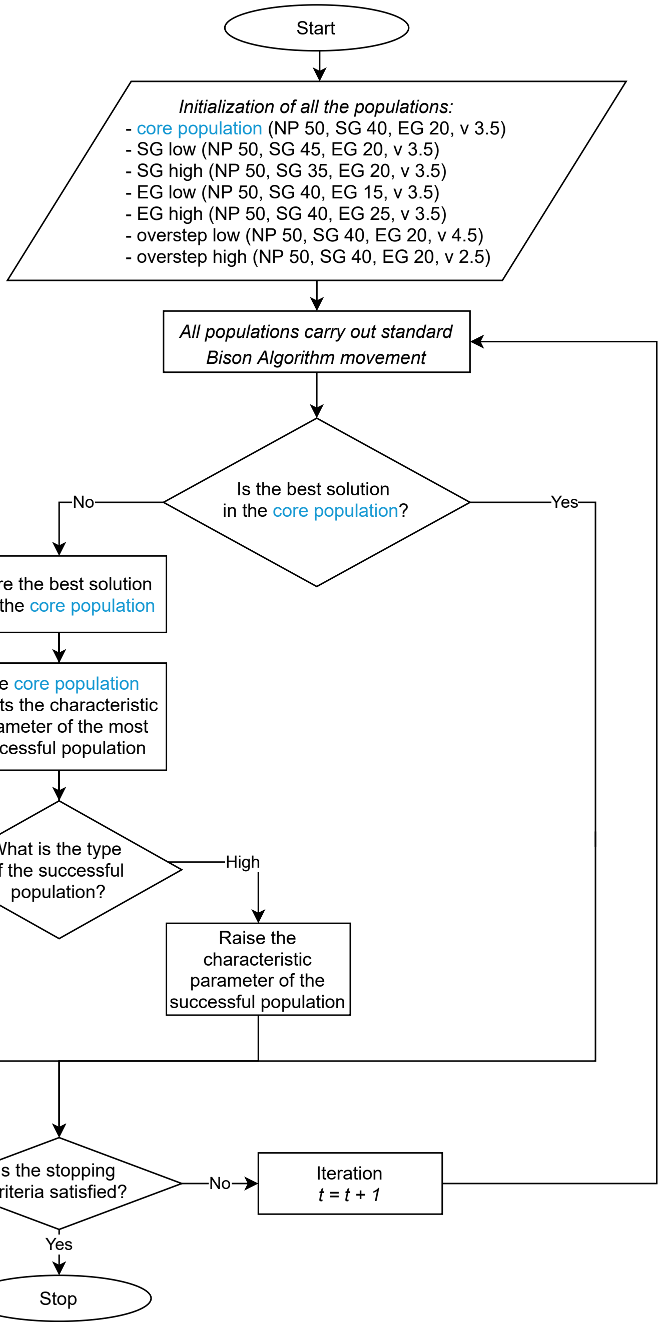


Fig. 6.10 Flowchart of the Self-Adaptive Bison Algorithm.

6.5 Bison Algorithm Modifications Vs. the Local Containment Problem

The problem of local optimum containment was the main motivation for the development of the Bison Algorithm. To establish how the algorithm actually deals with local optimum containment, this section analyses the behavior of the Bison Algorithm, and all its modifications, when facing the local containment challenge.

Figures 6.12–6.15 show the movement of the solutions, given the same initial conditions: a predefined initial population (illustrated in Figure 6.12 a) and the same starting run direction vector of the running group. The experiments solved the De Jong Function N.5 (Figure 6.11, Eq. 6.1) in 2 dimensions, with 24 local optima and the global one in [-32, -32] (the bottom left optimum in Figures 6.12–6.15). Following figures show how the particular Bison Algorithm modifications cover local optimum containment.

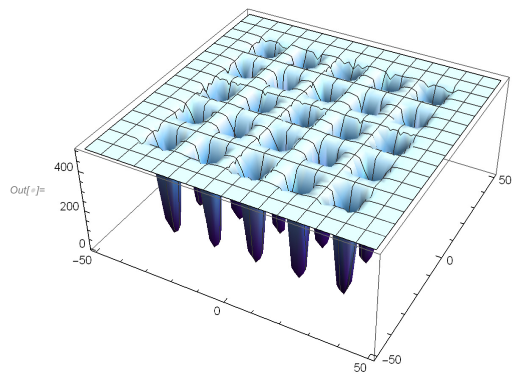


Fig. 6.11 De Jong Function Number 5.

$$f(\mathbf{x}) = (0.002 + \sum_{i=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6})^{-1}, \text{ where} \tag{6.1}$$

$$\mathbf{a} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix} \tag{6.2}$$

6.5.1 Original Bison Algorithm

Figure 6.12 models the behavior of the solutions in the original proposal of the Bison Algorithm [214]. The figure shows:

a) the initial distribution of the solutions common for all the subsequent scenarios,

b) the scattering of the running solutions,

c) their shift,

d) and vanishing of the bottom left running solution (around [-32, -32]), which transformed into the swarming solution of a better objective function value.

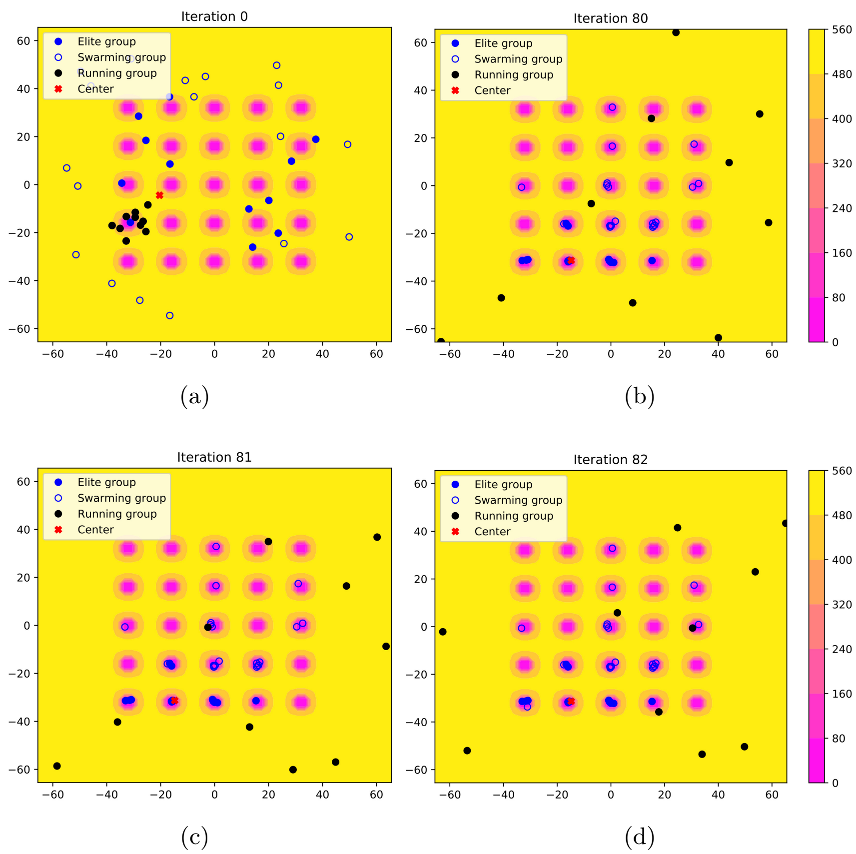
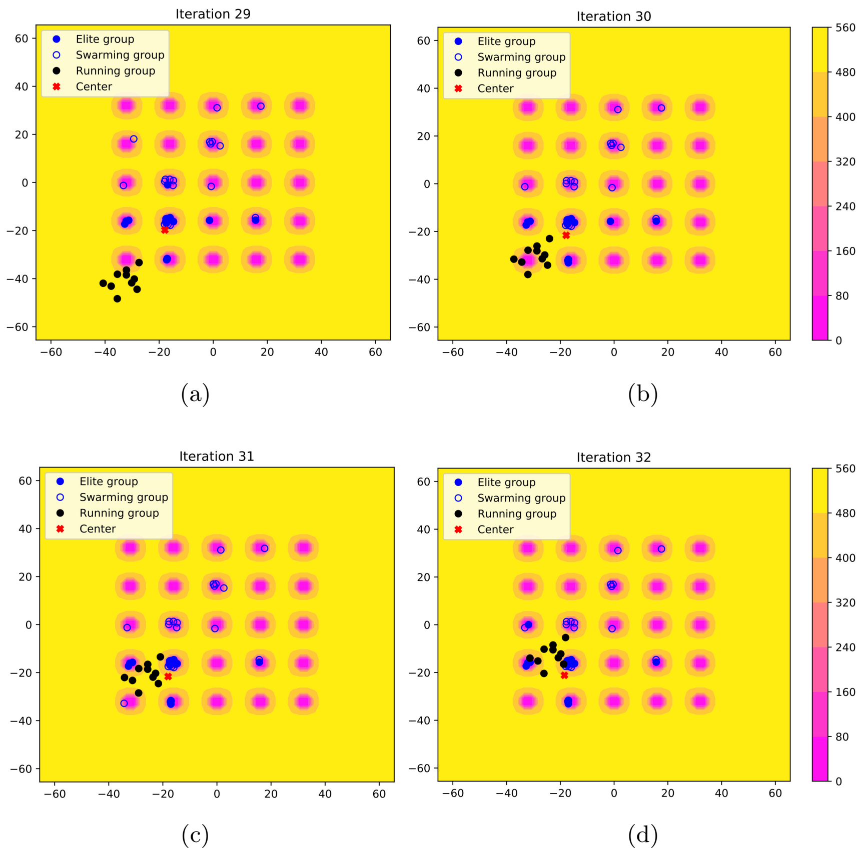


Fig. 6.12 Behaviour of the original Bison Algorithm when exposed to the local optimum containment problem.

6.5.2 Standard Bison Algorithm (with Coherent Running Group)

Figure 6.13 illustrates how the standard Bison Algorithm from Section 5 dealt with the local containment situation. This modification kept the running group together throughout the search process to exploit the benefits of a systematic group search. However, despite running through the global optimum area, the effect of the exploration group was minimal in Figure 6.13 c) and vanished in Figure 6.13 d).



(a) (b)

(c) (d)

Fig. 6.13 Behaviour of the standard Bison Algorithm when exposed to the local optimum containment problem.

6.5.3 Bison Algorithm with Run Support Strategy

To utilize the solutions better, the Run Support Strategy modification switches the target of swarming movement for the newly discovered solution, should the running solutions find something interesting. Figure 6.14 shows a neat example of a local optimum breakout with the help of the exploration group. When compared to the standard Bison Algorithm in Figure 6.13, the Run Support Strategy quite effectively shifted the population from three local optima areas to the newly discovered solution in Figure 6.14 d), and the global optimum in e) and f).

6.5.4 Bison Seeker Algorithm

Figure 6.15 illustrates the Bison Seeker Algorithm. This algorithm employs two behavior patterns of the running group. The running group explores the feasible solutions in Figure 6.15 a) and d). However, when the running group finds a promising solution, it transforms the behavior to exploit the promising area on its own. The difference between the Run Support Strategy and the Bison Seeker mechanism is in the acting group which examines the promising area. While Run Support Strategy sends the swarming group, the Bison Seeker Algorithm employs the running solutions. The seeking behavior can be seen in Figure 6.15 b) and c).

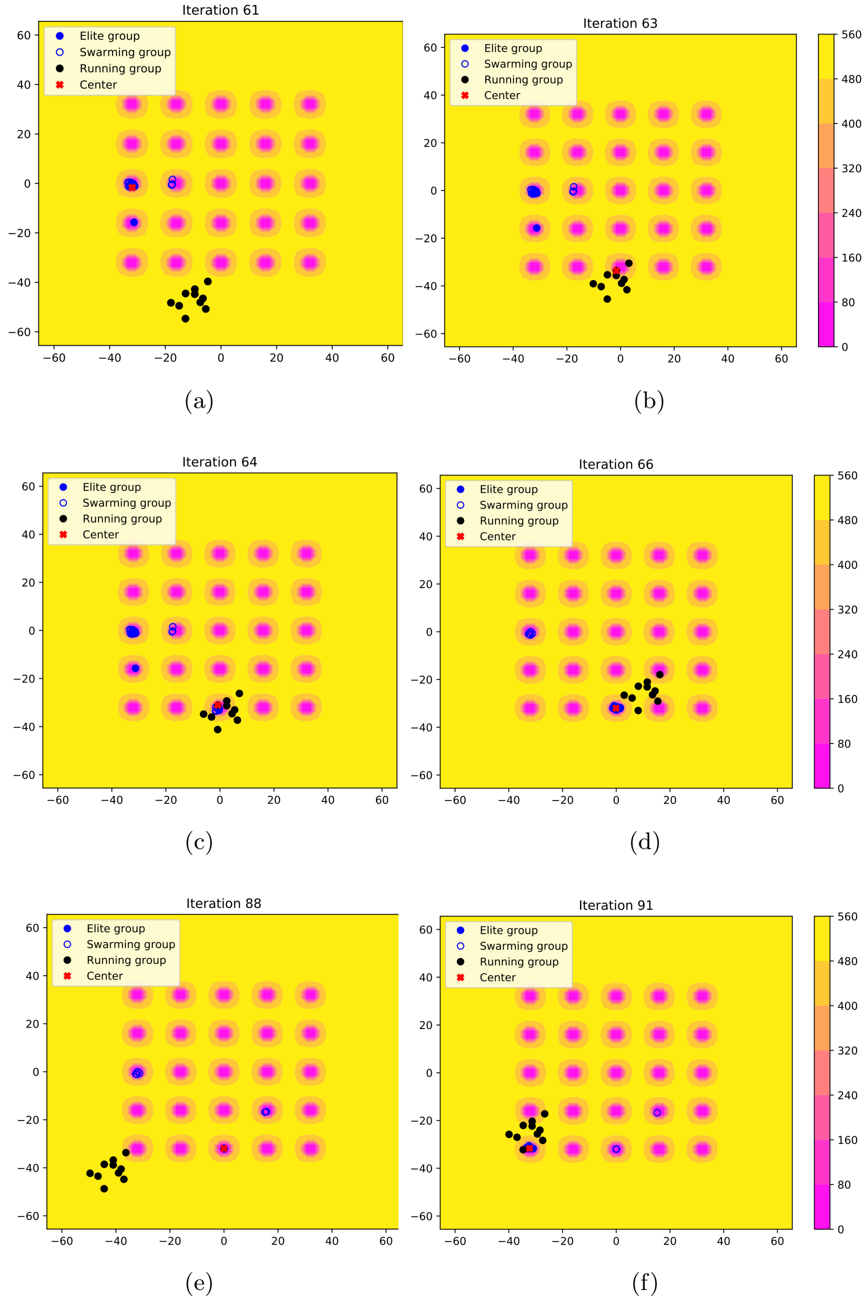


Fig. 6.14 Behaviour of the Bison Algorithm with the Run Support Strategy when exposed to the local optimum containment problem.

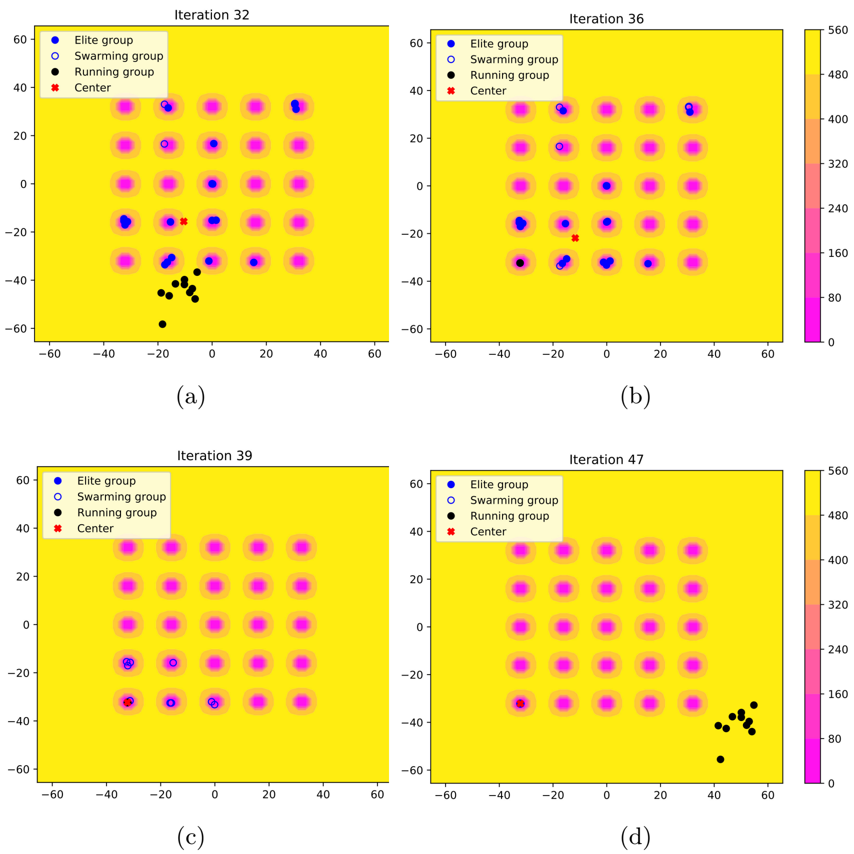


Fig. 6.15 Behaviour of the Bison Seeker Algorithm when exposed to the local optimum containment problem.

7 Performance of the Bison Algorithm

Following the guidelines in Section 3.3, the first step of performance evaluation should name the goal and the motivation of the experiment. Rather than focusing on superior performance, the goal of the proposed algorithm was to contribute to methodology oriented towards the particular problem with a potentially novel approach.

Nonetheless, it is essential to examine the optimization capabilities of a proposed algorithm. For this purpose, four popular optimizers out of the top 10 swarm algorithms were chosen. They were implemented from the EvoloPy optimization library [129], enabling the same template to be used for most functions of the algorithms. The parameter tuning process preceded the experiment, to provide the best parameter configuration setting for all the tested algorithms. The performance analysis criteria included: error values, convergence trends, change in population diversity, and computation complexity.

The ultimate goal of the experiment was not to prove the Bison Algorithm's superiority to the state-of-the-art optimizers. However, since the algorithms being compared were examined on the IEEE CEC benchmark test sets, the last Section 7.1.6 compares the Bison Algorithm with the winners of the competitions. In addition, since the original motivation of the Bison Algorithm was to design an algorithm with an escape mechanism from local optimum containment, the previous Section 6.5 analysed how the algorithm deals with the local optimum containment challenge in practice.

7.1 Comparison with Other Metaheuristics

The Bison Algorithm was compared to other swarm algorithms on the benchmark sets of IEEE CEC 2015 and IEEE CEC 2017 in 10, 30, 50, and 100 dimensions. Following the recommendations for results evaluation [25], each experiment took 51 independent runs, each consisting of $10,000 \times dimensions$ evaluations of the objective function.

The selected algorithms were: the Bison Algorithm (BIA), the Cuckoo Search (CS), Particle Swarm Optimization (PSO), the Bat Algorithm (BAT), and the Firefly Algorithm (FFA). The control parameters were tuned for each algorithm separately in Section 7.1.1. The implementations were derived from the EvoloPy optimization library [129] and from [138] and modified to fit the same template as the Bison Algorithm.

The statistical tests in the following sections include the Wilcoxon Rank-Sum test and the Friedman Rank test, both with the significance level set to 0.05. The Wilcoxon Rank-Sum tests (in Tables 7.2, 7.3, and 7.9) enumerate how many times a particular algorithm significantly outperformed all the other algorithms on a specific optimization problem. On the other hand, the Friedman Rank tests (Figures 7.3, 7.7, 7.6, 7.4) rank all the algorithms across all the solved problems and calculate the Nemenyi critical distance: all the optimizers ranked over the distance are of significantly worse performance than the first-ranked algorithm.

The complete results showing the objective function values of the mean solution and standard deviations of all the tested algorithms are presented in Appendix D. Standard statistic output of the Bison Algorithm showing the minimal, maximal, median, and mean solution qualities and standard deviations is shown in Appendix E.

7.1.1 Parameter Tuning

The parameter tuning experiment compared 20 parameter configurations from 11 literature sources [35, 129, 156, 210, 246, 271, 421, 434, 435, 436, 438]. Noticeably, in most cases, the selected configurations have already gone through the parameter tuning selection and recommended these parameter settings.

Each algorithm optimized the IEEE CEC 2017 benchmark test set with various parameter settings in 10 and 30 dimensions. The errors were tested for statistical significance (Wilcoxon Rank-Sum test and Friedman Rank test, $p<0.05$ both), and the following configurations were evaluated as the best.

The complete experiment was carried out as a part of the discussion about the importance of proper parameter configuration for fair comparison [217]. Interestingly, the paper showcased three comparison examples in which inappropriate parameter selection utterly shuffled the interpretation of the results.

Tab. 7.1 Control parameters used in experiments.

BIA		**CS**		**PSO**		**BAT**		**FFA**	
NP	50	NP	20	NP	50	NP	50	NP	50
Swarm group size	40	P_a	0.25	v_{max}	6	Loudness	1.5	α	0.5
Elite group size	20	α	0.01	W_{max}	0.9	Pulse rate	0.5	β	0.2
Overstep	3.5			W_{min}	0.2	Q_{min}	0	λ	1.0
Run support	0			C_1	2	Q_{max}	2		
				C_2	2				

7.1.2 Error Values Analysis

The most common method of performance analysis is to investigate the error values of the final solutions. This section examines them from several perspectives. The first set of tests analyzed the benchmarking testbed as a whole. Figures 7.1 and 7.2 depict the mean solution error of the tested algorithms on 45 IEEE CEC 2015 and 2017 benchmarking problems. Table 7.2 and Table 7.3 summarize the Wilcoxon Rank-Sum test results ($p<0.05$). The tests count the number of problems where one algorithm performed significantly better than all the remaining algorithms. Finally, the problems were compared with the Friedman Rank test in Figure 7.3. This test ranks the algorithms by another performance measure and sets the significance threshold as the Nemenyi critical distance; the algorithms over the distance performed significantly worse than the first-ranked algorithm. The Friedman Rank test is valid when the P-Value is lower than 0.05, as shown in Table 7.4.

After the performance analysis on the complete benchmark sets, the algorithms were compared with respect to the character of the solved problems. Since some problems from the IEEE CEC 2015 and 2017 test suites might be identical, examining both testbeds could bias the results. The class-oriented tests were, hence, performed solely on the IEEE CEC 2017.

Based on the benchmark definitions [25], the problems from CEC 2017 benchmark test set were classified into six classes: all problems, unimodal problems, multimodal problems, asymmetrical problems, problems with a huge amount of local optima, and problems with the second-best solution being far from the global best solution. The results were then examined with the Friedman Rank test (Figure 7.4) and the Wilcoxon Rank-Sum test (Figure 7.5) on the corresponding test sets across all dimensions. Table 7.5 defines the problem classes and the P-Values of the Friedman Rank test. Figure 7.5 shows the percentage of the problems in the examined class, where the algorithm delivered the best results (Wilcoxon, $p<0.05$).

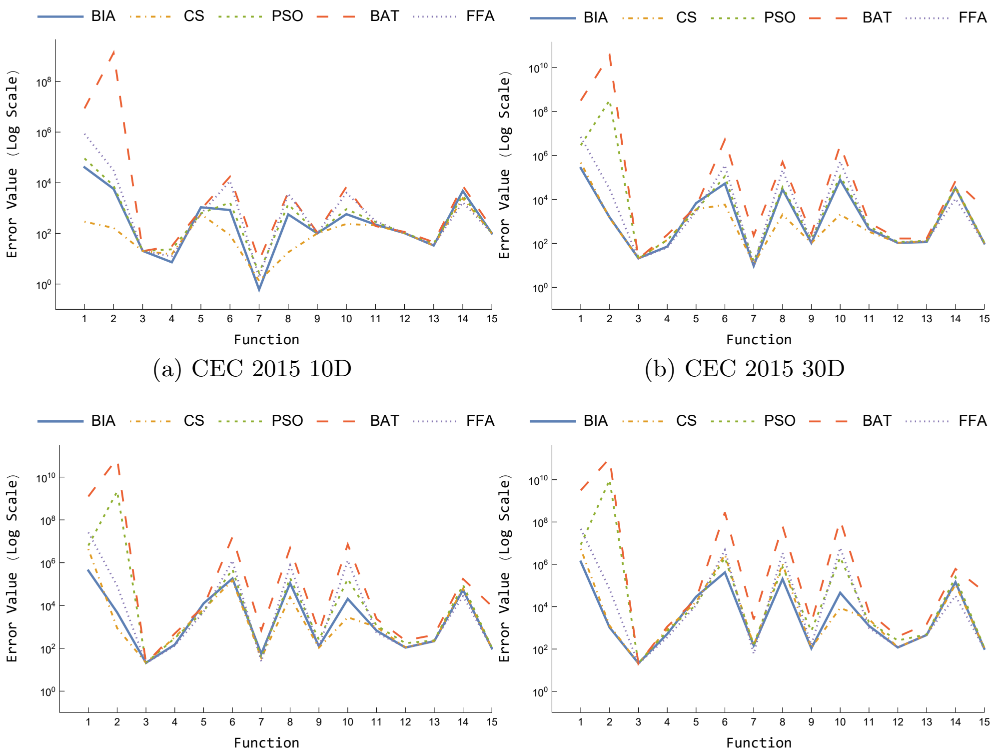
Analysis of the Whole Set of Benchmarking Problems IEEE CEC 2015 and CEC 2017

Tab. 7.2 Winning Algorithms on CEC 2015 (Wilcoxon Rank-Sum test, $p<0.05$).

	None	BIA	CS	PSO	BAT	FFA
10 dimensions	6	3	**5**	0	1	0
30 dimensions	4	**5**	3	0	1	2
50 dimensions	5	**4**	3	0	1	2
100 dimensions	4	**5**	0	0	1	5
Sum of wins	19	**17**	11	0	4	9

Tab. 7.3 Winning Algorithms on CEC 2017 (Wilcoxon Rank-Sum test, $p<0.05$).

	None	BIA	CS	PSO	BAT	FFA
10 dimensions	7	7	**13**	1	0	2
30 dimensions	3	**14**	6	1	0	6
50 dimensions	7	**10**	5	1	0	7
100 dimensions	6	**11**	2	1	0	10
Sum of wins	23	**42**	26	4	0	25



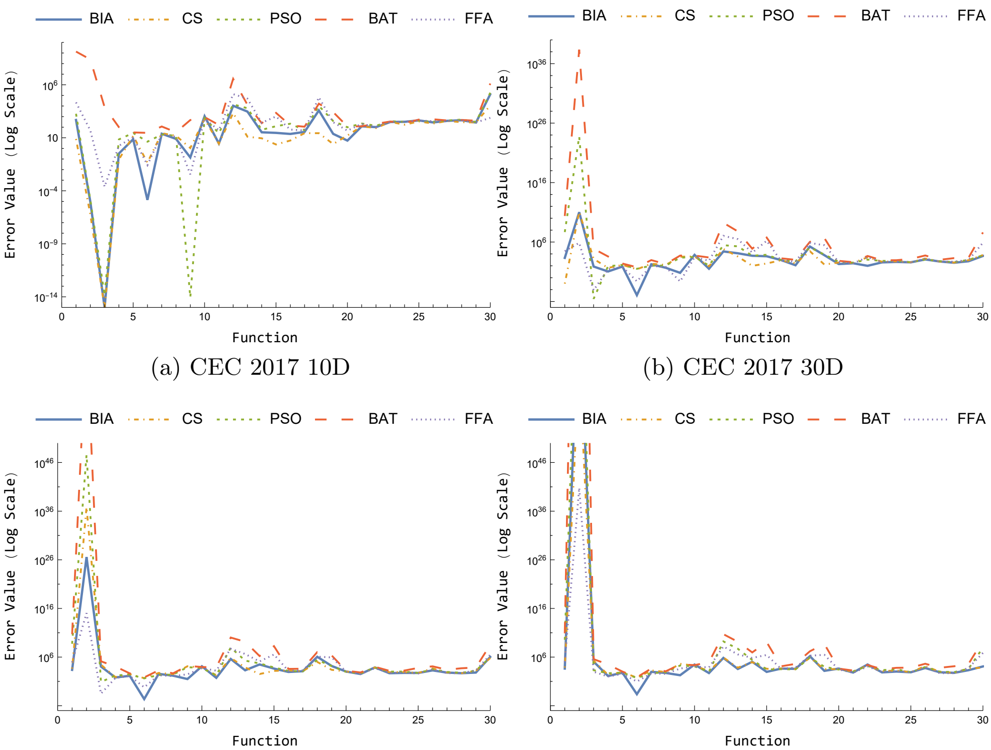
(a) CEC 2015 10D (b) CEC 2015 30D

(c) CEC 2015 50D (d) CEC 2015 100D

Fig. 7.1 Mean solution comparison of algorithms tested on the IEEE CEC 2015 test set.

Tab. 7.4 Friedman Rank test P-Values (significant, if p<0.05).

	10 D	30 D	50 D	100 D
CEC 2015	4.96E-05	2.79E-07	5.94E-07	2.42E-06
CEC 2017	2.67E-17	2.18E-20	9.41E-19	3.38E-16



(a) CEC 2017 10D

(b) CEC 2017 30D

(c) CEC 2017 50D

(d) CEC 2017 100D

Fig. 7.2 Mean solution comparison of algorithms tested on the IEEE CEC 2017 test set.

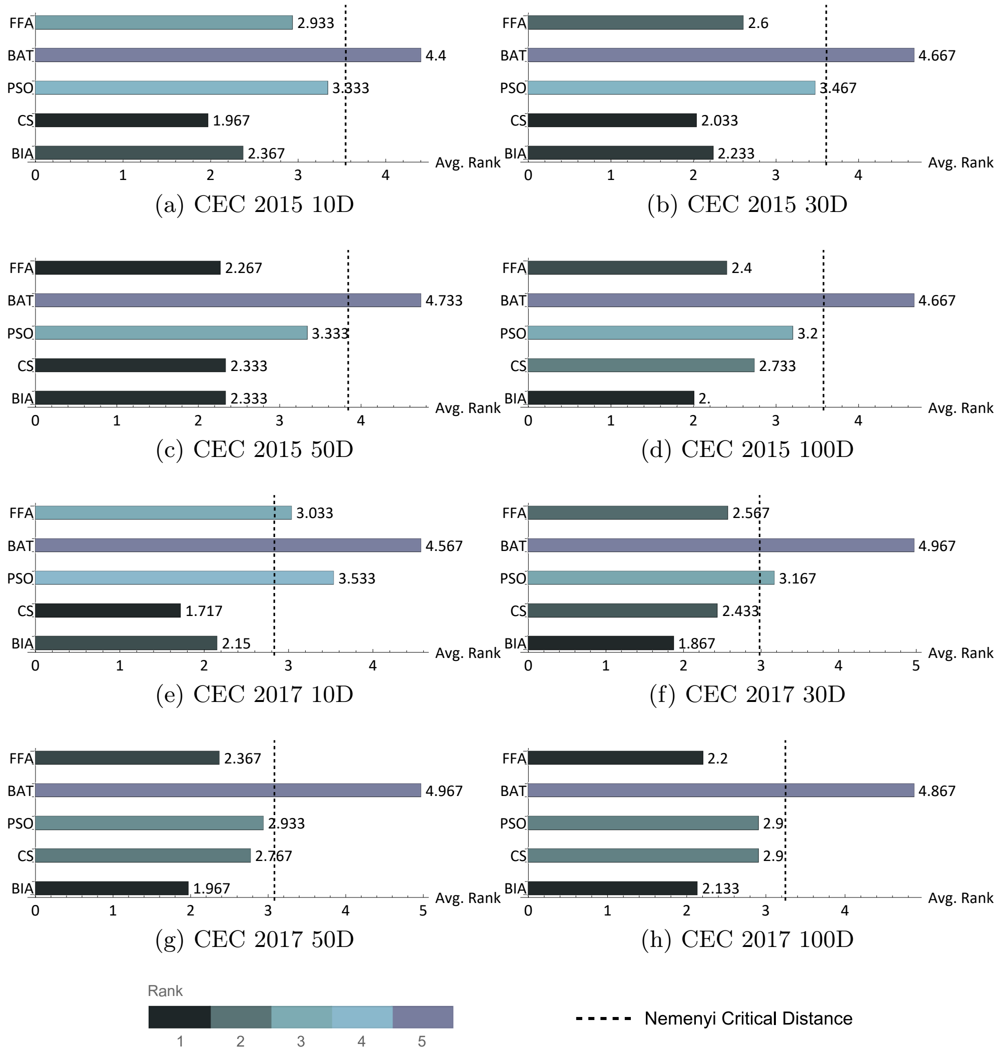
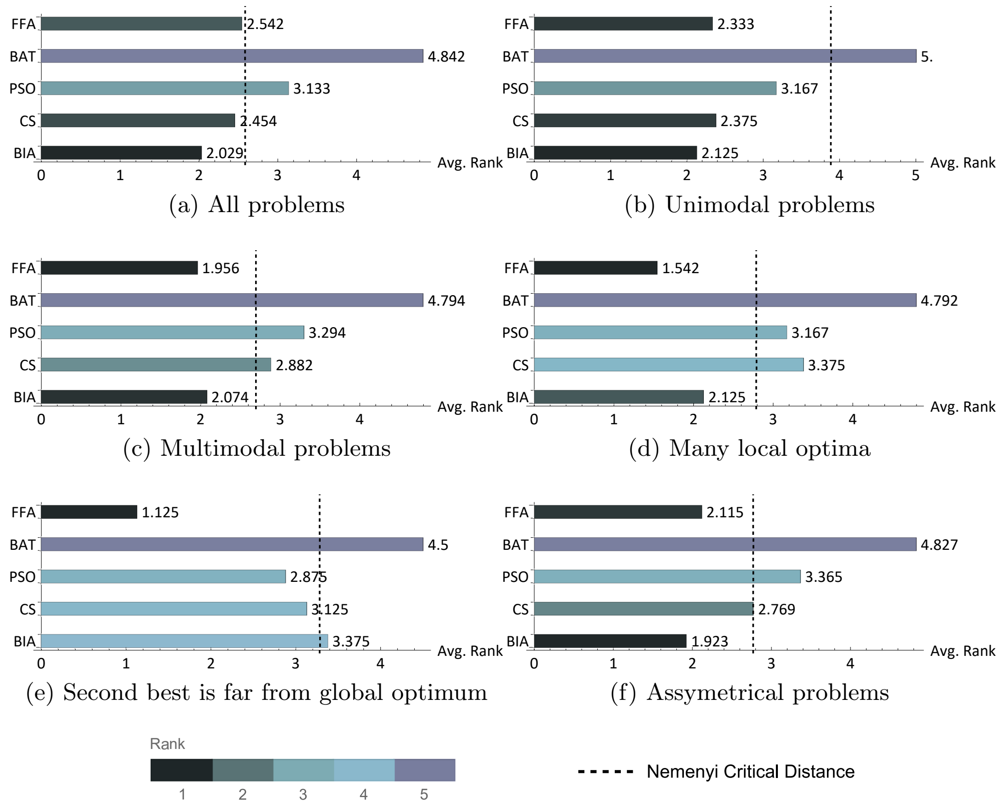


Fig. 7.3 Rank comparison of the BIA, CS, PSO, BAT, and FFA on benchmarks CEC 2015 and CEC 2017 (Friedman Rank Test, p<0.05).

Performance Analysis Based by the Problem Classes

Tab. 7.5 Friedman Rank Test P-Values ($p<0.05$) in CEC 2017 problem selection across all dimensions.

Problem feature	Problems	P-Value
All problems	F1-F30	1.72E-67
Unimodal problems	F1-F3	1.48E-07
Multimodal problems	F4-F10, F21-F30	2.06E-42
Many local optima	F4-F6, F8-F10	5.86E-19
Second best is far from global optimum	F5, F10	3.11E-05
Assymetrical problems	F6-8, F21-30	5.11E-34



(a) All problems

(b) Unimodal problems

(c) Multimodal problems

(d) Many local optima

(e) Second best is far from global optimum

(f) Assymetrical problems

Fig. 7.4 Performance measure featuring the ranks from the Friedman Rank test ($p<0.05$) based on the character of solved problems.

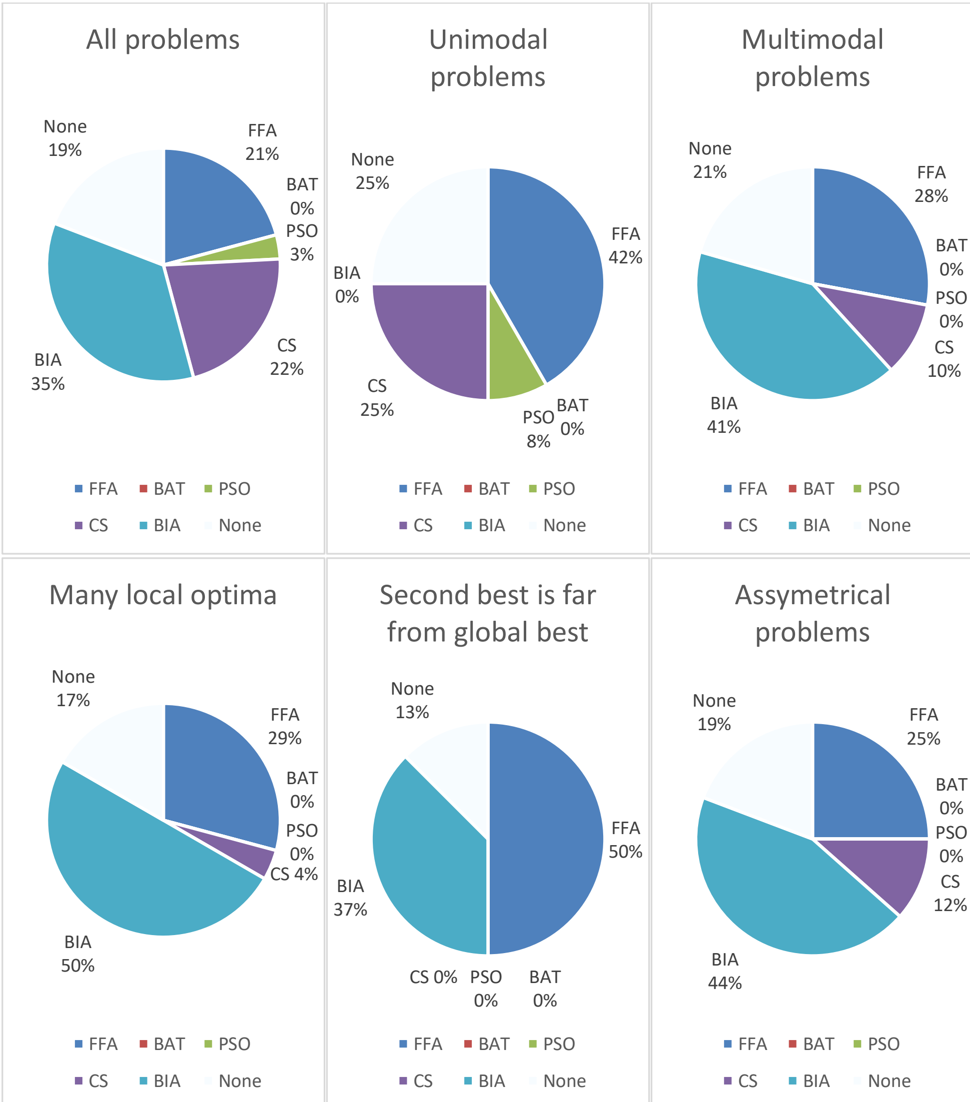


Fig. 7.5 Performance measure comparing the number of problems with a significantly better solution on the set with characteristic feature (Wilcoxon $p<0.05$).

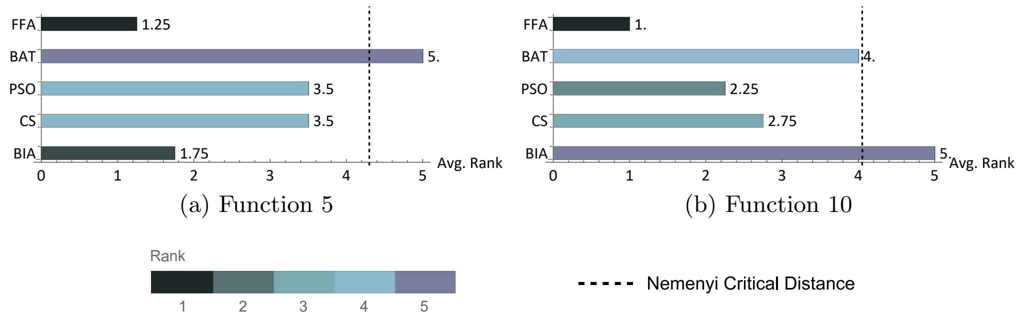
Discussion of Overall Results vs. Problem Class-Based Results

Based on the mean solution-oriented experiments (Figures 7.1 and 7.2), the Bison Algorithm excelled particularly in $F6$ and $F9$ of CEC 2017 over all of the tested dimensions and was generally quite successful when solving CEC 2015, though occasionally outperformed by the Cuckoo Search algorithm. According to the Wilcoxon Rank-Sum test, the Bison Algorithm had the best or second-best results and the highest rate sum of significant wins across all the tested dimensions when solving CEC 2015 and CEC 2017 problems (Table 7.2 and Table 7.3).

The Friedman Rank test ranked the Bison Algorithm (4x), Cuckoo Search (3x), and the Firefly Algorithm (1x) among the best ranks. The Bat Algorithm was, on the other hand, outperformed in all of the tested scenarios. Comparing the complete test set results with the problem class-based results yielded an interesting twist featuring the Firefly Algorithm in half of the problem classes and the Bison Algorithm in the rest. The Wilcoxon Rank-Sum test in Figure 7.5 showed the percentage of algorithm's wins on the individual problem classes.

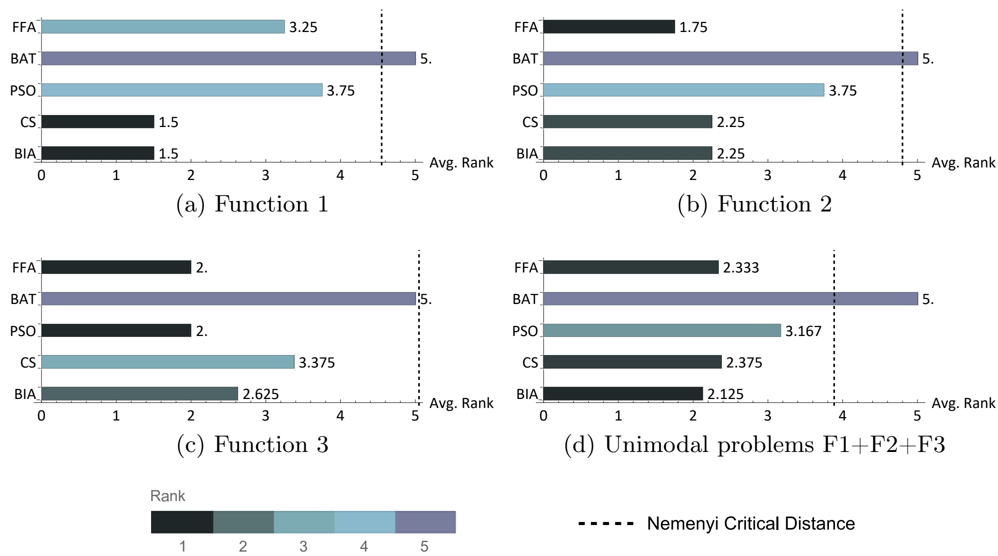
Consistently with both statistical tests (Figure 7.4 and Figure 7.5), the Bison Algorithm delivered the best results when solving asymmetrical problems and the complete IEEE CEC 2017 test set. The Firefly and Bison Algorithms switched the first and second ranks on multimodal and multiple local optima problems. However, there was a surprising imbalance between the Friedman Rank test and the Wilcoxon Rank-Sum tests comparing the problem class with the second-best solution far from the global one.

The Friedman Rank tests might explain the discrepancy in the problems individually in Figure 7.6. Testing Function 5 and Function 10 separately showed that while the Bison Algorithm ranked well on the first problem, it performed the worst on the other. Therefore, the overall Friedman Rank test in Figure 7.4 e), ended over the Nemenyi's critical distance, despite the success of the Wilcoxon Rank-Sum test.



(a) Function 5 (b) Function 10

Fig. 7.6 Friedman Rank test on the individual problem F5 and F10 ($p<0.05$).



(a) Function 1 (b) Function 2

(c) Function 3 (d) Unimodal problems F1+F2+F3

Fig. 7.7 Friedman Rank test ($p<0.05$) testing F1, F2, F3 separately, and F1+F2+F3.

Similarly, the unimodal problem class favored the Bison Algorithm with the Friedman Rank test despite no win in the Wilcoxon Rank-Sum test (compare Figure 7.4 b) and Figure 7.5 b)). The final rank of the Friedman Rank test is explained in Figure 7.7. Individual tests revealed that the Bison Algorithm never reached the first rank when solving an individual problem, yet ended in the final first place due to the inconsistency of other algorithms. In other words, the Bison Algorithm's consistent second performance resulted in the overall first rank.

7.1.3 Convergence Analysis

The convergences of the algorithms are presented in Figure 7.8 and Figure 7.9. Compared to other methods, the Bison Algorithm was able to regain convergence towards better solutions even after a period of population quality stagnation, as shown in Figure 7.8. This figure represents the convergence of all 51 runs optimizing Function 4 from the CEC 2017 benchmark. The Bison Algorithm offered a higher rate of sudden drops than the other algorithms. However, since the convergence data are highly problem-dependent, Figure 7.9 showed the mean convergences of all 15 functions in the CEC 2015 benchmark in 100 dimensions.

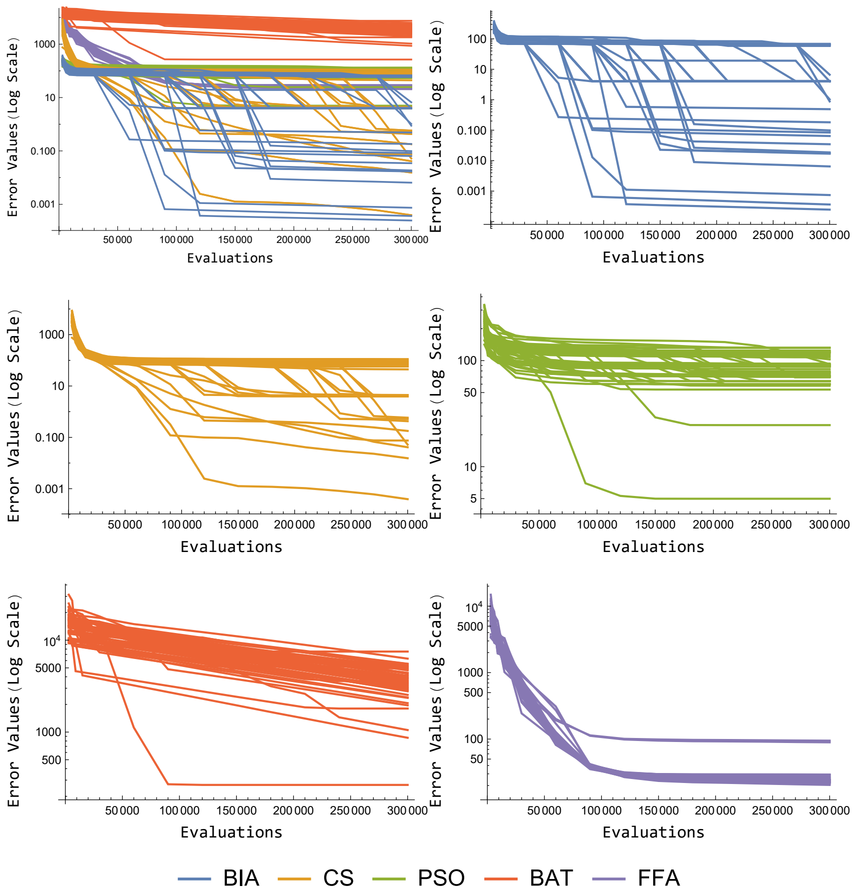
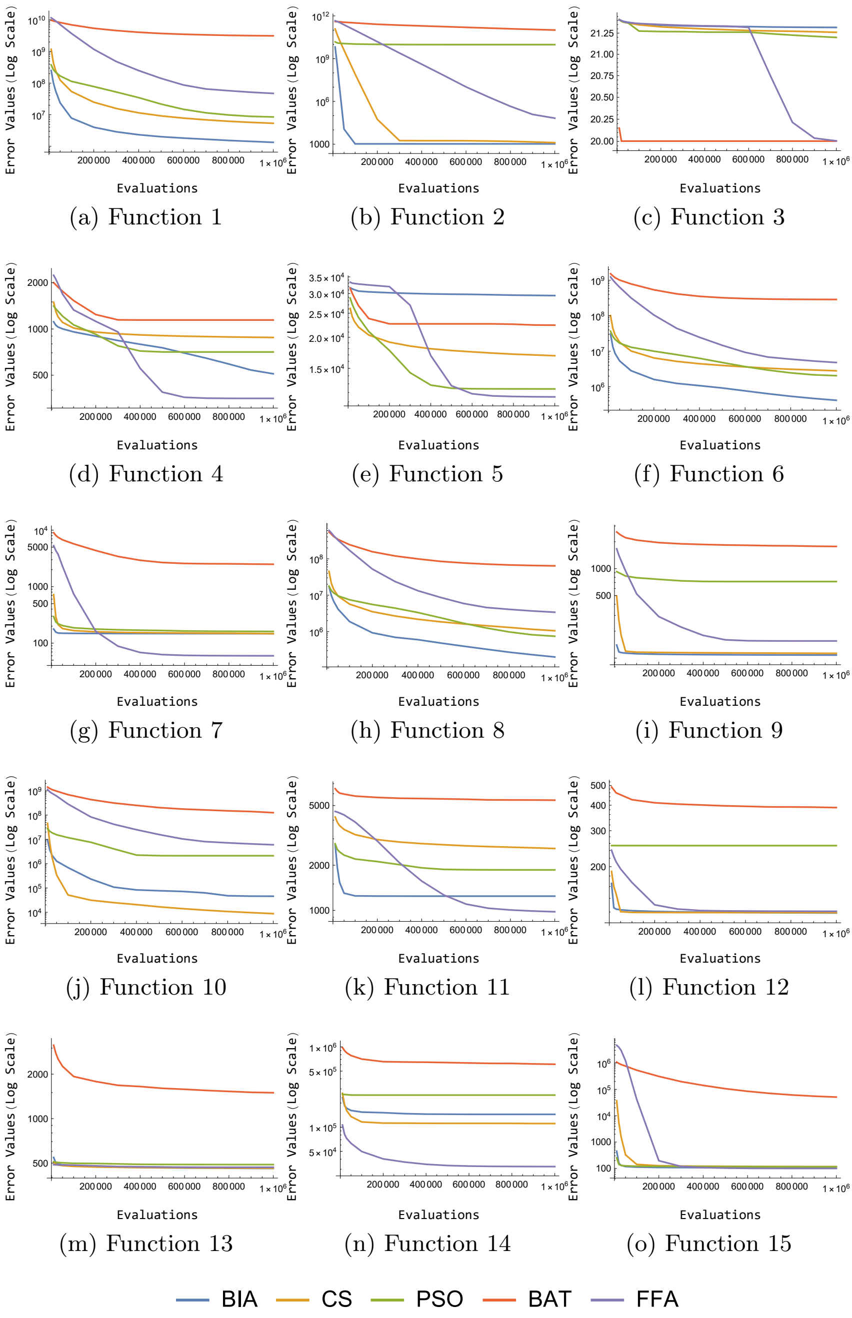


Fig. 7.8 Convergence of all runs of the swarm algorithms compared on IEEE CEC 2017 Function F4 in 30 dimensions.



(a) Function 1 (b) Function 2 (c) Function 3
(d) Function 4 (e) Function 5 (f) Function 6
(g) Function 7 (h) Function 8 (i) Function 9
(j) Function 10 (k) Function 11 (l) Function 12
(m) Function 13 (n) Function 14 (o) Function 15

Fig. 7.9 Mean convergences of benchmark test set IEEE CEC 2015 in 100 dimensions.

7.1.4 Population Diversity Analysis

The loss of diversity poses a significant optimization threat. This section studies the change in diversity of the algorithms examined throughout the optimization process. The diversity computation followed a metric by [316] in Equations 7.1 , 7.2. Tables 7.6 and 7.7 show the mean and median population diversities of the final populations from all the tested functions. Figure 7.10 represents the course of the mean diversities in the optimization process on the 100-dimensional set of CEC 2015 problems. The data are presented as a percentage relative to the theoretical maximum of the diversity value.

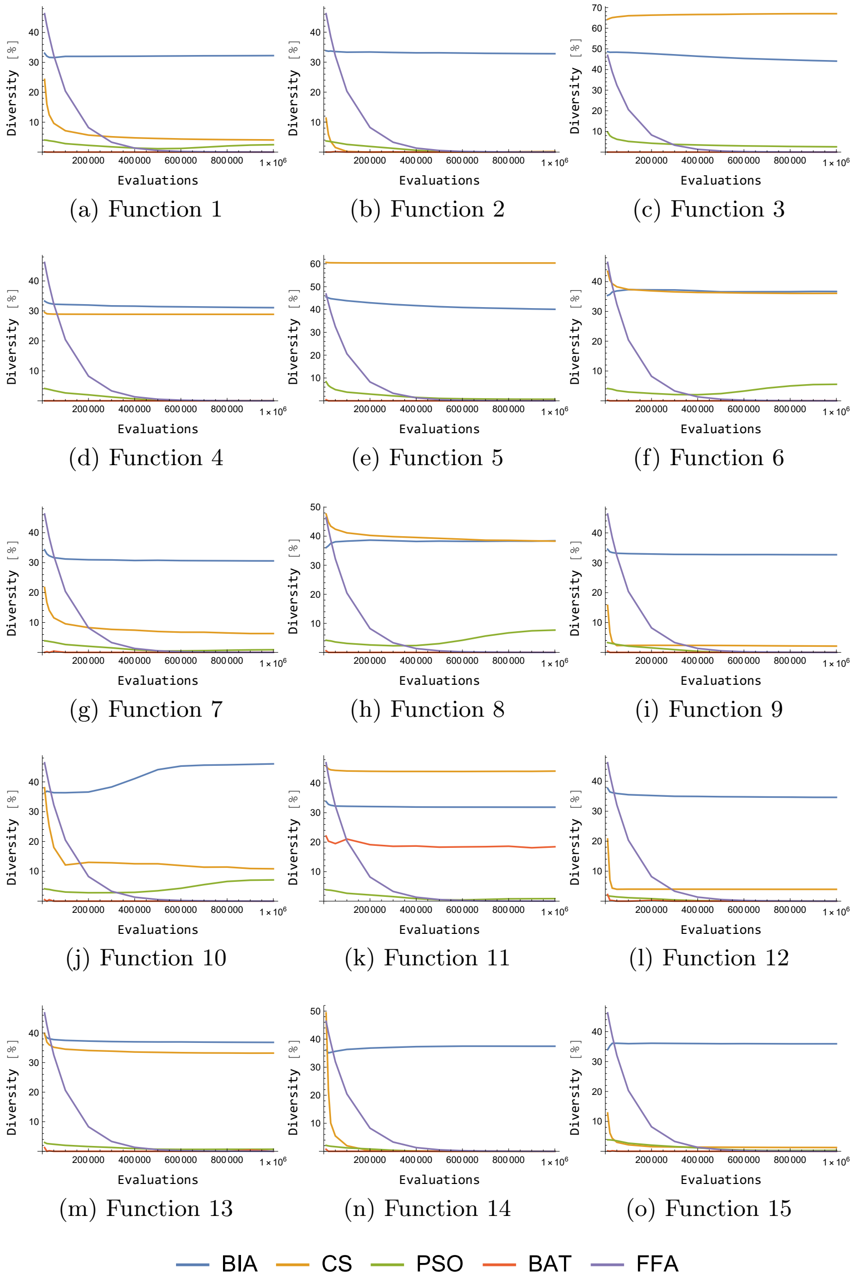
$$\text{Diversity} = \frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}(x_{i,j} - \overline{\mathbf{x}}_{\mathbf{j}})^2 \tag{7.1}$$

$$\overline{\mathbf{x}}_{\mathbf{j}} = \frac{1}{NP}\sum_{i=1}^{NP} x_{i,j} \tag{7.2}$$

Where:

- NP is the population size,
- D is the dimensionality of the problem,
- i and j are the population and dimension iterators respectively,
- $x_{i,j}$ is the vector value of the solution at the given dimension,
- and (x_j) is the corresponding mean of the solutions.

The population diversity investigation (Figure 7.10) revealed that the Bison Algorithm guarantees a stable level of diversity throughout the optimization process. While the diversities of Particle Swarm Optimization, the Bat Algorithm, and the Firefly Algorithm mostly gradually drop, and Cuckoo Search Optimization keeps a high diversity level in half of the problems, the Bison Algorithm holds the same diversity level in all the tested functions. Table 7.6 and Table 7.7 confirmed the lead of the bison population's diversity with mean and median diversity values.



(a) Function 1 (b) Function 2 (c) Function 3

(d) Function 4 (e) Function 5 (f) Function 6

(g) Function 7 (h) Function 8 (i) Function 9

(j) Function 10 (k) Function 11 (l) Function 12

(m) Function 13 (n) Function 14 (o) Function 15

Fig. 7.10 Mean diversity convergences of benchmark test set IEEE CEC 2015 in 100 dimensions.

Tab. 7.6 Mean and median values of mean diversities percentual to the theoretical maximal diversity value computed from all functions in benchmark CEC 2015.

	BIA		**CS**		**PSO**		**BAT**		**FFA**	
	Mean	*Median*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
10 D	33.72%	33.22%	24.96%	27.65%	2.03%	1.69%	0.22%	0.01%	0.01%	0.01%
30 D	34.83%	32.97%	25.83%	23.45%	1.66%	1.21%	0.18%	0.00%	0.01%	0.01%
50 D	35.23%	35.23%	24.75%	24.75%	1.56%	1.56%	1.04%	1.04%	0.01%	0.01%
100 D	36.10%	35.93%	22.45%	10.85%	1.92%	0.70%	1.23%	0.00%	0.01%	0.01%

Tab. 7.7 Mean and median values of mean diversities percentual to the theoretical maximal diversity value computed from all functions in benchmark CEC 2017.

	BIA		**CS**		**PSO**		**BAT**		**FFA**	
	Mean	*Median*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*	*Mean*	*Median*
10 D	33.50%	32.87%	21.99%	19.57%	1.38%	1.15%	1.39%	0.01%	0.01%	0.01%
30 D	33.14%	32.85%	26.11%	30.58%	0.75%	0.41%	0.02%	0.00%	0.01%	0.01%
50 D	33.97%	33.26%	25.97%	27.83%	0.96%	0.47%	0.01%	0.00%	0.01%	0.01%
100 D	34.43%	34.29%	24.65%	25.36%	1.31%	0.50%	0.00%	0.00%	0.01%	0.01%

7.1.5 Computational Complexity

This section studies the computational complexities of the algorithms compared by following the Evaluation Criteria for IEEE CEC 2017 [25]. Table 7.8 represents the results of the computational complexities, evaluated by Eq. 7.3, and ranked in Figure 7.11. The partial data for complexity computation are in Table D.1 in Appendix E.

$$Complexity = \frac{\hat{T}_2 - T_1}{T_0} \tag{7.3}$$

Where:

- T_0 is the computing time for computation of the mathematical operations defined in [25],
- T_1 is the computing time needed for 200,000 evaluations of Function 18, which is a hybrid optimization problem of the IEEE CEC 2017 benchmarking testbed [25],
- T_2 is a sequence of 5 computing times, that the algorithm needs to compute 200,000 evaluations of Function 18,
- and $\hat{T}_2$ is the mean value of all the computing times from T_2.

Tab. 7.8 Complexity computation (by Eq. 7.3) of the compared algorithms based on the Evaluation Criteria for IEEE CEC 2017.

Algorithm	**10 D**	**30 D**	**50 D**	**100 D**
Bison Algorithm	1.37	3.30	5.45	9.70
Cuckoo Search	0.61	0.69	0.81	0.96
Bat Algorithm	1.57	2.84	4.12	6.78
Particle Swarm Optimization	1.90	5.22	8.38	15.20
Firefly Algorithm	27.64	28.66	28.27	26.09

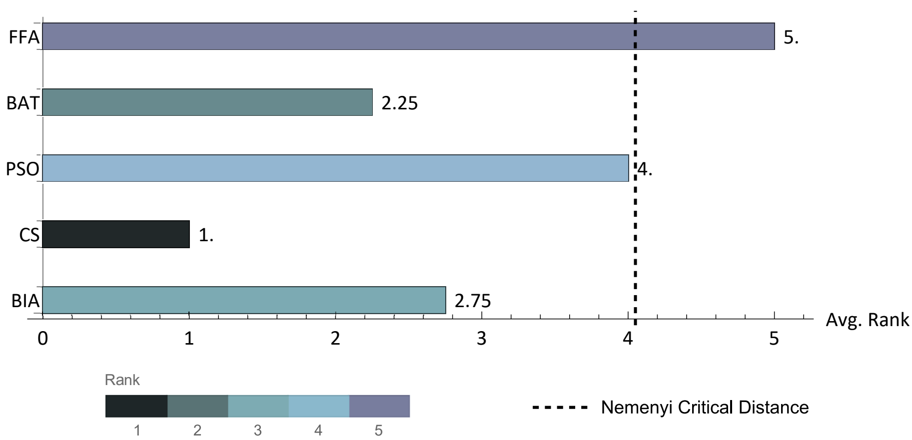


Fig. 7.11 Average rank of the overall computational complexity (Friedman Rank test, p<0.05).

The complexity computation results in Table 7.8 steadily ranked the Cuckoo Search the first, followed by the Bat Algorithm or the Bison Algorithm, Particle Swarm Optimization, and the Firefly Algorithm in last place.

However, there is a wide variety of optimization goals. While the computation time may be crucial for some problems, others may prefer a reliable quality of the final solutions. Therefore even the Firefly Algorithm, with the highest computational complexity of all metaheuristics compared, may be convenient for solving problems with unlimited time, as has been proven on the 100-dimensional IEEE CEC 2015 benchmarking testbed (Table 7.2).

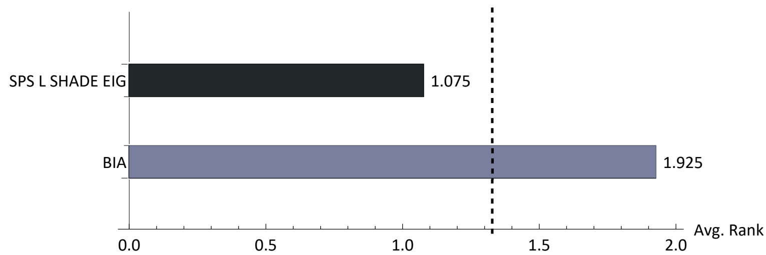
7.1.6 Comparison with Benchmark Winners

Compared to selected optimizers from the swarm family, the Bison Algorithm provided stable, competitive results. However, the introduction of novel metaheuristics should be linked with up-to-date performance analysis to find out how it stands against state-of-the-art methods. It is even more important when the aim of development is to create a tool of superior performance. Although the goal of the Bison Algorithm was of a different nature, this section compares the proposed optimizer with the winners of the CEC 2015 and CEC 2017 competitions: EBO with CMAR [226] and SPS L SHADE EIG. The codes of the competition winners were employed from the official IEEE CEC repository[1]. Both Table 7.9 and Figure 7.12 show that the competition winners generally outperformed the Bison Algorithm on the tested set of problems.

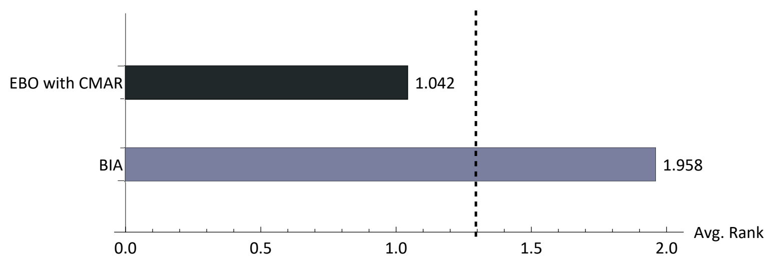
[1] `https://github.com/P-N-Suganthan/CEC2017-BoundContrained` and `https://github.com/P-N-Suganthan/CEC2015-Learning-Based`, accessed 08/2021

Tab. 7.9 Winning Algorithms on CEC 2015 and CEC 2017: BIA vs the competition winners (Wilcoxon Rank-Sum test, $p<0.05$).

	CEC 2015			**CEC 2017**		
	None	**SPS L SHADE EIG**	**BIA**	**None**	**EBO with CMAR**	**BIA**
10 dimensions	2	**13**	0	2	**28**	0
30 dimensions	1	**14**	0	1	**28**	1
50 dimensions	1	**13**	1	1	**29**	0
100 dimensions	0	**15**	0	0	**29**	1
Sum of wins	4	**55**	1	4	**114**	2



(a) CEC 2015
P-Value = 9.78E-24



(b) CEC 2017
P-Value = 2.24E-26

----- Nemenyi Critical Distance

Fig. 7.12 Friedman Rank test comparing the Bison Algorithm with the competition winners ($p<0.05$).

7.2 Performance of the Bison Algorithm Modifications

This section analyses the Bison Algorithm modifications and fulfillment of their goals. Table 7.10 represents the motivations of each bison variation and corresponding show-case analysis.

Tab. 7.10 Analysis of the Bison Algorithm modifications.

Version	Development motivation	Experiment	Section
Original Bison Algorithm		Mean solution error	Figure 7.13 in Section 7.2
Standard Bison Algorithm (with a coherent running group}	Potential benefit of group exploration	Thorough analysis on IEEE CEC 2015 and 2017 benchmark testbeds compared to six optimizers	7
Bison Seeker Algorithm	Better utilization of promising solutions	Success simulation experiment	7.2.1
Run Support Strategy	Better utilization of promising solutions	Success simulation experiment	7.2.1
Self-Adaptive Bison Algorithm	Robust parameter setting Detailed information about the inner dynamics of the algorithm	Comparison with the standard Bison Algorithm on CEC 2015/2017 Statistics of final parameter settings Convergence of individual parameters	7.2.2

Most Bison Algorithm modifications vary in the exploration mechanisms, conditioned mainly by finding an extraordinary solution. However, since the success of the running group depends on a stochastic element, it rarely happens. Table 7.11 shows the average number of iterations in which the running group found a better solution than the swarming group on the IEEE CEC 2017 test set. The exploration found a promising solution 1–786 times, but mostly in less than 0.20% of all the iterations. Nevertheless, in theory, the true success of the running group is needed once.

Since most algorithm modifications examined differ on these rare occasions, there are usually no significant performance differences in the mean error value criteria. Some modifications were, therefore, examined precisely at the moment of successful exploration. In the "success simulation experiment," further expanded in Section 7.2.1, both the Bison Seeker Algorithm and the Run Support Strategy generally outperformed the standard Bison Algorithm [216, 211]. However, the results were primarily insignificant on the complete set of the IEEE CEC 2017 benchmark, like in Figure 7.13.

Tab. 7.11 Successful running group statistics on CEC 2017: examining the minimum number of iterations in which the running group found a promising solution. Average of minimal encounters, average of maximal, average number of encounters, standard deviation of the means, median number of encounters, and percentage of all the iterations.

Dimensions	**Min**	**Max**	**Mean**	**Std mean**	**Median**	**Median** (%)
10 D	1.1	34.0	4.6	2.83	4.03	0.20%
30 D	1.6	49.0	5.0	4.43	4.63	0.08%
50 D	1.7	85.0	6.1	6.37	5.73	0.06%
100 D	2.6	786.0	7.7	8.99	6.87	0.03%

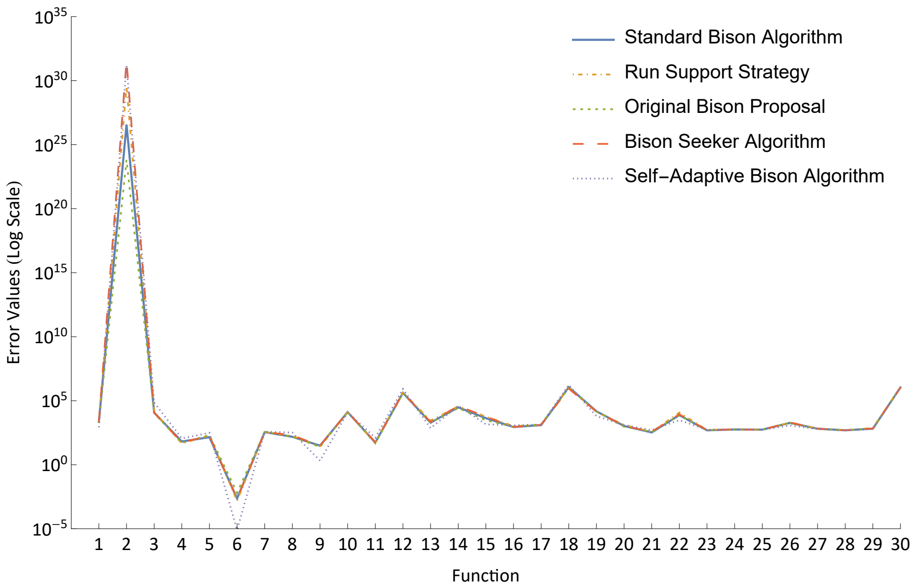


Fig. 7.13 Mean error values of all the Bison Algorithm variations solving the IEEE CEC 2017 benchmark test set in 50 dimensions.

7.2.1 Success Simulation Experiment

The success simulation experiments placed the running group solutions in the proximity of the global optimum. The Bison Seekers were randomly generated in 1/15 of the search space around the optimum, and the Run Support Strategy one run direction vector over the optimum location. The results were examined on three problems with known optimum locations: Schwefel's Function, Rastrigin's Function, and Easom's Function.

Table 7.12 summarizes the statistics of three Bison Algorithm variants during the success simulation experiments, examining the mean solution values, their standard deviations, and the optimum find rate. The optimum find rate (Figure 7.14) illustrates the ratio of all the cases, where the algorithm found the exact global optimum. Table 7.13 shows cases where one Bison Algorithm variant significantly outperformed the others according to the Wilcoxon Rank-Sum test ($\alpha = 0.05$). This table favored the Run Support Strategy as the most successful algorithm at utilizing the explored solutions.

It should be noted that besides the three problems studied, the original success simulation experiments in [216, 211] also examined Rosenbrock's Function with insignificant differences, as they are in Rastrigin's Function. On the other hand, the significant differences were in Schwefel's and Easom's Functions, modeling a many local optima problem, and a planar problem with global optimum surrounded by narrow monotonous neighborhood, respectively.

Tab. 7.12 Performance of the Run Support Strategy, Bison Seeker Algorithm, and standard Bison Algorithm on functions with simulated success (mean error, standard deviation, and optimum find rate).

	Run Support Strategy			Bison Seeker Algorithm			Standard Bison Algorithm		
Rastrigin	*avg*	*std*	*opt*	*avg*	*std*	*opt*	*avg*	*std*	*opt*
10D	3.33	1.96	(7%)	3.31	4.12	(2%)	3.97	3.39	(0%)
30D	19.27	6.38	(0%)	20.19	7.08	(0%)	22.00	14.69	(0%)
50D	45.24	12.49	(0%)	47.79	2.58E06	(0%)	44.64	10.39	(0%)
Schwefel									
10D	203.52	330.81	(53%)	342.8	544.85	(55%)	741.07	610.45	(27%)
30D	578.11	1070.85	(23%)	578.43	969.15	(18%)	3022.16	1156.57	(3%)
50D	1091.92	1785.85	(10%)	1152.8	1094.59	(2%)	5129.62	1349.55	(0%)
Easom									
10D	1.84	3.15	(73%)	1.38	2.83	(80%)	1.84	2.95	(70%)
30D	5.11	9.48	(73%)	19.63	11.1	(22%)	19.19	10.87	(20%)
50D	4.37	11.94	(80%)	45.27	6.48	(2%)	41.40	9.87	(3%)

Tab. 7.13 Wilcoxon Rank-Sum test (p=0.05), comparing three Bison Algorithm variants on a 3 functions testbed (Schwefel, Rastrigin, Easom).

	Run Support Strategy	Bison Seeker Algorithm	Standard Bison Algorithm	None	Decisive problems
10 D	1	0	0	2	Schwefel
30 D	2	0	0	1	Schwefel, Easom
50 D	2	0	0	1	Schwefel, Easom

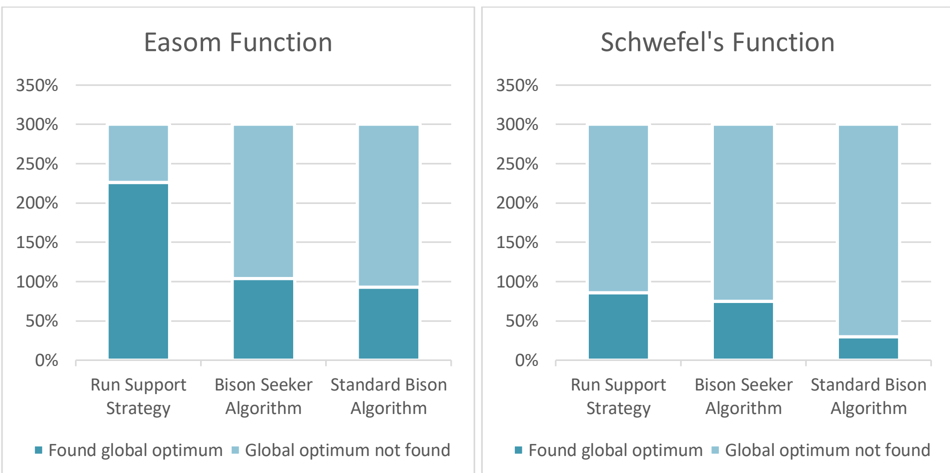


Fig. 7.14 Optimum find rate of the Run Support Strategy, the Bison Seeker Algorithm, and the standard Bison Algorithm in Easom's Function and Schwefel's Function in 10+30+50 dimensions.

7.2.2 Self-Adaptive Bison Algorithm Performance

This section analyses the asset of the Self-Adaptive Bison Algorithm (SA BIA). The experiments followed the IEEE CEC evaluation guidelines [25] with 51 runs and 10,000 $\times$ dimension objective function evaluations. To meet the maximum function evaluation limit, each sub-population consumed 1/7 of the budget.

First, the algorithm was compared to the standard Bison Algorithm. The comparison of mean error values of all the variations tested (Figure 7.13 in Section 7.2) implied minimal differences, insignificant with the Friedman Rank test (p>0.05). However, the standard Bison Algorithm provided significantly better results on most of the problems tested with the Wilcoxon Rank-Sum test ($\alpha = 0.05$) in Table 7.14, highlighting the problem-dependency of the results.

The algorithm was then analyzed for the final parameter configurations. Table 7.15 states the median values of the core population's last parameter configurations in CEC 2015. Generally, several recurrent configurations appeared across all the dimensions, which are enumerated in Figure 7.15. Interestingly, the most employed final configurations included the initial setting of the core population, formerly recommended by the Bison Algorithm parameter tuning experiment [210]. The second most frequent configuration of $SG = 49$ (the upper limit of possible SG values) boosted the exploitation factor and suppressed the exploration factor completely.

Since the optimal parameter configuration may evolve during optimization, the next set of tests studied the change in parameters in the process. Figure 7.16 shows the mean convergence of the overstep parameter values of the core population, solving each IEEE CEC 2015 problem separately. With two exceptions, the general course was quite similar, keeping the overstep parameter at 3.5. Figure 7.17 illustrates the mean convergence of the swarm group size and elite group size parameters. Neither figure shows surprising drops or rises in the mean convergences. Figure 7.18 summarizes the range of mean parameter values in one chart.

Finally, the populations were tested for their contribution – how often each population was used for its superior solutions. This experiment allowed a possible bias favoring the core population since the best solution is always shared with the core from the next iteration after the discovery. Nevertheless, it still provides valuable information about the usefulness of the individual subpopulations. Figure 7.16 shows the rate of the successful subpopulation in 10, 30, 50, and 100 dimensions in CEC 2015. Figure 7.19 sums up the percentual success rate of the subpopulations across all the dimensions tested. Based on these results, the most fruitful subpopulations were the core population, the SG high population, and the EG high population. On the other hand, the SG low and both overstep populations made a minimal difference.

These findings unlocked a new level of understanding of the inner dynamics of the algorithm. The Self-Adaptive version provided a robust approach for parameter tuning. Further studies may include different configurations of the initial subpopulations — set to the featured final parameter settings. Also, despite the favored exploitation, it might be interesting to add one purely explorative population.

Tab. 7.14 Wilcoxon Rank-Sum test (p=0.05) comparing the standard Bison Algorithm (BIA) and Self-Adaptive Bison Algorithm (SA BIA).

	CEC 2015			**CEC 2017**		
	None	BIA	SA BIA	None	BIA	SA BIA
10 dimensions	6	6	3	3	15	12
30 dimensions	5	8	2	9	11	10
50 dimensions	4	8	3	9	12	9
100 dimensions	1	9	5	8	14	8
Sum of wins	16	31	13	29	52	39

Tab. 7.15 Median parameter configurations of final core populations on CEC 2015.

	EG				**SG**			
	10D	30D	50D	100D	10D	30D	50D	100D
F1	20	20	20	20	49	49	49	49
F2	20	20	20	20	40	40	49	49
F3	20	20	20	20	40	40	40	40
F4	20	20	20	40	40	40	40	40
F5	1	20	20	20	40	40	40	40
F6	20	20	20	20	40	49	49	49
F7	20	20	20	40	49	49	40	40
F8	20	20	20	20	40	49	49	49
F9	20	20	20	40	49	49	49	40
F10	20	20	20	20	49	49	49	49
F11	20	20	40	40	40	40	40	40
F12	40	40	40	40	40	40	40	40
F13	20	32	40	20	49	40	40	40
F14	20	20	20	20	40	49	49	49
F15	20	20	20	20	49	49	49	49

Tab. 7.16 Mean usage of population groups within the Self-Adaptive Bison Algorithm solving the IEEE CEC 2015 benchmark.

	Core	**SG high**	**SG low**	**EG high**	**EG low**	**Overstep high**	**Overstep low**
10 D	261.27	11.14	0.65	4.43	4.13	0.98	1.75
30 D	812.78	21.57	0.65	13.33	1.61	0.73	1.57
50 D	1376.82	29.46	0.63	16.34	1.30	0.60	1.65
100 D	2801.07	31.95	0.67	17.81	1.56	0.54	1.75
Sum	5251.95	94.12	2.60	51.91	8.60	2.84	6.71
Percentual	97%	2%	0%	1%	0%	0%	0%

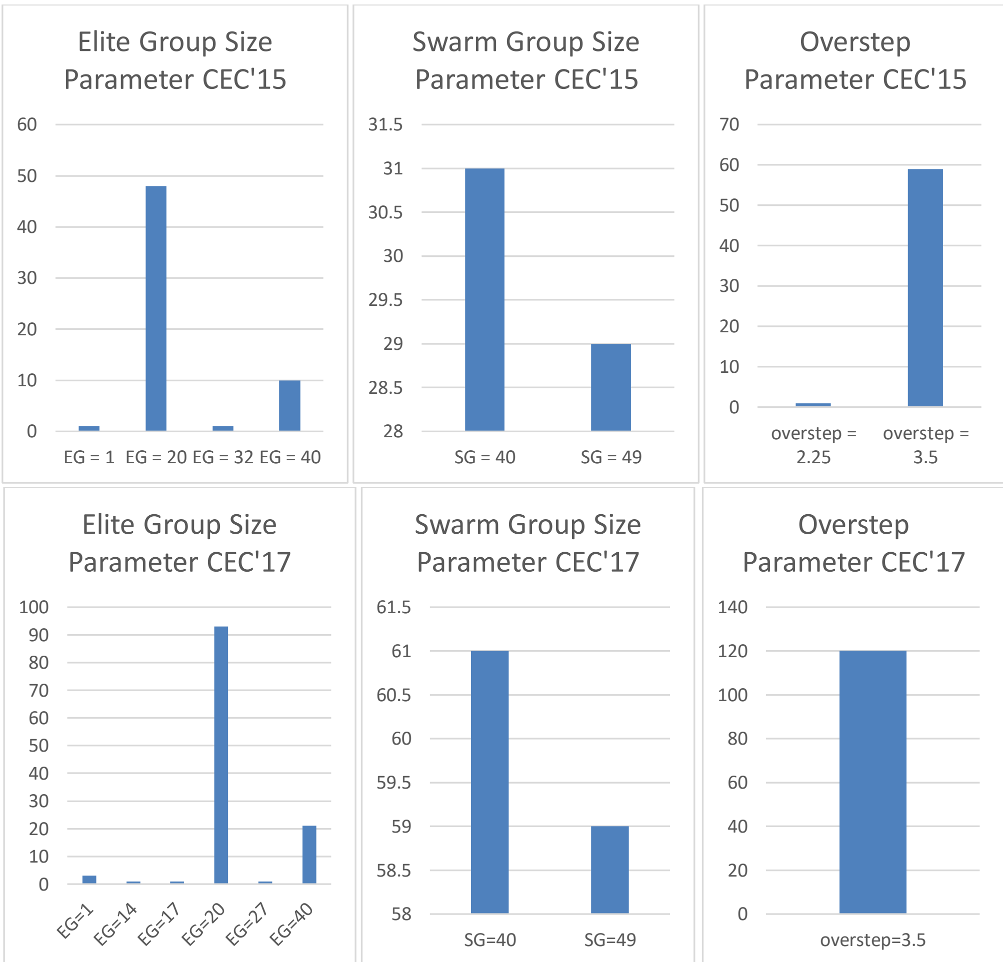


Fig. 7.15 Frequency of recurring median parameter configuration values of final core populations on CEC 2015 and 2017. (The unique overstep = 2.25m the parameter value was in CEC 2015 Function 5 in 100 dimensions.)

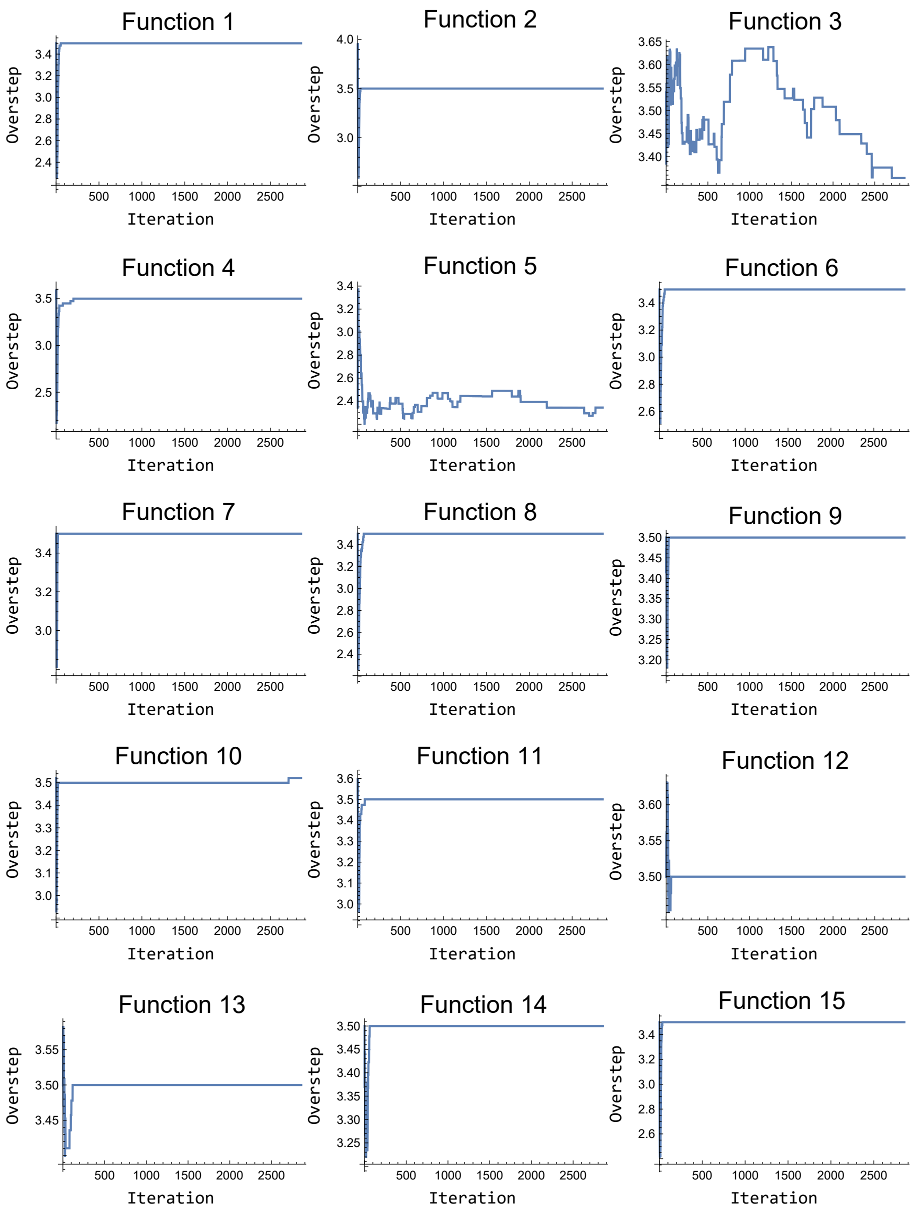


Fig. 7.16 Mean values of the core population's overstep parameter throughout the optimization process on the IEEE CEC 2015 benchmark testbed in 100 dimensions.

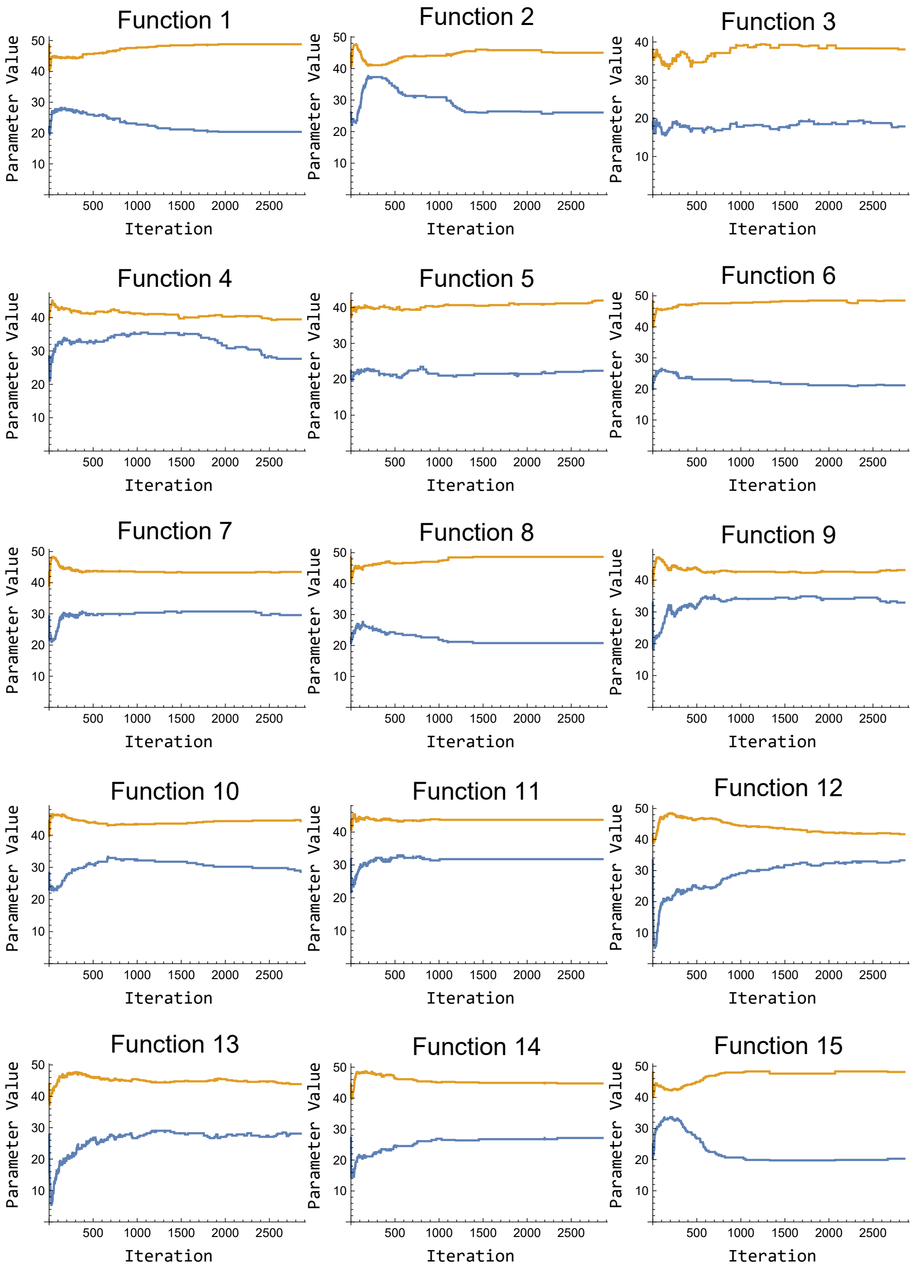
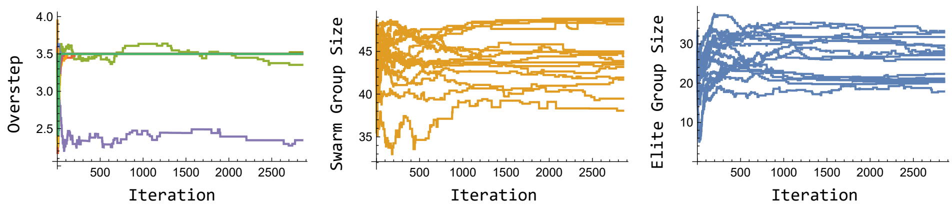


Fig. 7.17 Mean values of the core population's EG and SG parameters throughout the optimization process on the IEEE CEC 2015 benchmark testbed in 100 dimensions.



(a) Overstep (b) SG (c) EG

Fig. 7.18 All mean values of the overstep, swarm group size and elite group size parameters.

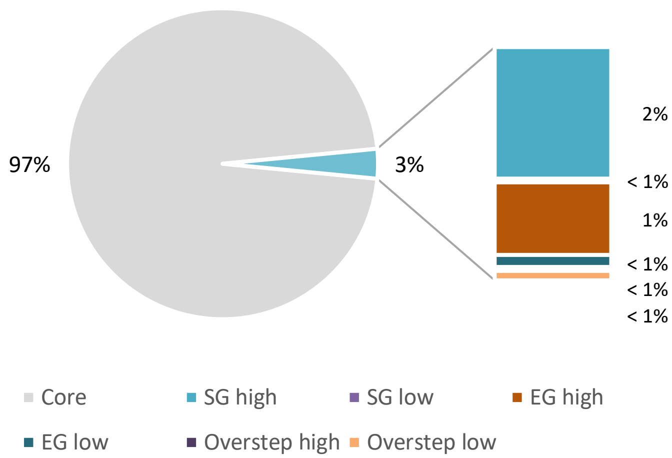


Fig. 7.19 Distribution of the leading population of the Self-Adaptive Bison Algorithm solving the CEC 2015 problem testbed across all dimensions and iterations.

8 Meaning For Applied Science

This section contemplates the meaning of the work for science and practice. The first part questions whether developing a novel metaheuristic algorithm was beneficial or merely intensified the stereotype of prejudice.

The main goal of Bison Algorithm development was not to disprove the No Free Lunch Theorem and create an algorithm with superior performance over all of the known optimizers but to show and direct further metaheuristic engineering towards meaningful development.

Furthermore, Section 8.2 explains how Bison Algorithm development followed the proposed recommendations and highlights its contribution. Section 8.3 examines the practical impact on applied science, that is, the applications. It describes how and where the algorithm was used. Finally, Section 8.4 explores how the work fulfilled its dissertation goals.

8.1 Does the World Need Yet Another Swarm Algorithm?

The excess of metaphor-based algorithms has been clearly stated as a major risk factor of current metaheuristics. More than 320 algorithms escalated into various troubles like the duality of algorithms, scepticism of any *"novel"* technique, or open disdain for metaheuristics. How could creating a new algorithm help?

It is important to acknowledge that even reviewers' scowls cannot ultimately end the development of new algorithms; very likely, new metaheuristics would arise despite generic opposition. Nevertheless, to prevent substandard production, it is vital to find a way of avoiding the recurrent mistakes that are often connected to new metaheuristic proposals. That is the goal of this thesis.

For meaningful development, the Author proposed rules for future design of novel swarm and metaheuristic algorithms. The Bison Algorithm was created as a proof of concept of the guidelines. Following the recommendations, the development was aimed at a common yet poignant problem of local optimum containment. As a result, an algorithm was born with interesting results for solving problems with many local optima.

One without the other would not make any difference. Suggesting a set of recommendations without their application would be just another invitation to improve metaheuristic practice with minimal impact. Developing a new algorithm without good scientific practice would, in fact, only reinforce the inappropriate "substandard" label on novel metaheuristics. But together, these two concepts may benefit new metaheuristics that are yet to be created.

Based on the Author's observations, metaheuristics development is at a crossroad. It can either continue in complete ignorance, leading to the total deterioration of metaheuristics or potential recommendations to ban future development. Or, it can reflect reservations and try to avoid common mistakes. Many contemporary scientists ([93, 27, 84, 123, 217, 218, 232]), the Author included, hope for the latter, and that was the content of this work. Ultimately, to answer the title question, the latter is what the world certainly needs.

8.2 Following the Recommendation Guidelines

This section evaluates how the development process followed the guidelines and recommendations from Section 3.3.

Guidelines for Algorithm Design

- ✓ Name motivation (not metaphor-based)
- ✓ Use standard vocabulary
- ✓ Share the source code of novel algorithms
- ✓ Describe algorithms with flowcharts for a better understanding
- ✓ Analyze components of the proposed algorithm individually
- ✓ Keep it simple

The guidelines for algorithm design instruct researchers to start development by naming the motivation. The motivation for the Bison Algorithm was the development of an escape tool from local optima containment, and this was stated in Section 5.1. The algorithm uses standard vocabulary for objective function, population, and solutions. Occasionally, it substitutes *"solution of the exploration running group* or *exploitation swarming group"* for *"runner"* or *"swarmer"* for a better comprehension of the methodology. The algorithm's source code is available at the Tomas Bata University Artificial Intelligence Laboratory's GitHub repository: `https://github.com/TBU-AILab/Bison-Algorithm`. The Self-Adaptive Bison Algorithm modification is available at: `https://github.com/TBU-AILab/Bison-Algorithm-OOP`. Both the standard version and all the modifications were described with Flowcharts (Figures 5.1, 6.3, 6.6, 6.4, and 6.10). The exploitation and exploration components were described separately in Sections 5.4, and 5.5.

Guidelines for Selection of Algorithms to be Compared and Benchmark

✓ Select algorithms to be compared with respect to the goal of the experiment

For performance-oriented comparison, compare algorithms with:

✓ Original version of the algorithm (first proposal)
✓ Reference version of the algorithm (the one that is modified)
✓ Best algorithms so far on the benchmark being examined (competition winner)
✓ Other algorithms operating on a similar principle

Select benchmark problems:

✓ Of broad characteristics without bias
✓ Prefer standard benchmark test sets

The selection of algorithms to be compared is connected to the goal of each experiment. In this case, two experiments were in place: a general comparison with other metaheuristics and an analysis of individual modifications. For the general metaheuristic comparison, the competing optimizers of the swarm algorithm family were selected from the top 10 most popular swarm optimizers from Section 3.1. These should fall into the class of other algorithms operating on a similar principle, although there are apparent differences. Furthermore, the algorithms selection also included the competition winners of the examined benchmark testbeds in Section 7.1.6. Since the development goal was tackling the local optimum problem, rather than winning the CEC competition, these results were complementary yet important. Comparison with what is currently state-of-the-art provides vital information for potential users. The Bison modifications were mainly compared to the standard Bison Algorithm as a reference version of the algorithm. The original proposal was included in an example of mean solution error comparison in Figure 7.13.

The experiments were carried out on standard benchmarks with broad characteristics. Moreover, the Bison Algorithm was also examined on problems liable to local optimum containment, to explore the initial motivation. Section 6.5 illustrates how the Bison modifications dealt with this problem in practice. The IEEE CEC 2017 benchmark included six problems characterized by a vast number of local optima. The Bison Algorithm was particularly successful in half of the tested cases (see Figure 7.5 b) in Section 7.1.2).

Guidelines for Experimental Setup

- ✓ Prefer own implementation over literature-based results
- ✓ Provide the same conditions for all the experiments
- ✓ Share the source codes of all the algorithms
- ✓ Tune the parameters of all the algorithms for the problem at hand with statistical tests
- ✓ Combine multiple performance measures

When examining the CPU execution time, all algorithms should:

- ✓ Be coded by the same programmer
- ✓ Be coded in the same programming language

✓ Share most functions
✓ Be examined on the same computer

There were no literature-based results. The competition winners' implementations were taken from the official IEEE CEC competition repository[1)]; others were derived from the EvoloPy library [129]. The open-source code of all the algorithms is available at `https://github.com/TBU-AILab/Bison-Algorithm-OOP`. All the experiments followed the CEC Evaluation guide and met the same objective function evaluation budget. The parameters of the swarm-based algorithms were tuned in the Parameter Tuning Experiment (Section 7.1.1) except for the competition winners, whose parameters were already tuned for the examined testbed.

The only experiment with CPU execution time was the Complexity Computation in Section 7.1.5. These experiments were carried out on the same computer, shared most of the functions, and were programmed by the Author to fit the same template in Python.

Guidelines for Results' Analysis

✓ Use statistical tests for significance
✓ Allow negative results
✓ Show results in context, provide interpretation
✓ Be cautious with generalization
✓ Depict the results in both graphs and tables
✓ Advocate assets and contribution of the algorithm (novelty/performance/methodology/challenge particular problem)

The result's analysis included the Wilcoxon Rank-Sum and Friedman Rank tests for significance and allowed for negative results. The results examined the whole set of 45 problems, but also multiple problem classes. Examination based on the character of the problems (in Section 7.1.2) provided extra context.

[1)] `https://github.com/P-N-Suganthan/CEC2017-BoundContrained` and `https://github.com/P-N-Suganthan/CEC2015-Learning-Based`, accessed 08/2021

Finally, it remains to advocate the asset and contribution of the algorithm, to which the Section 8 was dedicated. Based on the justification criteria mentioned in Section 3.3, novel algorithm development may be justified by at least one of the following:

? Novelty
X Superior performance
✓ Methodology
✓ Orientation towards a particular problem

So far, there is no tool to identify the similarity between a new optimizer and the rest of 320 metaheuristics. That is why this criterion is evaluated with a question mark. Nevertheless, to the Author's knowledge, there is no algorithm with the exploration and exploitation features similar to the Bison Algorithm. The novelty aspect, therefore, might as well benefit one of the contribution aspects. On the other hand, the algorithm did not meet the superior performance criterion since it did not outperform the competition winners. However, neither novelty nor superior performance was the main goal of the algorithm's development.

The advocacy of the proposed algorithm's development stands on 1) contribution to methodology and 2) building a tool to tackle a known optimization issue. The contribution to methodology lies in unique exploration and exploitation techniques. Moreover, the exploration method and utilization of found solutions stand as an independent block, which may be easily transferable to other optimizers, helping them escape local optima.

A secondary motivation for Bison Algorithm development was the introduction and advocacy of the last argument for the justification of novel algorithms development, that is, development aimed at fighting a particular optimization problem. The Bison Algorithm was developed with a mechanism to escape local optimum and was ultimately successful with this type of problem on the examined test set.

8.3 Applications of the Bison Algorithm

The most valuable contribution of swarm algorithms lies in quick solutions to complex real-life problems, including transportation, energy, logistics, or social networks [100]. Real-life problems need efficient real-time solutions. Although the metaheuristic approach does not guarantee finding the exact optimum, "good" solutions are often sufficient.

Because of the metaheuristics avalanche, many criticize both the quality and quantity of novel bio-inspired metaheuristics. But metaheuristics, even the novel ones, usually have their advocate in applications. Hence, the Author would like to highlight some implementations of the proposed algorithm.

The Bison Algorithm was successfully used to optimize 3 PID controllers – the water tank test, mass spring damper, DC motor, and their cascade versions in (Eqs. 8.1-8.3) [215]. The problems are defined as follows:

Water Tank Test

Goal: To maintain the desired water level $\dot{h}$ in the tank by changing the water inflow.

$$\dot{h} = \frac{1}{A}\left(q_{in} + q_{ex} - q_{em} - s \cdot \sqrt{2gh}\right), \tag{8.1}$$

Where:

- $\dot{h}$ is the water level in the tank,
- A is the surface area,
- q_{in} is a controllable water inflow,
- q_{ex} is the external water outflow,
- s is emergency water outflow,
- and $g = 9.81 m/s^2$ is the gravitational acceleration.

Mass Spring Damper

Goal: To maintain the desired position $s*$ of mass m_1 by managing the control force F.

$$\begin{cases} s_1 = v_1 \cdot t + \frac{1}{2}a_1 \cdot t^2 & v_1 = a_1 \cdot t \quad a_1 = \frac{1}{m_1}\left(k \cdot (s_2 - s_1) - v_1 \cdot y\right) \\ s_2 = v_2 \cdot t + \frac{1}{2}a_2 \cdot t^2 & v_2 = a_2 \cdot t \quad a_2 = \frac{1}{m_2}\left(k \cdot (F - s_2) - v_2 \cdot y\right) \end{cases} \tag{8.2}$$

Where:

- s_1, s_2 are the positions of masses, which are connected via spring,
- y is a constant point connected to the masses by another spring,
- and k is the stiffness constant.

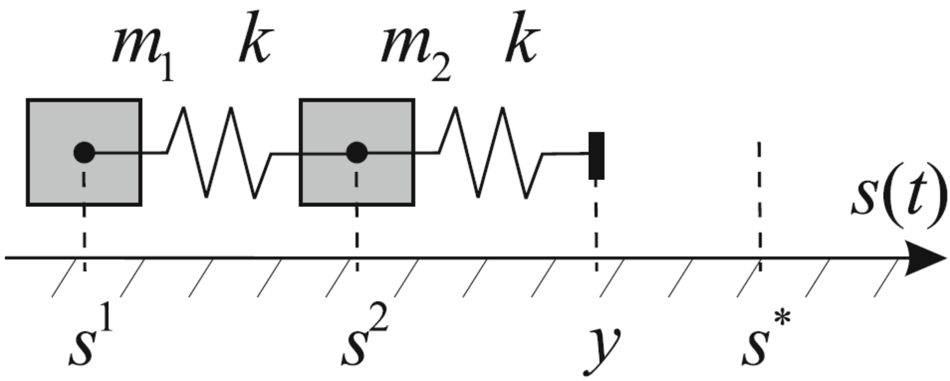


Fig. 8.1 Simulated mass spring damper.

DC Motor

Goal: to maintain the desired motor speed $\omega*$ by managing input voltage F.

$$\begin{cases} \dot{\omega} = \frac{1}{J}\left(K_t \cdot i - b \cdot \omega\right) \\ \dot{i} = \frac{1}{L}\left(-R \cdot i + V - K_e \cdot \omega\right) \end{cases} \tag{8.3}$$

Where:

- ω is the speed,
- J is the rotor's moment of inertia,
- b is viscous friction constant,
- K_t is motor torque constant,
- i is armature current,
- R is electric resistance,
- L is electric inductance,
- and K_e is the electromotive force constant.

The performance measurement included the current error, overshoot, oscillations, and suit (see Eq. 8.4). The simulations carried out 100 repetitions of $25,000$ objective function evaluations budget.

$$fitness = error \cdot \omega_e + over \cdot \omega_v + oscs \cdot \omega_o + suit \cdot \omega_s \tag{8.4}$$

The experiment compared five optimizers: the Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, the Cuckoo Search, and the Bison Algorithm, and examined the differences between standard versus cascade PID controllers. Figure 8.2 depicts the average results of the algorithms using both standard and cascade PID controllers. The engineering issues examined were minimization problems; thus, a lower bar indicates better results. Figure 8.3 summarizes the time consumption of each algorithm, and the best and worst result statistics of the optimizers: on how many of the six examined problem scenarios one algorithm deliver superior or inferior results to all the others. The paper concluded that the Bison Algorithm delivered top results (in a 5% range from the best-found results) in the majority of the tested problems.

The Bison Seeker Algorithm was applied as a hybrid method of symbolic regression in [258]. The algorithm outclassed basic symbolic regression even with a non-standard parameter setting of very few iterations and small populations. Figure 8.4 shows the percentual success rate of various instances of hybrid Bison

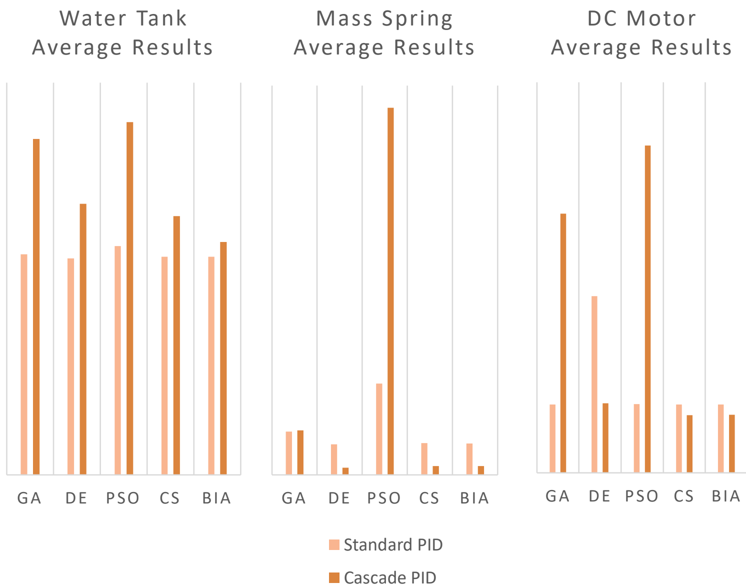


Fig. 8.2 Average fitness results of the Genetic Algorithm, Differential Evolution, Particle Swarm Optimization, the Cuckoo Search, and the Bison Algorithm designing standard and cascade PID controllers for three engineering problems.

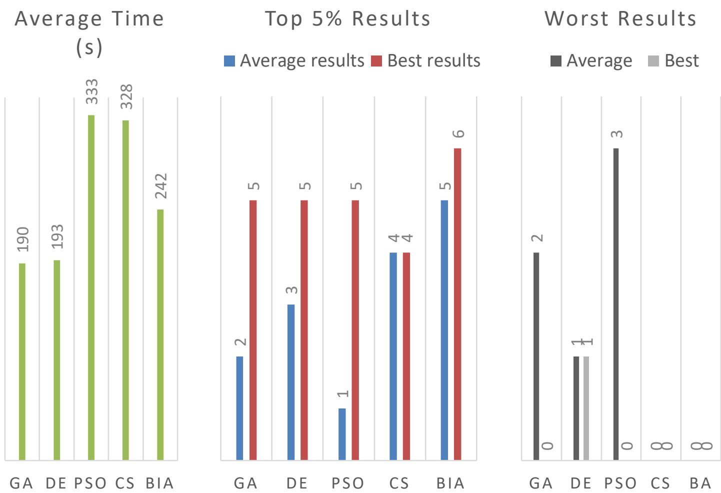


Fig. 8.3 Metaheuristic comparison: a) average time consumption of each algorithm during the experiment, b) how often one algorithm delivered a final solution of quality in the top 5%, c) how many times the algorithm delivered the worst results.

Seeker Algorithm Symbolic Regression compared to standard Symbolic Regression.

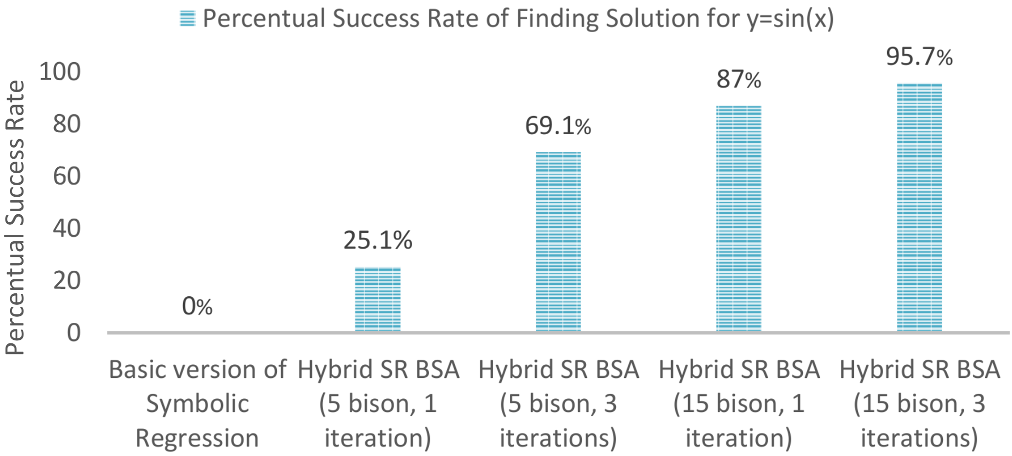


Fig. 8.4 Percentual success rate of finding the solution for $y = sin(x)$ comparing basic symbolic regression and various parameter variations of hybrid Bison Seeker Algorithm Symbolic Regression.

Dziwiński and Bartczuk [114] compared the Bison Algorithm with the Genetic Algorithm, Evolutionary Strategies, the Gravitational Search Algorithm, Differential Evolution, the Artificial Bee Colony, and the hybrid PSO and GA methods with fuzzy logic. They ranked the Bison Algorithm to the third overall place. Tolabi et al. [393] compared the Bison Algorithm on CEC 2017 with PSO, CS, and the Thief and Police Algorithm. In accordance with the No Free Lunch theorem, the Bison Algorithm kept the superiority of Functions 9 and 22 on the benchmark testbed.

8.4 Dissertation Goal Fulfillment

This section describes the steps taken to fulfill the dissertation goals, which were set as follows:

- ✓ **Map the current scene** of modern swarm algorithms, its trends, and challenges.
- ✓ **Investigate** the methods addressing the weaknesses of swarm algorithms.
- ✓ **Propose** a set of recommendations for new metaheuristics creation.
- ✓ **Proof of concept testing**: Implement the proposed recommendations and methods in a new swarm algorithm.
- ✓ **Evaluate the benefits** of the proposed algorithm for applied sciences.

There are multiple challenges in the development and modification of bio-inspired swarm algorithms. Mapping the current scene of modern metaheuristics and current trends revealed a variety of both optimization and existential problems (see Sections 2.3 and 3.2). To answer the former, Section 2.3.5 investigated the methods addressing the optimization problems, while Section 3.3. proposed guidelines to avoid the existential ones. Furthermore, the recommendations for novel metaheuristics development were applied to prove the concept, as a new swarm-based algorithm was proposed and tested. Section 6 introduced several algorithm modifications, including a self-adaptive variant, as a modern modification trend representative. Finally, Section 8 evaluated the benefits of the proposed algorithm and its applications.

9 Conclusion

Nowadays, most new metaheuristics go round in circles repeating the same mistakes and facing prejudicial disrespect regardless of the actual quality of the presented method. Many researchers, scientists, and practitioners stand up against common malpractice and try to influence future metaheuristics towards a better standard. Most recently, at the turn of 2020/2021, a great number of publications dedicated to benchmarking issues and fairness in comparison, were published. However, advising a better approach, or pointing out others' mistakes, is not as powerful as applying the change proposed.

This thesis describes the current scene of the swarm algorithms, the state-of-the-art optimization techniques, modification trends, and reservations about the pitfalls of novel metaheuristic development. Detecting two types of struggles: optimization problems like stagnation or premature convergence, and existential problems connected to the criticism mentioned above, the Author proposes a new standard for developing future metaheuristics. But most importantly, these recommendations are applied to a showcase development project of a new swarm-based algorithm.

Following the recommendations led to the creation of an algorithm designed to tackle local optimum containment. The Bison Algorithm proposes a systematical scanning of the search space independently of the exploitation process. The suggested technique offers a way out of stagnation caused by local optimum confinement. Yet, it should be easy to implement for all kinds of problems from discrete, continuous, to large-scale, or other optimizations.

The algorithm was thoroughly examined, tested, and compared to other swarm optimization methods on the sum of 45 functions of IEEE CEC 2015 and 2017. The results show that the proposed algorithm is exceptionally competent when solving problems with many local optima. The engagement of modern modification methods, including boosted exploration and the self-adaptive parameter approach, led to a deeper understanding of the inner dynamics of the algorithm.

Solving the recurrent problems of metaheuristic optimization may open the way for new challenges. The future might hold exciting discoveries like algorithm similarity detection systems, automatically assembled AI-based optimizers, neuroevolution, or new unexplored methods to tackle ubiquitous optimization problems.

This work did not aim to disclaim the No Free Lunch Theorem. It did not attempt to create a superlative optimizer that would solve every known possible problem. In fact, it aimed even higher. By setting preliminary rules and leading the way, this work presents one of many steps towards a meaningful development of metaheuristics yet to be created.

REFERENCES

[1] Abbass, H. MBO: Marriage in honey bees optimization a haplometrosis polygynous swarming approach. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 1, pp. 207–214, 2001. doi: 10.1109/CEC.2001.934391.

[2] Abdechiri, M., Meybodi, M. and Bahrami, H. Gases brownian motion optimization: An algorithm for optimization (GBMO). *Applied Soft Computing Journal*. 2013, 13, 5, pp. 2932–2946. doi: 10.1016/j.asoc.2012.03.068.

[3] Abdullah, A., Deris, S., Anwar, S. and Arjunan, S. N. V. An Evolutionary Firefly Algorithm for the Estimation of Nonlinear Biological Model Parameters. *PLOS ONE*. Mar 2013, 8, 3, pp. e56310. ISSN 1932-6203. doi: 10.1371/journal.pone.0056310.

[4] Abedinia, O., Amjady, N. and Ghasemi, A. A new metaheuristic algorithm based on shark smell optimization. *Complexity*. 2016, 21, 5, pp. 97–116. doi: 10.1002/cplx.21634.

[5] Abedinpourshotorban, H., Mariyam Shamsuddin, S., Beheshti, Z. and Jawawi, D. N. A. Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation*. Feb 2016, 26, pp. 8–22. ISSN 2210-6502. doi: 10.1016/j.swevo.2015.07.002.

[6] Aghay Kaboli, S., Selvaraj, J. and Rahim, N. Rain-fall optimization algorithm: A population based algorithm for solving constrained optimization problems. *Journal of Computational Science*. 2017, 19, pp. 31–42. doi: 10.1016/j.jocs.2016.12.010.

[7] Ahrari, A. and Atai, A. Grenade Explosion Method - A novel tool for optimization of multimodal functions. *Applied Soft Computing Journal*. 2010, 10, 4, pp. 1132–1140. doi: 10.1016/j.asoc.2009.11.032.

[8] Akbari, R., Mohammadi, A. and Ziarati, K. A novel bee swarm optimization algorithm for numerical function optimization. *Communications in Nonlinear Science and Numerical Simulation*. 2010, 15, 10, pp. 3142–3155. doi: 10.1016/j.cnsns.2009.11.003.

[9] Alatas, B. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. *Expert Systems with Applications*. 2011, 38, 10, pp. 13170–13180. doi: 10.1016/j.eswa.2011.04.126.

[10] Alauddin, M. Mosquito flying optimization (MFO). In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 79–84, 2016. doi: 10.1109/ICEEOT.2016.7754783.

[11] Almonacid, B. and Soto, R. Andean Condor Algorithm for cell formation problems. *Natural Computing*. Jun 2019, 18, 2, pp. 351–381. ISSN 1572-9796. doi: 10.1007/s11047-018-9675-0.

[12] Alsattar, H., Zaidan, A. and Zaidan, B. Novel meta-heuristic bald eagle search optimisation algorithm. *Artificial Intelligence Review*. 2020, 53, 3, pp. 2237–2264. doi: 10.1007/s10462-019-09732-5.

[13] Anandaraman, C., Sankar, A. V. M. and Natarajan, R. A new evolutionary algorithm based on bacterial evolution and its application for scheduling a flexible manufacturing system. *Jurnal Teknik Industri*. 2012, 14, 1, pp. 1–12.

[14] Anita and Yadav, A. AEFA: Artificial electric field algorithm for global optimization. *Swarm and Evolutionary Computation*. 2019, 48, pp. 93–108. doi: 10.1016/j.swevo.2019.03.013.

[15] Aranha, C., Camacho Villalón, C. L., Campelo, F., Dorigo, M., Ruiz, R., Sevaux, M., Sörensen, K. and Stützle, T. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence*. Nov 2021. ISSN 1935-3820. doi: 10.1007/s11721-021-00202-9. Available from: <`https://doi.org/10.1007/s11721-021-00202-9`>.

[16] Ardjmand, E. and Amin-Naseri, M. R. Unconscious Search - A New Structured Search Algorithm for Solving Continuous Engineering Optimization Problems Based on the Theory of Psychoanalysis. In Tan, Y., Shi, Y. and Ji, Z. (Ed.) *Advances in Swarm Intelligence*, Lecture Notes in Computer Science, pp. 233–242. Springer, 2012. doi: 10.1007/978-3-642-30976-2_28. ISBN 978-3-642-30976-2.

[17] Arnaout, J.-P. Worm Optimization: A novel optimization algorithm inspired by C. Elegans. In *Proceedings of the 2014 international conference on industrial engineering and operations management, Indonesia*, pp. 2499–2505, 2014.

[18] Arora, S. and Singh, S. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Computing*. 2019, 23, 3, pp. 715–734. doi: 10.1007/s00500-018-3102-4.

[19] Arrieta, A. B. et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *CoRR*. 2019, abs/1910.10045. Available from: `<http://arxiv.org/abs/1910.10045>`.

[20] Ashrafi, S. and Dariane, A. A novel and effective algorithm for numerical optimization: Melody Search (MS). In *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, pp. 109–114, 2011. doi: 10.1109/HIS.2011.6122089.

[21] Asil Gharebaghi, S. and Ardalan Asl, M. New meta-heuristic optimization algorithm using neuronal communication. *Iran University of Science & Technology*. 2017, 7, 3, pp. 413–431.

[22] Askarzadeh, A. Bird mating optimizer: An optimization algorithm inspired by bird mating strategies. *Communications in Nonlinear Science and Numerical Simulation*. 2014, 19, 4, pp. 1213–1228. doi: 10.1016/j.cnsns.2013.08.027.

[23] Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers and Structures*. 2016, 169, pp. 1–12. doi: 10.1016/j.compstruc.2016.03.001.

[24] Atashpaz-Gargari, E. and Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *2007 IEEE Congress on Evolutionary Computation*, pp. 4661–4667, 2007. doi: 10.1109/CEC.2007.4425083.

[25] Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J. and Qu, B. Y. Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. *Technical Report, Nanyang Technological University, Singapore*. 2016, 2016, pp. 34.

[26] Bansal, J., Sharma, H., Jadon, S. and Clerc, M. Spider Monkey Optimization algorithm for numerical optimization. *Memetic Computing*. 2014, 6, 1, pp. 31–47. doi: 10.1007/s12293-013-0128-0.

[27] Bartz-Beielstein, T. et al. Benchmarking in Optimization: Best Practice and Open Issues. *arXiv:2007.03488 [cs, math, stat]*. Dec 2020. Available from: `<http://arxiv.org/abs/2007.03488>`. arXiv: 2007.03488.

[28] Barzegar, B., Rahmani, A., Zamanifar, K. and Divsalar, A. Gravitational emulation local search algorithm for advanced reservation and scheduling in grid computing systems. In *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, pp. 1240–1245, 2009. doi: 10.1109/ICCIT.2009.319.

[29] Basu, S., Chaudhuri, C., Kundu, M., Nasipuri, M. and Basu, D. Text line extraction from multi-skewed handwritten documents. *Pattern Recognition*. 2007, 40, 6, pp. 1825–1839. doi: 10.1016/j.patcog.2006.10.002.

[30] Bayraktar, Z., Komurcu, M. and Werner, D. Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics. In *2010 IEEE Antennas and Propagation Society International Symposium*, 2010. doi: 10.1109/APS.2010.5562213.

[31] Beiranvand, H. and Rokrok, E. General Relativity Search Algorithm: A Global Optimization Approach. *International Journal of Computational Intelligence and Applications*. 2015, 14, 3. doi: 10.1142/S1469026815500170.

[32] Bellaachia, A. and Bari, A. Flock by leader: A novel machine learning biologically inspired clustering algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012, 7332 LNCS, PART 2, pp. 117–126. doi: 10.1007/978-3-642-31020-1_15.

[33] Benbouzid-Si Tayeb, F., Bessedik, M., Benbouzid, M., Cheurfi, H. and Blizak, A. Research on Permutation Flow-shop Scheduling Problem based on Improved Genetic Immune Algorithm with vaccinated offspring. In *Procedia Computer Science*, 112, pp. 427–436, 2017. doi: 10.1016/j.procs.2017.08.055.

[34] Benlic, U. and Hao, J.-K. *Hybrid Metaheuristics for the Graph Partitioning Problem*, pp. 157–185. Studies in Computational Intelligence. Springer, 2013. doi: 10.1007/978-3-642-30671-6_6. Available from: `<https://doi.org/10.1007/978-3-642-30671-6_6>`. ISBN 978-3-642-30671-6.

[35] Bergh, F. v. d. and Engelbrecht, A. P. A study of particle swarm optimization particle trajectories. *Inf. Sci.* 2006. doi: 10.1016/j.ins.2005.02.003.

[36] Berman, R. *American Bison*. Lerner Publications, Sep 2008. ISBN 978-0-8225-7513-9.

[37] Bernardino, H. S. and Barbosa, H. J. C. *Artificial Immune Systems for Optimization*, pp. 389–411. Studies in Computational Intelligence. Springer, 2009. doi: 10.1007/978-3-642-00267-0_14. Available from: `<https://doi.org/10.1007/978-3-642-00267-0_14>`. ISBN 978-3-642-00267-0.

[38] Beyer, H.-G. and Schwefel, H.-P. Evolution strategies–A comprehensive introduction. *Natural computing*. 2002, 1, 1, pp. 3–52.

[39] Birattari, M., Paquete, L., Strutzle, T. and Varrentrapp, K. Classification of Metaheuristics and Design of Experiments for the Analysis of Components. *Tech. Rep. AIDA-01-05*. 2001.

[40] Birbil, c. and Fang, S.-C. An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*. 2003, 25, 3, pp. 263–282. doi: 10.1023/A:1022452626305.

[41] Biswas, A., Mishra, K. K., Tiwari, S. and Misra, A. K. Physics-Inspired Optimization Algorithms: A Survey, Jun 2013. ISSN 2356-752X. Available from: <https://www.hindawi.com/journals/jopti/2013/438152/>. DOI: https://doi.org/10.1155/2013/438152.

[42] Bitam, S., Zeadally, S. and Mellouk, A. Fog computing job scheduling optimization based on bees swarm. *Enterprise Information Systems*. 2018, 12, 4, pp. 373–397. doi: 10.1080/17517575.2017.1304579.

[43] Biyanto, T., Fibrianto, H., Nugroho, G., Hatta, A., Listijorini, E., Budiati, T. and Huda, H. Duelist algorithm: An algorithm inspired by how duelist improve their capabilities in a duel. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2016, 9712 LNCS, pp. 39–47. doi: 10.1007/978-3-319-41000-5_4.

[44] Biyanto, T., Matradji, Irawan, S., Febrianto, H., Afdanny, N., Rahman, A., Gunawan, K., Pratama, J. and Bethiana, T. Killer Whale Algorithm: An Algorithm Inspired by the Life of Killer Whale. In *Procedia Computer Science*, 124, pp. 151–157, 2017. doi: 10.1016/j.procs.2017.12.141.

[45] Blum, C., Puchinger, J., Raidl, G. R. and Roli, A. A brief survey on hybrid metaheuristics. *Proceedings of BIOMA*. 2010, pp. 3–18.

[46] Bodaghi, M. and Samieefar, K. Meta-heuristic bus transportation algorithm. *Iran Journal of Computer Science*. 2019, 2, 1, pp. 23–32.

[47] Boettcher, S. and Percus, A. G. Extremal optimization: Methods derived from co-evolution. *arXiv preprint math/9904056*. 1999.

[48] Borji, A. and Hamide, M. A new approach to global optimization motivated by parliamentary political competitions. *International Journal of Innovative Computing, Information and Control*. 2009, 5, 6, pp. 1643–1653.

[49] Brabazon, A., Cui, W. and O'Neill, M. The raven roosting optimisation algorithm. *Soft Computing*. 2016, 20, 2, pp. 525–545. doi: 10.1007/s00500-014-1520-5.

[50] Brest, J., Zamuda, A., Boskovic, B., Maucec, M. S. and Zumer, V. Dynamic optimization using Self-Adaptive Differential Evolution. In *2009 IEEE Congress on Evolutionary Computation*, pp. 415–422, May 2009. doi: 10.1109/CEC.2009.4982976.

[51] Brownlee, J. A Note on Research Methodology and Benchmarking Optimization Algorithms. Technical report, Complex Intelligent Systems Laboratory (CIS), Centre for Information Technology Research (CITR), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, 2007.

[52] Burke, E. K., Curtois, T., Kendall, G., Hyde, M., Ochoa, G. and Vazquez-Rodriguez, J. A. Towards the Decathlon Challenge of Search Heuristics. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pp. 2205–2208, New York, NY, USA, 2009. Association for Computing Machinery. doi: 10.1145/1570256.1570303. Available from: <https://doi.org/10.1145/1570256.1570303>. ISBN 9781605585055.

[53] Bäck, T., Hoffmeister, F. and Schwefel, H.-P. A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann, 1991.

[54] CAI, X., CUI, Y. and TAN, Y. Predicted modified PSO with time-varying accelerator coefficients. *International Journal of Bio-Inspired Computation*. Jan 2009, 1, 1–2, pp. 50–60. ISSN 1758-0366. doi: 10.1504/IJBIC.2009.022773.

[55] CAMACHO VILLALÓN, C. L., STÜTZLE, T. and DORIGO, M. Grey Wolf, Firefly and Bat Algorithms: Three Widespread Algorithms that Do Not Contain Any Novelty. In DORIGO, M., STÜTZLE, T., BLESA, M. J., BLUM, C., HAMANN, H., HEINRICH, M. K. and STROBEL, V. (Ed.) *Swarm Intelligence*, pp. 121–133, Cham, 2020. Springer International Publishing. ISBN 978-3-030-60376-2.

[56] CANAYAZ, M. and KARCI, A. Cricket behaviour-based evolutionary computation technique in solving engineering optimization problems. *Applied Intelligence*. 2016, 44, 2, pp. 362–376. doi: 10.1007/s10489-015-0706-6.

[57] CARAVEO, C., VALDEZ, F. and CASTILLO, O. A new optimization metaheuristic algorithm based on self-defense mechanism of the plants with three reproduction operators. *Soft Computing*. 2018, 22, 15, pp. 4907–4920. doi: 10.1007/s00500-018-3188-8.

[58] CHAKRABORTY, A. and KAR, A. K. *Swarm Intelligence: A Review of Algorithms*, 10, pp. 475–494. Springer International Publishing, 2017. doi: 10.1007/978-3-319-50920-4_19. Available from: <http://link.springer.com/10.1007/978-3-319-50920-4_19>. ISBN 978-3-319-50919-8.

[59] CHAN, C., XUE, F., IP, W. and CHEUNG, C. A hyper-heuristic inspired by pearl hunting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012, 7219 LNCS, pp. 349–353. doi: 10.1007/978-3-642-34413-8_26.

[60] CHEN, C.-C., TSAI, Y.-C., LIU, I.-I., LAI, C.-C., YEH, Y.-T., KUO, S.-Y. and CHOU, Y.-H. A Novel Metaheuristic: Jaguar Algorithm with Learning Behavior. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1595–1600, 2016. doi: 10.1109/SMC.2015.282.

[61] Chen, G. Simplified particle swarm optimization algorithm based on particles classification. In *2010 Sixth International Conference on Natural Computation*, 5, pp. 2701–2705, Aug 2010. doi: 10.1109/ICNC.2010.5582563.

[62] Chen, H., Zhu, Y., Hu, K. and He, X. Hierarchical swarm model: A new approach to optimization. *Discrete Dynamics in Nature and Society*. 2010, 2010. doi: 10.1155/2010/379649.

[63] Chen, S. An analysis of locust swarms on large scale global optimization problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2009, 5865 LNAI, pp. 211–220. doi: 10.1007/978-3-642-10427-5_21.

[64] Chen, T.-., Tsai, P.-W., Chu, S.-A. and Pan, J.-S. A novel optimization approach: Bacterial-GA foraging. In *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, 2007. doi: 10.1109/ICICIC.2007.67.

[65] Chen, T., Pang, L., Du, J., Liu, Z. and Zhang, L. Artificial Searching Swarm Algorithm for solving constrained optimization problems. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 1, pp. 562–565, Nov 2009. doi: 10.1109/ICICISYS.2009.5357779.

[66] Chen, T., Wang, Y. and Li, J. Artificial tribe algorithm and its performance analysis. *Journal of Software*. 2012, 7, 3, pp. 651–656. doi: 10.4304/jsw.7.3.651-656.

[67] Cheng, M.-Y. and Prayogo, D. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. *Computers and Structures*. 2014, 139, pp. 98–112. doi: 10.1016/j.compstruc.2014.03.007.

[68] Cheng, S. Population diversity in particle swarm optimization: Definition, observation, control, and application. *University of Liverpool, England*. 2013.

[69] Cheraghalipour, A., Hajiaghaei-Keshteli, M. and Paydar, M. Tree Growth Algorithm (TGA): A novel approach for solving optimization

problems. *Engineering Applications of Artificial Intelligence*. 2018, 72, pp. 393–414. doi: 10.1016/j.engappai.2018.04.021.

[70] Chire Saire, J. and Túpac, V. An approach to real-coded quantum inspired evolutionary algorithm using particles filter. In *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, 2016. doi: 10.1109/LA-CCI.2015.7435984.

[71] Chu, S.-A., Tsai, P.-W. and Pan, J.-S. Cat swarm optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2006, 4099 LNAI, pp. 854–858. doi: 10.1007/11801603_94.

[72] Chu, X., Wu, T., Weir, J. D., Shi, Y., Niu, B. and Li, L. Learning–interaction–diversification framework for swarm intelligence optimizers: a unified perspective. *Neural Computing and Applications*. Mar 2020, 32, 6, pp. 1789–1809. ISSN 1433-3058. doi: 10.1007/s00521-018-3657-0.

[73] Chu, Y., Mi, H., Liao, H., Ji, Z. and Wu, Q. A Fast Bacterial Swarming Algorithm for high-dimensional function optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 3135–3140, 2008. doi: 10.1109/CEC.2008.4631222.

[74] Chuang, C.-L. and Jiang, J.-A. Integrated radiation optimization: Inspired by the gravitational radiation in the curvature of space-time. In *2007 IEEE Congress on Evolutionary Computation*, pp. 3157–3164, 2007. doi: 10.1109/CEC.2007.4424875.

[75] Chuang, L.-Y., Tsai, S.-W. and Yang, C.-H. Improved binary particle swarm optimization using catfish effect for feature selection. *Expert Systems with Applications*. 2011, 38, 10, pp. 12699–12707. doi: 10.1016/j.eswa.2011.04.057.

[76] Chuang, L.-Y., Tsai, S.-W. and Yang, C.-H. Chaotic catfish particle swarm optimization for solving global numerical optimization problems.

Applied Mathematics and Computation. Apr 2011, 217, 16, pp. 6900–6916. ISSN 0096-3003. doi: 10.1016/j.amc.2011.01.081.

[77] Civicioglu, P. Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers and Geosciences.* 2012, 46, pp. 229–247. doi: 10.1016/j.cageo.2011.12.011.

[78] Civicioglu, P. Backtracking Search Optimization Algorithm for numerical optimization problems. *Applied Mathematics and Computation.* 2013, 219, 15, pp. 8121–8144. doi: 10.1016/j.amc.2013.02.017.

[79] Civicioglu, P. Artificial cooperative search algorithm for numerical optimization problems. *Information Sciences.* 2013, 229, pp. 58–76. doi: 10.1016/j.ins.2012.11.013.

[80] Coelho, L. d. S., Sauer, J. G. and Rudek, M. Differential evolution optimization combined with chaotic sequences for image contrast enhancement. *Chaos, Solitons & Fractals.* Oct 2009, 42, 1, pp. 522–529. ISSN 0960-0779. doi: 10.1016/j.chaos.2009.01.012.

[81] Comellas, F. and Martínez-Navarro, J. Bumblebees: A multiagent combinatorial optimization algorithm inspired by social insect behaviour. In *GEC '09*, pp. 811–814, 2009. doi: 10.1145/1543834.1543949.

[82] Cortes, P., Garcia, J., Munuzuri, J. and Onieva, L. Viral systems: A new bio-inspired optimisation approach. *Computers and Operations Research.* 2008, 35, 9, pp. 2840–2860. doi: 10.1016/j.cor.2006.12.018.

[83] Cortes, P., Garcia, J., Onieva, L., Munuzuri, J. and Guadix, J. Viral system to solve optimization problems: An immune-inspired computational intelligence approach. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* 2008, 5132 LNCS, pp. 83–94. doi: 10.1007/978-3-540-85072-4_8.

[84] Csébfalvi, A. and Csébfalvi, G. Fair Comparison of Population-based Heuristic Approaches - The Evils of Competitive Testing. In *IJCCI 2012 -*

Proceedings of the 4th International Joint Conference on Computational Intelligence, pp. 306–309, Dec 2012. doi: 10.5220/0004168403060309. ISBN 978-989-8565-33-4.

[85] Cuevas, E., Gonzalez, M., Zaldivar, D., Perez-Cisneros, M. and Garcia, G. An algorithm for global optimization inspired by collective animal behavior. *Discrete Dynamics in Nature and Society*. 2012, 2012. doi: 10.1155/2012/638275.

[86] Cuevas, E., Cienfuegos, M., Zaldívar, D. and Perez-Cisneros, M. A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications*. 2013, 40, 16, pp. 6374–6384. doi: 10.1016/j.eswa.2013.05.041.

[87] Cuevas, E., Echavarría, A. and Ramírez-Ortegón, M. An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation. *Applied Intelligence*. 2014, 40, 2, pp. 256–272. doi: 10.1007/s10489-013-0458-0.

[88] Cui, X., Gao, J. and Potok, T. A flocking based algorithm for document clustering analysis. *Journal of Systems Architecture*. 2006, 52, 8–9, pp. 505–515. doi: 10.1016/j.sysarc.2006.02.003.

[89] Cui, Y. H., Guo, R., Rao, R. V. and Savsani, V. J. Harmony element algorithm: A naive initial searching range. In *International Conference on Advances in Mechanical Engineering*, pp. 1–6, 2008.

[90] Dai, C., Zhu, Y. and Chen, W. Seeker optimization algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2007, 4456 LNAI, pp. 167–176. doi: 10.1007/978-3-540-74377-4_18.

[91] Das, S., Biswas, A., Dasgupta, S. and Abraham, A. Bacterial foraging optimization algorithm: Theoretical foundations, analysis, and applications. *Studies in Computational Intelligence*. 2009, 203, pp. 23–55. doi: 10.1007/978-3-642-01085-9_2.

[92] Daskin, A. and Kais, S. Group leaders optimization algorithm. *Molecular Physics*. 2011, 109, 5, pp. 761–772. doi: 10.1080/00268976.2011.552444.

[93] Armas, J., Lalla-Ruiz, E., Tilahun, S. L. and Voss, S. Similarity in metaheuristics: a gentle step towards a comparison methodology. *Natural Computing*. Feb 2021. ISSN 1572-9796. doi: 10.1007/s11047-020-09837-9. Available from: <https://doi.org/10.1007/s11047-020-09837-9>.

[94] De Castro, L. N. and Von Zuben, F. J. The clonal selection algorithm with engineering applications. In *Proceedings of GECCO*, 2000, pp. 36–39, 2000.

[95] De Melo, V. Kaizen programming. In *GECCO '14*, pp. 895–902, 2014. doi: 10.1145/2576768.2598264.

[96] Oliveira, D. R., Lopes, H. S. and Parpinelli, R. S. *Bioluminescent swarm optimization algorithm*. INTECH Open Access Publisher, 2011.

[97] Deb, S., Fong, S. and Tian, Z. Elephant Search Algorithm for optimization problems. In *2015 Tenth International Conference on Digital Information Management (ICDIM)*, pp. 249–255, 2016. doi: 10.1109/ICDIM.2015.7381893.

[98] Dediu, A.-H., Lozano, M. and Martin-Vide, C. *Theory and Practice of Natural Computing: Third International Conference, TPNC 2014, Granada, Spain, December 9-11, 2014. Proceedings*. 8890. Springer, 2014.

[99] Del Acebo, E. and De La Rosa, J. Introducing bar systems: A class of swarm intelligence optimization algorithms. In *AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications*, pp. 18–23, 2008.

[100] Del Ser, J., Osaba, E., Molina, D., Yang, X.-S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P. N., Coello, C. A. C. and Herrera, F. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*. 2019, 48, pp. 220–250.

[101] Delahaye, D., Chaimatanan, S. and Mongeau, M. *Simulated Annealing: From Basics to Applications*, pp. 1–35. International Series

in Operations Research & Management Science. Springer International Publishing, 2019. doi: 10.1007/978-3-319-91086-4_1. Available from: <https://doi.org/10.1007/978-3-319-91086-4_1>. ISBN 978-3-319-91086-4.

[102] DHAHRI, H. and ALIMI, A. M. Opposition-based particle swarm optimization for the design of beta basis function neural network. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Jul 2010. doi: 10.1109/IJCNN.2010.5596501.

[103] DHIMAN, G. and KUMAR, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*. 2017, 114, pp. 48–70. doi: 10.1016/j.advengsoft.2017.05.014.

[104] DOŁAN, B. and ÖLMEZ, T. A new metaheuristic for numerical function optimization: Vortex Search algorithm. *Information Sciences*. 2015, 293, pp. 125–145. doi: 10.1016/j.ins.2014.08.053.

[105] DORIGO, M. Optimization, Learning and Natural Algorithms. *PhD Thesis, Politecnico di Milano*. 1992. Available from: <https://ci.nii.ac.jp/naid/10000136323/>.

[106] DORIGO, M. and DI CARO, G. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 2, pp. 1470–1477 Vol. 2, Jul 1999. doi: 10.1109/CEC.1999.782657.

[107] DORIGO, M., MANIEZZO, V. and COLORNI, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. 1996, 26, 1, pp. 29–41. doi: 10.1109/3477.484436.

[108] DRAGOI, E.-N. and DAFINESCU, V. Parameter control and hybridization techniques in differential evolution: a survey. *Artificial Intelligence Review*. 2016, 45, 4, pp. 447–470.

[109] Drias, H., Sadeg, S. and Yahi, S. Cooperative bees Swarm for solving the maximum weighted satisfiability problem. In *Computational Intelligence and Bioinspired Systems*, 3512, pp. 318–325, 2005. doi: 10.1007/11494669_39.

[110] Du, H., Wu, X. and Zhuang, J. Small-world optimization algorithm for function optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2006, 4222 LNCS-II, pp. 264–273. doi: 10.1007/11881223_33.

[111] Duan, H. and Qiao, P. Pigeon-inspired optimization: A new swarm intelligence optimizer for air robot path planning. *International Journal of Intelligent Computing and Cybernetics*. 2014, 7, 1, pp. 24–37. doi: 10.1108/IJICC-02-2014-0005.

[112] Dueck, G. New optimization heuristics; The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*. 1993, 104, 1, pp. 86–92. doi: 10.1006/jcph.1993.1010.

[113] Duman, E., Uysal, M. and Alkaya, A. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*. 2012, 217, pp. 65–77. doi: 10.1016/j.ins.2012.06.032.

[114] Dziwinski, P. and Bartczuk, L. A New Hybrid Particle Swarm Optimization and Genetic Algorithm Method Controlled by Fuzzy Logic. *IEEE Transactions on Fuzzy Systems*. 2020, 28, 6, pp. 1140–1154. doi: 10.1109/TFUZZ.2019.2957263.

[115] Eberhart, R. and Kennedy, J. New optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.

[116] Ebrahimi, A. and Khamehchi, E. Sperm whale algorithm: An effective metaheuristic algorithm for production optimization problems. *Journal of Natural Gas Science and Engineering*. 2016, 29, pp. 211–222. doi: 10.1016/j.jngse.2016.01.001.

[117] Eesa, A., Brifcani, A. and Orman, Z. Cuttlefish Algorithm – A Novel Bio-Inspired Optimization Algorithm. Available from: https://www.ijser.org/paper/Cuttlefish-Algorithm-A-Novel-Bio-Inspired-Optimization-Algorithm.html, 2014. ISSN 22295518.

[118] Eesa, A., Orman, Z. and Brifcani, A. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Systems with Applications*. 2015, 42, 5, pp. 2670–2679. doi: 10.1016/j.eswa.2014.11.009.

[119] Eiben, A. E., Hinterding, R. and Michalewicz, Z. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*. Jul 1999, 3, 2, pp. 124–141. ISSN 1941-0026. doi: 10.1109/4235.771166.

[120] Eita, M. and Fahmy, M. Group counseling optimization. *Applied Soft Computing Journal*. 2014, 22, pp. 585–604. doi: 10.1016/j.asoc.2014.03.043.

[121] El-Abd, M. Global-best brain storm optimization algorithm. *Swarm and Evolutionary Computation*. 2017, 37, pp. 27–44. doi: 10.1016/j.swevo.2017.05.001.

[122] El-Dosuky, M., El-Bassiouny, A., Hamza, T. and Rashad, M. New hoopoe heuristic optimization. *arXiv preprint arXiv:1211.6410*. 2012.

[123] Engelbrecht, A. P. Fitness function evaluations: A fair stopping condition? In *2014 IEEE Symposium on Swarm Intelligence*, pp. 1–8, Dec 2014. doi: 10.1109/SIS.2014.7011793.

[124] Erol, O. and Eksin, I. A new optimization method: Big Bang-Big Crunch. *Advances in Engineering Software*. 2006, 37, 2, pp. 106–111. doi: 10.1016/j.advengsoft.2005.04.005.

[125] Eskandar, H., Sadollah, A., Bahreininejad, A. and Hamdi, M. Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers and Structures*. 2012, 110–111, pp. 151–166. doi: 10.1016/j.compstruc.2012.07.010.

[126] Eusuff, M., Lansey, K. and Pasha, F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Engineering Optimization*. 2006, 38, 2, pp. 129–154. doi: 10.1080/03052150500384759.

[127] Fadakar, E. and Ebrahimi, M. A new metaheuristic football game inspired algorithm. In *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pp. 6–11, 2016. doi: 10.1109/CSIEC.2016.7482120.

[128] Fard, A. F. and Hajiaghaei-Keshteli, M. Red Deer Algorithm (RDA); a new optimization algorithm inspired by Red Deers' mating. In *International conference on industrial engineering, IEEE.,(2016 e)*, pp. 33–34, 2016.

[129] Faris, H., Aljarah, I., Mirjalili, S., Castillo, P. A. and Merelo, J. J. EvoloPy: An Open-source Nature-inspired Optimization Framework in Python:. In *Proceedings of the 8th International Joint Conference on Computational Intelligence*, pp. 171–177. SCITEPRESS - Science and Technology Publications, 2016. doi: 10.5220/0006048201710177. Available from: <https://www.scitepress.org/Papers/2016/60482/60482.pdf>. ISBN 978-989-758-201-1.

[130] Fathollahi-Fard, A., Hajiaghaei-Keshteli, M. and Tavakkoli-Moghaddam, R. The Social Engineering Optimizer (SEO). *Engineering Applications of Artificial Intelligence*. 2018, 72, pp. 267–293. doi: 10.1016/j.engappai.2018.04.009.

[131] Fathollahi-Fard, A., Hajiaghaei-Keshteli, M. and Tavakkoli-Moghaddam, R. Red deer algorithm (RDA): a new nature-inspired meta-heuristic. *Soft Computing*. 2020, 24, 19, pp. 14637–14665. doi: 10.1007/s00500-020-04812-z.

[132] Felipe, C. and Claus, A. *EC Bestiary: A bestiary of evolutionary, swarm and other metaphor-based algorithms*. Zenodo, Jun 2018. doi: 10.5281/zenodo.1293352. Available from: <https://doi.org/10.5281/zenodo.1293352>.

[133] Felipe, D., Goldbarg, E. and Goldbarg, M. Scientific algorithms for the Car Renter Salesman Problem. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 873–879, 2014. doi: 10.1109/CEC.2014.6900556.

[134] Feng, X., Ma, M. and Yu, H. Crystal energy optimization algorithm. *Computational Intelligence*. 2016, 32, 2, pp. 284–322. doi: 10.1111/coin.12053.

[135] Ferreira, C. *Gene expression programming in problem solving*, pp. 635–653. Springer, 2002.

[136] Filho, C., Neto, F., Lins, A., Nascimento, A. and Lima, M. A novel search algorithm based on fish school behavior. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2646–2651, 2008. doi: 10.1109/ICSMC.2008.4811695.

[137] Finck, S., Hansen, N., Ros, R. and Auger, A. *Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noisy Functions Contents*. Citeseer, 2009.

[138] Fister Jr, I., Yang, X.-S., Fister, I. and Brest, J. Memetic firefly algorithm for combinatorial optimization. *arXiv:1204.5165 [math]*. May 2012. Available from: `<http://arxiv.org/abs/1204.5165>`. arXiv: 1204.5165.

[139] Fister Jr., I., Yang, X.-S., Fister, I., Brest, J. and Fister, D. A Brief Review of Nature-Inspired Algorithms for Optimization. *arXiv:1307.4186 [cs]*. Jul 2013. Available from: `<http://arxiv.org/abs/1307.4186>`. arXiv: 1307.4186.

[140] Flores, J., Lopez, R. and Barrera, J. Gravitational interactions optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2011, 6683 LNCS, pp. 226–237. doi: 10.1007/978-3-642-25566-3_17.

[141] Formato, R. Central force optimization: A new nature inspired computational framework for multidimensional search and optimization. *Stud-*

ies in Computational Intelligence. 2008, 129, pp. 221–238. doi: 10.1007/978-3-540-78987-1_21.

[142] GANDOMI, A. Interior search algorithm (ISA): A novel approach for global optimization. *ISA Transactions*. 2014, 53, 4, pp. 1168–1183. doi: 10.1016/j.isatra.2014.03.018.

[143] GANDOMI, A. and ALAVI, A. Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*. 2012, 17, 12, pp. 4831–4845. doi: 10.1016/j.cnsns.2012.05.010.

[144] GAO, G.-G. W. X.-Z., ZENGER, K. and COELHO, L. d. S. A novel metaheuristic algorithm inspired by rhino herd behavior. *Proceedings of the 9th EUROSIM congress on modelling and simulation, EUROSIM 2016, the 57th SIMS conference on simulation and modelling SIMS 2016*. 2018.

[145] GHAEMI, M. and FEIZI-DERAKHSHI, M.-R. Forest optimization algorithm. *Expert Systems with Applications*. 2014, 41, 15, pp. 6676–6687. doi: 10.1016/j.eswa.2014.05.009.

[146] GHERAIBIA, Y. and MOUSSAOUI, A. Penguins Search Optimization Algorithm (PeSOA). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2013, 7906 LNAI, pp. 222–231. doi: 10.1007/978-3-642-38577-3_23.

[147] GHORBANI, N. and BABAEI, E. Exchange market algorithm. *Applied Soft Computing Journal*. 2014, 19, pp. 177–187. doi: 10.1016/j.asoc.2014.02.006.

[148] GOGNA, A. and TAYAL, A. Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*. 2013, 25, 4, pp. 503–526.

[149] GONÇALVES, M., LOPEZ, R. and MIGUEL, L. Search group algorithm: A new metaheuristic method for the optimization of truss structures. *Computers and Structures*. 2015, 153, pp. 165–184. doi: 10.1016/j.compstruc.2015.03.003.

[150] Gonzalez-Fernandez, Y. and Chen, S. Leaders and followers-A new metaheuristic to avoid the bias of accumulated information. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 776–783, 2015. doi: 10.1109/CEC.2015.7256970.

[151] Greensmith, J., Aickelin, U. and Cayzer, S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In *Artificial Immune Systems*, 3627, pp. 153–167, 2005. doi: 10.1007/11536444_12.

[152] Grefenstette, J. J. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*. Jan 1986, 16, 1, pp. 122–128. ISSN 2168-2909. doi: 10.1109/TSMC.1986.289288.

[153] Guo, P. and Zhu, L. Ant colony optimization for continuous domains. In *2012 8th International Conference on Natural Computation*, pp. 758–762. IEEE, May 2012. doi: 10.1109/ICNC.2012.6234538. Available from: `<http://ieeexplore.ieee.org/document/6234538/>`. ISBN 978-1-4577-2133-5.

[154] Hajiaghaei-Keshteli, M. and Aminnayeri, M. Solving the integrated scheduling of production and rail transportation problem by Keshtel algorithm. *Applied Soft Computing Journal*. 2014, 25, pp. 184–203. doi: 10.1016/j.asoc.2014.09.034.

[155] Hamadneh, N., Tilahun, S., Sathasivam, S. and Choon, O. Prey-Predator algorithm as a new optimization technique using in radial basis function neural networks. *Research Journal of Applied Sciences*. 2013, 8, 7, pp. 383–387. doi: 10.3923/rjasci.2013.383.387.

[156] Harrison, K. R., Ombuki-Berman, B. M. and Engelbrecht, A. P. Optimal parameter regions for particle swarm optimization algorithms. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 349–356, Jun 2017. doi: 10.1109/CEC.2017.7969333.

[157] Hasançebi, O. and Kazemzadeh Azad, S. An efficient metaheuristic algorithm for engineering optimization: SOPT. *Iran University of Science & Technology*. 2012, 2, pp. 479–487.

[158] Hatamlou, A. Heart: A novel optimization algorithm for cluster analysis. *Progress in Artificial Intelligence*. 2014, 2, 2–3, pp. 167–173. doi: 10.1007/s13748-014-0046-5.

[159] Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*. Feb 2013, 222, pp. 175–184. ISSN 0020-0255. doi: 10.1016/j.ins.2012.08.023.

[160] Havens, T., Spain, C., Salmon, N. and Keller, J. Roach infestation optimization. In *2008 IEEE Swarm Intelligence Symposium*, 2008. doi: 10.1109/SIS.2008.4668317.

[161] He, Q. and Han, C. An Improved Particle Swarm Optimization Algorithm with Disturbance Term. In Huang, D.-S., Li, K. and Irwin, G. W. (Ed.) *Computational Intelligence and Bioinformatics*, Lecture Notes in Computer Science, pp. 100–108. Springer, 2006. doi: 10.1007/11816102_11. ISBN 978-3-540-37282-0.

[162] He, S., Wu, Q. H., Wen, J. Y., Saunders, J. R. and Paton, R. C. A particle swarm optimizer with passive congregation. *Biosystems*. Dec 2004, 78, 1, pp. 135–147. ISSN 0303-2647. doi: 10.1016/j.biosystems.2004.08.003.

[163] He, S., Wu, Q. and Saunders, J. Group search optimizer: An optimization algorithm inspired by animal searching behavior. *IEEE Transactions on Evolutionary Computation*. 2009, 13, 5, pp. 973–990. doi: 10.1109/TEVC.2009.2011992.

[164] Hedayatzadeh, R., Salmassi, F., Keshtgari, M., Akbari, R. and Ziarati, K. Termite colony optimization: A novel approach for optimizing continuous problems. In *2010 18th Iranian Conference on Electrical Engineering*, pp. 553–558, 2010. doi: 10.1109/IRANIANCEE.2010.5507009.

[165] Heidari, A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. and Chen, H. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*. 2019, 97, pp. 849–872. doi: 10.1016/j.future.2019.02.028.

[166] Holland, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. A Bradford Book, reprint edition edition, Apr 1992. ISBN 978-0-262-58111-0.

[167] Hooker, J. N. Testing heuristics: We have it all wrong. *Journal of heuristics*. 1995, 1, 1, pp. 33–42.

[168] Hosseini, E. Laying chicken algorithm: A new meta-heuristic approach to solve continuous programming problems. *J Appl Computat Math*. 2017, 6, 344, pp. 2.

[169] Hsiao, Y.-T., Chuang, C.-L., Jiang, J.-A. and Chien, C.-C. A novel optimization algorithm: Space gravitational optimization. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, 3, pp. 2323–2328, 2005.

[170] Hu, T. C., Kahng, A. B. and Tsao, C.-W. A. Old bachelor acceptance: A new class of non-monotone threshold accepting methods. *ORSA Journal on Computing*. 1995, 7, 4, pp. 417–425.

[171] Huan, T., Kulkarni, A., Kanesan, J., Huang, C. and Abraham, A. Ideology algorithm: a socio-inspired optimization methodology. *Neural Computing and Applications*. 2017, 28, pp. 845–876. doi: 10.1007/s00521-016-2379-4.

[172] Huang, C., Li, Y. and Yao, X. A Survey of Automatic Parameter Tuning Methods for Metaheuristics. *IEEE Transactions on Evolutionary Computation*. Apr 2020, 24, 2, pp. 201–216. ISSN 1941-0026. doi: 10.1109/TEVC.2019.2921598.

[173] Huang, G. Artificial infectious disease optimization: A SEIQR epidemic dynamic model-based function optimization algorithm. *Swarm and Evolutionary Computation*. 2016, 27, pp. 31–67. doi: 10.1016/j.swevo.2015.09.007.

[174] Husseinzadeh Kashan, A. League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships. *Applied Soft Computing Journal*. 2014, 16, pp. 171–200. doi: 10.1016/j.asoc.2013.12.005.

[175] Husseinzadeh Kashan, A. A new metaheuristic for optimization: Optics inspired optimization (OIO). *Computers and Operations Research*. 2015, 55, pp. 99–125. doi: 10.1016/j.cor.2014.10.011.

[176] Häckel, S. and Dippold, P. The Bee Colony-inspired Algorithm (BCiA): A two-stage approach for solving the vehicle routing problem with time windows. In *GECCO '09*, pp. 25–32, 2009. doi: 10.1145/1569901.1569906.

[177] Ibrahim, M. K. and Ali, R. S. Novel optimization algorithm inspired by camel traveling behavior. *Iraqi Journal for Electrical And Electronic Engineering*. 2016, 12, 2, pp. 167–177.

[178] Ince, D. C., Hatton, L. and Graham-Cumming, J. The case for open computer programs. *Nature*. 2012, 482, 7386, pp. 485–488.

[179] Irizarry, R. A generalized framework for solving dynamic optimization problems using the artificial chemical process paradigm: Applications to particulate processes and discrete dynamic systems. *Chemical Engineering Science*. Nov 2005, 60, 21, pp. 5663–5681. ISSN 0009-2509. doi: 10.1016/j.ces.2005.05.028.

[180] Islam, M. R., Saifullah, C. M. K. and Mahmud, M. R. Chemical reaction optimization: survey on variants. *Evolutionary Intelligence*. Sep 2019, 12, 3, pp. 395–420. ISSN 1864-5917. doi: 10.1007/s12065-019-00246-1.

[181] Jacovi, M. The shark-search algorithm. An application: Tailored Web site mapping. *Computer Networks*. 1998, 30, 1–7, pp. 317–326.

[182] Jadbabaie, A., Lin, J. and Morse, A. S. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*. 2003, 48, pp. 988–1001.

[183] Jaderyan, M. and Khotanlou, H. Virulence Optimization Algorithm. *Applied Soft Computing Journal*. 2016, 43, pp. 596–618. doi: 10.1016/j.asoc.2016.02.038.

[184] Jadon, S. S., Tiwari, R., Sharma, H. and Bansal, J. C. Hybrid artificial bee colony algorithm with differential evolution. *Applied Soft Computing*. 2017, 58, pp. 11–24.

[185] Jahani, E. and Chizari, M. Tackling global optimization problems with a novel algorithm – Mouth Brooding Fish algorithm. *Applied Soft Computing Journal*. 2018, 62, pp. 987–1002. doi: 10.1016/j.asoc.2017.09.035.

[186] Jain, M., Singh, V. and Rani, A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation*. 2019, 44, pp. 148–175. doi: 10.1016/j.swevo.2018.02.013.

[187] Jamous, R., Tharwat, A. and Bayoum, B. I. Modifications of Particle Swarm Optimization Techniques and Its Application on Stock Market: A Survey. *Int J Adv Comput Sci Appl*. 2015. doi: 10.14569/IJACSA.2015.060315.

[188] Jau, Y.-M., Su, K.-L., Wu, C.-J. and Jeng, J.-T. Modified quantum-behaved particle swarm optimization for parameters estimation of generalized nonlinear multi-regressions model based on Choquet integral with outliers. *Applied Mathematics and Computation*. Sep 2013, 221, pp. 282–295. ISSN 0096-3003. doi: 10.1016/j.amc.2013.06.050.

[189] Javidy, B., Hatamlou, A. and Mirjalili, S. Ions motion algorithm for solving optimization problems. *Applied Soft Computing Journal*. 2015, 32, pp. 72–79. doi: 10.1016/j.asoc.2015.03.035.

[190] Jiang, Q., Wang, L., Hei, X., Fei, R., Yang, D., Zou, F., Li, H., Cao, Z. and Lin, Y. Optimal approximation of stable linear systems with a

novel and efficient optimization algorithm. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 840–844, 2014. doi: 10.1109/CEC.2014.6900366.

[191] Jin, G.-G. and Tran, T.-D. A nature-inspired evolutionary algorithm based on spiral movements. In *Proceedings of SICE Annual Conference 2010*, pp. 1643–1647, 2010.

[192] Jin, X. and Reynolds, R. Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: A cultural algorithm approach. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 3, pp. 1672–1678, 1999. doi: 10.1109/CEC.1999.785475.

[193] Juang, Y.-T., Tung, S.-L. and Chiu, H.-C. Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions. *Information Sciences*. Oct 2011, 181, 20, pp. 4539–4549. ISSN 0020-0255. doi: 10.1016/j.ins.2010.11.025.

[194] Juarez, J. R. C., Wang, H.-J., Lai, Y.-C. and Liang, Y.-C. Virus optimization algorithm (VOA): A novel metaheuristic for solving continuous optimization problems. In *Proceedings of the 2009 Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS 2009)*, pp. 2166–2174, 2009.

[195] Jung, S. Queen-bee evolution for genetic algorithms. *Electronics Letters*. 2003, 39, 6, pp. 575–576. doi: 10.1049/el:20030383.

[196] Kadavy, T., Pluhacek, M., Viktorin, A. and Senkerik, R. Partial population restart of firefly algorithm using complex network analysis. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, Nov 2017. doi: 10.1109/SSCI.2017.8285418.

[197] Kadavy, T., Pluhacek, M., Viktorin, A. and Senkerik, R. Multi-swarm Optimization Algorithm Based on Firefly and Particle Swarm Optimization Techniques. In *International Conference on Artificial Intelligence and Soft Computing*, pp. 405–416. Springer, 2018.

[198] Kallioras, N., Lagaros, N. and Avtzis, D. Pity beetle algorithm – A new metaheuristic inspired by the behavior of bark beetles. *Advances in Engineering Software*. 2018, 121, pp. 147–166. doi: 10.1016/j.advengsoft.2018.04.007.

[199] Kallrath, J. and Milone, E. F. *Brief Review of Mathematical Optimization*, 2009, pp. 351–368. Springer, 2009 edition, 2009. doi: 10.1007/978-1-4419-0699-1. ISBN ISBN: 9781441906991.

[200] Karaboga, D. An idea based on honey bee swarm for numerical optimization. Technical report, Technical report-tr06, Erciyes university, Engineering Faculty, 2005.

[201] Karci, A. Theory of saplings growing up algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2007, 4431 LNCS, PART 1, pp. 450–460. doi: 10.1007/978-3-540-71618-1_50.

[202] Kaveh, A. and Bakhshpoori, T. Water Evaporation Optimization: A novel physically inspired optimization algorithm. *Computers and Structures*. 2016, 167, pp. 69–85. doi: 10.1016/j.compstruc.2016.01.008.

[203] Kaveh, A. and Dadras, A. A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software*. 2017, 110, pp. 69–84. doi: 10.1016/j.advengsoft.2017.03.014.

[204] Kaveh, A. and Farhoudi, N. A new optimization method: Dolphin echolocation. *Advances in Engineering Software*. 2013, 59, pp. 53–70. doi: 10.1016/j.advengsoft.2013.03.004.

[205] Kaveh, A. and Ilchi Ghazaan, M. Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints. *Acta Mechanica*. 2017, 228, 1, pp. 307–322. doi: 10.1007/s00707-016-1725-z.

[206] Kaveh, A. and Khayatazad, M. A new meta-heuristic method: Ray Optimization. *Computers and Structures*. 2012, 112–113, pp. 283–294. doi: 10.1016/j.compstruc.2012.09.003.

[207] Kaveh, A. and Mahdavi, V. Colliding bodies optimization: A novel meta-heuristic method. *Computers & Structures*. 2014. doi: 10.1016/J.COMPSTRUC.2014.04.005.

[208] Kaveh, A. and Talatahari, S. A novel heuristic optimization method: charged system search. *Acta Mechanica*. Sep 2010, 213, 3, pp. 267–289. ISSN 1619-6937. doi: 10.1007/s00707-009-0270-4.

[209] Kaveh, A. and Zolghadr, A. A novel meta-heuristic algorithm: tug of war optimization. *Iran University of Science & Technology*. 2016, 6, 4, pp. 469–492.

[210] Kazikova, A., Pluhacek, M. and Senkerik, R. Tuning of the Bison Algorithm Control Parameters. In Noelle, L., Burger, A., Tholen, C., Werner, J. and Wellhausen, J. (Ed.) *32ND European Conference on Modelling and Simulation (ECMS 2018)*, pp. 156–162, 2018. ISBN 978-0-9932440-7-0.

[211] Kazikova, A., Pluhacek, M. and Senkerik, R. Regarding the Behavior of Bison Runners Within the Bison Algorithm. *MENDEL*. Jun. 2018, 24, 1, pp. 63–70. doi: 10.13164/mendel.2018.1.063. Available from: <https://mendel-journal.org/index.php/mendel/article/view/24>.

[212] Kazikova, A., Pluhacek, M., Viktorin, A. and Senkerik, R. New Running Technique for the Bison Algorithm. In Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R. and Zurada, J. M. (Ed.) *Artificial Intelligence and Soft Computing*, Lecture Notes in Computer Science, pp. 417–426. Springer International Publishing, 2018. doi: 10.1007/978-3-319-91253-0_39. ISBN 978-3-319-91253-0.

[213] Kazikova, A., Oplatkova, Z. K., Pluhacek, M. and Senkerik, R. Border Strategies of the Bison Algorithm. In Iacono, M., Palmieri, F., Gribaudo, M. and Ficco, M. (Ed.) *Proceedings of the 33rd International ECMS Conference on Modelling and Simulation (ECMS 2019)*, 33 / *Communications of the ECMS*, pp. 43–49, 2019. ISBN 978-3-937436-65-4.

[214] Kazikova, A., Pluhacek, M., Senkerik, R. and Viktorin, A. Proposal of a New Swarm Optimization Method Inspired in Bison Behavior. In Matoušek, R. (Ed.) *Recent Advances in Soft Computing*, Advances in Intelligent Systems and Computing, pp. 146–156. Springer International Publishing, 2019. doi: 10.1007/978-3-319-97888-8_13. ISBN 978-3-319-97888-8.

[215] Kazikova, A., Łapa, K., Pluhacek, M. and Senkerik, R. Cascade PID Controller Optimization Using Bison Algorithm. In Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R. and Zurada, J. M. (Ed.) *Artificial Intelligence and Soft Computing*, Lecture Notes in Computer Science, pp. 406–416. Springer International Publishing, 2020. doi: 10.1007/978-3-030-61401-0_38. ISBN 978-3-030-61401-0.

[216] Kazikova, A., Pluhacek, M., Kadavy, T. and Senkerik, R. Introducing the Run Support Strategy for the Bison Algorithm. In Zelinka, I., Brandstetter, P., Trong Dao, T., Hoang Duy, V. and Kim, S. B. (Ed.) *AETA 2018 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*, Lecture Notes in Electrical Engineering, pp. 272–282. Springer International Publishing, 2020. doi: 10.1007/978-3-030-14907-9_27. ISBN 978-3-030-14907-9.

[217] Kazikova, A., Pluhacek, M. and Senkerik, R. Why Tuning the Control Parameters of Metaheuristic Algorithms Is So Important for Fair Comparison? *MENDEL*. Dec 2020, 26, 2, pp. 9–16. doi: 10.13164/mendel.2020.2.009.

[218] Kazikova, A., Pluhacek, M. and Senkerik, R. How Does the Number of Objective Function Evaluations Impact Our Understanding of Metaheuristics Behavior? *IEEE ACCESS*. 2021, 9, pp. 44032–44048. ISSN 2169-3536. doi: 10.1109/ACCESS.2021.3066135.

[219] Kennedy, J. and Eberhart, R. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4, pp. 1942–1948 vol.4, Nov 1995. doi: 10.1109/ICNN.1995.488968.

[220] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. Optimization by Simulated Annealing. *Science*. May 1983, 220, 4598, pp. 671–680. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.220.4598.671.

[221] Klein, C. and Coelho, L. Meerkats-inspired algorithm for global optimization problems. In *ESANN*, pp. 679–684, 2018.

[222] Klein, C., Mariani, V. and Coelho, L. Cheetah based optimization algorithm: A novel swarm intelligence paradigm. In *ESANN 2018*, pp. 685–690, 2018.

[223] Kora, P. and Krishna, K. S. R. Hybrid firefly and particle swarm optimization algorithm for the detection of bundle branch block. *International Journal of the Cardiovascular Academy*. 2016, 2, 1, pp. 44–48.

[224] Krishnanand, K. and Ghose, D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005*, 2005, pp. 84–91, 2005. doi: 10.1109/SIS.2005.1501606.

[225] Kumar, A., Misra, R. and Singh, D. Butterfly optimizer. In *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI)*, 2016. doi: 10.1109/WCI.2015.7495523. Available from: <https://ieeexplore.ieee.org/abstract/document/7495523>.

[226] Kumar, A., Misra, R. K. and Singh, D. Improving the local search capability of Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat Phase. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1835–1842, Jun 2017. doi: 10.1109/CEC.2017.7969524.

[227] Kundu, S. Gravitational clustering: A new approach based on the spatial distribution of the points. *Pattern Recognition*. 1999, 32, 7, pp. 1149–1160. doi: 10.1016/S0031-3203(98)00143-5.

[228] Kuo, R. J. and Hong, C. W. Integration of Genetic Algorithm and Particle Swarm Optimization for Investment Portfolio Optimization. *Applied*

Mathematics & Information Sciences. Nov 2013, 7, 6, pp. 2397–2408. ISSN 1935-0090, 2325-0399. doi: 10.12785/amis/070633.

[229] LAGUNA, M. Editor's Note on the MIC 2013 Special Issue of the Journal of Heuristics (Volume 22, Issue 4, August 2016). *Journal of Heuristics*. Oct 2016, 22, 5, pp. 665–666. ISSN 1572-9397. doi: 10.1007/s10732-016-9318-5.

[230] LAM, A. and LI, V. Chemical-reaction-inspired metaheuristic for optimization. *IEEE Transactions on Evolutionary Computation*. 2010, 14, 3, pp. 381–399. doi: 10.1109/TEVC.2009.2033580.

[231] LAMPINEN, J. and ZELINKA, I. On stagnation of the differential evolution algorithm. In *Proceedings of MENDEL*, pp. 76–83, 2000.

[232] LATORRE, A., MOLINA, D., OSABA, E., DEL SER, J. and HERRERA, F. Fairness in Bio-inspired Optimization Research: A Prescription of Methodological Guidelines for Comparing Meta-heuristics. *arXiv:2004.09969 [cs]*. Apr 2020. Available from: `<http://arxiv.org/abs/2004.09969>`. arXiv: 2004.09969.

[233] LEE, K. and GEEM, Z. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*. 2005, 194, 36–38, pp. 3902–3933. doi: 10.1016/j.cma.2004.09.007.

[234] LENORD MELVIX, J. Greedy Politics Optimization: Metaheuristic inspired by political strategies adopted during state assembly elections. In *2014 IEEE International Advance Computing Conference (IACC)*, pp. 1157–1162, 2014. doi: 10.1109/IAdCC.2014.6779490.

[235] LI, B. and JIANG, W. Optimizing complex functions by chaos search. *Cybernetics and Systems*. 1998, 29, 4, pp. 409–419. doi: 10.1080/019697298125678.

[236] LI, M., ZHAO, H., WENG, X. and HAN, T. Cognitive behavior optimization algorithm for solving optimization problems. *Applied Soft Computing Journal*. 2016, 39, pp. 199–222. doi: 10.1016/j.asoc.2015.11.015.

[237] Li, M., Zhao, H., Weng, X. and Han, T. A novel nature-inspired algorithm for optimization: Virus colony search. *Advances in Engineering Software*. 2016, 92, pp. 65–88. doi: 10.1016/j.advengsoft.2015.11.004.

[238] Liu, C., Yan, X., Liu, C. and Wu, H. The wolf colony algorithm and its application. *Chinese Journal of Electronics*. 2011, 20, 2, pp. 212–216.

[239] Liu, J. and Lampinen, J. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Computing*. Jun 2005, 9, 6, pp. 448–462. ISSN 1433-7479. doi: 10.1007/s00500-004-0363-x.

[240] Liu, Y. and Passino, K. Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors. *Journal of Optimization Theory and Applications*. 2002, 115, 3, pp. 603–628. doi: 10.1023/A:1021207331209.

[241] Liu, Y., Qin, Z., Shi, Z. and Lu, J. Center particle swarm optimization. *Neurocomputing*. Jan 2007, 70, 4, pp. 672–679. ISSN 0925-2312. doi: 10.1016/j.neucom.2006.10.002.

[242] Liu, Y. and Qin, G. A Modified Particle Swarm Optimization Algorithm for Reliability Redundancy Optimization Problem. *Journal of Computers*. Sep 2014, 9, 9, pp. 2124–2131. ISSN 1796-203X. doi: 10.4304/jcp.9.9.2124-2131.

[243] López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M. and Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*. 2016, 3, pp. 43–58.

[244] Luo, F., Zhao, J. and Dong, Z. A new metaheuristic algorithm for real-parameter optimization: Natural aggregation algorithm. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 94–103, 2016. doi: 10.1109/CEC.2016.7743783.

[245] Lučić, P. and Teodorović, D. Transportation modeling: An artificial life approach. *Proceedings of the International Conference on Tools with Artificial Intelligence*. 2002, pp. 216–223. doi: 10.1109/TAI.2002.1180807.

[246] Maca, P. and Pech, P. The Inertia Weight Updating Strategies in Particle Swarm Optimisation Based on the Beta Distribution. *Mathematical Problems in Engineering*. Mar 2015, 2015, pp. 790465. ISSN 1024-123X. doi: 10.1155/2015/790465.

[247] Maciel, O., Valdivia, A., Oliva, D., Cuevas, E., Zaldívar, D. and Perez-Cisneros, M. A novel hybrid metaheuristic optimization method: hypercube natural aggregation algorithm. *Soft Computing*. 2020, 24, 12, pp. 8823–8856. doi: 10.1007/s00500-019-04416-2.

[248] Maia, R., De Castro, L. and Caminhas, W. OptBees - A bee-inspired algorithm for solving continuous optimization problems. In *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, pp. 142–151, 2013. doi: 10.1109/BRICS-CCI-CBIC.2013.33.

[249] Malakooti, B., Kim, H. and Sheikh, S. Bat intelligence search with application to multi-objective multiprocessor scheduling optimization. *International Journal of Advanced Manufacturing Technology*. 2012, 60, 9–12, pp. 1071–1086. doi: 10.1007/s00170-011-3649-z.

[250] Mansouri, T., Farasat, A., Menhaj, M. and Reza Sadeghi Moghadam, M. ARO: A new model free optimization algorithm for real time applications inspired by the asexual reproduction. *Expert Systems with Applications*. 2011, 38, 5, pp. 4866–4874. doi: 10.1016/j.eswa.2010.09.084.

[251] McCaffrey, J. Generation of pairwise test sets using a simulated bee colony algorithm. In *2009 IEEE International Conference on Information Reuse Integration*, pp. 115–119, 2009. doi: 10.1109/IRI.2009.5211598.

[252] Mehrabian, A. and Lucas, C. A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*. 2006, 1, 4, pp. 355–366. doi: 10.1016/j.ecoinf.2006.07.003.

[253] Melin, P., Astudillo, L., Castillo, O., Valdez, F. and Garcia, M. Optimal design of type-2 and type-1 fuzzy tracking controllers for au-

tonomous mobile robots under perturbed torques using a new chemical optimization paradigm. *Expert Systems with Applications*. 2013, 40, 8, pp. 3185–3195. doi: 10.1016/j.eswa.2012.12.032.

[254] Memari, A., Ahmad, R., Akbari Jokar, M. R. and Abdul Rahim, A. R. A New Modified Firefly Algorithm for Optimizing a Supply Chain Network Problem. *Applied Sciences*. Jan 2019, 9, 1, pp. 7. doi: 10.3390/app9010007.

[255] Meng, X., Liu, Y., Gao, X. and Zhang, H. A new bio-inspired algorithm: Chicken swarm optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2014, 8794, pp. 86–94. doi: 10.1007/978-3-319-11857-4_10.

[256] Meng, X.-B., Gao, X., Lu, L., Liu, Y. and Zhang, H. A new bio-inspired optimisation algorithm: Bird Swarm Algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*. 2016, 28, 4, pp. 673–687. doi: 10.1080/0952813X.2015.1042530.

[257] Merrikh-Bayat, F. The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature. *Applied Soft Computing Journal*. 2015, 33, pp. 292–303. doi: 10.1016/j.asoc.2015.04.048.

[258] Merta, J. Hybrid Symbolic Regression with the Bison Seeker Algorithm. *MENDEL*. Jun 2019, 25, 1, pp. 79–86. ISSN 2571-3701. doi: 10.13164/mendel.2019.1.079.

[259] Meyer, J.-A. and Wilson, S. W. Task differentiation in Polistes wasp colonies: a model for self-organizing groups of robots. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. 1991.

[260] Min, H. and Wang, Z. Design and analysis of group escape behavior for distributed autonomous mobile robots. In *2011 IEEE International*

Conference on Robotics and Automation, pp. 6128–6135, 2011. doi: 10.1109/ICRA.2011.5980123.

[261] MINHAS, F. and ARIF, M. MOX: A novel global optimization algorithm inspired from Oviposition site selection and egg hatching inhibition in mosquitoes. *Applied Soft Computing Journal*. 2011, 11, 8, pp. 4614–4625. doi: 10.1016/j.asoc.2011.07.020.

[262] MIRJALILI, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*. 2015, 89, pp. 228–249. doi: 10.1016/j.knosys.2015.07.006.

[263] MIRJALILI, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*. 2016, 96, pp. 120–133. doi: 10.1016/j.knosys.2015.12.022.

[264] MIRJALILI, S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*. 2016, 27, 4, pp. 1053–1073. doi: 10.1007/s00521-015-1920-1.

[265] MIRJALILI, S. and LEWIS, A. The Whale Optimization Algorithm. *Advances in Engineering Software*. 2016, 95, pp. 51–67. doi: 10.1016/j.advengsoft.2016.01.008.

[266] MIRJALILI, S., MIRJALILI, S. and HATAMLOU, A. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications*. 2016, 27, 2, pp. 495–513. doi: 10.1007/s00521-015-1870-7.

[267] MIRJALILI, S., GANDOMI, A., MIRJALILI, S., SAREMI, S., FARIS, H. and MIRJALILI, S. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*. 2017, 114, pp. 163–191. doi: 10.1016/j.advengsoft.2017.07.002.

[268] MIRJALILI, S. The Ant Lion Optimizer. *Advances in Engineering Software*. May 2015, 83, pp. 80–98. ISSN 0965-9978. doi: 10.1016/j.advengsoft.2015.01.010.

[269] Mirjalili, S., Mirjalili, S. M. and Lewis, A. Grey Wolf Optimizer. *Advances in Engineering Software*. Mar 2014, 69, pp. 46–61. ISSN 0965-9978. doi: 10.1016/j.advengsoft.2013.12.007.

[270] Mo, H. and Xu, L. Magnetotactic bacteria optimization algorithm for multimodal optimization. In *2013 IEEE Symposium on Swarm Intelligence (SIS)*, pp. 240–247, 2013. doi: 10.1109/SIS.2013.6615185.

[271] Mo, Y.-b., Ma, Y.-z. and Zheng, Q.-y. Optimal Choice of Parameters for Firefly Algorithm. In *2013 Fourth International Conference on Digital Manufacturing Automation*, pp. 887–892, Jun 2013. doi: 10.1109/ICDMA.2013.210.

[272] Moein, S. and Logeswaran, R. KGMO: A swarm optimization algorithm based on the kinetic energy of gas molecules. *Information Sciences*. 2014, 275, pp. 127–144. doi: 10.1016/j.ins.2014.02.026.

[273] Moez, H., Kaveh, A. and Taghizadieh, N. Natural Forest Regeneration Algorithm: A New Meta-Heuristic. *Iranian Journal of Science and Technology, Transactions of Civil Engineering*. Dec 2016, 40, 4, pp. 311–326. ISSN 2364-1843. doi: 10.1007/s40996-016-0042-z.

[274] Moghdani, R. and Salimifard, K. Volleyball Premier League Algorithm. *Applied Soft Computing Journal*. 2018, 64, pp. 161–185. doi: 10.1016/j.asoc.2017.11.043.

[275] Molina, D., Poyatos, J., Del Ser, J., Garcia, S., Hussain, A. and Herrera, F. Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration versus Algorithmic Behavior, Critical Analysis and Recommendations. *arXiv:2002.08136 [cs]*. Feb 2020. Available from: <http://arxiv.org/abs/2002.08136>. arXiv: 2002.08136.

[276] Monismith Jr., D. and Mayfield, B. Slime mold as a model for numerical optimization. In *2008 IEEE Swarm Intelligence Symposium*, 2008. doi: 10.1109/SIS.2008.4668295.

[277] Montiel, O., Castillo, O., Melin, P., Díaz, A. and Sepúlveda, R. Human evolutionary model: A new approach to optimization. *Information Sciences*. 2007, 177, 10, pp. 2075–2098. doi: 10.1016/j.ins.2006.09.012.

[278] Moosavian, N. and Kasaee Roodsari, B. Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. *Swarm and Evolutionary Computation*. 2014, 17, pp. 14–24. doi: 10.1016/j.swevo.2014.02.002.

[279] Mora-Gutierrez, R., Ramírez-Rodríguez, J. and Rincón-Garcia, E. An optimization algorithm inspired by musical composition. *Artificial Intelligence Review*. 2014, 41, 3, pp. 301–315. doi: 10.1007/s10462-011-9309-8.

[280] Mozaffari, A., Goudarzi, A., Fathi, A. and Samadian, P. Bio-inspired methods for fast and robust arrangement of thermoelectric modulus. *International Journal of Bio-Inspired Computation*. 2013, 5, 1, pp. 19–34. doi: 10.1504/IJBIC.2013.053056.

[281] Mozaffari, A., Fathi, A. and Behzadipour, S. The great salmon run: a novel bio-inspired algorithm for artificial system design and optimisation. *International Journal of Bio-Inspired Computation*. Jan 2012, 4, 5, pp. 286–301. ISSN 1758-0366. doi: 10.1504/IJBIC.2012.049889.

[282] Mucherino, A. and Seref, O. Monkey search: A novel metaheuristic search for global optimization. In *AIP Conference Proceedings*, 953, pp. 162–173, 2007. doi: 10.1063/1.2817338.

[283] Murase, H. Finite element inverse analysis using a photosynthetic algorithm. *Computers and Electronics in Agriculture*. 2000, 29, 1–2, pp. 115–123. doi: 10.1016/S0168-1699(00)00139-3.

[284] Mutazono, A., Sugano, M. and Murata, M. Frog call-inspired self-organizing anti-phase synchronization for wireless sensor networks. In *2009 2nd International Workshop on Nonlinear Dynamics and Synchronization*, pp. 81–88, 2009. doi: 10.1109/inds.2009.5227977.

[285] Muthiah-Nakarajan, V. and Noel, M. Galactic Swarm Optimization: A new global optimization metaheuristic inspired by galactic motion. *Applied Soft Computing Journal*. 2016, 38, pp. 771–787. doi: 10.1016/j.asoc.2015.10.034.

[286] Muñoz, M., López, J. and Caicedo, E. An artificial beehive algorithm for continuous optimization. *International Journal of Intelligent Systems*. 2009, 24, 11, pp. 1080–1093. doi: 10.1002/int.20376.

[287] Müller, S., Marchetto, J., Airaghi, S. and Koumoutsakos, P. Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*. 2002, 6, 1, pp. 16–29. doi: 10.1109/4235.985689.

[288] Naderi, B., Khalili, M. and Khamseh, A. Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. *International Journal of Production Research*. 2014, 52, 9, pp. 2667–2681. doi: 10.1080/00207543.2013.871389.

[289] Nara, K., Takeyama, T. and Kim, H. New evolutionary algorithm based on sheep flocks heredity model and its application to scheduling problem. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, 6, pp. VI–503–VI–508, 1999.

[290] Neri, F. and Tirronen, V. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*. Feb 2010, 33, 1, pp. 61–106. ISSN 1573-7462. doi: 10.1007/s10462-009-9137-2.

[291] Neshat, M., Sepidnam, G. and Sargolzaei, M. Swallow swarm optimization algorithm: A new method to optimization. *Neural Computing and Applications*. 2013, 23, 2, pp. 429–454. doi: 10.1007/s00521-012-0939-9.

[292] Nguyen, H. and Bhanu, B. Zombie Survival Optimization: A swarm intelligence algorithm inspired by zombie foraging. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 987–990, 2012.

[293] Nguyen, T. T., Yang, S. and Branke, J. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*. Oct 2012, 6, pp. 1–24. ISSN 22106502. doi: 10.1016/j.swevo.2012.05.001.

[294] Nicoară, E. S. Mechanisms to Avoid the Premature Convergence of Genetic Algorithms., 2009.

[295] Nishida, T. Y. *Membrane algorithms: approximate algorithms for NP-complete optimization problems*, pp. 303–314. Springer, 2006.

[296] Niu, B. and Wang, H. Bacterial colony optimization: Principles and foundations. *Communications in Computer and Information Science*. 2012, 304 CCIS, pp. 501–506. doi: 10.1007/978-3-642-31837-5_73.

[297] Obagbuwa, I. and Adewumi, A. An improved cockroach swarm optimization. *Scientific World Journal*. 2014, 2014. doi: 10.1155/2014/375358.

[298] Ochoa, P., Castillo, O. and Soria, J. A fuzzy differential evolution method with dynamic adaptation of parameters for the optimization of fuzzy controllers. In *2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW)*, pp. 1–6, Jun 2014. doi: 10.1109/NORBERT.2014.6893893.

[299] Odili, J., Kahar, M. and Anwar, S. African Buffalo Optimization: A Swarm-Intelligence Technique. In *Procedia Computer Science*, 76, pp. 443–448, 2015. doi: 10.1016/j.procs.2015.12.291.

[300] Oftadeh, R., Mahjoob, M. and Shariatpanahi, M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Computers and Mathematics with Applications*. 2010, 60, 7, pp. 2087–2098. doi: 10.1016/j.camwa.2010.07.049.

[301] Oliveto, P. S., Paixão, T., Perez Heredia, J., Sudholt, D. and Trubenova, B. How to Escape Local Optima in Black Box Optimisation: When Non-elitism Outperforms Elitism. *Algorithmica*. 2018, 80, 5, pp. 1604–1633. ISSN 0178-4617. doi: 10.1007/s00453-017-0369-2.

[302] Omidvar, R., Parvin, H. and Rad, F. SSPCO optimization algorithm (See-See Partridge Chicks Optimization). In *2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pp. 101–106, 2016. doi: 10.1109/MICAI.2015.22.

[303] Osaba, E., Diaz, F. and Onieva, E. Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. *Applied Intelligence*. 2014, 41, 1, pp. 145–166. doi: 10.1007/s10489-013-0512-y.

[304] Pan, Q. and He, Y. Recent advances in self-propelled particles. *Science China Chemistry*. Oct 2017, 60, 10, pp. 1293–1304. ISSN 1869-1870. doi: 10.1007/s11426-017-9115-8.

[305] Pan, W.-T. A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems*. 2012, 26, pp. 69–74. doi: 10.1016/j.knosys.2011.07.001.

[306] Parpinelli, R. and Lopes, H. An eco-inspired evolutionary algorithm applied to numerical optimization. In *2011 Third World Congress on Nature and Biologically Inspired Computing*, pp. 466–471, 2011. doi: 10.1109/NaBIC.2011.6089631.

[307] Passino, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*. Jun 2002, 22, 3, pp. 52–67. ISSN 1941-000X. doi: 10.1109/MCS.2002.1004010.

[308] Pattnaik, S., Bakwad, K., Sohi, B., Ratho, R. and Devi, S. Swine Influenza Models Based Optimization (SIMBO). *Applied Soft Computing Journal*. 2013, 13, 1, pp. 628–653. doi: 10.1016/j.asoc.2012.07.010.

[309] Pellerin, E., Pigeon, L. and Delisle, S. Self-adaptive parameters in genetic algorithms. In *Data Mining and Knowledge Discovery: Theory, Tools, and Technology VI*, 5433, pp. 53–64. International Society for Optics and Photonics, Apr 2004. doi: 10.1117/12.542156.

[310] Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S. and Zaidi, M. - *The Bees Algorithm — A Novel Tool for Complex Optimisation*

Problems, pp. 454–459. Elsevier Science Ltd, Jan 2006. doi: 10.1016/B978-008045157-2/50081-X. ISBN 978-0-08-045157-2.

[311] Pham, Q. T. Competitive evolution: A natural approach to operator selection. In Yao, X. (Ed.) *Progress in Evolutionary Computation*, Lecture Notes in Computer Science, pp. 49–60. Springer, 1995. doi: 10.1007/3-540-60154-6_47. ISBN 978-3-540-49528-4.

[312] Pierezan, J., Maidl, G., Massashi Yamao, E., Santos Coelho, L. and Cocco Mariani, V. Cultural coyote optimization algorithm applied to a heavy duty gas turbine operation. *Energy Conversion and Management*. 2019, 199. doi: 10.1016/j.enconman.2019.111932.

[313] Pierezan, J. and Dos Santos Coelho, L. Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, Jul 2018. doi: 10.1109/CEC.2018.8477769.

[314] Pinto, P., Runkler, T. A. and Sousa, J. M. *Wasp swarm optimization of logistic systems*, pp. 264–267. Springer, 2005.

[315] Piotrowski, A. P. and Napiorkowski, J. J. Some metaheuristics should be simplified. *Information Sciences*. Feb 2018, 427, pp. 32–62. ISSN 0020-0255. doi: 10.1016/j.ins.2017.10.039.

[316] Polakova, R., Tvrdík, J., Bujok, P. and Matoušek, R. Population-size adaptation through diversity-control mechanism for differential evolution. In *MENDEL, 22th International Conference on Soft Computing*, pp. 49–56, 2016.

[317] Premaratne, U., Samarabandu, J. and Sidhu, T. A new biologically inspired optimization algorithm. In *2009 International Conference on Industrial and Information Systems (ICIIS)*, pp. 279–284, 2009. doi: 10.1109/ICIINFS.2009.5429852.

[318] Puchinger, J. and Raidl, G. R. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification.

In Mira, J. and alvarez, J. R. (Ed.) *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, Lecture Notes in Computer Science, pp. 41–53. Springer, 2005. doi: 10.1007/11499305_5. ISBN 978-3-540-31673-2.

[319] Punnathanam, V. and Kotecha, P. Yin-Yang-pair Optimization: A novel lightweight optimization algorithm. *Engineering Applications of Artificial Intelligence*. 2016, 54, pp. 62–79. doi: 10.1016/j.engappai.2016.04.004.

[320] Puris, A., Bello, R., Molina, D. and Herrera, F. Variable mesh optimization for continuous optimization problems. *Soft Computing*. 2012, 16, 3, pp. 511–525. doi: 10.1007/s00500-011-0753-9.

[321] Purnomo, H. D. Soccer Game Optimization: Fundamental Concept. *Jurnal Sistem Komputer*. 2014, 4, 1, pp. 25–36.

[322] Quan, H. and Shi, X. A surface-simplex swarm evolution algorithm. *Wuhan University Journal of Natural Sciences*. 2017, 22, 1, pp. 38–50. doi: 10.1007/s11859-017-1214-9.

[323] Quijano, N. and Passino, K. M. Honey Bee Social Foraging Algorithms for Resource Allocation, Part I: Algorithm and Theory. In *2007 American Control Conference*, pp. 3383–3388, Jul 2007. doi: 10.1109/ACC.2007.4282167.

[324] Quijano, N. and Passino, K. Honey bee social foraging algorithms for resource allocation: Theory and application. *Engineering Applications of Artificial Intelligence*. 2010, 23, 6, pp. 845–861. doi: 10.1016/j.engappai.2010.05.004.

[325] Rabanal, P., Rodríguez, I. and Rubio, F. Using river formation dynamics to design heuristic algorithms. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2007, 4618 LNCS, pp. 163–177. doi: 10.1007/978-3-540-73554-0_16.

[326] RAHMANI, R. and YUSOF, R. A new simple, fast and efficient algorithm for global optimization over continuous search-space problems: Radial Movement Optimization. *Applied Mathematics and Computation*. 2014, 248, pp. 287–300. doi: 10.1016/j.amc.2014.09.102.

[327] RAJA, N. S. M., MANIC, K. S. and RAJINIKANTH, V. Firefly algorithm with various randomization parameters: an analysis. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pp. 110–121. Springer, 2013.

[328] RAJABIOUN, R. Cuckoo optimization algorithm. *Applied Soft Computing Journal*. 2011, 11, 8, pp. 5508–5518. doi: 10.1016/j.asoc.2011.05.008.

[329] RAJAKUMAR, B. R. The Lion's Algorithm: A New Nature-Inspired Search Algorithm. *Procedia Technology*. Jan 2012, 6, pp. 126–135. ISSN 2212-0173. doi: 10.1016/j.protcy.2012.10.016.

[330] RAJAKUMAR, B. Lion algorithm for standard and large scale bilinear system identification: A global optimization based on Lion's social behavior. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2116–2123, 2014. doi: 10.1109/CEC.2014.6900561.

[331] RAJAKUMAR, R., DHAVACHELVAN, P. and VENGATTARAMAN, T. A survey on nature inspired meta-heuristic algorithms with its domain specifications. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pp. 1–6, Oct 2016. doi: 10.1109/CESYS.2016.7889811.

[332] RAKHSHANI, H. and RAHATI, A. Snap-drift cuckoo search: A novel cuckoo search optimization algorithm. *Applied Soft Computing Journal*. 2017, 52, pp. 771–794. doi: 10.1016/j.asoc.2016.09.048.

[333] RAMOS, A., PETIT, O., LONGOUR, P., PASQUARETTA, C. and SUEUR, C. Collective decision making during group movements in European bison, Bison bonasus. *Animal Behaviour*. Nov 2015, 109, pp. 149–160. ISSN 0003-3472. doi: 10.1016/j.anbehav.2015.08.016.

[334] RAO, R., SAVSANI, V. and VAKHARIA, D. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization

problems. *CAD Computer Aided Design*. 2011, 43, 3, pp. 303–315. doi: 10.1016/j.cad.2010.12.015.

[335] Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. GSA: A Gravitational Search Algorithm. *Information Sciences*. 2009, 179, 13, pp. 2232–2248. doi: 10.1016/j.ins.2009.03.004.

[336] Ray, T. and Liew, K. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*. 2003, 7, 4, pp. 386–396. doi: 10.1109/TEVC.2003.814902.

[337] Razmjooy, N., Khalilpour, M. and Ramezani, M. A New Meta-Heuristic Optimization Algorithm Inspired by FIFA World Cup Competitions: Theory and Its Application in PID Designing for AVR System. *Journal of Control, Automation and Electrical Systems*. Aug 2016, 27, 4, pp. 419–440. ISSN 2195-3899. doi: 10.1007/s40313-016-0242-6.

[338] Rbouh, I. and Imrani, A. A. E. Hurricane-based Optimization Algorithm. *AASRI Procedia*. Jan 2014, 6, pp. 26–33. ISSN 2212-6716. doi: 10.1016/j.aasri.2014.05.005.

[339] Rechenberg, I. *Evolutionsstrategie—Optimierung technischer Systeme nach Prinzipien der biologischen Information*. Fromman Verlag, Freiburg, Germany, 1973.

[340] Rechenberg, I. *Evolutionsstrategien*, pp. 83–114. Springer, 1978.

[341] Reddy K, S., Panwar, L., Panigrahi, B. and Kumar, R. Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets. *Engineering Optimization*. 2019, 51, 3, pp. 369–389. doi: 10.1080/0305215X.2018.1463527.

[342] Reynolds, C. W. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*. Aug 1987, 21, 4, pp. 25–34. ISSN 0097-8930. doi: 10.1145/37402.37406.

[343] Rohlfshagen, P. and Yao, X. Attributes of Dynamic Combinatorial Optimisation. In Li, X. et al. (Ed.) *Simulated Evolution and Learning*, Lecture Notes in Computer Science, pp. 442–451. Springer, 2008. doi: 10.1007/978-3-540-89694-4_45. ISBN 978-3-540-89694-4.

[344] Ruiz, R. Heuristic Scheduling: Running away from the bio-inspired tsunami. Available from: https://github.com/fcampelo/EC-Bestiary/blob/master/bestiary_references/Presentation_RRuiz.pdf, 2012.

[345] Sacco, W., Filho, H. and De Oliveira, C. A populational particle collision algorithm applied to a nuclear reactor core design optimization. In *Joint International Topical Meeting on Mathematics and Computations for Supercomputing in Nuclear Applications, M&C+ SNA 2007 (CD-ROM)*, 2007.

[346] Sadollah, A., Bahreininejad, A., Eskandar, H. and Hamdi, M. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing Journal*. 2013, 13, 5, pp. 2592–2612. doi: 10.1016/j.asoc.2012.11.026.

[347] Salcedo-Sanz, S., Del Ser, J., Landa-Torres, I., Gil-López, S. and Portilla-Figueras, J. The Coral Reefs Optimization Algorithm: A Novel Metaheuristic for Efficiently Solving Optimization Problems. *Scientific World Journal*. 2014, 2014. doi: 10.1155/2014/739768.

[348] Salgotra, R. and Singh, U. The naked mole-rat algorithm. *Neural Computing and Applications*. 2019, 31, 12, pp. 8837–8857. doi: 10.1007/s00521-019-04464-7.

[349] Salih, S. and Alsewari, A. A new algorithm for normal and large-scale optimization problems: Nomadic People Optimizer. *Neural Computing and Applications*. 2020, 32, 14, pp. 10359–10386. doi: 10.1007/s00521-019-04575-1.

[350] Salimi, H. Stochastic Fractal Search: A powerful metaheuristic algorithm. *Knowledge-Based Systems*. 2015, 75, pp. 1–18. doi: 10.1016/j.knosys.2014.07.025.

[351] Samareh Moosavi, S. and Khatibi Bardsiri, V. Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. *Engineering Applications of Artificial Intelligence*. 2017, 60, pp. 1–15. doi: 10.1016/j.engappai.2017.01.006.

[352] Saremi, S., Mirjalili, S. and Lewis, A. Grasshopper Optimisation Algorithm: Theory and application. *Advances in Engineering Software*. 2017, 105, pp. 30–47. doi: 10.1016/j.advengsoft.2017.01.004.

[353] Sato, T. and Hagiwara, M. Bee System: Finding solution by a concentrated search. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 4, pp. 3954–3959, 1997.

[354] Savsani, P. and Savsani, V. Passing vehicle search (PVS): A novel metaheuristic algorithm. *Applied Mathematical Modelling*. 2016, 40, 5–6, pp. 3951–3978. doi: 10.1016/j.apm.2015.10.040.

[355] Sayed, G., Tharwat, A. and Hassanien, A. Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection. *Applied Intelligence*. 2019, 49, 1, pp. 188–205. doi: 10.1007/s10489-018-1261-8.

[356] Schwefel, H.-P. Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. *Diploma thesis, Technical Univ. of Berlin*. 1965.

[357] Shah-Hosseini, H. The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation*. 2009, 1, 1–2, pp. 71–79. doi: 10.1504/IJBIC.2009.022775.

[358] Shah-Hosseini, H. Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation. *In-*

ternational Journal of Computational Science and Engineering. 2011, 6, 1–2, pp. 132–140. doi: 10.1504/IJCSE.2011.0412021.

[359] Sharafi, Y., Khanesar, M. and Teshnehlab, M. COOA: Competitive optimization algorithm. *Swarm and Evolutionary Computation*. 2016, 30, pp. 39–63. doi: 10.1016/j.swevo.2016.04.002.

[360] Shareef, H., Ibrahim, A. and Mutlag, A. Lightning search algorithm. *Applied Soft Computing Journal*. 2015, 36, pp. 315–333. doi: 10.1016/j.asoc.2015.07.028.

[361] Sharma, A. A new optimizing algorithm using reincarnation concept. In *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 281–288, Nov 2010. doi: 10.1109/CINTI.2010.5672231.

[362] Shayeghi, H. and Dadashpour, J. Anarchic society optimization based PID control of an automatic voltage regulator (AVR) system. *Electrical and Electronic Engineering*. 2012, 2, 4, pp. 199–207.

[363] Shen, J. and Li, J. The principle analysis of light ray optimization algorithm. In *2010 Second International Conference on Computational Intelligence and Natural Computing*, 2, pp. 154–157, 2010. doi: 10.1109/CINC.2010.5643764.

[364] Shi, Y. Brain storm optimization algorithm. In *Proceedings of the Second international conference on Advances in swarm intelligence - Volume Part I*, ICSI'11, pp. 303–309. Springer-Verlag, Jun 2011. ISBN 978-3-642-21514-8.

[365] Simon, D. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*. 2008, 12, 6, pp. 702–713. doi: 10.1109/TEVC.2008.919004.

[366] Smit, S. K. *Parameter tuning and scientific testing in evolutionary algorithms*. Vrije Universiteit, 2012.

[367] Sörensen, K. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*. 2015, 22, 1, pp. 3–18.

[368] Sörensen, K. Metaphors and metaheuristics: A match made in hell. Available from: https://github.com/fcampelo/EC-Bestiary/blob/master/bestiary_references/Sorensen2018%20-%20plenary%20slides.pdf, 2018.

[369] Sörensen, K. and Glover, F. Metaheuristics. *Encyclopedia of operations research and management science*. 2013, 62, pp. 960–970.

[370] Storn, R. and Price, K. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*. 1997, 11, 4, pp. 341–359. doi: 10.1023/A:1008202821328.

[371] Storn, R., Price, K. and others. Differential evolution–a simple and efficient adaptive scheme for global optimization over continuous spaces: technical report TR-95-012. *International Computer Science, Berkeley, California*. 1995.

[372] Stützle, T. and López-Ibáñez, M. *Automated Design of Metaheuristic Algorithms*, pp. 541–579. Springer International Publishing, Cham, 2019. doi: $\mathbb{10.1007/978-3-319-91086-4}_17$. ISBN 978-3-319-91086-4.

[373] Su, M.-C., Su, S.-Y. and Zhao, Y.-X. A swarm-inspired projection algorithm. *Pattern Recognition*. 2009, 42, 11, pp. 2764–2786. doi: 10.1016/j.patcog.2009.03.020.

[374] Su, S., Wang, J., Fan, W. and Yin, X. Good lattice swarm algorithm for constrained engineering design optimization. In *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 6421–6424, 2007. doi: 10.1109/WICOM.2007.1575.

[375] Subashini, P., Dhivyaprabha, T. and Krishnaveni, M. Synergistic fibroblast optimization. *Advances in Intelligent Systems and Computing*. 2017, 517, pp. 285–294. doi: 10.1007/978-981-10-3174-8_25.

[376] Subramanian, C., Sekar, A. and Subramanian, K. A new engineering optimization method: African Wild Dog Algorithm. *International Journal of Soft Computing*. 2013, 8, 3, pp. 163–170. doi: 10.3923/ijscomp.2013.163.170.

[377] Sulaiman, M., Salhi, A., Selamoglu, B. and Kirikchi, O. A plant propagation algorithm for constrained engineering optimisation problems. *Mathematical Problems in Engineering*. 2014, 2014. doi: 10.1155/2014/627416.

[378] Sur, C., Sharma, S. and Shukla, A. Egyptian vulture optimization algorithm - A new nature inspired meta-heuristics for knapsack problem. *Advances in Intelligent Systems and Computing*. 2013, 209 AISC, pp. 227–237. doi: 10.1007/978-3-642-37371-8_26.

[379] Taherdangkoo, M., Yazdi, M. and Bagheri, M. Stem cells optimization algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2011, 6840 LNBI, pp. 394–403. doi: 10.1007/978-3-642-24553-4_52.

[380] Taherdangkoo, M., Hossein Shirzadi, M., Yazdi, M. and Hadi Bagheri, M. A robust clustering method based on blind, naked mole-rats (BNMR) algorithm. *Swarm and Evolutionary Computation*. 2013, 10, pp. 1–11. doi: 10.1016/j.swevo.2013.01.001.

[381] Taillard, e. and Voss, S. Popmusic - partial optimization metaheuristic under special intensification conditions. *Operations Research/ Computer Science Interfaces Series*. 2002, 15, pp. 613–629. doi: 10.1007/978-1-4615-1507-4.

[382] Talbi, E.-G. *Metaheuristics: from design to implementation.* 74. John Wiley & Sons, 2009.

[383] Tamura, K. and Yasuda, K. Primary study of spiral dynamics inspired optimization. *IEEJ Transactions on Electrical and Electronic Engineering*. 2011, 6, SUPPL. 1, pp. S98–S100. doi: 10.1002/tee.20628.

[384] Tan, Y. and Zhu, Y. Fireworks algorithm for optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2010, 6145 LNCS, PART 1, pp. 355–364. doi: 10.1007/978-3-642-13495-1_44.

[385] Tang, D., Dong, S., Jiang, Y., Li, H. and Huang, Y. ITGO: Invasive tumor growth optimization algorithm. *Applied Soft Computing Journal*. 2015, 36, pp. 670–698. doi: 10.1016/j.asoc.2015.07.045.

[386] Tang, R., Fong, S., Yang, X.-S. and Deb, S. Wolf search algorithm with ephemeral memory. In *Seventh International Conference on Digital Information Management (ICDIM 2012)*, pp. 165–172, 2012. doi: 10.1109/ICDIM.2012.6360147.

[387] Tanyildizi, E. and Demir, G. Golden sine Algorithm: A novel math-inspired algorithm. *Advances in Electrical and Computer Engineering*. 2017, 17, 2, pp. 71–78. doi: 10.4316/AECE.2017.02010.

[388] Tayarani, M. and Akbarzadeh. T., N. Magnetic optimization algorithms a new synthesis. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 2659–2664, 2008. doi: 10.1109/CEC.2008.4631155.

[389] Teodorovic, D. and Dell'Orco, M. Bee colony optimization–a cooperative learning approach to complex transportation problems. *Advanced OR and AI methods in transportation*. 2005, 51, pp. 60.

[390] Thammano, A. and Moolwong, J. A new computational intelligence technique based on human group formation. *Expert Systems with Applications*. 2010, 37, 2, pp. 1628–1634. doi: 10.1016/j.eswa.2009.06.046.

[391] Ting, T., Man, K., Guan, S.-U., Nayel, M. and Wan, K. Weightless Swarm Algorithm (WSA) for dynamic optimization problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012, 7513 LNCS, pp. 508–515. doi: 10.1007/978-3-642-35606-3_60.

[392] Ting, T., Yang, X.-S., Cheng, S. and Huang, K. *Hybrid metaheuristic algorithms: past, present, and future*, pp. 71–83. Springer, 2015.

[393] Tolabi, H. B., Ara, A. L. and Hosseini, R. A new thief and police algorithm and its application in simultaneous reconfiguration with optimal allocation of capacitor and distributed generation units. *Energy*. Jul 2020, 203, pp. 117911. ISSN 0360-5442. doi: 10.1016/j.energy.2020.117911.

[394] Tongchim, S. and Chongstitvatana, P. Parallel genetic algorithm with parameter adaptation. *Information Processing Letters*. Apr 2002, 82, 1, pp. 47–54. ISSN 0020-0190. doi: 10.1016/S0020-0190(01)00286-1.

[395] Torabi, S. and Safi-Esfahani, F. Improved Raven Roosting Optimization algorithm (IRRO). *Swarm and Evolutionary Computation*. 2018, 40, pp. 144–154. doi: 10.1016/j.swevo.2017.11.006.

[396] Tran, T. and Ng, K. A water-flow algorithm for flexible flow shop scheduling with intermediate buffers. *Journal of Scheduling*. 2011, 14, 5, pp. 483–500. doi: 10.1007/s10951-010-0205-x.

[397] Tsai, H.-C. and Lin, Y.-H. Modification of the fish swarm algorithm with particle swarm optimization formulation and communication behavior. *Applied Soft Computing Journal*. 2011, 11, 8, pp. 5367–5374. doi: 10.1016/j.asoc.2011.05.022.

[398] Tzanetos, A. and Dounias, G. A new metaheuristic method for optimization: Sonar inspired optimization. *Communications in Computer and Information Science*. 2017, 744, pp. 417–428. doi: 10.1007/978-3-319-65172-9_35.

[399] Ursem, R., Krink, T., Jensen, M. and Michalewicz, Z. Analysis and modeling of control tasks in dynamic systems. *IEEE Transactions on Evolutionary Computation*. Aug 2002, 6, 4, pp. 378–389. ISSN 1941-0026. doi: 10.1109/TEVC.2002.802871.

[400] Uymaz, S. A., Tezel, G. and Yel, E. Artificial algae algorithm (AAA) for nonlinear global optimization. *Applied Soft Computing*. Jun 2015, 31, pp. 153–171. ISSN 1568-4946. doi: 10.1016/j.asoc.2015.03.003.

[401] Vicsek, T., Czirk, A., Ben-Jacob, E., Cohen, I. and Shochet, O. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*. 1995, 75, 6, pp. 1226–1229. doi: 10.1103/PhysRevLett.75.1226.

[402] Victoire, T. A. A. and Jeyakumar, A. E. Hybrid PSO–SQP for economic dispatch with valve-point effect. *Electric Power Systems Research*. 2004, 71, 1, pp. 51–59.

[403] Viktorin, A., Senkerik, R., Pluhacek, M., Kadavy, T. and Zamuda, A. Distance based parameter adaptation for Success-History based Differential Evolution. *Swarm and Evolutionary Computation*. Nov 2019, 50, pp. 100462. ISSN 2210-6502. doi: 10.1016/j.swevo.2018.10.013.

[404] Walton, S., Hassan, O., Morgan, K. and Brown, M. Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos, Solitons and Fractals*. 2011, 44, 9, pp. 710–718. doi: 10.1016/j.chaos.2011.06.004.

[405] Wang, G.-G., Deb, S. and Coelho, L. Elephant Herding Optimization. In *2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 1–5, 2016. doi: 10.1109/ISCBI.2015.8.

[406] Wang, G.-G., Deb, S. and Dos Santos Coelho, L. Earthworm optimisation algorithm: A bio-inspired metaheuristic algorithm for global optimisation problems. *International Journal of Bio-Inspired Computation*. 2018, 12, 1, pp. 1–22. doi: 10.1504/ijbic.2018.093328.

[407] Wang, G.-G., Deb, S. and Cui, Z. Monarch butterfly optimization. *Neural Computing and Applications*. 2019, 31, 7, pp. 1995–2014. doi: 10.1007/s00521-015-1923-y.

[408] Wang, G.-G., Gao, X.-Z., Zenger, K. and S. Coelho, L. A Novel Metaheuristic Algorithm inspired by Rhino Herd Behavior. In *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation (EUROSIM 2016), The 57th SIMS Conference on Simulation and Modelling (SIMS 2016); Linköping electronic conference proceedings*, pp.

1026–1033, Dec 2018. doi: 10.3384/ecp171421026. Available from: <https://ep.liu.se/ecp/142/151/ecp17142151.pdf>.

[409] Wang, J. and Wang, D. Particle swarm optimization with a leader and followers. *Progress in Natural Science*. 2008, 18, 11, pp. 1437–1443. doi: 10.1016/j.pnsc.2008.03.029.

[410] Wang, P., Zhu, Z. and Huang, S. Seven-spot ladybird optimization: A novel and efficient metaheuristic algorithm for numerical optimization. *The Scientific World Journal*. 2013, 2013. doi: 10.1155/2013/378515.

[411] Wedde, H., Farooq, M. and Zhang, Y. BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2004, 3172 LNCS, pp. 83–94. doi: 10.1007/978-3-540-28646-2_8.

[412] Wei, Z., Cui, Z. and Zeng, J. Social cognitive optimization algorithm with reactive power optimization of power system. In *2010 International Conference on Computational Aspects of Social Networks*, pp. 11–14, 2010. doi: 10.1109/CASoN.2010.10.

[413] Weibo, W., Quanyuan, F. and Yongkang, Z. A novel particle swarm optimization algorithm with stochastic focusing search for real-parameter optimization. In *2008 11th IEEE Singapore International Conference on Communication Systems*, pp. 583–587, 2008. doi: 10.1109/ICCS.2008.4737251.

[414] Weyland, D. A rigorous analysis of the harmony search algorithm: How the research community can be misled by a novel methodology. *International Journal of Applied Metaheuristic Computing (IJAMC)*. 2010, 1, 2, pp. 50–60.

[415] Weyland, D. A critical analysis of the harmony search algorithm—How not to solve sudoku. *Operations Research Perspectives*. 2015, 2, pp. 97–105. ISSN 2214-7160. doi:

https://doi.org/10.1016/j.orp.2015.04.001. Available from: https://www.sciencedirect.com/science/article/pii/S221471601500010X.

[416] Wu, G. Across neighborhood search for numerical optimization. *Information Sciences*. 2016, 329, pp. 597–618. doi: 10.1016/j.ins.2015.09.051.

[417] Wu, S. and Banzhaf, W. A hierarchical cooperative evolutionary algorithm. In *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 23–30, 2010. doi: 10.1145/1830483.1830527.

[418] Xie, L., Zeng, J. and Cui, Z. General framework of artificial physics optimization algorithm. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pp. 1321–1326, 2009. doi: 10.1109/NABIC.2009.5393736.

[419] Xie, X.-F., Zhang, W.-J. and Yang, Z.-L. Adaptive particle swarm optimization on individual level. In *6th International Conference on Signal Processing, 2002.*, 2, pp. 1215–1218 vol.2, Aug 2002. doi: 10.1109/ICOSP.2002.1180009.

[420] Xu, Y., Cui, Z. and Zeng, J. Social emotional optimization algorithm for nonlinear constrained optimization problems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2010, 6466 LNCS, pp. 583–590. doi: 10.1007/978-3-642-17563-3_68.

[421] Xue, F., Cai, Y., Cao, Y., Cui, Z. and Li, F. Optimal parameter settings for bat algorithm. *International Journal of Bio-Inspired Computation*. May 2015, 7, 2, pp. 125–128. ISSN 1758-0366. doi: 10.1504/IJBIC.2015.069304.

[422] Yampolskiy, R. and EL-Barkouky, A. Wisdom of artificial crowds algorithm for solving NP-hard problems. *International Journal of Bio-Inspired Computation*. 2011, 3, 6, pp. 358–369. doi: 10.1504/IJBIC.2011.043624.

[423] YAN, G.-W., HAO, Z.-J. and XIE, J. A novel atmosphere clouds model optimization algorithm. *Journal of Computers (Taiwan)*. 2013, 24, 3, pp. 26–39.

[424] YANG, C., TU, X. and CHEN, J. Algorithm of marriage in honey bees optimization based on the wolf pack search. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*, pp. 462–467, 2007. doi: 10.1109/IPC.2007.104.

[425] YANG, F.-C. and WANG, Y.-P. Water flow-like algorithm for object grouping problems. *Journal of the Chinese Institute of Industrial Engineers*. 2007, 24, 6, pp. 475–488. doi: 10.1080/10170660709509062.

[426] YANG, M., LI, C., CAI, Z. and GUAN, J. Differential Evolution With Auto-Enhanced Population Diversity. *IEEE Transactions on Cybernetics*. Feb 2015, 45, 2, pp. 302–315. ISSN 2168-2275. doi: 10.1109/TCYB.2014.2339495.

[427] YANG, S., JIANG, J. and YAN, G. A dolphin partner optimization. In *2009 WRI Global Congress on Intelligent Systems*, 1, pp. 124–128, 2009. doi: 10.1109/GCIS.2009.464.

[428] YANG, S. and SATO, Y. Fitness Predator Optimizer to avoid premature convergence for multimodal problems. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 258–263, Oct 2014. doi: 10.1109/SMC.2014.6973917.

[429] YANG, X.-S. Engineering optimizations via nature-inspired virtual bee algorithms. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, 3562, pp. 317–323, 2005. doi: 10.1007/11499305_33.

[430] YANG, X.-S. Firefly algorithms for multimodal optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2009, 5792 LNCS, pp. 169–178. doi: 10.1007/978-3-642-04944-6_14.

[431] Yang, X.-S. Flower pollination algorithm for global optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012, 7445 LNCS, pp. 240–249. doi: 10.1007/978-3-642-32894-7_27.

[432] Yang, X.-S. and Deb, S. Eagle strategy using Levy walk and firefly algorithms for stochastic optimization. *Studies in Computational Intelligence*. 2010, 284, pp. 101–111. doi: 10.1007/978-3-642-12538-6_9.

[433] Yang, X.-S., Lees, J. and Morley, C. Application of virtual ant algorithms in the optimization of CFRP shear strengthened precracked structures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2006, 3991 LNCS-I, pp. 834–837. doi: 10.1007/11758501_117.

[434] Yang, X.-S. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008. ISBN 978-1-905986-10-1.

[435] Yang, X.-S. Firefly Algorithm, Levy Flights and Global Optimization. In Bramer, M., Ellis, R. and Petridis, M. (Ed.) *Research and Development in Intelligent Systems XXVI*, pp. 209–218. Springer, 2010. doi: 10.1007/978-1-84882-983-1_15. ISBN 978-1-84882-983-1.

[436] Yang, X.-S. A New Metaheuristic Bat-Inspired Algorithm. *arXiv:1004.4170 [physics]*. Apr 2010. Available from: <http://arxiv.org/abs/1004.4170>. arXiv: 1004.4170.

[437] Yang, X.-S. *Chapter 15 - Other Algorithms and Hybrid Algorithms*, pp. 213–226. Elsevier, Jan 2014. doi: 10.1016/B978-0-12-416743-8.00015-4. ISBN 978-0-12-416743-8.

[438] Yang, X.-S. and Deb, S. Engineering Optimisation by Cuckoo Search. *arXiv:1005.2908 [math, stat]*. Dec 2010. Available from: <http://arxiv.org/abs/1005.2908>. arXiv: 1005.2908.

[439] Yang, X.-S. and Deb, S. Cuckoo Search via Levy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pp. 210–214, Dec 2009. doi: 10.1109/NABIC.2009.5393690.

[440] Yang, X.-S., Deb, S., Fong, S., He, X. and Zhao, Y.-X. From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. *Computer*. 2016, 49, 9, pp. 52–59.

[441] Yu, J. and Li, V. A social spider algorithm for global optimization. *Applied Soft Computing Journal*. 2015, 30, pp. 614–627. doi: 10.1016/j.asoc.2015.02.014.

[442] Zarand, G., Pazmandi, F., Pal, K. and Zimanyi, G. Using hysteresis for optimization. *Physical Review Letters*. 2002, 89, 15, pp. 150201/1–150201/4. doi: 10.1103/PhysRevLett.89.150201.

[443] Zelinka, I. *SOMA—self-organizing migrating algorithm*, pp. 167–217. Springer, 2004.

[444] Zhang, H., Kennedy, D. D., Rangaiah, G. P. and Bonilla-Petriciolet, A. Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data. *Fluid Phase Equilibria*. Feb 2011, 301, 1, pp. 33–45. ISSN 0378-3812. doi: 10.1016/j.fluid.2010.10.025.

[445] Zhang, J., Xiao, M., Gao, L. and Pan, Q. Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. *Applied Mathematical Modelling*. 2018, 63, pp. 464–490. doi: 10.1016/j.apm.2018.06.036.

[446] Zhang, L., Dahlmann, C. and Zhang, Y. Human-Inspired Algorithms for continuous function optimization. In *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 1, pp. 318–321, 2009. doi: 10.1109/ICICISYS.2009.5357838.

[447] Zhang, Q., Wang, R., Yang, J., Ding, K., Li, Y. and Hu, J. Collective decision optimization algorithm: A new heuristic optimization method. *Neurocomputing*. 2017, 221, pp. 123–137. doi: 10.1016/j.neucom.2016.09.068.

[448] Zhang, Q., Wang, R., Yang, J., Lewis, A., Chiclana, F. and Yang, S. Biology migration algorithm: a new nature-inspired heuristic methodology

for global optimization. *Soft Computing*. 2019, 23, 16, pp. 7333–7358. doi: 10.1007/s00500-018-3381-9.

[449] Zhang, W., Luo, Q. and Zhou, Y. A method for training RBF neural networks based on population migration algorithm. In *2009 International Conference on Artificial Intelligence and Computational Intelligence*, 1, pp. 165–169, 2009. doi: 10.1109/AICI.2009.35.

[450] Zhang, X., Chen, W. and Dai, C. Application of oriented search algorithm in reactive power optimization of power system. In *2008 Third International conference on electric utility deregulation and restructuring and power technologies*, pp. 2856–2861, 2008. doi: 10.1109/DRPT.2008.4523896.

[451] Zhang, X., Sun, B., Mei, T. and Wang, R. Post-disaster restoration based on fuzzy preference relation and Bean Optimization Algorithm. In *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, pp. 271–274, 2010. doi: 10.1109/YCICT.2010.5713097.

[452] Zhang, Y., Wang, S. and Ji, G. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications, Oct 2015. ISSN 1024-123X. Available from: <https://www.hindawi.com/journals/mpe/2015/931256/>. DOI: https://doi.org/10.1155/2015/931256.

[453] Zhao, Z., Cui, Z., Zeng, J. and Yue, X. Artificial plant optimization algorithm for constrained optimization problems. In *2011 Second International Conference on Innovations in Bio-inspired Computing and Applications*, pp. 120–123, 2011. doi: 10.1109/IBICA.2011.34.

[454] Zheng, M., Liu, G.-X., Zhou, C.-G., Liang, Y.-C. and Wang, Y. Gravitation field algorithm and its application in gene cluster. *Algorithms for Molecular Biology*. 2010, 5, 1. doi: 10.1186/1748-7188-5-32.

[455] Zheng, Y.-J. Water wave optimization: A new nature-inspired metaheuristic. *Computers and Operations Research*. 2015, 55, pp. 1–11. doi: 10.1016/j.cor.2014.10.008.

[456] ZHENG, Y.-J., LING, H.-F. and XUE, J.-Y. Ecogeography-based optimization: Enhancing biogeography-based optimization with ecogeographic barriers and differentiations. *Computers and Operations Research*. 2014, 50, pp. 115–127. doi: 10.1016/j.cor.2014.04.013.

[457] ZHU, C. and NI, J. Cloud model-based differential evolution algorithm for optimization problems. In *2012 Sixth International Conference on Internet Computing for Science and Engineering*, pp. 55–59, 2012. doi: 10.1109/ICICSE.2012.57.

[458] ZHU, G.-Y. and ZHANG, W.-B. Optimal foraging algorithm for global optimization. *Applied Soft Computing Journal*. 2017, 51, pp. 294–313. doi: 10.1016/j.asoc.2016.11.047.

[459] ZUNGERU, A., ANG, L.-M. and SENG, K. Termite-hill: Performance optimized swarm intelligence based routing algorithm for wireless sensor networks. *Journal of Network and Computer Applications*. 2012, 35, 6, pp. 1901–1917. doi: 10.1016/j.jnca.2012.07.014.

[460] ZYL, E. V. and ENGELBRECHT, A. A Subspace-Based Method for PSO Initialization. In *2015 IEEE Symposium Series on Computational Intelligence*, pp. 226–233, Dec 2015. doi: 10.1109/SSCI.2015.42.

PUBLICATIONS OF THE AUTHOR

Journal Publications

[P.1] KAZIKOVA, A., PLUHACEK, M. and SENKERIK, R. How does the number of objective function evaluations impact our understanding of metaheuristics behavior?. *IEEE Access*. 2021, vol. 9, pp. 44032-44048. [accessed 2022-06-08]. ISSN 2169-3536. Available from: `https://ieeexplore.ieee.org/document/9378523`.

[P.2] KAZIKOVA, A., PLUHACEK, M. and SENKERIK, R. Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison?. *Mendel*. 2020, vol. 26, iss. 2, pp. 9-16. [accessed. 2022-06-08]. ISSN 1803-3814. Available from: `https://mendel-journal.org/index.php/mendel/article/view/120`.

[P.3] KAZIKOVA, A., PLUHACEK, M. and SENKERIK, R. Regarding the behavior of bison runners within the Bison algorithm. *Mendel*. 2018, vol. 24, iss. 1, pp. 63-70. [accessed 2022-06-08]. ISSN 1803-3814. Available from: `https://mendel-journal.org/index.php/mendel/article/view/24`.

[P.4] PLUHACEK, Michal, KAZIKOVA, A., KADAVY, T., VIKTORIN, A. and SENKERIK, R. Relation of neighborhood size and diversity loss rate in particle swarm optimization with ring topology. *Mendel* . 2021, vol. 27, iss. 2, pp. 74-79. [accessed 2022-06-08]. ISSN 1803-3814. Available from: `https://mendel-journal.org/index.php/mendel/article/view/151/162`.

Conference Proceedings

[P.5] KAZIKOVA, A., KOMINKOVA OPLATKOVA, Z., PLUHACEK, M. and SENKERIK, R. Border strategies of the bison algorithm. In: *Proceedings of the 33rd International ECMS Conference on Modelling and Simulation (ECMS 2019)* . Napoli: European Council for Modelling and Simulation, 2019, pp. 43-49. [accessed 2022-06-08]. ISSN 2522-2414.

Available from: http://www.scs-europe.net/dlib/2019/2019-0043.htm.

[P.6] KAZIKOVA, A., Krystian ŁAPA, PLUHACEK, M. and SENKERIK, R. Cascade PID controller optimization using bison algorithm. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* . Zakopane: Springer Science and Business Media Deutschland GmbH, 2020, pp. 406-416. [accessed 2022-06-08]. ISSN 0302-9743. Available from: https://link.springer.com/chapter/10.1007/978-3-030-61401-0_38.

[P.7] KAZIKOVA, A., PLUHACEK, M., KADAVY, T. and SENKERIK, R. Introducing the run support strategy for the bison algorithm. In: *Lecture Notes in Electrical Engineering* . Ostrava: Springer Verlag, 2020, pp. 272-282. [accessed 2022-06-08]. ISSN 1876-1100. Available from: https://link.springer.com/chapter/10.1007/978-3-030-14907-9_27.

[P.8] KAZIKOVA, A., PLUHACEK, M., VIKTORIN, A. and SENKERIK, R. New running technique for the bison algorithm. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Zakopane: Springer Verlag, 2018, pp. 417-426. [accessed 2022-06-08]. ISSN 0302-9743. Available from: https://link.springer.com/chapter/10.1007/978-3-319-91253-0_39.

[P.9] KAZIKOVA, A., PLUHACEK, M. and SENKERIK, R. Performance of the bison algorithm on benchmark IEEE CEC 2017. In: *Advances in Intelligent Systems and Computing*. Springer Verlag, 2019, pp. 445-454. [accessed 2022-06-08]. ISSN 2194-5357. Available from: https://link.springer.com/chapter/10.1007/978-3-319-91189-2_44.

[P.10] KAZIKOVA, A., PLUHACEK, M., SENKERIK, R. and VIKTORIN, A. Proposal of a new swarm optimization method inspired in bison behavior. In: *Advances in Intelligent Systems and Computing*. Brno: Springer Verlag, 2019, pp. 146-156. [accessed 2022-06-08]. ISSN

2194-5357. Available from: https://link.springer.com/chapter/10.1007/978-3-319-97888-8_13.

[P.11] KAZIKOVA, A., PLUHACEK, M. and SENKERIK, R. Tuning of the bison algorithm control parameters. In: *Proceedings - European Council for Modelling and Simulation, ECMS*. Wilhelmshaven: European Council for Modelling and Simulation, 2018, pp. 156-162. [accessed 2022-06-08]. ISSN 2522-2414. Available from: http://www.scs-europe.net/dlib/2018/2018-0156.htm.

[P.12] SENKERIK, Roman, VIKTORIN, A., KADAVY, T., PLUHACEK, M., KAZIKOVA, A., DIEP, Q. B. and ZELINKA, I. Population diversity analysis in adaptive differential evolution variants with unconventional randomization schemes. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Zakopane: Springer Verlag, 2019, pp. 506-518. [accessed 2022-06-08]. ISSN 0302-9743. Available from: https://link.springer.com/chapter/10.1007/978-3-030-20912-4_46.

[P.13] PLUHACEK, Michal, KAZIKOVA, A., KADAVY, T., VIKTORIN, A. and SENKERIK, R.. Explaining SOMA: The relation of stochastic perturbation to population diversity and parameter space coverage. In: *GECCO 2021 Companion - Proceedings of the 2021 Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, Inc, 2021, pp. 1944-1952. [accessed 2022-06-08]. Available from: https://dl.acm.org/doi/10.1145/3449726.3463211.

[P.14] PLUHACEK, Michal, VIKTORIN, A., KADAVY, T. and KAZIKOVA, A. On the common population diversity measures in metaheuristics and their limitations. In: *2021 IEEE Symposium Series on Computational Intelligence, SSCI 2021 – Proceedings*. Orlando, FL: Institute of Electrical and Electronics Engineers Inc., 2021 [accessed 2022-06-08]. Available from: https://ieeexplore.ieee.org/document/9660135.

CURRICULUM VITAE

Anežka Kazíková

INFORMATION

Born 9th May 1990
Czech Republic

kazikova@utb.cz
+420 776 021 238

ADDRESS

Nová Lhota 282
696 74

LANGUAGES

Czech • maternity language
English • intermediate

SKILLS

PROGRAMMING
> Advanced knowledge
Python • C • C++

> Basic knowledge
C# • Java • PHP • MySQL
MatLab • Mathematica

WEB TECHNOLOGIES
HTML • CSS • JavaScript

INTEREST

Artificial Intelligence
Swarm Algorithms
Optimization

EDUCATION

2015 - NOW

Tomas Bata University in Zlín
/ Faculty of Applied Informatics
Doctoral studies: Engineering Informatics
Member of A.I. laboratory (ailab.fai.cz)

2013 - 2015

Tomas Bata University in Zlín / Faculty of Applied Informatics
Field of study: Engineering Informatics

2010 - 2013

Tomas Bata University in Zlín / Faculty of Applied Informatics
Field of study: Information and Control Technologies

WORK EXPERIENCE

2015 - NOW

Freelance programming (NetArk.cz s.r.o.)

2010 - 2016

Website development

ACHIEVEMENTS

2013 • EdHouse Award for Bachelor's Thesis
2015 • Dean's Award for Master's Thesis
2015 • Passed with honors

LIST OF APPENDICES

APPENDIX A: LIST OF METAHEURISTICS

Tab. A.1 List of all metaheuristics from [275] sorted by number of citations of their original proposal publications in the Scopus Database 30/1/2020-1/11/2021.

	Algorithm	**Acronym**	**Year**	**Reference Paper**	**▼ Scopus ▼ Citations**	**Classification**
1	Genetic Algorithms	GA	1975	[166]	34159	Breeding based
2	Simulated Annealing	SA.1	1983	[220]	26959	Chemistry based
3	Differential Evolution	DE	1997	[370]	14769	Breeding based
4	Particle Swarm Optimization	PSO	1995	[115]	10938	Swarm based
5	Ant Colony Optimization	ACO	1996	[107]	8075	Swarm based
6	Self-Driven Particles	SPP	1995	[401]	4195	Physics based
7	Cuckoo Search	CS	2009	[439]	3702	Swarm based
8	Grey Wolf Optimizer	GWO	2014	[269]	3433	Swarm based
9	Artificial Bee Colony	ABC	2005	[200]	3423	Swarm based
10	Gravitational Search Algorithm	GSA	2009	[335]	3237	Physics based
11	Bat Inspired Algorithm	BAT	2010	[436]	2468	Swarm based
12	Bacterial Foraging Optimization	BFOA	2002	[307]	2183	Swarm based
13	Evolution Strategies	ES	1973	[339]	2093	Breeding based
14	Biogeography based Optimization	BBO	2008	[365]	2082	Breeding based
15	Firefly Algorithm	FA	2009	[430]	2081	Swarm based
16	Whale Optimization Algorithm	WOA	2016	[265]	1962	Swarm based
17	Teaching-Learning based Optimization Algorithm	TLBO	2011	[334]	1688	Human based
18	Imperialist Competitive Algorithm	ICA	2007	[24]	1458	Human based
19	Harmony Search	HS	2005	[233]	1281	Physics based
20	Moth Flame Optimization Algorithm	MFO	2015	[262]	1002	Swarm based
21	Ant Lion Optimizer	ALO	2015	[268]	970	Swarm based
22	Flower Pollination Algorithm	FPA	2012	[431]	951	Plant based
23	Krill Herd	KH	2012	[143]	950	Swarm based
24	Sine Cosine Algorithm	SCA.2	2016	[264]	893	Swarm based
25	Fruit Fly Optimization Algorithm	FOA	2012	[305]	874	Swarm based
26	Weed Colonization Optimization	IWO	2006	[252]	835	Breeding based
27	Salp Swarm Algorithm	SSA.2	2017	[267]	810	Swarm based
28	Big Bang Big Crunch	BBBC	2006	[124]	745	Physics based
29	Clonal Selection Algorithm	CSA.1	2000	[94]	738	Miscellaneous
30	Dragonfly Algorithm	DA	2016	[263]	708	Swarm based
31	Shuffled Frog-Leaping Algorithm	SFLA	2006	[126]	683	Swarm based
32	Bees Algorithm	BA	2006	[310]	682	Swarm based
33	Charged Systems Search	CSS	2010	[208]	669	Physics based
34	Grasshopper Optimisation Algorithm	GOA	2017	[352]	606	Swarm based
35	Cuckoo Optimization Algorithm	COA	2011	[328]	594	Swarm based
36	Symbiosis Organisms Search	SOS	2014	[67]	586	Swarm based
37	Electromagnetism Mechanism Optimization	EMO	2003	[40]	571	Physics based
38	Crow Search Algorithm	CSA	2016	[23]	569	Swarm based

	Algorithm	Acronym	Year	Reference Paper	▼ Scopus ▼ Citations	Classification
39	Backtracking Search Optimization	BSO.3	2013	[79]	562	Miscellaneous
40	Fireworks Algorithm Optimization	FAO	2010	[384]	541	Miscellaneous
41	Multi-Verse Optimizer	MVO	2016	[266]	523	Physics based
42	Group Search Optimizer	GSO.1	2009	[163]	517	Swarm based
43	The Great Deluge Algorithm	TGD	1993	[112]	497	Miscellaneous
44	Water Cycle Algorithm	WCA.2	2012	[125]	494	Physics based
45	Black Hole Optimization	BH	2013	[159]	457	Physics based
46	Modified Cuckoo Search	MCS	2011	[404]	415	Swarm based
47	Chemical Reaction Optimization Algorithm	CRO.1	2010	[230]	365	Chemistry based
48	Cat Swarm Optimization	CSO	2006	[71]	339	Swarm based
49	Brain Storm Optimization Algorithm	BSO.2	2011	[364]	324	Human based
50	Mine Blast Algorithm	MBA	2013	[346]	322	Miscellaneous
51	Ray Optimization	RO	2012	[206]	319	Physics based
52	Social Behavior Optimization Algorithm	SBO.1	2003	[336]	310	Human based
53	Self-Organizing Migrating Algorithm	SOMA	2004	[443]	295	Breeding based
54	Harry's Hawk Optimization Algorithm	HHO	2019	[165]	295	Swarm based
55	Marriage In Honey Bees Optimization	MHBO	2001	[1]	290	Breeding based
56	Colliding Bodies Optimization	CBO	2014	[207]	284	Physics based
57	Bacterial Chemotaxis Optimization	BCO.2	2002	[287]	281	Swarm based
58	Social Spider Optimization	SSO.2	2013	[86]	279	Swarm based
59	Bee Colony Optimization	BCO	2005	[389]	278	Swarm based
60	Differential Search Algorithm	DSA	2012	[77]	270	Miscellaneous
61	Intelligence Water Drops Algorithm	IWD	2009	[357]	269	Physics based
62	Glowworm Swarm Optimization	GSO	2005	[224]	250	Swarm based
63	Chicken Swarm Optimization	CSO.1	2014	[255]	247	Swarm based
64	Dolphin Echolocation	DE.1	2013	[204]	239	Swarm based
65	Pigeon Inspired Optimization	PIO	2014	[111]	227	Swarm based
66	Virtual Bees Algorithm	VBA	2005	[429]	225	Swarm based
67	Water Wave Optimization Algorithm	WWA	2015	[455]	224	Physics based
68	Chaos Optimization Algorithm	COA.4	1998	[235]	213	Miscellaneous
69	Stochastic Fractal Search	SFS.1	2015	[350]	204	Miscellaneous
70	Interior Search Algorithm	ISA	2014	[142]	201	Miscellaneous
71	Dendritic Cells Algorithm	DCA	2005	[151]	197	Breeding based
72	Spider Monkey Optimization	SMO	2014	[26]	196	Swarm based
73	Lion Optimization Algorithm	LOA	2014	[330]	188	Swarm based
74	Eagle Strategy	ES.1	2010	[432]	171	Swarm based
75	Lightning Search Algorithm	LSA	2015	[360]	165	Physics based
76	Social Spider Algorithm	SSA	2015	[441]	164	Swarm based
77	Artificial Chemical Reaction Optimization Algorithm	ACROA	2011	[9]	158	Chemistry based
78	Extremal Optimization	EO	1999	[47]	155	Miscellaneous
79	BeeHive Algorithm	BHA	2004	[411]	153	Swarm based
80	Cuttlefish Algorithm	CFA	2015	[118]	153	Swarm based
81	Hunting Search	HuS	2010	[300]	152	Swarm based
82	Catfish Optimization Algorithm	CAO	2011	[75]	150	Swarm based
83	Regular Butterfly Optimization Algorithm	RBOA	2019	[18]	150	Swarm based
84	Monkey Search	MS	2007	[282]	144	Swarm based
85	Spotted Hyena Optimizer	SHO	2017	[103]	142	Swarm based
86	Bird Swarm Algorithm	BSA	2016	[256]	141	Swarm based
87	Galaxy based Search Algorithm	GBSA	2011	[358]	136	Physics based
88	Elephant Herding Optimization	EHO	2016	[405]	136	Swarm based
89	Monarch Butterfly Optimization	MBO.1	2019	[407]	136	Swarm based
90	Migrating Birds Optimization	MBO.2	2012	[113]	135	Swarm based

	Algorithm	Acronym	Year	Reference Paper	▼ Scopus ▼ Citations	Classification
91	Cultural Algorithms	CA	1999	[192]	132	Human based
92	Squirrel Search Algorithm	SSA.1	2019	[186]	129	Swarm based
93	Vortex Search Algorithm	VS	2015	[104]	127	Physics based
94	Shark Search Algorithm	SA	1998	[181]	127	Swarm based
95	Wolf Search Algorithm	WSA.1	2012	[386]	125	Swarm based
96	Fish School Search	FSS	2008	[136]	116	Swarm based
97	Artificial Reaction Algorithm	ARA	2013	[253]	108	Chemistry based
98	Wasp Colonies Algorithm	WCA	1991	[259]	108	Swarm based
99	Bee Swarm Optimization	BSO	2010	[8]	107	Swarm based
100	Virus Colony Search	VCS	2016	[237]	107	Swarm based
101	Thermal Exchange Optimization	TEO	2017	[203]	106	Chemistry based
102	League Championship Algorithm	LCA.1	2014	[174]	103	Human based
103	Water Evaporation Optimization	WEO	2016	[202]	102	Physics based
104	Wolf Pack Search	WPS	2007	[424]	97	Swarm based
105	Bees Swarm Optimization Algorithm	BSOA	2005	[109]	95	Swarm based
106	Search Group Algorithm	SGA.2	2015	[149]	93	Miscellaneous
107	Grenade Explosion Method	GEM	2010	[7]	88	Miscellaneous
108	Fish Swarm Algorithm	FSA	2011	[397]	88	Swarm based
109	Artificial Cooperative Search	ACS	2013	[78]	87	Miscellaneous
110	Bee System	BS.1	2002	[245]	87	Swarm based
111	Shark Smell Optimization	SSO	2016	[4]	86	Swarm based
112	States Matter Optimization Algorithm	SMS	2014	[87]	85	Physics based
113	Artificial Algae Algorithm	AAA	2015	[400]	85	Swarm based
114	Wind Driven Optimization	WDO	2010	[30]	84	Miscellaneous
115	Rain-Fall Optimization Algorithm	RFOA	2017	[6]	83	Physics based
116	Exchange Market Algorithm	EMA	2014	[147]	83	Miscellaneous
117	Bird Mating Optimization	BMO	2014	[22]	82	Breeding based
118	Water Flow Algorithm	WFA.1	2007	[29]	82	Physics based
119	Ions Motion Optimization Algoirthm	IMO	2015	[189]	82	Chemistry based
120	Queen-Bee Evolution	QBE	2003	[195]	81	Breeding based
121	River Formation Dynamics	RFD	2007	[325]	80	Physics based
122	FIFA World Cup Competitions	FIFAAO	2016	[337]	80	Human based
123	Flocking Base Algorithms	FBA	2006	[88]	80	Swarm based
124	Forest Optimization Algorithm	FOA.1	2014	[145]	77	Plant based
125	Central Force Optimization	CFO	2008	[141]	76	Physics based
126	Coyote Optimization Algorithm	CCOA	2018	[313]	76	Swarm based
127	Seeker Optimization Algorithm	SOA	2007	[90]	76	Swarm based
128	Spiral Dynamics Optimization	SO	2011	[383]	75	Physics based
129	Coral Reefs Optimization	CRO	2014	[347]	73	Breeding based
130	Electromagnetic Field Optimization	EFO	2016	[5]	70	Physics based
131	Small World Optimization	SWO	2006	[110]	70	Miscellaneous
132	Optics Inspired Optimization	OIO	2015	[175]	69	Physics based
133	African Buffalo Optimization	ABO	2015	[299]	69	Swarm based
134	Human Evolutionary Model	HEM	2007	[277]	68	Human based
135	Lion Algorithm	LA	2012	[329]	67	Swarm based
136	Soccer League Competition	SLC	2014	[278]	63	Human based
137	Magnetic Optimization Algorithm	MFO.2	2008	[388]	62	Physics based
138	Passing Vehicle Search	PVS	2016	[354]	59	Miscellaneous
139	Fast Bacterial Swarming Algorithm	FBSA	2008	[73]	59	Swarm based

	Algorithm	Acronym	Year	Reference Paper	▼ Scopus ▼ Citations	Classification
140	Social Engineering Optimization	SEO	2018	[130]	58	Miscellaneous
141	Runner Root Algorithm	RRA	2015	[257]	55	Plant based
142	Bees Life Algorithm	BLA	2018	[42]	55	Swarm based
143	Snap-Drift Cuckoo Search	SDCS	2017	[332]	55	Swarm based
144	Galactic Swarm Optimization	GSO.2	2016	[285]	54	Physics based
145	Roach Infestation Problem	RIO	2008	[160]	54	Swarm based
146	Gases Brownian Motion Optimization	GBMO	2013	[2]	53	Chemistry based
147	Earthworm Optimization Algorithm	EOA	2018	[406]	51	Breeding based
148	Global-Best Brain Storm Optimization Algorithm	GBSO	2017	[121]	51	Human based
149	Satin Bowerbird Optimizer	SBO	2017	[351]	51	Swarm based
150	Ecogeography-Based Optimization	EBO	2014	[456]	50	Breeding based
151	PopMusic Algorithm	PopMusic	2002	[381]	50	Physics based
152	Vibrating Particle Systems Algorithm	VPO	2017	[205]	50	Physics based
153	Bee System	BS	1997	[353]	50	Swarm based
154	Water Flow-Like Algorithms	WFA	2007	[425]	49	Physics based
155	Old Bachelor Acceptance	OBA	1995	[170]	49	Human based
156	Gravitational Clustering Algorithm	GCA	1999	[227]	48	Physics based
157	Hysteresis for Optimization	HO	2002	[442]	48	Physics based
158	Wolf Colony Algorithm	WCA.1	2011	[238]	47	Swarm based
159	Ying-Yang Pair Optimization	YYOP	2016	[319]	46	Miscellaneous
160	Dolphin Partner Optimization	DPO	2009	[427]	45	Swarm based
161	Golden Ball Algorithm	GBA	2014	[303]	44	Human based
162	Artificial Chemical Process	ACP	2005	[179]	40	Chemistry based
163	Social Emotional Optimization Algorithm	SEA	2010	[420]	40	Human based
164	Volleyball Premier League Algorithm	VPL	2018	[274]	40	Human based
165	Collective Animal Behavior	CAB	2012	[85]	40	Swarm based
166	Swallow Swarm Optimization	SSO.1	2013	[291]	40	Swarm based
167	Termite Hill Algorithm	TA	2012	[459]	40	Swarm based
168	Sperm Whale Algorithm	SWA	2016	[116]	39	Swarm based
169	Anarchic Society Optimization	ASO	2012	[362]	38	Human based
170	Across Neighbourhood Search	ANS	2016	[416]	38	Miscellaneous
171	Keshtel Algorithm	KA	2014	[154]	38	Miscellaneous
172	Radial Movement Optimization	RMO	2014	[326]	37	Physics based
173	Space Gravitational Algorithm	SGA	2005	[169]	37	Physics based
174	Tree Growth Algorithm	TGA	2018	[69]	37	Plant based
175	Membrane Algorithms	MA	2006	[295]	37	Miscellaneous
176	Artificial Physics Optimization	APO	2009	[418]	36	Physics based
177	Penguins Search Optimization Algorithm	PSOA	2013	[146]	36	Swarm based
178	Paddy Field Algorithm	PFA	2009	[317]	35	Plant based
179	Chaotic Dragonfly Algorithm	CDA	2019	[355]	35	Swarm based
180	Elephant Search Algorithm	ESA	2016	[97]	34	Swarm based
181	Artificial Electric Field Algorithm	AEFA	2019	[14]	32	Physics based
182	Egyptian Vulture Optimization Algorithm	EV	2013	[378]	32	Swarm based
183	Goose Team Optimization	GTO	2008	[409]	32	Swarm based
184	The Great Salmon Run Algorithm	TGSR	2012	[281]	30	Swarm based
185	Kaizen Programming	KP	2014	[95]	30	Miscellaneous
186	Kinetic Gas Molecules Optimization	KGMO	2014	[272]	29	Chemistry based
187	Social Cognitive Optimization Algorithm	SCOA	2010	[412]	29	Human based

	Algorithm	Acronym	Year	Reference Paper	▼ Scopus ▼ Citations	Classification
188	Viral Systems Optimization	VSO	2008	[82]	29	Swarm based
189	Queuing Search Algorithm	QSA.1	2018	[445]	28	Human based
190	Plant Propagation Algorithm	PPA.1	2014	[377]	28	Plant based
191	Honeybee Social Foraging	HSF	2010	[324]	28	Swarm based
192	Termite Colony Optimization	TCO	2010	[164]	28	Swarm based
193	Wisdom of Artificial Crowds	WAC	2011	[422]	27	Human based
194	Natural Aggregation Algorithm	NAA	2016	[244]	27	Swarm based
195	Gravitational Field Algorithm	GFA	2010	[454]	26	Physics based
196	Tug Of War Optimization	TWO	2016	[209]	26	Human based
197	Variable Mesh Optimization	VMO	2012	[320]	24	Breeding based
198	Cognitive Behavior Optimization Algorithm	COA.3	2016	[236]	24	Human based
199	Group Counseling Optimization	GCO	2014	[120]	24	Human based
200	Simulated Bee Colony	SBC	2009	[251]	24	Swarm based
201	Raven Roosting Optimization Algorithm	RRO	2016	[49]	24	Swarm based
202	Eco-Inspired Evolutionary Algorithm	EEA	2011	[306]	23	Breeding based
203	Melody Search	MS.1	2011	[20]	23	Physics based
204	Integrated Radiation Optimization	IRO	2007	[74]	23	Chemistry based
205	Group Leaders Optimization Algorithm	GLOA	2011	[92]	23	Human based
206	Bacterial-GA Foraging	BGAF	2007	[64]	23	Swarm based
207	Pity Beetle Algorithm	PBA	2018	[198]	23	Swarm based
208	Gene Expression	GE	2002	[135]	22	Breeding based
209	Sheep Flock Heredity Model	SFHM	1999	[289]	22	Breeding based
210	Swine Influenza Models based Optimization	SIMBO	2013	[308]	22	Breeding based
211	Killer Whale Algorithm	KWA	2017	[44]	21	Swarm based
212	Method of Musical Composition	MMC	2014	[279]	20	Physics based
213	Water-Flow Algorithm Optimization	WFO	2011	[396]	20	Physics based
214	Collective Decision Optimization Algorithm	CDOA	2017	[447]	20	Human based
215	Hierarchical Swarm Model	HSM	2010	[62]	20	Swarm based
216	Red Deer Algorithm	RDA	2020	[131]	20	Swarm based
217	Slime Mould Algorithm	SMA	2008	[276]	20	Swarm based
218	Cricket Behavior-Based Algorithm	CBBE	2016	[56]	19	Swarm based
219	Invasive Tumor Optimization Algorith	ITGO	2015	[385]	19	Swarm based
220	Mouth Breeding Fish Algorithm	MBF	2018	[185]	19	Swarm based
221	Swarm Inspired Projection Algorithm	SIP	2009	[373]	19	Swarm based
222	Photosynthetic Algorithm	PA	2000	[283]	18	Chemistry based
223	Competitive Optimization Algorithm	COOA	2016	[359]	18	Human based
224	Simple Optimization	SOPT	2012	[157]	18	Miscellaneous
225	Binary Whale Optimization Algorithm	BWOA	2019	[341]	18	Swarm based
226	Optimal Foraging Algorithm	OFA	2017	[458]	18	Swarm based
227	Prey Predator Algorithm	PPA	2013	[155]	18	Swarm based
228	Human-Inspired Algorithms	HIA	2009	[446]	17	Human based
229	Parliamentary Optimization Algorithm	POA	2009	[48]	17	Human based
230	Heart Optimization	HO.1	2014	[158]	17	Miscellaneous
231	Bumblebees	BB	2009	[81]	17	Swarm based
232	Cultural Coyote Optimization Algorithm	CCOA	2019	[312]	17	Swarm based
233	Magnetotactic Bacteria Optimization Algorithm	MBO	2013	[270]	17	Swarm based
234	Naked Moled Rat	NMR	2019	[348]	17	Swarm based
235	Ideology Algorithm	IA	2017	[171]	16	Human based
236	Saplings Growing Up Algorithm	SGA.1	2007	[201]	16	Plant based

	Algorithm	Acronym	Year	Reference Paper	▼ Scopus ▼ Citations	Classification
237	Bee Colony-Inspired Algorithm	BCIA	2009	[176]	16	Swarm based
238	Football Game Inspired Algorithms	FCA.1	2016	[127]	15	Human based
239	Human Group Formation	HGF	2010	[390]	15	Human based
240	Self-Defense Mechanism Of The Plants Algorithm	SDMA	2018	[57]	15	Plant based
241	Cheetah Based Algorithm	CBA	2018	[222]	15	Swarm based
242	Weightless Swarm Algorithm	WSA	2012	[391]	15	Swarm based
243	Gravitational Interactions Algorithm	GIO	2011	[140]	14	Physics based
244	Duelist Optimization Algorithm	DOA	2016	[43]	14	Human based
245	Pearl Hunting Algorithm	PHA	2012	[59]	14	Miscellaneous
246	Wasp Swarm Optimization	WSO	2005	[314]	14	Swarm based
247	Meerkats Inspired Algorithm	MIA	2018	[221]	14	Swarm based
248	Asexual Reproduction Optimization	ARO	2011	[250]	13	Breeding based
249	Bean Optimization Algorithm	BOA	2010	[451]	13	Breeding based
250	SuperBug Algorithm	SuA	2012	[13]	13	Breeding based
251	Leaders and Followers Algorithm	LFA	2015	[150]	13	Human based
252	Artificial Plants Optimization Algorithm	APO.1	2011	[453]	13	Plant based
253	Golden Sine Algorithm	GSA.1	2017	[387]	13	Miscellaneous
254	Blind, Naked Mole-Rats Algorithm	BNMR	2013	[380]	13	Swarm based
255	Modified Cockroach Swarm Optimization	MCSO	2014	[297]	13	Swarm based
256	Artificial Raindrop Algorithm	RDA.1	2014	[190]	12	Miscellaneous
257	Laying Chicken Algorithm	LCA	2017	[168]	12	Swarm based
258	Bat Intelligence	BI	2012	[249]	12	Swarm based
259	Stem Cells Algorithm	SCA	2011	[379]	11	Breeding based
260	Virulence Optimization Algorithm	VOA	2016	[183]	11	Breeding based
261	Hurricane based Optimization Algorithm	HO.2	2014	[338]	11	Physics based
262	Oriented Search Algorithm	OSA	2008	[450]	11	Human based
263	Consultant Guide Search	CGS	2010	[417]	11	Swarm based
264	Frog Call Inspired Algorithm	FCA	2009	[284]	11	Swarm based
265	Virtual Ants Algorithm	VAA	2006	[433]	11	Swarm based
266	Gravitational Emulation Local Search	GELS	2009	[28]	10	Physics based
267	Synergistic Fibroblast Optimization	SFO	2017	[375]	10	Chemistry based
268	Bar Systems	BS.2	2008	[99]	10	Miscellaneous
269	Cloud Model-Based Algorithm	CMBDE	2012	[457]	10	Miscellaneous
270	Animal Behavior Hunting	ABH	2014	[288]	10	Swarm based
271	Flock by Leader	FL	2012	[32]	10	Swarm based
272	Good Lattice Swarm Optimization	GLSO	2007	[374]	10	Swarm based
273	Locust Swarms Optimization	LSO	2009	[63]	10	Swarm based
274	The Great Salmon Run Algorithm	TGSR	2013	[280]	10	Swarm based
275	Sonar Inspired Optimization	SIO	2017	[398]	9	Physics based
276	Unconscious Search	US	2012	[16]	9	Human based
277	Artificial Beehive Algorithm	ABA	2009	[286]	9	Swarm based
278	Biology Migration Algorithm	BMA	2019	[448]	9	Swarm based
279	Butterfly Optimizer	BO	2016	[225]	9	Swarm based
280	Seven-Spot Labybird Optimization	LBO	2013	[410]	9	Swarm based
281	General Relativity Search Algorithm	GRSA	2015	[31]	8	Physics based
282	Scientifics Algoritmhs	SA.2	2014	[133]	8	Miscellaneous
283	Group Escape Behavior	GEB	2011	[260]	8	Swarm based
284	Bioluminiscent Swarm Optimization Algorithm	BSO.1	2011	[96]	8	Swarm based

	Algorithm	Acronym	Year	Reference Paper	▼ Scopus ▼ Citations	Classification
285	Improved Genetic Immune Algorithm	IGIA	2017	[33]	7	Breeding based
286	Virus Optimization Algorithm	VOA.1	2009	[194]	7	Swarm based
287	Mox Optimization Algorithm	MOX	2011	[261]	7	Swarm based
288	Population Migration Algorithm	PMA	2009	[449]	7	Swarm based
289	Light Ray Optimization	LRO	2010	[363]	6	Physics based
290	Spiral Optimization Algorithm	SPOA	2010	[191]	6	Physics based
291	Camel Travelling Behavior	COA.1	2016	[177]	6	Swarm based
292	Artificial Tribe Algorithm	ATA	2012	[66]	6	Swarm based
293	Bison Algorithm	BIA	2019	[214]	6	Swarm based
294	Nomadic People Optimizer	NPO	2020	[349]	6	Swarm based
295	Surface-Simplex Swarm Evolution Algorithm	SSSE	2017	[322]	6	Swarm based
296	Quantum Superposition Algorithm	QSA	2016	[70]	5	Physics based
297	Neuronal Communication Algorithm	NCA	2017	[21]	5	Miscellaneous
298	Andean Condor Algorithm	ACA	2019	[11]	5	Swarm based
299	African Wild Dog Algorithm	AWDA	2013	[376]	5	Swarm based
300	Jaguar Algorithm	JA	2016	[60]	5	Swarm based
301	Mosquito Flying Optimization	MFO.1	2016	[10]	5	Swarm based
302	Reincarnation Concept Optimization Algorithm	ROA	2010	[361]	5	Swarm based
303	Bus Transportation Behavior	BTA	2019	[46]	4	Human based
304	Greedy Politics Optimization Algorithm	GPO	2014	[234]	4	Human based
305	Soccer Game Optimization	SGO	2014	[321]	4	Human based
306	Natural Forest Regeneration Algorithm	NFR	2016	[273]	4	Plant based
307	Artificial Searching Swarm Algorithm	ASSA	2009	[65]	4	Swarm based
308	Bacterial Colony Optimization	BCO.1	2012	[296]	4	Swarm based
309	Worm Optimization	WO	2014	[17]	4	Swarm based
310	Immune-Inspired Computational Intelligence	ICI	2008	[83]	3	Breeding based
311	Crystal Energy Optimization Algorithm	CEO	2016	[134]	3	Physics based
312	Particle Collision Algorithm	PCA	2007	[345]	3	Physics based
313	Rhino Herd Behavior	RHB	2018	[144]	3	Swarm based
314	OptBees	OB	2013	[248]	3	Swarm based
315	Hoopoe Heuristic Optimization	HHO.1	2012	[122]	3	Swarm based
316	Zombie Survival Optimization	ZSO	2012	[292]	3	Swarm based
317	Artificial Infections Disease Optimization	AIDO	2016	[173]	2	Breeding based
318	Harmony Elements Algorithm	HEA	2008	[89]	2	Physics based
319	Stochastic Focusing Search	SFS	2008	[413]	2	Human based
320	Bald Eagle Search	BES	2020	[12]	2	Swarm based
321	See-See Partridge Chicks Optimization	SSPCO	2016	[302]	2	Swarm based
322	Atmosphere Clouds Model	ACM	2013	[423]	1	Miscellaneous
323	Hypercube Natural Aggregation Algorithm	HYNAA	2020	[247]	0	Swarm based

APPENDIX B: INSTRUCTIONS FOR THE SWARM INTELLIGENCE JOURNAL SUBMISSIONS OF NOVEL NATURAL METAPHOR ARTICLES

Swarm Intelligence manuscript No.
(will be inserted by the editor)

Swarm Intelligence: A few things you need to know if you want to publish in this journal

Marco Dorigo

Received: date / Accepted: date

Starting with Volume 11 the following actions will be implemented:

- Submission letter. At submission time, all the authors will be asked to declare that the manuscript is not submitted to another journal/conference, that it is free from plagiarism, that it was edited for language, and that a spell checker was used. Papers whose linguistic quality is too low will be rejected without being sent to referees. In the submission letter, the authors are also asked to state in one or two paragraphs what are the main contributions of their manuscript and to suggest at least three possible referees with whom they have no publications or projects in common and with whom they do not share their af liations.
- "Natural metaphor articles". There is a relatively recent trend that consists in taking a natural system/process and use it as a metaphor to generate an algorithm whose components have names taken from the natural system/process used as metaphor. This algorithm is often advertised as a "new natural metaphor algorithm" and used to solve a specif c problem (most of the time an optimization problem). Unfortunately, this approach has become so common that there are now hundreds of so-called "new" algorithms that are submitted (and unfortunately often also published) to journals and conferences every year. The problem is that it often takes a lot of work and ef ort for editors, and sometime referees, to understand why the authors are using the proposed metaphor, what is really new and what is the same as the old with just a new name, and whether the proposed algorithm is just a small incremental improvement of a known algorithm or a radically new idea. The number of such manuscripts submitted to Swarm Intelligence has greatly increased in the last few years. I have therefore asked the associate editors to pay particular attention to these "natural metaphor" inspired manuscripts and to send them to referees only if the manuscript seems to be of very high quality. In other words, I have asked the associate editors to increase the number of manuscripts that they reject directly so as to decrease the work load on referees, who are a precious resource that we need to protect. However, this is not enough and

IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

we have therefore recently decided that manuscripts submitted to Swarm Intelligence should refrain from "abusing" the natural metaphor approach. For example, optimization algorithms inspired by a natural phenomenon should explain what the natural phenomenon is optimizing and how. The natural phenomenon should be, therefore, a process that is already scientifically understood. Analogies (and nature-based inspiration) need to be matched to a clear, mathematically formal, explanation in terms of computational concepts, such as solutions, objective functions, neighborhoods, perturbations, and so on. It is the responsibility of the authors, and not of the referees, to explain how new nature-inspired proposals differ from already existing methods [1]. Being inspired by a different metaphor is not enough. In particular, it is not acceptable that the motivation for writing an article is the "new metaphor" [2]. Any manuscript that does not follow these guidelines will be rejected with a simple reference to this document as motivation.

– Experimental methodology. Most of swarm intelligence research is empirical in nature. Whenever this is the case, it is required that the evaluation of the proposed solution is done following a strict experimental protocol that includes (i) making data sets and the implementation of the used algorithms available to the readers, and (ii) using an appropriate number of experiment repetitions and appropriate statistical procedures to compare results. The performance of swarm intelligence algorithms, especially when used to solve continuous or combinatorial optimization problems, is often strongly dependent on the value of the algorithm parameters. Such values should be set using either sound statistical procedures as those from statistical design [3] or automatic parameter tuning procedures [4, 5, 6]; in all cases the data sets used for the tuning and evaluation phases should be clearly identified and the procedures used for setting the parameters must be reproducible.[1]
  Another important point is that authors should design their experiments to be as fair as possible. It is not acceptable to simply quote results published from other articles, to compare your new algorithm's results to these and say that your algorithm is better if the algorithms are not evaluated under the same experimental conditions. Finally, experimental comparisons should not be devoted solely to show that the authors' new algorithm is the best performing. In fact, way more interesting is to understand and explain why algorithms perform better (or worse). Competitive testing without new insights about the reasons behind the performance of algorithms is of little value and should be avoided [7].

November 2016 Marco Dorigo

References

1. D. Weyland. A rigorous analysis of the harmony search algorithm — How the research community can be misled by a "novel" methodology. International Journal of Applied Metaheuristic Computing, 1(2):50–60, 2010.

[1] Data sets, implemented algorithms and procedures used for setting the parameters must be submitted as supplementary material.

2. K. Sörensen. Metaheuristics: The metaphor exposed. International Transactions in Operational Research, 22(1):3–18, 2015.
3. D. C. Montgomery. Design and Analysis of Experiments. John Wiley & Sons, New York, 2012.
4. F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research, 36(1):267–306, 2009.
5. F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Learning and Intelligent Optimization: 5th International Conference, LION 5. Selected Papers, pages 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
6. M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, in press, 2016.
7. J. N. Hooker. Testing heuristics: We have it all wrong. Journal of Heuristics, 1(1):33–42, 1995.

These additional instructions for manuscript submission for the Swarm Intelligence Journal since Volume 11 are available at: https://media.springer.com/full/springer-instructions-for-authors-assets/pdf/1593723_Additional_submission_instructions.pdf, accessed 01/11/2021.

APPENDIX C: JOURNAL OF HEURISTIC POLICIES ON HEURISTIC SEARCH RESEARCH

The instructions for heuristic research for the Journal of Heuristics are available at: https://www.springer.com/journal/10732/updates/17199246, accessed 01/11/2021.

Journal of Heuristic Policies on Heuristic Search Research

These general policies apply to manuscripts submitted to the Journal of Heuristics and that belong to the general areas of heuristic search for optimization, including but not limited to metaheuristics, hyperheuristics, and matheuristics.

Metaphor-based methodologies

1. The optimization literature is rich with heuristic search methodologies for both discrete and continuous spaces. Proposing new paradigms is only acceptable if they contain innovative basic ideas, such as those that are embedded in classical frameworks like genetic algorithms, tabu search, and simulated annealing. The Journal of Heuristics avoids the publication of articles that repackage and embed old ideas in methods that are claimed to be based on metaphors of natural or man-made systems and processes. These so-called "novel" methods employ analogies that range from intelligent water drops, musicians playing jazz, imperialist societies, leapfrogs, kangaroos, all types of swarms and insects and even mine blast processes (Sörensen, 2013). If a researcher uses a metaphor to stimulate his or her own ideas about a new method, the method must nevertheless be translated into metaphor-free language, so that the strategies employed can be clearly understood, and their novelty is made clearly visible. (See items 2 and 3 below.) Metaphors are cheap and easy to come by. Their use to "window dress" a method is not acceptable.
2. The Journal of Heuristics is interested in advancing the area of heuristic search by publishing articles that, as mentioned by Sörensen (2013), adequately frame the methodology being applied within the existing optimization literature. Adequately framing a method entails deconstructing it and describing its components, measuring their contribution, and making connections to other procedures where these and/or similar components appear. Contributions must provide a clear explanation on how the components were adapted to the specific problem that is being solved. Implementations should be explained by employing standard optimization terminology, where a solution is called a "solution" and not something else related to some obscure metaphor (e.g., harmony, flies, bats, countries, etc.). In short, the journal embraces a component-based view of heuristic search.
3. The Journal of Heuristics fully endorses Sörensen's view that metaphor-based "novel" methods should not be published if they cannot demonstrate a contribution to their field. Renaming existing concepts does not count as a contribution. Even though these methods are often called "novel", many present no new ideas, except for the occasional marginal variant of an already existing methodology. These methods should not take the journal space of truly innovative ideas and research. Since they do not use the standard optimization vocabulary, they are unnecessarily difficult to understand.
4. The Journal of Heuristics considers new methodologies only if they are scientifically tested by following the principles outlined by Hooker (1995). Scientific testing entails the construction of controlled experiments to isolate the effects of algorithmic components as well as to investigate how problem characteristics influence the behavior of those components. The journal considers that there is little gain for the scientific community for yet another search method whose polished implementation is narrowly tested on benchmark instances of a single problem class.

Competitive testing and up-the-wall game

5. The Journal of Heuristics does not endorse the up-the-wall-game (Burke, et al. 2009). The idea of the up-the-wall game is to develop and apply a proposed search procedure to existing benchmark problems in order to compare it with other players. The goal is to get further "up the wall" than the other players. Although some competition among researchers or research groups could stimulate innovation, the ultimate goal of science is to understand (Burke, et al. 2009). True innovation in heuristic-search research is not achieved from yet another method that performs better than its competitors if there is no understanding as to why the method performs well (Sörensen, 2013).

6. The Journal of Heuristics favors the publication of meaningful insights over procedures that are tuned to perform better than others on a set of benchmark instances. In other words, the journal finds no value in conclusions stating that procedure X outperformed procedure Y if there is no insight related as to why this happened (Sörensen, 2013). Competitive testing fails to yield insight in the performance of algorithms (Hooker, 1995). The journal strives to assess the value of experimental results by their contribution to our understanding of heuristic search instead of whether they show that the polished implementation of a proposed method is able to win a race against the state of the art.

Development of customized solutions

7. The need for developing a customized solution to a problem must be justified. General-purpose solvers based on exact and heuristic methodologies should be tried first if the goal of the project is to solve a specific problem that requires a search procedure. If these general-purpose optimizers perform adequately for the application being considered, there is no need for a specialized procedure.

8. When the contribution is centered on developing a customized solution for a particular problem (e.g., those submitted to the area of Real-World Applications), considerable effort must be made to assess solution quality. Acceptable practices include but are not limited to measuring optimality gaps with lower or upper bounds and comparing solutions against known results or against results found with general-purpose optimizers. It is not acceptable to simply compare several versions of the same proposed solution method.

Statistically valid experiments and parameter tuning

9. The Journal of Heuristics requires that the authors conduct statistically valid computational experiments in order to support their statements about the performance of proposed procedures. Statistical validity refers to both the design of experiments and the analysis of the data. Barr, et al. (1995) present guidelines on how to design and perform statistically valid experiments.

10. For procedures that require parameter tuning, the available data must be partitioned into a training and a test set. Tuning should be performed in the training set only. Procedures that are tuned to solve a particular set of problems and that are not able to demonstrate their merit outside the chosen set of instances are of little interest.

References

Barr, R. S., B. L. Golden, J. P. Kelly, M. G. C. Resende and W. R. Stewart (1995) "Designing and Reporting on Computational Experiments with Heuristic Methods," *Journal of Heuristics*, vol. 1, no. 1, pp. 9-32

Burke, E. K., T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, and J. A. Vazquez-Rodriguez (2009) "Towards the Decathlon Challenge of Search Heuristics," http://www.cs.stir.ac.uk/~goc/papers/GECCO09Decwk1005-ochoaATS.pdf

Hooker, J. N. (1995) "Testing Heuristics: We Have It All Wrong," *Journal of Heuristics*, vol. 1, no. 1, 33-44.

Sörensen, K. (2013) "Metaheuristics – The Metaphor Exposed," *International Transactions in Operational Research*, http://onlinelibrary.wiley.com/doi/10.1111/itor.12001/abstract

APPENDIX D: COMPLETE RESULTS OF ALL THE ALGORITHMS TESTED

Tab. D.1 Complexity computation of examined swarm algorithms.

10 dimensions				
Algorithm	*T0*	*T1*	*Mean T2*	*Complexity*
BIA	2.68	1.07	4.75	1.37
CS	2.68	1.13	2.76	0.61
PSO	2.68	1.15	6.24	1.90
BAT	2.68	1.13	5.33	1.57
FFA	2.68	1.19	75.19	27.64
30 dimensions				
Algorithm	*T0*	*T1*	*Mean T2*	*Complexity*
BIA	2.63	1.26	9.93	3.30
CS	2.63	1.34	3.16	0.69
PSO	2.63	1.34	15.06	5.22
BAT	2.63	1.37	8.85	2.84
FFA	2.63	1.35	76.76	28.66
50 dimensions				
Algorithm	*T0*	*T1*	*Mean T2*	*Complexity*
BIA	2.63	1.66	16.02	5.45
CS	2.63	1.74	3.88	0.81
PSO	2.63	1.74	23.81	8.38
BAT	2.63	1.74	12.58	4.12
FFA	2.63	1.71	76.18	28.27
100 dimensions				
Algorithm	*T0*	*T1*	*Mean T2*	*Complexity*
BIA	2.84	3.47	31.00	9.70
CS	2.84	3.38	6.10	0.96
PSO	2.84	3.40	46.53	15.20
BAT	2.84	3.39	22.61	6.78
FFA	2.84	3.21	77.22	26.09

In Tables D.2-D.9 were highlighted algorithms, which performed significantly better according to the Wilcoxon Rank-Sum test (p=0.05) when compared to all the remaining swarm algorithms.

Tab. D.2 Complete statistics of all examined swarm algorithms on CEC 2015 in 10 dimensions.

	BIA		CS		PSO		BAT		FFA	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	$4.07E+04$	$3.60E+04$	$2.88E+02$	$3.25E+02$	$9.12E+04$	$3.99E+05$	$8.58E+06$	$7.19E+06$	$8.56E+05$	$8.51E+05$
f_2	$5.71E+03$	$5.91E+03$	$1.64E+02$	$1.97E+02$	$7.56E+03$	$6.91E+03$	$1.35E+09$	$6.45E+08$	$3.03E+04$	$4.71E+04$
f_3	$2.04E+01$	$6.57E-02$	$2.02E+01$	$5.56E-02$	$2.01E+01$	$1.17E-01$	$2.00E+01$	$5.15E-04$	$2.00E+01$	$1.80E-04$
f_4	$7.25E+00$	$5.45E+00$	$1.51E+01$	$4.93E+00$	$2.34E+01$	$8.26E+00$	$3.13E+01$	$1.26E+01$	$1.24E+01$	$5.35E+00$
f_5	$1.07E+03$	$3.12E+02$	$5.73E+02$	$1.55E+02$	$6.17E+02$	$2.06E+02$	$8.79E+02$	$2.58E+02$	$5.93E+02$	$2.50E+02$
f_6	$8.35E+02$	$1.09E+03$	$8.66E+01$	$5.08E+01$	$1.65E+03$	$1.98E+03$	$1.74E+04$	$2.54E+04$	$1.17E+04$	$1.67E+04$
f_7	$6.06E-01$	$6.15E-01$	$1.33E+00$	$2.61E-01$	$2.80E+00$	$1.24E+00$	$8.39E+00$	$1.56E+00$	$1.79E+00$	$4.58E-01$
f_8	$5.62E+02$	$5.39E+02$	$1.78E+01$	$1.43E+01$	$1.41E+03$	$1.41E+03$	$4.15E+03$	$3.42E+03$	$3.70E+03$	$6.32E+03$
f_9	$1.00E+02$	$4.96E-02$	$1.00E+02$	$7.12E-02$	$1.00E+02$	$1.77E-01$	$1.09E+02$	$4.20E+00$	$1.00E+02$	$5.39E-02$
f_{10}	$5.73E+02$	$3.89E+02$	$2.35E+02$	$1.90E+01$	$9.04E+02$	$8.29E+02$	$6.82E+03$	$6.25E+03$	$3.97E+03$	$3.54E+03$
f_{11}	$2.22E+02$	$1.30E+02$	$2.08E+02$	$1.36E+02$	$2.83E+02$	$7.09E+01$	$1.94E+02$	$7.45E+01$	$3.06E+02$	$1.01E+02$
f_{12}	$1.02E+02$	$5.45E-01$	$1.03E+02$	$6.34E-01$	$1.02E+02$	$8.94E-01$	$1.15E+02$	$4.96E+00$	$1.02E+02$	$5.79E-01$
f_{13}	$3.32E+01$	$2.88E+00$	$3.32E+01$	$2.27E+00$	$4.00E+01$	$3.61E+00$	$4.52E+01$	$2.80E+00$	$3.42E+01$	$3.61E+00$
f_{14}	$4.72E+03$	$2.95E+03$	$2.44E+03$	$1.09E+03$	$3.07E+03$	$3.50E+03$	$7.47E+03$	$1.34E+03$	$1.68E+03$	$2.28E+03$
f_{15}	$1.00E+02$	$0.00E+00$	$1.00E+02$	$0.00E+00$	$1.01E+02$	$4.20E+00$	$1.60E+02$	$1.69E+01$	$1.00E+02$	$1.45E-03$

Tab. D.3 Complete statistics of all examined swarm algorithms on CEC 2015 in 30 dimensions.

	BIA		CS		PSO		BAT		FFA	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	$2.66E+05$	$1.91E+05$	$4.70E+05$	$3.35E+05$	$2.89E+06$	$7.71E+06$	$3.10E+08$	$1.93E+08$	$6.94E+06$	$3.75E+06$
f_2	$1.53E+03$	$1.64E+03$	$1.42E+03$	$1.27E+03$	$3.21E+08$	$7.54E+08$	$3.61E+10$	$1.12E+10$	$2.94E+04$	$3.71E+04$
f_3	$2.10E+01$	$4.33E-02$	$2.09E+01$	$5.74E-02$	$2.05E+01$	$3.17E-01$	$2.00E+01$	$3.09E-04$	$2.00E+01$	$4.03E-04$
f_4	$7.13E+01$	$5.71E+01$	$1.44E+02$	$2.68E+01$	$1.41E+02$	$2.81E+01$	$2.45E+02$	$5.66E+01$	$6.40E+01$	$1.65E+01$
f_5	$6.74E+03$	$2.96E+02$	$3.53E+03$	$2.73E+02$	$3.17E+03$	$6.46E+02$	$4.65E+03$	$1.00E+03$	$2.79E+03$	$5.70E+02$
f_6	$5.25E+04$	$2.99E+04$	$5.78E+03$	$3.43E+03$	$1.12E+05$	$8.70E+04$	$5.20E+06$	$5.16E+06$	$3.53E+05$	$2.88E+05$
f_7	$9.53E+00$	$2.45E+00$	$1.14E+01$	$1.39E+00$	$1.49E+01$	$1.45E+01$	$2.27E+02$	$5.09E+01$	$1.07E+01$	$1.78E+00$
f_8	$2.88E+04$	$1.74E+04$	$2.04E+03$	$1.32E+03$	$3.60E+04$	$2.66E+04$	$5.06E+05$	$7.71E+05$	$2.17E+05$	$1.47E+05$
f_9	$1.05E+02$	$1.84E+01$	$1.04E+02$	$3.26E-01$	$1.50E+02$	$9.74E+01$	$2.69E+02$	$4.00E+01$	$1.24E+02$	$5.56E+01$
f_{10}	$7.73E+04$	$5.70E+04$	$2.06E+03$	$5.78E+02$	$1.15E+05$	$4.54E+05$	$2.94E+06$	$3.08E+06$	$5.26E+05$	$4.92E+05$
f_{11}	$4.50E+02$	$1.22E+02$	$3.38E+02$	$1.24E+02$	$5.97E+02$	$3.25E+02$	$8.69E+02$	$2.49E+02$	$4.36E+02$	$2.34E+01$
f_{12}	$1.05E+02$	$5.93E-01$	$1.08E+02$	$1.00E+00$	$1.16E+02$	$1.06E+01$	$1.67E+02$	$8.26E+00$	$1.07E+02$	$8.59E-01$
f_{13}	$1.16E+02$	$3.20E+00$	$1.24E+02$	$2.84E+00$	$1.31E+02$	$5.85E+00$	$1.66E+02$	$1.45E+01$	$1.26E+02$	$4.62E+00$
f_{14}	$3.30E+04$	$9.27E+02$	$3.26E+04$	$1.10E+03$	$3.80E+04$	$4.51E+03$	$6.83E+04$	$7.15E+03$	$1.08E+04$	$1.01E+04$
f_{15}	$1.00E+02$	$0.00E+00$	$1.00E+02$	$0.00E+00$	$1.10E+02$	$8.53E+00$	$4.05E+03$	$4.51E+03$	$1.00E+02$	$1.09E-03$

Tab. D.4 Complete statistics of all examined swarm algorithms on CEC 2015 in 50 dimensions.

	BIA		CS		PSO		BAT		FFA	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	$4.27E+05$	$2.17E+05$	$4.33E+06$	$1.39E+06$	$6.34E+06$	$1.14E+07$	$1.24E+09$	$6.72E+08$	$2.60E+07$	$1.11E+07$
f_2	$4.74E+03$	$3.37E+03$	$8.74E+02$	$1.61E+03$	$2.15E+09$	$1.48E+09$	$7.50E+10$	$2.50E+10$	$8.23E+04$	$9.54E+04$
f_3	$2.11E+01$	$3.10E-02$	$2.11E+01$	$5.79E-02$	$2.09E+01$	$2.41E-01$	$2.00E+01$	$3.42E-02$	$2.00E+01$	$4.93E-04$
f_4	$1.48E+02$	$1.28E+02$	$3.33E+02$	$5.64E+01$	$3.01E+02$	$2.79E+01$	$4.95E+02$	$7.45E+01$	$1.32E+02$	$2.87E+01$
f_5	$1.26E+04$	$3.79E+02$	$6.67E+03$	$3.61E+02$	$5.34E+03$	$8.32E+02$	$8.96E+03$	$1.99E+03$	$5.03E+03$	$7.76E+02$
f_6	$1.83E+05$	$1.66E+05$	$1.50E+05$	$8.66E+04$	$4.87E+05$	$6.64E+05$	$1.59E+07$	$1.89E+07$	$1.25E+06$	$7.03E+05$
f_7	$5.50E+01$	$3.28E+01$	$3.54E+01$	$2.00E+01$	$3.48E+01$	$2.26E+01$	$6.87E+02$	$1.43E+02$	$2.39E+01$	$2.42E+00$
f_8	$1.11E+05$	$7.31E+04$	$2.48E+04$	$1.36E+04$	$1.93E+05$	$2.29E+05$	$4.85E+06$	$4.90E+06$	$7.63E+05$	$4.85E+05$
f_9	$1.23E+02$	$8.05E+01$	$1.07E+02$	$5.16E-01$	$2.51E+02$	$1.64E+02$	$5.75E+02$	$9.29E+01$	$1.19E+02$	$5.78E+01$
f_{10}	$2.04E+04$	$1.83E+04$	$2.82E+03$	$4.40E+02$	$1.73E+05$	$5.19E+05$	$7.18E+06$	$1.22E+07$	$1.27E+06$	$6.69E+05$
f_{11}	$6.70E+02$	$6.13E+01$	$9.90E+02$	$6.12E+02$	$1.16E+03$	$4.07E+02$	$2.35E+03$	$4.29E+02$	$5.27E+02$	$5.73E+01$
f_{12}	$1.08E+02$	$8.29E-01$	$1.12E+02$	$1.27E+00$	$1.73E+02$	$2.12E+01$	$2.35E+02$	$1.64E+01$	$1.11E+02$	$1.35E+00$
f_{13}	$2.17E+02$	$4.09E+00$	$2.22E+02$	$4.39E+00$	$2.34E+02$	$7.07E+00$	$4.21E+02$	$7.40E+01$	$2.23E+02$	$4.79E+00$
f_{14}	$5.42E+04$	$1.01E+04$	$6.23E+04$	$8.35E+03$	$9.15E+04$	$1.98E+04$	$1.75E+05$	$1.51E+04$	$2.72E+04$	$2.92E+04$
f_{15}	$1.00E+02$	$0.00E+00$	$1.00E+02$	$1.25E-01$	$1.08E+02$	$6.66E+00$	$9.33E+03$	$1.08E+04$	$1.00E+02$	$1.07E-03$

Tab. D.5 Complete statistics of all examined swarm algorithms on CEC 2015 in 100 dimensions.

	BIA		CS		PSO		BAT		FFA	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	1.34E+06	4.70E+05	5.35E+06	1.70E+06	8.49E+06	4.60E+06	3.14E+09	2.42E+09	4.73E+07	1.22E+07
f_2	1.04E+03	1.76E+03	1.28E+03	2.01E+03	9.63E+09	6.14E+09	1.01E+11	8.09E+10	6.65E+04	8.26E+04
f_3	2.13E+01	2.06E−02	2.13E+01	3.47E−02	2.12E+01	1.09E−01	2.00E+01	7.61E−03	2.00E+01	7.08E−04
f_4	5.11E+02	3.18E+02	8.81E+02	7.60E+01	7.08E+02	6.05E+01	1.14E+03	1.18E+02	3.53E+02	5.45E+01
f_5	2.94E+04	4.93E+02	1.69E+04	5.51E+02	1.24E+04	1.41E+03	2.24E+04	3.87E+03	1.15E+04	1.29E+03
f_6	4.15E+05	1.07E+05	2.81E+06	7.34E+05	2.05E+06	8.57E+05	2.87E+08	2.32E+08	4.81E+06	2.02E+06
f_7	1.47E+02	3.32E+01	1.47E+02	3.33E+01	1.61E+02	5.49E+01	2.49E+03	1.03E+03	5.98E+01	1.26E+01
f_8	2.01E+05	1.08E+05	1.06E+06	4.18E+05	7.46E+05	4.98E+05	6.41E+07	8.18E+07	3.41E+06	1.29E+06
f_9	1.08E+02	5.71E−01	1.13E+02	8.35E−01	7.18E+02	2.64E+02	1.77E+03	2.11E+02	1.56E+02	1.49E+02
f_{10}	4.56E+04	1.61E+05	8.74E+03	4.30E+03	2.12E+06	3.62E+06	1.26E+08	1.91E+08	6.00E+06	2.38E+06
f_{11}	1.24E+03	4.64E+02	2.59E+03	1.11E+03	1.86E+03	1.16E+03	5.43E+03	2.99E+02	9.79E+02	1.38E+02
f_{12}	1.19E+02	8.88E−01	1.19E+02	1.24E+00	2.54E+02	2.62E+01	3.89E+02	2.60E+01	1.21E+02	1.36E+00
f_{13}	4.60E+02	5.41E+00	4.62E+02	5.92E+00	4.89E+02	1.37E+01	1.50E+03	3.16E+02	4.69E+02	5.39E+00
f_{14}	1.45E+05	5.02E+04	1.11E+05	8.98E+03	2.51E+05	4.98E+04	6.08E+05	6.78E+04	3.23E+04	4.77E+04
f_{15}	1.04E+02	6.29E+00	1.16E+02	5.26E+00	1.16E+02	8.00E+00	5.09E+04	1.60E+05	1.00E+02	1.50E−03

Tab. D.6 Complete statistics of all examined swarm algorithms on CEC 2017 in 10 dimensions.

	BIA		CS		PSO		BAT		FFA	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	4.78E+02	6.57E+02	7.78E+00	1.24E+01	1.84E+03	2.41E+03	1.36E+09	6.72E+08	2.18E+04	3.07E+04
f_2	9.51E−06	1.25E−05	5.97E−07	5.61E−07	1.45E−05	1.67E−05	2.47E+08	4.19E+08	4.98E+01	8.18E+01
f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.34E−14	2.44E−14	7.80E+03	2.57E+03	2.44E−04	1.09E−04
f_4	3.29E−01	2.76E−01	8.99E−02	2.40E−01	7.01E+00	1.63E+01	1.01E+02	6.30E+01	1.78E+00	8.76E−01
f_5	7.27E+00	6.28E+00	1.50E+01	5.27E+00	2.85E+01	9.05E+00	3.22E+01	1.20E+01	1.10E+01	5.48E+00
f_6	1.39E−05	5.92E−05	4.59E−02	6.00E−02	4.31E+00	5.69E+00	2.92E+01	5.28E+00	2.70E−02	2.54E−02
f_7	2.43E+01	8.37E+00	2.70E+01	5.13E+00	2.02E+01	5.26E+00	1.17E+02	3.75E+01	2.11E+01	6.14E+00
f_8	7.89E+00	7.25E+00	1.56E+01	5.50E+00	1.58E+01	7.40E+00	3.34E+01	1.27E+01	1.23E+01	5.47E+00
f_9	1.41E−01	3.79E−01	1.09E+00	6.01E+00	8.92E−15	3.09E−14	4.75E+02	2.03E+02	3.56E−03	1.76E−02
f_{10}	1.08E+03	3.42E+02	6.04E+02	1.48E+02	7.15E+02	2.64E+02	8.93E+02	2.96E+02	6.03E+02	2.84E+02
f_{11}	2.89E+00	2.62E+00	2.75E+00	1.18E+00	2.71E+01	1.37E+01	1.62E+02	7.63E+01	5.21E+01	4.26E+01
f_{12}	1.05E+04	8.57E+03	2.16E+03	1.57E+03	1.43E+04	1.19E+04	3.66E+06	6.04E+06	1.32E+05	1.09E+05
f_{13}	2.98E+03	2.64E+03	1.27E+01	5.79E+00	6.15E+03	4.71E+03	1.30E+04	1.07E+04	4.94E+04	6.61E+04
f_{14}	3.32E+01	7.77E+00	9.03E+00	4.93E+00	5.68E+01	3.94E+01	2.41E+02	1.85E+02	2.61E+02	1.75E+02
f_{15}	2.92E+01	1.90E+01	2.35E+00	1.02E+00	1.24E+02	1.68E+02	2.25E+03	1.60E+03	1.04E+03	9.68E+02
f_{16}	2.31E+01	4.81E+01	5.53E+00	8.08E+00	2.13E+02	1.27E+02	1.49E+02	1.02E+02	5.95E+01	6.38E+01
f_{17}	3.59E+01	2.08E+01	2.75E+01	6.52E+00	5.23E+01	2.55E+01	1.11E+02	2.64E+01	4.95E+01	2.89E+01
f_{18}	3.82E+03	4.03E+03	2.73E+01	6.33E+00	1.04E+04	1.03E+04	1.66E+04	2.23E+04	6.63E+04	6.77E+04
f_{19}	2.36E+01	2.39E+01	2.59E+00	7.53E−01	3.84E+02	6.65E+02	2.68E+03	2.93E+03	2.95E+02	3.51E+02
f_{20}	5.45E+00	1.09E+01	1.65E+01	7.53E+00	9.81E+01	6.12E+01	1.16E+02	2.71E+01	4.72E+01	2.64E+01
f_{21}	1.21E+02	4.12E+01	1.22E+02	4.55E+01	2.22E+02	3.79E+01	1.23E+02	1.14E+01	2.10E+02	2.33E+01
f_{22}	1.02E+02	1.15E+01	8.32E+01	3.01E+01	1.36E+02	2.08E+02	2.09E+02	1.15E+02	1.01E+02	7.48E−01
f_{23}	1.02E+02	1.15E+01	8.32E+01	3.01E+01	1.36E+02	2.08E+02	2.09E+02	1.15E+02	1.01E+02	7.48E−01
f_{24}	3.09E+02	5.36E+00	3.17E+02	5.06E+00	3.67E+02	2.81E+01	3.43E+02	3.38E+01	3.16E+02	6.44E+00
f_{25}	3.23E+02	4.21E+01	1.74E+02	9.76E+01	3.25E+02	1.13E+02	2.52E+02	5.15E+01	3.38E+02	3.49E+01
f_{26}	4.36E+02	1.96E+01	3.50E+02	1.10E+02	4.17E+02	5.04E+01	5.21E+02	4.32E+01	4.30E+02	2.18E+01
f_{27}	2.83E+02	5.89E+01	2.21E+02	9.48E+01	3.29E+02	1.84E+02	5.60E+02	7.85E+01	3.65E+02	1.05E+02
f_{28}	3.99E+02	5.29E+00	3.89E+02	9.20E−01	4.32E+02	3.19E+01	4.31E+02	1.29E+01	3.77E+02	2.52E+01
f_{29}	4.54E+02	1.48E+02	2.97E+02	3.52E+01	5.22E+02	1.40E+02	4.85E+02	5.45E+01	4.51E+02	3.66E+01
f_{30}	2.77E+02	2.13E+01	2.80E+02	2.16E+01	3.15E+02	4.77E+01	4.08E+02	4.28E+01	2.79E+02	4.73E+01

Tab. D.7 Complete statistics of all examined swarm algorithms on CEC 2017 in 30 dimensions.

	BIA		**CS**		**PSO**		**BAT**		**FFA**	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	$2.15E+03$	$2.16E+03$	$1.00E-01$	$2.58E-01$	$4.71E+07$	$2.38E+08$	$2.41E+10$	$9.39E+09$	$2.91E+04$	$2.22E+04$
f_2	$1.04E+11$	$5.38E+11$	$1.20E+11$	$5.41E+11$	$5.42E+23$	$3.87E+24$	$2.29E+38$	$1.40E+39$	$7.16E+05$	$4.14E+06$
f_3	$7.54E+01$	$6.99E+01$	$1.46E+03$	$9.14E+02$	$2.77E-04$	$3.56E-04$	$7.37E+04$	$1.95E+04$	$7.80E-03$	$2.10E-03$
f_4	$1.11E+01$	$2.13E+01$	$4.43E+01$	$3.59E+01$	$8.30E+01$	$2.54E+01$	$3.74E+03$	$1.39E+03$	$2.65E+01$	$1.35E+01$
f_5	$7.50E+01$	$6.12E+01$	$1.43E+02$	$2.95E+01$	$1.45E+02$	$2.80E+01$	$2.41E+02$	$4.53E+01$	$6.56E+01$	$1.93E+01$
f_6	$1.16E-03$	$6.20E-03$	$2.60E+01$	$1.09E+01$	$3.14E+01$	$1.09E+01$	$5.69E+01$	$7.26E+00$	$2.48E-01$	$2.12E-01$
f_7	$1.83E+02$	$3.31E+01$	$2.22E+02$	$3.87E+01$	$1.06E+02$	$2.51E+01$	$9.13E+02$	$2.03E+02$	$9.77E+01$	$1.93E+01$
f_8	$4.70E+01$	$4.79E+01$	$1.38E+02$	$2.52E+01$	$1.09E+02$	$2.12E+01$	$1.94E+02$	$4.10E+01$	$6.18E+01$	$2.00E+01$
f_9	$6.56E+00$	$7.44E+00$	$3.36E+03$	$1.38E+03$	$2.30E+03$	$8.13E+02$	$5.25E+03$	$1.53E+03$	$2.35E-01$	$4.23E-01$
f_{10}	$6.97E+03$	$2.67E+02$	$3.78E+03$	$2.36E+02$	$3.18E+03$	$6.18E+02$	$5.32E+03$	$9.52E+02$	$2.79E+03$	$5.74E+02$
f_{11}	$3.20E+01$	$2.57E+01$	$8.34E+01$	$2.50E+01$	$1.09E+02$	$3.66E+01$	$2.53E+03$	$1.33E+03$	$7.13E+02$	$3.67E+02$
f_{12}	$2.61E+04$	$1.42E+04$	$6.66E+04$	$5.21E+04$	$2.94E+05$	$1.36E+06$	$1.71E+09$	$1.05E+09$	$1.09E+07$	$8.78E+06$
f_{13}	$1.28E+04$	$9.36E+03$	$3.64E+03$	$3.73E+03$	$1.88E+05$	$8.68E+05$	$6.15E+07$	$2.17E+08$	$3.06E+06$	$3.56E+06$
f_{14}	$4.89E+03$	$4.70E+03$	$9.92E+01$	$2.37E+01$	$5.05E+03$	$5.18E+03$	$7.56E+04$	$1.20E+05$	$1.64E+04$	$2.09E+04$
f_{15}	$4.05E+03$	$4.63E+03$	$2.45E+02$	$1.25E+02$	$7.10E+03$	$9.35E+03$	$5.71E+06$	$2.31E+07$	$1.59E+06$	$1.60E+06$
f_{16}	$9.35E+02$	$4.53E+02$	$8.68E+02$	$2.05E+02$	$9.36E+02$	$2.68E+02$	$1.82E+03$	$5.58E+02$	$6.84E+02$	$2.56E+02$
f_{17}	$1.25E+02$	$1.07E+02$	$2.58E+02$	$1.07E+02$	$5.73E+02$	$1.87E+02$	$1.01E+03$	$2.60E+02$	$5.27E+02$	$2.08E+02$
f_{18}	$1.75E+05$	$1.49E+05$	$2.83E+04$	$1.40E+04$	$1.22E+05$	$1.01E+05$	$1.08E+06$	$1.27E+06$	$6.98E+05$	$5.12E+05$
f_{19}	$6.22E+03$	$6.72E+03$	$1.64E+02$	$1.53E+02$	$7.91E+03$	$9.42E+03$	$4.73E+06$	$8.41E+06$	$2.89E+05$	$1.45E+05$
f_{20}	$1.99E+02$	$1.35E+02$	$3.63E+02$	$1.19E+02$	$4.82E+02$	$1.79E+02$	$6.84E+02$	$1.22E+02$	$4.47E+02$	$1.54E+02$
f_{21}	$2.69E+02$	$5.80E+01$	$3.18E+02$	$4.62E+01$	$3.36E+02$	$3.04E+01$	$4.09E+02$	$5.18E+01$	$2.61E+02$	$1.40E+01$
f_{22}	$1.00E+02$	$5.84E-01$	$2.74E+03$	$1.98E+03$	$1.10E+03$	$1.65E+03$	$4.51E+03$	$1.03E+03$	$2.90E+03$	$7.55E+02$
f_{23}	$3.81E+02$	$1.35E+01$	$4.91E+02$	$2.92E+01$	$6.62E+02$	$8.07E+01$	$8.40E+02$	$7.31E+01$	$4.24E+02$	$2.32E+01$
f_{24}	$4.48E+02$	$1.85E+01$	$5.56E+02$	$2.83E+01$	$7.02E+02$	$6.64E+01$	$8.75E+02$	$1.11E+02$	$4.96E+02$	$1.79E+01$
f_{25}	$3.88E+02$	$2.07E+00$	$3.87E+02$	$5.69E+00$	$3.92E+02$	$1.10E+01$	$1.43E+03$	$7.61E+02$	$3.80E+02$	$3.44E+00$
f_{26}	$1.18E+03$	$7.39E+02$	$1.65E+03$	$1.05E+03$	$1.14E+03$	$1.35E+03$	$5.49E+03$	$9.17E+02$	$1.69E+03$	$2.29E+02$
f_{27}	$5.32E+02$	$1.15E+01$	$5.22E+02$	$1.02E+01$	$6.02E+02$	$8.58E+01$	$1.02E+03$	$1.04E+02$	$5.00E+02$	$2.27E-04$
f_{28}	$3.29E+02$	$4.76E+01$	$3.50E+02$	$5.58E+01$	$4.22E+02$	$4.17E+01$	$1.97E+03$	$5.67E+02$	$5.00E+02$	$2.17E-04$
f_{29}	$5.85E+02$	$1.74E+02$	$8.90E+02$	$1.10E+02$	$9.93E+02$	$2.35E+02$	$2.27E+03$	$3.75E+02$	$9.40E+02$	$1.77E+02$
f_{30}	$4.53E+03$	$1.65E+03$	$6.49E+03$	$3.28E+03$	$5.96E+03$	$3.51E+03$	$4.13E+07$	$6.05E+07$	$7.64E+05$	$3.89E+05$

Tab. D.8 Complete statistics of all examined swarm algorithms on CEC 2017 in 50 dimensions.

	BIA		**CS**		**PSO**		**BAT**		**FFA**	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	$2.35E+03$	$2.52E+03$	$4.24E+03$	$6.36E+03$	$5.48E+08$	$1.06E+09$	$5.16E+10$	$2.25E+10$	$8.23E+04$	$1.06E+05$
f_2	$3.65E+26$	$2.61E+27$	$6.96E+36$	$4.97E+37$	$3.80E+47$	$2.71E+48$	$3.63E+71$	$2.58E+72$	$1.10E+15$	$7.85E+15$
f_3	$1.16E+04$	$3.16E+03$	$2.72E+04$	$5.91E+03$	$7.59E+00$	$4.40E+00$	$1.37E+05$	$3.50E+04$	$3.04E-02$	$6.55E-03$
f_4	$6.50E+01$	$5.03E+01$	$7.80E+01$	$4.83E+01$	$1.65E+02$	$1.07E+02$	$1.17E+04$	$5.56E+03$	$4.94E+01$	$1.70E+01$
f_5	$1.42E+02$	$1.19E+02$	$3.15E+02$	$4.04E+01$	$2.52E+02$	$3.18E+01$	$4.46E+02$	$5.70E+01$	$1.33E+02$	$3.33E+01$
f_6	$2.32E-03$	$4.30E-03$	$4.64E+01$	$9.61E+00$	$4.28E+01$	$6.44E+00$	$6.67E+01$	$8.40E+00$	$6.16E-01$	$5.56E-01$
f_7	$3.54E+02$	$7.21E+01$	$5.49E+02$	$7.60E+01$	$2.04E+02$	$4.08E+01$	$1.87E+03$	$3.33E+02$	$1.86E+02$	$3.40E+01$
f_8	$1.52E+02$	$1.25E+02$	$3.22E+02$	$3.85E+01$	$2.57E+02$	$3.50E+01$	$4.61E+02$	$6.33E+01$	$1.23E+02$	$2.65E+01$
f_9	$3.08E+01$	$3.81E+01$	$1.31E+04$	$3.78E+03$	$8.47E+03$	$1.44E+03$	$1.55E+04$	$3.73E+03$	$2.89E+02$	$2.04E+03$
f_{10}	$1.29E+04$	$4.59E+02$	$6.93E+03$	$3.88E+02$	$5.67E+03$	$8.69E+02$	$9.20E+03$	$1.98E+03$	$5.19E+03$	$1.03E+03$
f_{11}	$5.40E+01$	$2.93E+01$	$1.88E+02$	$3.77E+01$	$1.64E+02$	$3.55E+01$	$7.43E+03$	$3.81E+03$	$1.62E+03$	$5.35E+02$
f_{12}	$4.15E+05$	$2.78E+05$	$6.58E+05$	$6.63E+05$	$9.27E+07$	$3.53E+08$	$1.04E+10$	$6.53E+09$	$5.23E+07$	$3.43E+07$
f_{13}	$1.93E+03$	$2.99E+03$	$7.51E+03$	$6.44E+03$	$1.88E+05$	$1.30E+06$	$1.41E+09$	$3.36E+09$	$4.30E+06$	$4.31E+06$
f_{14}	$3.05E+04$	$2.25E+04$	$3.28E+02$	$8.58E+01$	$5.00E+04$	$6.81E+04$	$1.65E+06$	$2.54E+06$	$1.27E+05$	$1.06E+05$
f_{15}	$4.34E+03$	$4.03E+03$	$1.46E+03$	$9.94E+02$	$7.27E+03$	$6.32E+03$	$1.78E+08$	$5.74E+08$	$3.42E+06$	$3.79E+06$
f_{16}	$8.86E+02$	$5.30E+02$	$1.83E+03$	$2.16E+02$	$1.46E+03$	$3.23E+02$	$3.74E+03$	$1.11E+03$	$1.43E+03$	$4.06E+02$
f_{17}	$1.25E+03$	$5.12E+02$	$1.24E+03$	$1.80E+02$	$1.20E+03$	$3.06E+02$	$4.01E+03$	$1.03E+03$	$2.02E+03$	$6.38E+02$
f_{18}	$1.15E+06$	$5.83E+05$	$1.02E+05$	$4.47E+04$	$2.58E+05$	$1.85E+05$	$1.06E+07$	$1.49E+07$	$1.24E+06$	$8.32E+05$
f_{19}	$1.49E+04$	$5.68E+03$	$2.41E+03$	$2.09E+03$	$1.44E+04$	$9.25E+03$	$2.18E+06$	$3.50E+06$	$8.54E+05$	$1.97E+05$
f_{20}	$1.07E+03$	$5.28E+02$	$1.12E+03$	$2.27E+02$	$9.38E+02$	$3.17E+02$	$1.57E+03$	$2.98E+02$	$1.10E+03$	$2.60E+02$
f_{21}	$3.34E+02$	$1.23E+02$	$4.78E+02$	$3.76E+01$	$4.82E+02$	$4.74E+01$	$6.95E+02$	$1.01E+02$	$3.32E+02$	$2.45E+01$
f_{22}	$7.49E+03$	$6.51E+03$	$7.53E+03$	$4.27E+02$	$6.68E+03$	$8.09E+02$	$9.98E+03$	$2.14E+03$	$5.26E+03$	$9.38E+02$
f_{23}	$4.88E+02$	$1.89E+01$	$7.75E+02$	$5.34E+01$	$1.02E+03$	$1.28E+02$	$1.52E+03$	$1.49E+02$	$5.70E+02$	$2.85E+01$
f_{24}	$5.66E+02$	$4.38E+01$	$8.44E+02$	$6.21E+01$	$1.06E+03$	$1.26E+02$	$1.62E+03$	$2.02E+02$	$6.51E+02$	$4.16E+01$
f_{25}	$5.50E+02$	$2.80E+01$	$5.32E+02$	$5.01E+01$	$5.50E+02$	$3.34E+01$	$5.99E+03$	$4.40E+03$	$4.33E+02$	$1.11E+01$
f_{26}	$1.86E+03$	$6.58E+02$	$4.74E+03$	$8.96E+02$	$3.08E+03$	$2.86E+03$	$1.26E+04$	$1.69E+03$	$2.80E+03$	$3.55E+02$
f_{27}	$6.50E+02$	$4.17E+01$	$7.22E+02$	$9.16E+01$	$8.83E+02$	$1.74E+02$	$2.53E+03$	$2.66E+02$	$5.00E+02$	$2.89E-04$
f_{28}	$4.90E+02$	$2.68E+01$	$4.92E+02$	$2.66E+01$	$5.84E+02$	$1.87E+02$	$5.05E+03$	$2.16E+03$	$5.00E+02$	$2.71E-04$
f_{29}	$6.59E+02$	$1.80E+02$	$1.50E+03$	$2.09E+02$	$1.64E+03$	$3.69E+02$	$7.38E+03$	$1.49E+03$	$1.94E+03$	$4.79E+02$
f_{30}	$1.10E+06$	$2.14E+05$	$6.61E+05$	$9.03E+04$	$8.86E+05$	$3.85E+05$	$3.34E+08$	$4.90E+08$	$3.04E+06$	$2.05E+06$

Tab. D.9 Complete statistics of all examined swarm algorithms on CEC 2017 in 100 dimensions.

	BIA		**CS**		**PSO**		**BAT**		**FFA**	
	avg	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*	*avg*	*std*
f_1	$4.34E+03$	$4.93E+03$	$6.95E+03$	$7.81E+03$	$3.62E+09$	$3.22E+09$	$7.77E+10$	$6.34E+10$	$9.38E+04$	$1.14E+05$
f_2	$2.25E+81$	$1.59E+82$	$2.04E+66$	$9.38E+66$	$1.18E+95$	$6.17E+95$	$4.60E+164$	$2.34E+165$	$6.32E+40$	$4.51E+41$
f_3	$1.18E+05$	$1.67E+04$	$1.84E+05$	$2.00E+04$	$2.03E+03$	$7.16E+02$	$3.77E+05$	$1.24E+05$	$9.70E+02$	$4.18E+03$
f_4	$1.29E+02$	$5.15E+01$	$2.28E+02$	$6.49E+01$	$4.69E+02$	$2.09E+02$	$3.01E+04$	$2.35E+04$	$1.02E+02$	$2.25E+01$
f_5	$6.19E+02$	$2.89E+02$	$8.23E+02$	$7.68E+01$	$6.44E+02$	$6.27E+01$	$1.05E+03$	$1.60E+02$	$3.47E+02$	$4.89E+01$
f_6	$2.37E-02$	$2.09E-02$	$5.79E+01$	$8.14E+00$	$5.02E+01$	$3.99E+00$	$6.98E+01$	$5.43E+00$	$9.49E+00$	$1.21E+01$
f_7	$9.28E+02$	$1.15E+02$	$1.83E+03$	$1.90E+02$	$5.12E+02$	$1.16E+02$	$4.46E+03$	$5.70E+02$	$4.42E+02$	$6.25E+01$
f_8	$5.54E+02$	$3.05E+02$	$8.54E+02$	$7.36E+01$	$6.99E+02$	$7.43E+01$	$1.20E+03$	$1.41E+02$	$3.57E+02$	$6.11E+01$
f_9	$1.87E+02$	$3.02E+02$	$4.18E+04$	$7.07E+03$	$1.92E+04$	$2.17E+03$	$3.06E+04$	$4.44E+03$	$4.49E+03$	$7.47E+03$
f_{10}	$2.93E+04$	$4.46E+02$	$1.68E+04$	$6.02E+02$	$1.22E+04$	$1.20E+03$	$2.23E+04$	$4.91E+03$	$1.12E+04$	$1.07E+03$
f_{11}	$5.23E+02$	$2.15E+02$	$1.30E+03$	$4.76E+02$	$9.72E+02$	$1.61E+02$	$8.93E+04$	$5.95E+04$	$7.60E+03$	$2.26E+03$
f_{12}	$6.77E+05$	$2.42E+05$	$8.24E+05$	$3.29E+05$	$1.66E+09$	$2.37E+09$	$5.08E+10$	$3.97E+10$	$1.52E+08$	$7.28E+07$
f_{13}	$3.63E+03$	$2.66E+03$	$6.68E+03$	$5.62E+03$	$5.13E+07$	$1.55E+08$	$1.41E+09$	$5.87E+09$	$3.56E+06$	$2.15E+06$
f_{14}	$1.08E+05$	$3.54E+04$	$9.24E+04$	$4.93E+04$	$2.55E+05$	$1.06E+05$	$9.10E+06$	$1.42E+07$	$6.84E+05$	$5.09E+05$
f_{15}	$9.06E+02$	$1.16E+03$	$3.89E+03$	$3.78E+03$	$2.88E+03$	$2.93E+03$	$1.22E+09$	$3.05E+09$	$2.70E+06$	$1.57E+06$
f_{16}	$4.34E+03$	$2.20E+03$	$4.50E+03$	$3.03E+02$	$3.27E+03$	$6.82E+02$	$1.25E+04$	$3.01E+03$	$3.95E+03$	$8.28E+02$
f_{17}	$3.86E+03$	$1.04E+03$	$3.25E+03$	$3.10E+02$	$2.99E+03$	$5.27E+02$	$4.04E+04$	$1.19E+05$	$7.15E+03$	$1.63E+03$
f_{18}	$1.52E+06$	$7.71E+05$	$7.81E+05$	$2.50E+05$	$8.03E+05$	$4.04E+05$	$1.57E+07$	$2.96E+07$	$1.95E+06$	$8.29E+05$
f_{19}	$1.49E+03$	$2.23E+03$	$3.24E+03$	$3.69E+03$	$1.00E+04$	$1.37E+04$	$4.98E+08$	$1.97E+09$	$2.89E+06$	$4.19E+05$
f_{20}	$4.35E+03$	$6.19E+02$	$3.52E+03$	$2.18E+02$	$2.74E+03$	$5.30E+02$	$4.24E+03$	$6.10E+02$	$2.84E+03$	$5.26E+02$
f_{21}	$6.68E+02$	$3.26E+02$	$1.03E+03$	$8.18E+01$	$1.15E+03$	$1.31E+02$	$1.72E+03$	$2.00E+02$	$6.11E+02$	$6.37E+01$
f_{22}	$2.99E+04$	$4.29E+03$	$1.84E+04$	$6.56E+02$	$1.52E+04$	$1.60E+03$	$2.40E+04$	$4.28E+03$	$1.16E+04$	$1.05E+03$
f_{23}	$7.22E+02$	$3.05E+01$	$1.23E+03$	$8.88E+01$	$2.07E+03$	$2.53E+02$	$3.25E+03$	$2.66E+02$	$9.45E+02$	$6.42E+01$
f_{24}	$1.07E+03$	$4.15E+01$	$1.84E+03$	$1.02E+02$	$3.14E+03$	$4.87E+02$	$6.21E+03$	$4.63E+02$	$1.24E+03$	$5.67E+01$
f_{25}	$7.65E+02$	$6.05E+01$	$8.04E+02$	$7.83E+01$	$7.80E+02$	$9.01E+01$	$6.73E+03$	$6.02E+03$	$7.15E+02$	$4.64E+01$
f_{26}	$5.72E+03$	$2.22E+03$	$1.41E+04$	$1.35E+03$	$9.20E+03$	$6.37E+03$	$4.30E+04$	$5.51E+03$	$7.63E+03$	$5.68E+02$
f_{27}	$7.65E+02$	$2.98E+01$	$9.65E+02$	$8.95E+01$	$1.03E+03$	$1.46E+02$	$6.36E+03$	$8.06E+02$	$5.00E+02$	$2.81E-04$
f_{28}	$5.74E+02$	$9.71E+01$	$5.71E+02$	$2.98E+01$	$8.18E+02$	$3.22E+02$	$1.30E+04$	$7.80E+03$	$5.00E+02$	$2.50E-04$
f_{29}	$2.04E+03$	$6.94E+02$	$4.38E+03$	$2.49E+02$	$3.86E+03$	$5.30E+02$	$3.95E+04$	$2.52E+04$	$5.86E+03$	$1.18E+03$
f_{30}	$1.19E+04$	$7.59E+03$	$7.00E+03$	$4.25E+03$	$8.68E+07$	$2.28E+08$	$2.45E+09$	$3.99E+09$	$7.15E+06$	$4.58E+06$

APPENDIX E: STATISTIC OUTPUT OF THE STANDARD BISON ALGORITHM

Tab. E.1 Statistics of the Bison Algorithm on CEC 2015 in 10 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$1.75E+03$	$1.44E+05$	$4.07E+04$	$2.75E+04$	$3.60E+04$
f_2	$1.95E-02$	$2.40E+04$	$5.71E+03$	$3.88E+03$	$5.91E+03$
f_3	$2.02E+01$	$2.05E+01$	$2.04E+01$	$2.04E+01$	$6.57E-02$
f_4	$9.95E-01$	$2.34E+01$	$7.25E+00$	$5.97E+00$	$5.45E+00$
f_5	$3.73E+00$	$1.52E+03$	$1.07E+03$	$1.10E+03$	$3.12E+02$
f_6	$1.01E+01$	$4.71E+03$	$8.35E+02$	$3.55E+02$	$1.09E+03$
f_7	$1.94E-02$	$2.67E+00$	$6.06E-01$	$3.78E-01$	$6.15E-01$
f_8	$8.50E+00$	$2.21E+03$	$5.62E+02$	$3.81E+02$	$5.39E+02$
f_9	$1.00E+02$	$1.00E+02$	$1.00E+02$	$1.00E+02$	$4.96E-02$
f_{10}	$2.37E+02$	$2.25E+03$	$5.73E+02$	$4.64E+02$	$3.89E+02$
f_{11}	$7.99E-01$	$3.00E+02$	$2.22E+02$	$3.00E+02$	$1.30E+02$
f_{12}	$1.01E+02$	$1.03E+02$	$1.02E+02$	$1.02E+02$	$5.45E-01$
f_{13}	$2.88E+01$	$4.08E+01$	$3.32E+01$	$3.27E+01$	$2.88E+00$
f_{14}	$1.00E+02$	$1.16E+04$	$4.72E+03$	$5.56E+03$	$2.95E+03$
f_{15}	$1.00E+02$	$1.00E+02$	$1.00E+02$	$1.00E+02$	$0.00E+00$

Tab. E.2 Statistics of the Bison Algorithm on CEC 2015 in 30 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$5.79E+04$	$9.19E+05$	$2.66E+05$	$1.99E+05$	$1.91E+05$
f_2	$2.24E-01$	$5.95E+03$	$1.53E+03$	$1.09E+03$	$1.64E+03$
f_3	$2.09E+01$	$2.11E+01$	$2.10E+01$	$2.10E+01$	$4.33E-02$
f_4	$1.79E+01$	$1.76E+02$	$7.13E+01$	$3.38E+01$	$5.71E+01$
f_5	$5.85E+03$	$7.30E+03$	$6.74E+03$	$6.73E+03$	$2.96E+02$
f_6	$8.53E+03$	$1.47E+05$	$5.25E+04$	$4.45E+04$	$2.99E+04$
f_7	$3.86E+00$	$1.39E+01$	$9.53E+00$	$1.00E+01$	$2.45E+00$
f_8	$4.89E+03$	$8.51E+04$	$2.88E+04$	$2.40E+04$	$1.74E+04$
f_9	$1.02E+02$	$2.34E+02$	$1.05E+02$	$1.03E+02$	$1.84E+01$
f_{10}	$8.64E+03$	$2.90E+05$	$7.73E+04$	$6.27E+04$	$5.70E+04$
f_{11}	$3.01E+02$	$6.64E+02$	$4.50E+02$	$4.95E+02$	$1.22E+02$
f_{12}	$1.04E+02$	$1.07E+02$	$1.05E+02$	$1.05E+02$	$5.93E-01$
f_{13}	$1.09E+02$	$1.24E+02$	$1.16E+02$	$1.16E+02$	$3.20E+00$
f_{14}	$3.14E+04$	$3.68E+04$	$3.30E+04$	$3.30E+04$	$9.27E+02$
f_{15}	$1.00E+02$	$1.00E+02$	$1.00E+02$	$1.00E+02$	$0.00E+00$

Tab. E.3 Statistics of the Bison Algorithm on CEC 2015 in 50 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$1.38E+05$	$1.08E+06$	$4.27E+05$	$3.72E+05$	$2.17E+05$
f_2	$3.65E+01$	$1.38E+04$	$4.74E+03$	$4.43E+03$	$3.37E+03$
f_3	$2.11E+01$	$2.12E+01$	$2.11E+01$	$2.11E+01$	$3.10E-02$
f_4	$3.58E+01$	$3.93E+02$	$1.48E+02$	$7.36E+01$	$1.28E+02$
f_5	$1.15E+04$	$1.33E+04$	$1.26E+04$	$1.26E+04$	$3.79E+02$
f_6	$2.05E+04$	$1.21E+06$	$1.83E+05$	$1.50E+05$	$1.66E+05$
f_7	$7.41E+00$	$9.31E+01$	$5.50E+01$	$7.62E+01$	$3.28E+01$
f_8	$1.15E+04$	$3.53E+05$	$1.11E+05$	$9.14E+04$	$7.31E+04$
f_9	$1.04E+02$	$6.29E+02$	$1.23E+02$	$1.04E+02$	$8.05E+01$
f_{10}	$5.16E+03$	$1.00E+05$	$2.04E+04$	$1.35E+04$	$1.83E+04$
f_{11}	$5.56E+02$	$8.46E+02$	$6.70E+02$	$6.77E+02$	$6.13E+01$
f_{12}	$1.06E+02$	$1.10E+02$	$1.08E+02$	$1.09E+02$	$8.29E-01$
f_{13}	$2.00E+02$	$2.24E+02$	$2.17E+02$	$2.17E+02$	$4.09E+00$
f_{14}	$4.95E+04$	$7.84E+04$	$5.42E+04$	$4.95E+04$	$1.01E+04$
f_{15}	$1.00E+02$	$1.00E+02$	$1.00E+02$	$1.00E+02$	$0.00E+00$

Tab. E.4 Statistics of the Bison Algorithm on CEC 2015 in 100 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$7.88E+05$	$3.06E+06$	$1.34E+06$	$1.23E+06$	$4.70E+05$
f_2	$2.13E-01$	$1.08E+04$	$1.04E+03$	$3.91E+02$	$1.76E+03$
f_3	$2.13E+01$	$2.14E+01$	$2.13E+01$	$2.13E+01$	$2.06E-02$
f_4	$1.28E+02$	$9.16E+02$	$5.11E+02$	$6.35E+02$	$3.18E+02$
f_5	$2.78E+04$	$3.02E+04$	$2.94E+04$	$2.95E+04$	$4.93E+02$
f_6	$1.94E+05$	$7.68E+05$	$4.15E+05$	$3.96E+05$	$1.07E+05$
f_7	$1.85E+01$	$1.66E+02$	$1.47E+02$	$1.59E+02$	$3.32E+01$
f_8	$6.05E+04$	$6.59E+05$	$2.01E+05$	$1.88E+05$	$1.08E+05$
f_9	$1.07E+02$	$1.10E+02$	$1.08E+02$	$1.08E+02$	$5.71E-01$
f_{10}	$3.92E+03$	$7.48E+05$	$4.56E+04$	$6.02E+03$	$1.61E+05$
f_{11}	$3.02E+02$	$1.87E+03$	$1.24E+03$	$1.40E+03$	$4.64E+02$
f_{12}	$1.16E+02$	$1.21E+02$	$1.19E+02$	$1.19E+02$	$8.88E-01$
f_{13}	$4.47E+02$	$4.69E+02$	$4.60E+02$	$4.61E+02$	$5.41E+00$
f_{14}	$1.00E+02$	$1.80E+05$	$1.45E+05$	$1.68E+05$	$5.02E+04$
f_{15}	$1.00E+02$	$1.21E+02$	$1.04E+02$	$1.00E+02$	$6.29E+00$

Tab. E.5 Statistics of the Bison Algorithm on CEC 2017 in 10 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$2.51E-02$	$2.28E+03$	$4.78E+02$	$1.36E+02$	$6.57E+02$
f_2	$1.42E-07$	$8.14E-05$	$9.51E-06$	$7.16E-06$	$1.25E-05$
f_3	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
f_4	$1.29E-01$	$1.30E+00$	$3.29E-01$	$2.33E-01$	$2.76E-01$
f_5	$0.00E+00$	$2.89E+01$	$7.27E+00$	$4.97E+00$	$6.28E+00$
f_6	$0.00E+00$	$3.94E-04$	$1.39E-05$	$0.00E+00$	$5.92E-05$
f_7	$1.14E+01$	$3.93E+01$	$2.43E+01$	$2.72E+01$	$8.37E+00$
f_8	$9.95E-01$	$3.05E+01$	$7.89E+00$	$5.97E+00$	$7.25E+00$
f_9	$0.00E+00$	$1.74E+00$	$1.41E-01$	$0.00E+00$	$3.79E-01$
f_{10}	$6.95E+00$	$1.47E+03$	$1.08E+03$	$1.13E+03$	$3.42E+02$
f_{11}	$2.81E-03$	$9.67E+00$	$2.89E+00$	$2.00E+00$	$2.62E+00$
f_{12}	$6.71E+02$	$3.24E+04$	$1.05E+04$	$9.01E+03$	$8.57E+03$
f_{13}	$8.09E+01$	$9.94E+03$	$2.98E+03$	$2.41E+03$	$2.64E+03$
f_{14}	$1.62E+01$	$4.97E+01$	$3.32E+01$	$3.35E+01$	$7.77E+00$
f_{15}	$5.08E+00$	$1.08E+02$	$2.92E+01$	$2.68E+01$	$1.90E+01$
f_{16}	$3.98E-02$	$1.98E+02$	$2.31E+01$	$7.28E-01$	$4.81E+01$
f_{17}	$2.84E+00$	$8.27E+01$	$3.59E+01$	$2.88E+01$	$2.08E+01$
f_{18}	$2.75E+01$	$1.43E+04$	$3.82E+03$	$2.41E+03$	$4.03E+03$
f_{19}	$5.81E+00$	$1.74E+02$	$2.36E+01$	$2.08E+01$	$2.39E+01$
f_{20}	$0.00E+00$	$4.46E+01$	$5.45E+00$	$1.31E+00$	$1.09E+01$
f_{21}	$1.00E+02$	$2.28E+02$	$1.21E+02$	$1.00E+02$	$4.12E+01$
f_{22}	$1.00E+02$	$1.82E+02$	$1.02E+02$	$1.01E+02$	$1.15E+01$
f_{23}	$3.00E+02$	$3.31E+02$	$3.09E+02$	$3.09E+02$	$5.36E+00$
f_{24}	$1.00E+02$	$3.90E+02$	$3.23E+02$	$3.34E+02$	$4.21E+01$
f_{25}	$3.98E+02$	$4.50E+02$	$4.36E+02$	$4.45E+02$	$1.96E+01$
f_{26}	$0.00E+00$	$3.93E+02$	$2.83E+02$	$3.00E+02$	$5.89E+01$
f_{27}	$3.92E+02$	$4.19E+02$	$3.99E+02$	$3.99E+02$	$5.29E+00$
f_{28}	$3.00E+02$	$6.46E+02$	$4.54E+02$	$5.05E+02$	$1.48E+02$
f_{29}	$2.35E+02$	$3.39E+02$	$2.77E+02$	$2.73E+02$	$2.13E+01$
f_{30}	$9.57E+02$	$9.77E+05$	$1.28E+05$	$3.52E+03$	$3.13E+05$

Tab. E.6 Statistics of the Bison Algorithm on CEC 2017 in 30 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$2.86E+00$	$1.04E+04$	$2.15E+03$	$1.56E+03$	$2.16E+03$
f_2	$6.82E-02$	$3.77E+12$	$1.04E+11$	$2.82E+05$	$5.38E+11$
f_3	$4.09E+00$	$2.77E+02$	$7.54E+01$	$6.08E+01$	$6.99E+01$
f_4	$2.47E-04$	$6.79E+01$	$1.11E+01$	$4.00E+00$	$2.13E+01$
f_5	$1.89E+01$	$1.78E+02$	$7.50E+01$	$3.68E+01$	$6.12E+01$
f_6	$1.14E-13$	$4.44E-02$	$1.16E-03$	$4.32E-05$	$6.20E-03$
f_7	$4.60E+01$	$2.30E+02$	$1.83E+02$	$1.90E+02$	$3.31E+01$
f_8	$1.09E+01$	$1.79E+02$	$4.70E+01$	$2.69E+01$	$4.79E+01$
f_9	$0.00E+00$	$4.01E+01$	$6.56E+00$	$3.82E+00$	$7.44E+00$
f_{10}	$6.27E+03$	$7.44E+03$	$6.97E+03$	$6.97E+03$	$2.67E+02$
f_{11}	$6.09E+00$	$8.88E+01$	$3.20E+01$	$2.10E+01$	$2.57E+01$
f_{12}	$5.51E+03$	$7.59E+04$	$2.61E+04$	$2.60E+04$	$1.42E+04$
f_{13}	$2.29E+02$	$3.45E+04$	$1.28E+04$	$1.15E+04$	$9.36E+03$
f_{14}	$9.22E+01$	$1.73E+04$	$4.89E+03$	$3.17E+03$	$4.70E+03$
f_{15}	$6.51E+00$	$1.87E+04$	$4.05E+03$	$2.47E+03$	$4.63E+03$
f_{16}	$1.91E+01$	$1.69E+03$	$9.35E+02$	$1.06E+03$	$4.53E+02$
f_{17}	$1.42E+01$	$5.12E+02$	$1.25E+02$	$7.31E+01$	$1.07E+02$
f_{18}	$2.22E+04$	$7.32E+05$	$1.75E+05$	$1.57E+05$	$1.49E+05$
f_{19}	$4.67E+01$	$2.61E+04$	$6.22E+03$	$4.34E+03$	$6.72E+03$
f_{20}	$2.68E+01$	$7.70E+02$	$1.99E+02$	$1.55E+02$	$1.35E+02$
f_{21}	$2.13E+02$	$3.75E+02$	$2.69E+02$	$2.32E+02$	$5.80E+01$
f_{22}	$1.00E+02$	$1.02E+02$	$1.00E+02$	$1.00E+02$	$5.84E-01$
f_{23}	$3.51E+02$	$4.26E+02$	$3.81E+02$	$3.79E+02$	$1.35E+01$
f_{24}	$4.22E+02$	$5.43E+02$	$4.48E+02$	$4.47E+02$	$1.85E+01$
f_{25}	$3.84E+02$	$3.98E+02$	$3.88E+02$	$3.87E+02$	$2.07E+00$
f_{26}	$2.00E+02$	$3.11E+03$	$1.18E+03$	$1.33E+03$	$7.39E+02$
f_{27}	$5.12E+02$	$5.61E+02$	$5.32E+02$	$5.32E+02$	$1.15E+01$
f_{28}	$3.00E+02$	$4.14E+02$	$3.29E+02$	$3.00E+02$	$4.76E+01$
f_{29}	$4.18E+02$	$1.10E+03$	$5.85E+02$	$5.60E+02$	$1.74E+02$
f_{30}	$2.49E+03$	$1.04E+04$	$4.53E+03$	$4.14E+03$	$1.65E+03$

Tab. E.7 Statistics of the Bison Algorithm on CEC 2017 in 50 dimensions.

	min	max	$mean$	$median$	std
f_1	$2.45E+00$	$9.37E+03$	$2.35E+03$	$1.70E+03$	$2.52E+03$
f_2	$6.76E+13$	$1.86E+28$	$3.65E+26$	$3.51E+19$	$2.61E+27$
f_3	$5.38E+03$	$2.07E+04$	$1.16E+04$	$1.15E+04$	$3.16E+03$
f_4	$4.40E-03$	$1.48E+02$	$6.50E+01$	$6.66E+01$	$5.03E+01$
f_5	$3.68E+01$	$3.78E+02$	$1.42E+02$	$7.16E+01$	$1.19E+02$
f_6	$6.06E-06$	$2.00E-02$	$2.32E-03$	$5.10E-04$	$4.30E-03$
f_7	$7.92E+01$	$4.47E+02$	$3.54E+02$	$3.70E+02$	$7.21E+01$
f_8	$3.78E+01$	$3.65E+02$	$1.52E+02$	$7.46E+01$	$1.25E+02$
f_9	$7.23E-01$	$1.53E+02$	$3.08E+01$	$1.49E+01$	$3.81E+01$
f_{10}	$1.18E+04$	$1.37E+04$	$1.29E+04$	$1.29E+04$	$4.59E+02$
f_{11}	$2.72E+01$	$1.80E+02$	$5.40E+01$	$4.61E+01$	$2.93E+01$
f_{12}	$8.81E+04$	$1.35E+06$	$4.15E+05$	$3.26E+05$	$2.78E+05$
f_{13}	$2.96E+01$	$1.77E+04$	$1.93E+03$	$7.26E+02$	$2.99E+03$
f_{14}	$1.93E+03$	$9.87E+04$	$3.05E+04$	$2.62E+04$	$2.25E+04$
f_{15}	$3.66E+01$	$1.50E+04$	$4.34E+03$	$3.26E+03$	$4.03E+03$
f_{16}	$3.67E+02$	$2.53E+03$	$8.86E+02$	$7.24E+02$	$5.30E+02$
f_{17}	$2.55E+02$	$1.91E+03$	$1.25E+03$	$1.38E+03$	$5.12E+02$
f_{18}	$2.44E+04$	$2.58E+06$	$1.15E+06$	$1.07E+06$	$5.83E+05$
f_{19}	$2.58E+03$	$2.65E+04$	$1.49E+04$	$1.51E+04$	$5.68E+03$
f_{20}	$4.08E+01$	$1.95E+03$	$1.07E+03$	$1.29E+03$	$5.28E+02$
f_{21}	$2.42E+02$	$5.68E+02$	$3.34E+02$	$2.57E+02$	$1.23E+02$
f_{22}	$1.00E+02$	$1.41E+04$	$7.49E+03$	$1.26E+04$	$6.51E+03$
f_{23}	$4.55E+02$	$5.28E+02$	$4.88E+02$	$4.87E+02$	$1.89E+01$
f_{24}	$5.04E+02$	$8.48E+02$	$5.66E+02$	$5.59E+02$	$4.38E+01$
f_{25}	$4.61E+02$	$6.15E+02$	$5.50E+02$	$5.61E+02$	$2.80E+01$
f_{26}	$3.00E+02$	$2.96E+03$	$1.86E+03$	$1.99E+03$	$6.58E+02$
f_{27}	$5.78E+02$	$7.46E+02$	$6.50E+02$	$6.43E+02$	$4.17E+01$
f_{28}	$4.59E+02$	$5.94E+02$	$4.90E+02$	$4.97E+02$	$2.68E+01$
f_{29}	$3.59E+02$	$1.11E+03$	$6.59E+02$	$6.23E+02$	$1.80E+02$
f_{30}	$7.73E+05$	$1.60E+06$	$1.10E+06$	$1.05E+06$	$2.14E+05$

Tab. E.8 Statistics of the Bison Algorithm on CEC 2017 in 100 dimensions.

	min	*max*	*mean*	*median*	*std*
f_1	$1.07E+01$	$2.55E+04$	$4.34E+03$	$2.76E+03$	$4.93E+03$
f_2	$2.90E+47$	$1.13E+83$	$2.25E+81$	$1.47E+65$	$1.59E+82$
f_3	$9.01E+04$	$1.58E+05$	$1.18E+05$	$1.17E+05$	$1.67E+04$
f_4	$8.29E+00$	$2.35E+02$	$1.29E+02$	$1.45E+02$	$5.15E+01$
f_5	$1.38E+02$	$9.42E+02$	$6.19E+02$	$7.65E+02$	$2.89E+02$
f_6	$9.30E-04$	$8.78E-02$	$2.37E-02$	$1.90E-02$	$2.09E-02$
f_7	$2.65E+02$	$1.05E+03$	$9.28E+02$	$9.48E+02$	$1.15E+02$
f_8	$1.17E+02$	$8.92E+02$	$5.54E+02$	$7.14E+02$	$3.05E+02$
f_9	$2.27E+01$	$2.12E+03$	$1.87E+02$	$1.08E+02$	$3.02E+02$
f_{10}	$2.82E+04$	$3.00E+04$	$2.93E+04$	$2.93E+04$	$4.46E+02$
f_{11}	$1.64E+02$	$8.26E+02$	$5.23E+02$	$6.17E+02$	$2.15E+02$
f_{12}	$2.14E+05$	$1.16E+06$	$6.77E+05$	$6.41E+05$	$2.42E+05$
f_{13}	$5.50E+01$	$1.17E+04$	$3.63E+03$	$2.87E+03$	$2.66E+03$
f_{14}	$6.23E+04$	$2.68E+05$	$1.08E+05$	$1.00E+05$	$3.54E+04$
f_{15}	$3.73E+01$	$5.18E+03$	$9.06E+02$	$4.52E+02$	$1.16E+03$
f_{16}	$1.17E+03$	$7.41E+03$	$4.34E+03$	$4.30E+03$	$2.20E+03$
f_{17}	$8.67E+02$	$4.80E+03$	$3.86E+03$	$4.26E+03$	$1.04E+03$
f_{18}	$3.67E+05$	$4.18E+06$	$1.52E+06$	$1.54E+06$	$7.71E+05$
f_{19}	$4.59E+01$	$1.08E+04$	$1.49E+03$	$4.47E+02$	$2.23E+03$
f_{20}	$2.14E+03$	$5.03E+03$	$4.35E+03$	$4.51E+03$	$6.19E+02$
f_{21}	$3.23E+02$	$1.12E+03$	$6.68E+02$	$4.26E+02$	$3.26E+02$
f_{22}	$1.00E+02$	$3.13E+04$	$2.99E+04$	$3.06E+04$	$4.29E+03$
f_{23}	$6.60E+02$	$7.98E+02$	$7.22E+02$	$7.20E+02$	$3.05E+01$
f_{24}	$9.89E+02$	$1.18E+03$	$1.07E+03$	$1.06E+03$	$4.15E+01$
f_{25}	$6.39E+02$	$8.96E+02$	$7.65E+02$	$7.67E+02$	$6.05E+01$
f_{26}	$3.00E+02$	$1.21E+04$	$5.72E+03$	$5.95E+03$	$2.22E+03$
f_{27}	$6.89E+02$	$8.20E+02$	$7.65E+02$	$7.69E+02$	$2.98E+01$
f_{28}	$4.89E+02$	$9.82E+02$	$5.74E+02$	$5.50E+02$	$9.71E+01$
f_{29}	$1.06E+03$	$5.53E+03$	$2.04E+03$	$1.86E+03$	$6.94E+02$
f_{30}	$4.89E+03$	$4.99E+04$	$1.19E+04$	$9.06E+03$	$7.59E+03$