Hindawi
Journal of Optimization
Volume 2017, Article ID 3828420, 25 pages
https://doi.org/10.1155/2017/3828420

Research Article

Hydrological Cycle Algorithm for Continuous Optimization Problems

Ahmad Wedyan, Jacqueline Whalley, and Ajit Narayanan

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand

Correspondence should be addressed to Ahmad Wedyan; awedyan@aut.ac.nz

Received 2 August 2017; Revised 13 November 2017; Accepted 22 November 2017; Published 17 December 2017

Academic Editor: Efren Mezura-Montes

A new nature-inspired optimization algorithm called the Hydrological Cycle Algorithm (HCA) is proposed based on the continuous movement of water in nature. In the HCA, a collection of water drops passes through various hydrological water cycle stages, such as flow, evaporation, condensation, and precipitation. Each stage plays an important role in generating solutions and avoiding premature convergence. The HCA shares information by direct and indirect communication among the water drops, which improves solution quality. Similarities and differences between HCA and other water-based algorithms are identified, and the implications of these differences on overall performance are discussed. A new topological representation for problems with a continuous domain is proposed. In proof-of-concept experiments, the HCA is applied on a variety of benchmarked continuous numerical functions. The results were found to be competitive in comparison to a number of other algorithms and validate the effectiveness of HCA. Also demonstrated is the ability of HCA to escape from local optima solutions and converge to global solutions. Thus, HCA provides an alternative approach to tackling various types of multimodal continuous optimization problems as well as an overall framework for water-based particle algorithms in general.

1. Introduction

Optimization is the process of making something as good as possible. In computer science and operations research, an optimization problem can be defined as the problem of finding the best solution or better than previously known best solution among a set of feasible solutions by trying different variations of the input [1]. In mathematics and engineering, researchers use optimization algorithms to develop and improve new ideas and models that can be expressed as a function of certain variables. Optimization problems can be categorized as either continuous or discrete, based on the nature of the variables in the optimization function. In continuous optimization, each variable may have any value within a defined and specific range, and they are commonly real numbers. For discrete optimization, each variable has a value from a finite set of possible values or a permutation of a set of numbers, commonly as integer numbers [2].

A variety of continuous optimization problems (COPs) exist in the literature for benchmarking purposes [3]. In these functions, the values for various continuous variables need to be obtained such that the function is either minimized or maximized. These functions have various structures and can be categorized as either unimodal or multimodal. A unimodal function has only one local optimum point, whereas a multimodal function has several local optimum points with one or more points as a global optimum. Each function may consist of a single variable or may be multivariate. Typically, a unimodal function has one variable, while a multimodal function has multiple variables. Mathematically, a continuous objective function can be represented as follows:

$$\text{minimize} \quad f(X) : R^n \to \mathbb{R}, \tag{1}$$

where X is a vector that consists of n decision variables:

$$X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}. \tag{2}$$

A solution x^* is called a global minimizer of a problem $f(x)$, if $f(x^*) \le f(x)$ for all $x \in X$. Conversely, it is called a global maximizer if $f(x^*) \ge f(x)$ for all $x \in X$.

These functions are commonly used to evaluate the characteristics of new algorithms and to measure an algorithm's overall performance, convergence rate, robustness, precision, and its ability to escape from local optima solutions [3]. These empirical results can be used to compare any new algorithm with existing optimization algorithms and to gain understanding of the algorithm's behavior on different kinds of problem. For instance, multimodal functions can be used to check the ability of an algorithm to escape local optima and explore new regions, especially when the global optimal is surrounded by many local optima. Similarly, flat surface functions add more difficulty to reaching global optima as the flatness does not give any indication of the direction towards global optima [3].

Practically, the number of variables and the domain of each variable (the function dimensionality) affect the complexity of the function. In addition, other factors such as the number of local/global optima points and the distribution of local optima compared with global optima are crucial issues in determining the complexity of a function, especially when a global minimum point is located very close to local minima points [3]. Some of these functions represent real-life optimization problems, while others are artificial problems. Furthermore, a function can be classified as unconstrained or constrained. An unconstrained function has no restriction on the values of its variables.

Due to the presence of a large variety of optimization problems, it is hard to find a suitable algorithm that can solve all types of optimization problems effectively. On the other hand, the No-Free-Lunch (NFL) theorems posit that no particular algorithm performs better than all other algorithms for all problems and that what an algorithm gains in performance on some problems can be lost on other problems [5]. Consequently, we can infer that some algorithms are more applicable and compatible with particular classes of optimization problems than others. Therefore, there is always a need to design and evaluate new algorithms for their potential to improve performance on certain types of optimization problem. For these reasons, the main objective of this paper is to develop a new optimization algorithm and determine its performance on various benchmarked problems to gain understanding of the algorithm and evaluate its applicability to other unsolved problems.

Many established nature-inspired algorithms, especially water-based algorithms, have proven successful in solving COPs. Nevertheless, these water-based algorithms do not take into account the broader concepts of the hydrological water cycle, how the water drops and paths are formed, or the fact that water is a renewable and recyclable resource. That is, previous water-based algorithms partially utilized some stages of the natural water cycle and demonstrated only some of the important aspects of a full hydrological water cycle for solving COPs.

The main contribution of this paper is to delve deeper into hydrological theory and the fundamental concepts surrounding water droplets to inform the design of a new optimization algorithm. This algorithm provides a new hydrological-based conceptual framework within which existing and future work in water-based algorithms can be located. In this paper, we present a new nature-inspired optimization algorithm called the Hydrological Cycle Algorithm (HCA) for solving optimization problems. HCA simulates nature's hydrological water cycle. More specifically, it involves a collection of water drops passing through different phases such as flow (runoff), evaporation, condensation, and precipitation to generate a solution. It can be considered as a swarm intelligence optimization algorithm for some parts of the cycle when a collection of water drops moves through the search space. But it can also be considered an evolutionary algorithm for other parts of the cycle when information is exchanged and shared. By using the full hydrological water cycle as a conceptual framework, we show that previous water-based algorithms have predominantly only used swarm-like aspects inspired by precipitation and flow. HCA, however, uses all four stages that will form a complete water-based approach to solving optimization problems efficiently. In particular, we show that for certain problems HCA leads to improved performance and solution quality. In particular, we demonstrate how the condensation stage is utilized to allow information sharing among the particles. The information sharing helps in improving the performance and reducing the number of iterations needed to reach the optimal solution. Our claims for HCA are to be judged not on improved efficiency but on improved conceptual bioinspiration as well as contribution to the area of water-based algorithm. However, we also show that HCA is competitive in comparison to several other nature-inspired algorithms, both water-based and non-water-based.

In summary, the paper presents a new overall conceptual framework for water-based algorithms which applies the hydrological cycle in its entirety to continuous optimization problems as a proof-of-concept (i.e., validates the full hydrological inspired framework). The benchmarked mathematical test functions chosen in this paper are useful for evaluating performance characteristics of any new approach, especially the effectiveness of exploration and exploitation processes. Solving these functions (i.e., finding the global-optimal solution) is challenging for most algorithms, even with a few variables, because their artificial landscapes may contain many local optimal solutions. Our minimal criterion for success is that the HCA algorithm performance is comparable to other well-known algorithms including those that only partially adopt the full HCA framework. We maintain that our results are competitive and that therefore water-inspired researchers and non-water-inspired researchers lose nothing by adopting the richer bioinspiration that comes with HCA in its entirety rather than adopting only parts of the full water cycle. The HCA's performance is evaluated in terms of finding optimal solutions and computational effort (number of iterations). The algorithm's behavior, its convergence rate, and its ability to deal with multimodal functions are also investigated and analyzed. The main aim of this paper is not to prove the superiority of HCA over other algorithms. Instead, the aim is to provide researchers in science, engineering, operations research, and machine learning with a new water-based tool for dealing with continuous variables in convex, unconstrained, and nonlinear problems. Further details and features of the HCA are described in Section 3.

The remainder of this paper is organized as follows. Section 2 discusses various related swarm intelligence techniques used to tackle COPs and distinguishes HCA from related algorithms. Section 3 describes the hydrological cycle in nature and explains the proposed HCA. Section 4 outlines the experimental setup, problem representation, results obtained, and the comparative evaluation conducted with other algorithms. Finally, Section 5 presents concluding remarks and outlines plans for further research.

2. Previous Swarm Intelligence Approaches for Solving Continuous Problems

Many metaheuristic optimization algorithms have been designed and their performance has been evaluated on benchmark functions. Among these are nature-inspired algorithms, which have received considerable attention. Each algorithm uses a different approach and representation to solve the target problem. It is worth mentioning that our focus is on studies that involve solving unconstrained continuous functions with few variables. If a new algorithm cannot deal with these cases, there is little chance of it dealing with more complex functions.

Ant colony optimization (ACO) is one of the best-known techniques for solving discrete optimization problems and has been extended to deal with COPs in many studies. For example, in [6], the authors divided the function domain into a certain number of regions representing the trial solution space for the ants to explore. Then they generated two types of ants (global and local ants) and distributed them to explore these regions. The global ants were responsible for updating the fitness of the regions globally, whereas the local ants did the same locally. The approach incorporated mutation and crossover operations to improve the solution quality.

Socha and Dorigo [7] extended the ACO algorithm to tackle continuous domain problems. The extension facilitated consideration of continuous probability instead of discrete probability and incorporated the Gaussian probability density function to choose the next point. The pheromone evaporation procedure was modified and old solutions were removed from the candidate solutions pool and replaced with new and better solutions. A weight associated with each ant solution represented the fitness of the solution. Other ACO inspired approaches with various implementations that do not follow the original ACO algorithm structure have also been developed to solve COPs [8–11]. In further work, the performance of ACO was evaluated on continuous function optimization [12]. One of the problems with ACO is that reinforcement of paths is the only method for sharing information between the ants in an indirect way. A heavily pheromone-reinforced path is naturally considered a good path for other ants to follow. No other mechanism exists to allow ants to share information for exploration purposes except indirectly through the evaporation of pheromone over time. Exploration diminishes over time with ACO, which suits problems where there may be only one global solution. But if the aim is to find a number of alternative solutions, the rate of evaporation has to be strictly controlled to ensure that no single path dominates. Evaporation control can lead to other side-effects in the classic ACO paradigm, such as longer convergence time, resulting in the need to add possibly nonintuitive information sharing mechanisms, such as global updates, for exploration purposes.

Although strictly speaking they are not a swarm intelligence approach, genetic algorithms (GAs) [13] have been applied to many COPs. A GA uses simple operations such as crossover and mutation to manipulate and direct the population and to help escape from local optima. The values of the variables are encoded using binary or real numbers [14]. However, as with other algorithms, GA performance can depend critically on the initial population's values, which are usually random. This was clarified by Maaranen et al. [15], who investigated the importance and effect of the initial population values on the quality of the final solution. In [16] the authors also focused on the choice of the initial population values and modified the algorithm to intensify the search in the most promising area of the solution space. The main problem with standard GAs for optimization continues to be the lack of "long-termism," where fitness for exploitation may need to be secondary to allow for adequate exploration to assess the problem space for alternative solutions. One possible way to deal with this problem is to hybridize the GA with other approaches, such as in [17] where a hybrid GA and particle swarm optimization (PSO) is proposed for solving multimodal functions.

James and Russell [18] examined the performance of the PSO algorithm on some continuous optimization functions. Later, researchers, such as Chen et al. [19], modified the PSO algorithm to help overcome the problem of premature convergence. Additionally, PSO is used for global optimization in [20]. The PSO is also hybridized with ACO to improve the performance when dealing with high dimension multimodal functions [21]. Further, a comparative study between GA and PSO on numerical benchmark problems can be found in [22]. Other versions of the PSO algorithm with crossover and mutation operators to improve the performance have also been proposed, such as in [23]. Once such GA operators are introduced, the question arises as to what advantages a PSO approach has over a standard evolutionary approach to the same problem.

The bees algorithm (BA) has been used to solve a number of benchmark functions [24]. This algorithm simulates the behavior of swarms of honey bees during food foraging. In BA, scout bees are sent to search for food sources by moving randomly from one place to another. On returning to the hive, they share the information with the other bees about location, distance, and quality of the food source. The algorithm produces good results for several benchmark functions. However, the information sharing is direct and confined to subsets of bees, leading to the possibility of convergence to a local optimum or divergence away from the global optimum.

The grenade explosion method (GSM) is an evolutionary algorithm which has been used to optimize real-valued problems [25]. The GSM is based on the mechanism associated with a grenade explosion, which is intended to overcome problems associated with "crowding" where candidate solutions have converged to a local minimum. When a grenade explodes, shrapnel pieces are thrown to destroy objects in the vicinity of the explosion. In the GSM, losses are calculated

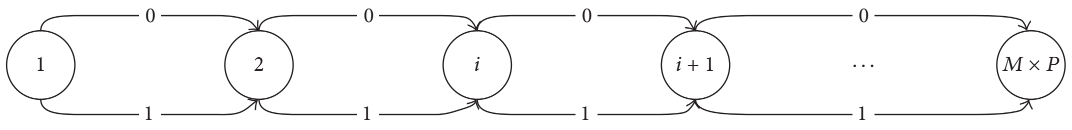


Figure 1: Representation of a continuous problem in the IWD algorithm.

for each piece of shrapnel, and the effect of each piece of shrapnel is calculated. The highest loss effect represents valuable objects existing in that area. Then, another grenade is thrown to the area to cause greater loss, and so on. Overall, the GSM was able to find the global minimum faster than other algorithms in seven out of eight benchmarked tests. However, the algorithm requires the maintenance of nonintuitive territory radius preventing shrapnel pieces from coming too close to each other and forcing shrapnel to spread uniformly over the search space. Also required is an explosion range for the shrapnel. These control parameters work well with problem spaces containing many local optima so that global optima can be found after suitable exploration. However, the relationship between these control parameters and problem spaces of unknown topology is not clear.

2.1. Water-Based Algorithms. Algorithms inspired by water flow have become increasingly common for tackling optimization problems. Two of the best-known examples are intelligent water drops (IWD) and the water cycle algorithm (WCA). These algorithms have been shown to be successful in many applications including various COPs. The IWD algorithm is a swarm-based algorithm inspired by the ground flow of water in nature and the actions and reactions that occur during that process [26]. In IWD, a water drop has two properties: movement velocity and the amount of soil it carries. These properties are changed and updated for each water drop throughout its movement from one point to another. The velocity of a water drop is affected and determined only by the amount of soil between two points, while the amount of soil it carries is equal to the amount of soil being removed from a path [26].

The IWD has also been used to solve COPs. Shah-Hosseini [27] utilized binary variable values in the IWD and created a directed graph of $(M \times P)$ nodes, in which M identifies the number of variables and P identifies the precision (number of bits for each variable), which was assumed to be 32. The graph comprised $2 \times (M \times P)$ directed edges connecting nodes, with each edge having a value of zero or one, as depicted in Figure 1. One of the advantages of this representation is that it is intuitively consistent with the natural idea of water flowing in a specific direction.

Also, the IWD algorithm was augmented with a mutation-based local search to enhance its solution quality. In this process, an edge is selected at random from a solution and replaced by another edge which is kept if it improves the fitness value, and the process was repeated 100 times for each solution. This mutation is applied to all the generated solutions, and then the soil is updated on the edges belonging to the best solution. However, the variables' values have to be converted from binary to real numbers to be evaluated. Also, once mutation (but not crossover) is introduced into IWD, the question arises as what advantages IWD has over standard evolutionary algorithms that use simpler notations for representing continuous numbers. Detailed differences between IWD and our HCA are described in Section 3.9.

Various researchers have applied the WCA to COPs [28, 29]. The WCA is also based on the ground water flow through rivers and streams to the sea. The algorithm constructs a solution using a population of streams, where each stream is a candidate solution. It initially generates random solutions for the problem by distributing the streams into different positions according to the problem's dimensions. The best stream is considered a sea and a specific number of streams are considered rivers. Then the streams are made to flow towards the positions of the rivers and sea. The position of the sea indicates the global-best solution, whereas those of the rivers indicate the local best solution points. In each iteration, the position of each stream may be swapped with any of the river positions or (counterintuitively) the sea position according to the fitness of that stream. Therefore, if a stream solution is better than that of a river, that stream becomes a river, and the corresponding river becomes a stream. The same process occurs between rivers and sea. Evaporation occurs in the rivers/streams when the distance between a river/stream and the sea is very small (close to zero). Following evaporation, new streams are generated with different positions in a process that can be described as rain. These processes help the algorithm to escape from local optima.

WCA treats streams, rivers, and the sea as set of moving points in a multidimensional space and does not consider stream/river formation (movements of water and soils). Therefore, the overall structure of the WCA is quite similar to that of the PSO algorithm. For instance, the PSO algorithm has *Gbest* and *Pbest* points that indicate the global and local solutions, with the *Gbest* point guiding the *Pbest* points to converge to that point. In a similar manner, the WCA has a number of rivers that represent the local optimal solutions, and a sea position represents the global-optimal solution. The sea is used as a guide for the streams and rivers. Moreover, in PSO, a particle at the *Pbest* point updates its position towards the *Gbest* point. Analogously, the WCA exchanges the positions of rivers with that of the sea when the fitness of the river is better. The main difference is the consideration of the evaporation and raining stages that help the algorithm to escape from local optima.

An improved version of WCA called dual-system (DSWCA) has been presented for solving constrained engineering optimization problems [30]. The DSWCA improvements involved adding two cycles (outer and inner). The outer cycle is in accordance with the WCA and is designed to perform exploration. The inner cycle which is based on the ocean cycle was designed as an exploitation process. The confluence between rivers process is also included to improve convergence speed. The improvements aim to help the original WCA to avoid falling into the local optimal solutions. The DSWCA was able to provide better results than WCA. In a further extension of WCA, Heidari et al. proposed a chaotic WCA (CWCA) that incorporates thirteen chaotic patterns with WCA movement process to improve WCA's performance and deal with the premature convergence problem [31]. The experimental results demonstrated improvement in controlling the premature convergence problem. However, WCA and its extensions introduce many additional features, many of which are not compatible with nature-inspiration, to improve performance and make WCA competitive with non-water-based algorithms. Detailed differences between WCA and our HCA are described in Section 3.9.

The review of previous approaches shows that nature-inspired swarm approaches tend to introduce evolutionary concepts of mutation and crossover for sharing information, to adopt nonplausible global data structures to prevent convergence to local optima or add more centralized control parameters to preserve the balance between exploration and exploitation in increasingly difficult problem domains, thereby compromising self-organization. When such approaches are modified to deal with COPs, complex and unnatural representations of numbers may also be required that add to the overheads of the algorithm as well as leading to what may appear to be "forced" and unnatural ways of dealing with the complexities of a continuous problem.

In swarm algorithms, a number of cooperative homogenous entities explore and interact with the environment to achieve a goal, which is usually to find the global-optimal solution to a problem through exploration and exploitation. Cooperation can be accomplished by some form of communication that allows the swarm members to share exploration and exploitation information to produce high-quality solutions [32, 33]. Exploration aims to acquire as much information as possible about promising solutions, while exploitation aims to improve such promising solutions. Controlling the balance between exploration and exploitation is usually considered critical when searching for more refined solutions as well as more diverse solutions. Also important is the amount of exploration and exploitation information to be shared, and when.

Information sharing in nature-inspired algorithms usually includes metrics for evaluating the usefulness of information and mechanisms for how it should be shared. In particular, these metrics and mechanisms should not contain more knowledge or require more intelligence than can be plausibly assigned to the swarm entities, where the emphasis is on emergence of complex behavior from relatively simple entities interacting in nonintelligent ways with each other. Information sharing can be done either randomly or nonrandomly among entities. Different approaches are used to share information such as direct or indirect interaction. Two sharing models are: one-to-one, in which explicit interaction occurs between entities, and many-to-many (broadcasting), in which interaction occurs indirectly between the entities through the environment [34].

Usually, either direct or indirect interaction between entities is employed in an algorithm for information sharing, and the use of both types in the same algorithm is often neglected. For instance, the ACO algorithm uses indirect communication for information sharing, where ants lay amounts of pheromone on paths depending on the usefulness of what they have explored. The more promising the path, the more the pheromone added, leading other ants to exploiting this path further. Pheromone is not directed at any other ant in particular. In the artificial bee colony (ABC) algorithm, bees share information directly (specifically, the direction and distance to flowers) in a kind of waggle dancing [35]. The information received by other bees depends on which bee they observe and not the information gathered from all bees. In a GA, the information exchange occurs directly in the form of crossover operations between selected members (chromosomes and genes) of the population. The quality of information shared depends on the members chosen and there is no overall store of information available to all members. In PSO, on the other hand, the particles have their own information as well as access to global information to guide their exploitation. This can lead to competitive and cooperative optimization techniques [36]. However, no attempt is made in standard PSO to share information possessed by individual particles. This lack of individual-to-individual communication can lead to early convergence of the entire population of particles to a nonoptimal solution. Selective information sharing between individual particles will not always avoid this narrow exploitation problem but, if undertaken properly, could allow the particles to find alternative and more diverse solution spaces where the global optimum lies.

As can be seen from the above discussion, the distinctions between communication (direct and indirect) and information sharing (random or nonrandom) can become blurred. In HCA, we utilize both direct and indirect communication to share information among selected members of the whole population. Direct communication occurs in the condensation stage of the water cycle, where some water drops collide with each other when they evaporate into the cloud. On the other hand, indirect communication occurs between the water drops and the environment via the soil and path depth. Utilizing information sharing helps to diversify the search space and improve the overall performance through better exploitation. In particular, we show how HCA with direct and indirect communication suits continuous problems, where individual particles share useful information directly when searching a continuous space.

Furthermore, in some algorithms, indirect communication is used not just to find a possible solution but also to reinforce an already found promising solution. Such reinforcement may lead the algorithm to getting stuck in the same solution, which is known as *stagnation behavior* [37]. Such reinforcement can also cause a premature convergence in some problems. An important feature used in HCA to avoid this problem is the path depth. The depths of paths are constantly changing, where deep paths will have less chance of being chosen compared with shallow paths. More details will be provided about this feature in Section 3.3.

In summary, this paper aims to introduce a novel nature-inspired approach for dealing with COPs which also introduces a continuous number representation that is natural for the algorithm. The cyclic nature of the algorithm contributes to self-organization as the system converges to the solution through direct and indirect communication between particles, feedback from the environment, and internal self-evaluation. Finally, our HCA addresses some of the weaknesses of other similar algorithms. HCA is inspired by the full water cycle, not parts of it as exemplified by IWD and WCA.

3. The Hydrological Cycle Algorithm

The water cycle, also known as the hydrologic cycle, describes the continuous movement of water in nature [38]. It is renewable because water particles are constantly circulating from the land to the sky and vice versa. Therefore, the amount of water remains constant. Water drops move from one place to another by natural physical processes, such as evaporation, precipitation, condensation, and runoff. In these processes, the water passes through different states.

3.1. Inspiration. The water cycle starts when surface water gets heated by the sun, causing water from different sources to change into vapor and evaporate into the air. Then the water vapor cools and condenses as it rises to become drops. These drops combine and collect with each other and eventually become so saturated and dense that they fall back because of the force of gravity in a process called precipitation. The resulting water flows on the surface of the Earth into ponds, lakes, or oceans where it again evaporates back into the atmosphere. Then, the entire cycle is repeated.

In the flow (runoff) stage, the water drops start moving from one location to another based on Earth's gravity and the ground topography. When the water is scattered, it chooses a path with less soil and fewer obstacles. Assuming no obstacles exist, water drops will take the shortest path towards the center of the Earth (i.e., oceans) owing to the force of gravity. Water drops have force because of this gravity and this force causes changes in motion depending on the topology. As the water flows in rivers and streams, it picks up sediments and transports them away from their original location. These processes are known as erosion and deposition. They help to create the topography of the ground by making new paths with more soil removal, or the fading of others as they become less likely to be chosen owing to too much soil deposition. These processes are highly dependent on water velocity. For instance, a river is continually picking up and dropping off sediments from one point to another. When the river flow is fast, soil particles are picked up, and the opposite happens when the river flow is slow. Moreover, there is a strong relationship between the water velocity and the suspension load (the amount of dissolved soil in the water) that it currently carries. The water velocity is higher when only a small amount of soil is being carried and lower when larger amounts are carried.

In addition, the term "flow rate" or "discharge" can be defined as the volume of water in a river passing a defined point over a specific period. Mathematically, flow rate is calculated based on the following equation [39]:

$$\begin{gathered} Q = V \times A \Longrightarrow \\ [Q = V \times (W \times D)], \end{gathered} \tag{3}$$

where Q is flow rate, A is cross-sectional area, V is velocity, W is width, and D is depth. The cross-sectional area is calculated by multiplying the depth by the width of the stream. The velocity of the flowing water is defined as the quantity of discharge divided by the cross-sectional area. Therefore, there is an inverse relationship between the velocity and the cross-sectional area [40]. The velocity increases when the cross-sectional area is small, and vice versa. If we assume that the river has a constant flow rate (velocity × cross-sectional area = constant) and that the width of the river is fixed, then we can conclude that the velocity is inversely proportional to the depth of the river, in which case the velocity decreases in deep water and vice versa. Therefore, the amount of soil deposited increases as the velocity decreases. This explains why less soil is taken from deep water and more soil is taken from shallow water. A deep river will have water moving more slowly than a shallow river.

In addition to river depth and force of gravity, there are other factors that affect the flow velocity of water drops and cause them to accelerate or decelerate. These factors include obstructions, amount of soil existing in the path, topography of the land, variations in channel cross-sectional area, and the amount of dissolved soil being carried (the suspension load).

Water evaporation increases with temperature, with maximum evaporation at 100°C (boiling point). Conversely, condensation increases when the water vapor cools towards 0°C. The evaporation and condensation stages play important roles in the water cycle. Evaporation is necessary to ensure that the system remains in balance. In addition, the evaporation process helps to decrease the temperature and keeps the air moist. Condensation is a very important process as it completes the water cycle. When water vapor condenses, it releases the same amount of thermal energy into the environment that was needed to make it a vapor. However, in nature, it is difficult to measure the precise evaporation rate owing to the existence of different sources of evaporation; hence, estimation techniques are required [38].

The condensation process starts when the water vapor rises into the sky; then, as the temperature decreases in the higher layer of the atmosphere, the particles with less temperature have more chance to condense [41]. Figure 2(a) illustrates the situation where there are no attractions (or connections) between the particles at high temperature. As the temperature decreases, the particles collide and agglomerate to form small clusters, leading to clouds, as shown in Figure 2(b).

An interesting phenomenon in which some of the large water drops may eliminate other smaller water drops occurs during the condensation process as some of the water drops—known as collectors—fall from a higher layer to a lower layer in the atmosphere (in the cloud). Figure 3 depicts the action of a water drop falling and colliding with smaller

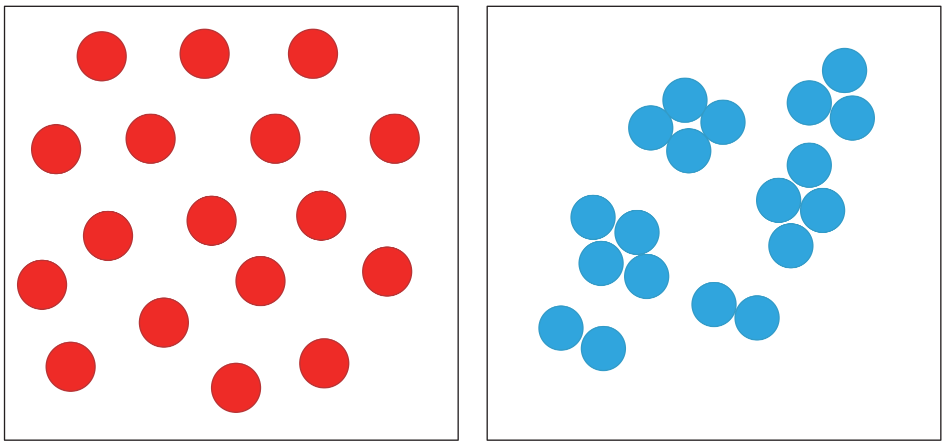

(a) Hot water (b) Cold water

Figure 2: Illustration of the effect of temperature on water drops.

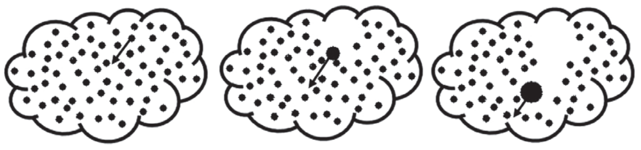


Figure 3: A collector water drop in action.

water drops. Consequently, the water drop grows as a result of coalescence with the other water drops [42].

Based on this phenomenon, only water drops that are sufficiently large will survive the trip to the ground. Of the numerous water drops in the cloud, only a portion will make it to the ground.

3.2. The Proposed Algorithm. Given the brief description of the hydrological cycle above, the IWD algorithm can be interpreted as covering only one stage of the overall water cycle—the flow stage. Although the IWD algorithm takes into account some of the characteristics and factors that affect the flowing water drops through rivers, and the actions and reactions along the way, the IWD algorithm neglects other factors that could be useful in solving optimization problems through adoption of other key concepts taken from the full hydrological process.

Studying the full hydrological water cycle gave us the inspiration to design a new and potentially more powerful algorithm within which the original IWD algorithm can be considered a subpart. We divide our algorithm into four main stages: Flow (Runoff), Evaporation, Condensation, and Precipitation.

The HCA can be described formally as consisting of a set of artificial water drops (WDs), such that

$$\mathrm{HCA} = \{\mathrm{WD}_1, \mathrm{WD}_2, \mathrm{WD}_3, \ldots, \mathrm{WD}_n\}, \quad \text{where } n \geq 1. \tag{4}$$

The characteristics associated with each water drop are as follows:

(i) V^{WD}: the velocity of the water drop.

(ii) $\mathrm{Soil}^{\mathrm{WD}}$: the amount of soil the water drop carries.

(iii) ψ^{WD}: the solution quality of the water drop.

The input to the HCA is a graph representation of a problems' solution space. The graph can be a fully connected undirected graph $G = (N, E)$, where N is the set of nodes and E is the set of edges between the nodes. In order to find a solution, the water drops are distributed randomly over the nodes of the graph and then traverse the graph (G), according to various rules, via the edges (E) searching for the best or optimal solution. The characteristics associated with each edge are the initial amount of soil on the edge and the edge depth.

The initial amount of soil is the same for all the edges. The depth measures the distance from the water surface to the riverbed, and it can be calculated by dividing the length of the edge by the amount of soil. Therefore, the depth varies and depends on the amount of soil that exists on the path and its length. The depth of the path increases when more soil is removed over the same length. A deep path can be interpreted as either the path being lengthy or there being less soil. Figures 4 and 5 illustrate the relationship between path depth, soil, and path length.

The HCA goes through a number of cycles and iterations to find a solution to a problem. One iteration is considered complete when all water drops have generated solutions based on the problem constraints. A water drop iteratively constructs a solution for the problem by continuously moving between the nodes. Each iteration consists of specific steps (which will be explained below). On the other hand, a cycle represents a varied number of iterations. A cycle is considered as being complete when the temperature reaches a specific value, which makes the water drops evaporate, condense, and precipitate again. This procedure continues until the termination condition is met.

3.3. Flow Stage (Runoff). This stage represents the construction stage where the HCA explores the search space. This is carried out by allowing the water drops to flow (scattered or regrouped) in different directions and constructing various solutions. Each water drop constructs a

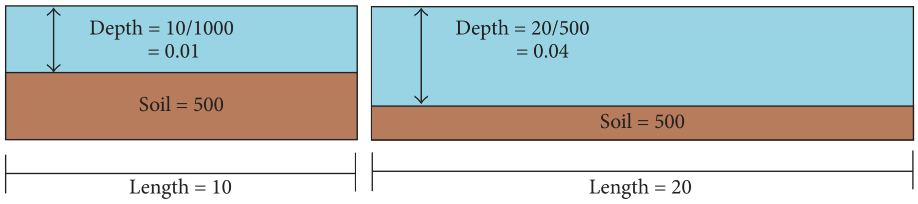


Figure 4: Depth increases with length increases for the same amount of soil.

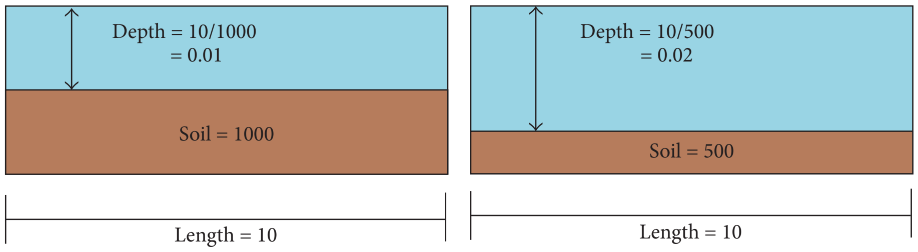


Figure 5: Depth increases when the amount of soil is reduced over the same length.

solution incrementally by moving from one point to another. Whenever a water drop wants to move towards a new node, it has to choose between different numbers of nodes (various branches). It calculates the probability of all the unvisited nodes and chooses the highest probability node taking into consideration the problem constraints. This can be described as a state transition rule that determines the next node to visit. In HCA, the node with the highest probability will be selected. The probability of the node is calculated using

$$P^{WD}{}_i(j) = \frac{f(\text{Soil}(i,j))^2 \times g(\text{Depth}(i,j))}{\sum_{k \notin vc(\text{WD})} \left(f(\text{Soil}(i,k))^2 \times g(\text{Depth}(i,k)) \right)}, \quad (5)$$

where $P^{WD}{}_i(j)$ is the probability of choosing node j from node i. $f(\text{Soil}(i,j))$ is equal to the inverse of the soil between i and j and is calculated using

$$f(\text{Soil}(i,j)) = \frac{1}{\varepsilon + \text{Soil}(i,j)}. \quad (6)$$

$\varepsilon = 0.01$ is a small value that is used to prevent division by zero. The second factor of the transition rule is the inverse of depth, which is calculated based on

$$g(\text{Depth}(i,j)) = \frac{1}{\text{Depth}(i,j)}. \quad (7)$$

$\text{Depth}(i,j)$ is the depth between nodes i and j and is calculated by dividing the length of the path by the amount of soil. This can be expressed as follows:

$$\text{Depth}(i,j) = \frac{\text{Length}(i,j)}{\text{Soil}(i,j)}. \quad (8)$$

The depth value can be normalized to be in the range [1–100], as it might be very small. The depth of the path is constantly changing because it is influenced by the amount of existing soil, where the depth is inversely proportional to the amount of the existing soil in the path of a fixed length. Water drops will choose paths with less depth over other paths. This enables exploration of new paths and diversifies the generated solutions. Assume the following scenario (depicted by Figure 6), where water drops have to choose between nodes A or B after node S. Initially, both edges have the same length and same amount of soil (line thickness represents soil amount). Consequently, the depth values for the two edges will be the same. After some water drops chose node A to visit, the edge (S-A) as a result has less soil and is deeper. According to the new state transition rule, the next water drops will be forced to choose node B because edge (S-B) will be shallower than (S-A). This technique provides a way to avoid stagnation and explore the search space efficiently.

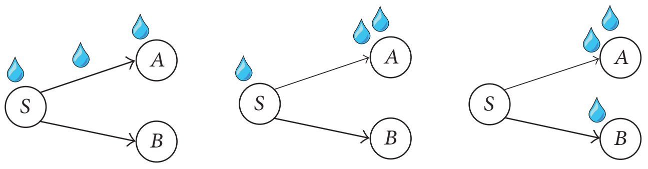


Figure 6: The relationship between soil and depth over the same length.

3.3.1. Velocity Update. After selecting the next node, the water drop moves to the selected node and marks it as visited. Each water drop has a variable velocity (V). The velocity of a water drop might be increased or decreased while it is moving based on the factors mentioned above (the amount of soil existing on the path, the depth of the path, etc.). Mathematically, the velocity of a water drop at time $(t+1)$ can be calculated using

$$V_{t+1}^{WD} = \left[K \times V_t^{WD} \right] + \alpha \left(\frac{V^{WD}}{\text{Soil}(i,j)} \right) + \sqrt[2]{\frac{V^{WD}}{\text{Soil}^{WD}}} + \left(\frac{100}{\psi^{WD}} \right) + \sqrt[2]{\frac{V^{WD}}{\text{Depth}(i,j)}}. \quad (9)$$

Equation (9) defines how the water drop updates its velocity after each movement. Each part of the equation has an effect on the new velocity of the water drop, where V_t^{WD} is the current water drop velocity and K is a uniformly distributed random number between $[0, 1]$ that refers to the roughness coefficient. The roughness coefficient represents factors that may cause the water drop velocity to decrease. Owing to difficulties measuring the exact effect of these factors, some randomness is considered with the velocity.

The second term of the expression in (9) reflects the relationship between the amount of soil existing on a path and the velocity of a water drop. The velocity of the water drops increases when they traverse paths with less soil. Alpha (α) is a relative influence coefficient that emphasizes this part in the velocity update equation. The third term of the expression represents the relationship between the amount of soil that a water drop is currently carrying and its velocity. Water drop velocity increases with decreasing amount of soil being carried. This gives weak water drops a chance to increase their velocity, as they do not hold much soil. The fourth term of the expression refers to the inverse ratio of a water drop's fitness, with velocity increasing with higher fitness. The final term of the expression indicates that a water drop will be slower in a deep path than in a shallow path.

3.3.2. Soil Update. Next, the amount of soil existing on the path and the depth of that path are updated. A water drop can remove (or add) soil from (or to) a path while moving based on its velocity. This is expressed by

$$\mathrm{Soil}(i,j) = \begin{cases} [PN * \mathrm{Soil}(i,j)] - \Delta\mathrm{Soil}(i,j) - \sqrt[2]{\dfrac{1}{\mathrm{Depth}(i,j)}} & \text{if } V^{\mathrm{WD}} \ge \mathrm{Avg}\left(\mathrm{all}_V{}^{\mathrm{WDS}}\right) \ (\text{Erosion}) \\ [PN * \mathrm{Soil}(i,j)] + \Delta\mathrm{Soil}(i,j) + \sqrt[2]{\dfrac{1}{\mathrm{Depth}(i,j)}} & \text{else } (\text{Deposition}). \end{cases} \tag{10}$$

PN represents a coefficient (i.e., sediment transport rate or gradation coefficient) that may affect the reduction in the amount of soil. The soil can be removed only if the current water drop velocity is greater than the average of all other water drops. Otherwise, an amount of soil is added (soil deposition). Consequently, the water drop is able to transfer an amount of soil from one place to another and usually transfers it from fast to slow parts of the path. The increasing soil amount on some paths favors the exploration of other paths during the search process and avoids entrapment in local optimal solutions. The amount of soil existing between node i and node j is calculated and changed using

$$\Delta\mathrm{Soil}(i,j) = \frac{1}{\mathrm{time}_{i,j}^{\mathrm{WD}}}, \tag{11}$$

such that

$$\mathrm{time}_{i,j}^{\mathrm{WD}} = \frac{\mathrm{Distance}(i,j)}{V_{t+1}^{\mathrm{WD}}}. \tag{12}$$

Equation (11) shows that the amount of soil being removed is inversely proportional to time. Based on the second law of motion, time is equal to distance divided by velocity. Therefore, time is proportional to velocity and inversely proportional to path length. Furthermore, the amount of soil being removed or added is inversely proportional to the depth of the path. This is because shallow soils will be easy to remove compared to deep soils. Therefore, the amount of soil being removed will decrease as the path depth increases. This factor facilitates exploration of new paths because less soil is removed from the deep paths as they have been used many times before. This may extend the time needed to search for and reach optimal solutions. The depth of the path needs to be updated when the amount of soil existing on the path changes using (8).

3.3.3. Soil Transportation Update. In the HCA, we consider the fact that the amount of soil a water drop carries reflects its solution quality. This can be done by associating the quality of the water drop's solution with its carrying soil value. Figure 7 illustrates the fact that a water drop with more carrying soil has a better solution.

To simulate this association, the amount of soil that the water drop is carrying is updated with a portion of the soil being removed from the path divided by the last solution quality found by that water drop. Therefore, the water drop with a better solution will carry more soil. This can be expressed as follows:

$$\mathrm{Soil}^{\mathrm{WD}} = \mathrm{Soil}^{\mathrm{WD}} + \frac{\Delta\mathrm{Soil}(i,j)}{\psi^{\mathrm{WD}}}. \tag{13}$$

The idea behind this step is to let a water drop preserves its previous fitness value. Later, the amount of carrying soil value is used in updating the velocity of the water drops.

3.3.4. Temperature Update. The final step that occurs at this stage is updating of the temperature. The new temperature value depends on the solution quality generated by the water drops in the previous iterations. The temperature will be updated as follows:

$$\mathrm{Temp}(t+1) = \mathrm{Temp}(t) + +\Delta\mathrm{Temp}, \tag{14}$$

where

$$\Delta\mathrm{Temp} = \begin{cases} \beta * \left(\dfrac{\mathrm{Temp}(t)}{\Delta D}\right) & \Delta D > 0 \\ \dfrac{\mathrm{Temp}(t)}{10} & \text{otherwise} \end{cases} \tag{15}$$

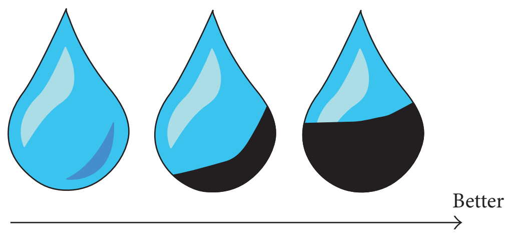


Figure 7: Relationship between the amount of carrying soil and its quality.

and where coefficient β is determined based on the problem. The difference (ΔD) is calculated based on

$$\Delta D = \text{MaxValue} - \text{MinValue}, \tag{16}$$

such that

$$\begin{aligned} \text{MaxValue} &= \max\left[\text{Solutions Quality}\left(\text{WDs}\right)\right], \\ \text{MinValue} &= \min\left[\text{Solutions Quality}\left(\text{WDs}\right)\right]. \end{aligned} \tag{17}$$

According to (16), increase in temperature will be affected by the difference between the best solution (MinValue) and the worst solution (MaxValue). Less difference means that there was not a lot of improvement in the last few iterations and, as a result, the temperature will increase more. This means that there is no need to evaporate the water drops as long as they can find different solutions in each iteration. The flow stage will repeat a few times until the temperature reaches a certain value. When the temperature is sufficient, the evaporation stage begins.

3.4. Evaporation Stage. In this stage, the number of water drops that will evaporate (the evaporation rate) when the temperature reaches a specific value is first determined. The evaporation rate is chosen randomly between one and the total number of water drops:

$$\text{ER} = \text{Random}\left(1, N\right), \tag{18}$$

where ER is evaporation rate, and N is number of water drops. According to the evaporation rate, a specific number of water drops is selected to evaporate. The selection is done using the roulette-wheel selection method, taking into consideration the solution quality of each water drop. Only the selected water drops evaporate and go to the next stage.

3.5. Condensation Stage. As the temperature decreases, water drops draw closer to each other and then collide and combine or bounce off. These operations are useful as they allow the water drops to be in direct physical contact with each other. This stage represents the collective and cooperative behavior between all the water drops. A water drop can generate a solution without direct communication (by itself); however, communication between water drops may lead to the generation of better solutions in the next cycle. In HCA, physical contact is used for direct communication between water drops, whereas the amount of soil on the edges can be considered indirect communication between the water drops. Direct communication (information exchange) has been proven to improve solution quality significantly when applied by other algorithms [37].

The condensation stage is considered to be a problem-dependent stage that can be redesigned to fit the problem specification. For example, one way of implementing this stage to fit the Travelling Salesman Problem (TSP) is to use local improvement methods. Using these methods enhances the solution quality and generates better results in the next cycle (i.e., minimizes the total cost of the solution), with the hypothesis that it will reduce the number of iterations needed and help to reach the optimal/near-optimal solution faster. Equation (19) shows the evaporated water (EW) selected from the overall population, where n represents the evaporation rate (number of evaporated water drops):

$$\text{EW} = \left\{\text{WD}_1, \text{WD}_2, \ldots, \text{WD}_n\right\}. \tag{19}$$

Initially, all the possible combinations (as pairs) are selected from the evaporated water drops to share their information with each other via collision. When two water drops collide, there is a chance either of the two water drops merging (coalescence) or of them bouncing off each other. In nature, determining which operation will take place is based on factors such as the temperature and the velocity of each water drop. In this research, we considered the similarity between the water drops' solutions to determine which process will occur. When the similarity is greater than or equal to 50% between two water drops' solutions, they merge. Otherwise, they collide and bounce off each other. Mathematically, this can be represented as follows:

$$\text{OP}\left(\text{WD}_1, \text{WD}_2\right) = \begin{cases} \text{Bounce}\left(\text{WD}_1, \text{WD}_2\right), & \text{Similarity} < 50\% \\ \text{Merge}\left(\text{WD}_1, \text{WD}_2\right), & \text{Similarity} \geq 50\%. \end{cases} \tag{20}$$

We use the Hamming distance [43] to count the number of different positions in two series that have the same length to obtain a measure of the similarity between water drops. For example, the Hamming distance between 12468 and 13458 is two. When two water drops collide and merge, one water drop (i.e., the collector) will become more powerful by eliminating the other one in the process also acquiring the characteristics of the eliminated water drop (i.e., its velocity):

$$\text{WD1}_{\text{Vel}} = \text{WD1}_{\text{Vel}} + \text{WD2}_{\text{Vel}}. \tag{21}$$

On the other hand, when two water drops collide and bounce off, they will share information with each other about the goodness of each node and how much a node contributes to their solutions. The bounce-off operation generates information that is used later to refine the quality of the water drops' solutions in the next cycle. The information is available to all water drops and helps them to choose a node that has a better contribution from all the possible nodes at the flow stage. The information consists of the weight of each node. The weight measures a node occurrence within a solution over the water drop solution quality. The idea behind sharing these details is to emphasize the best nodes found up to that point. Within this exchange, the water drops will favor those nodes in the next cycle. Mathematically, the weight can be calculated as follows:

$$\text{Weight}\,(\text{node}) = \frac{\text{Node}_{\text{occurrence}}}{\text{WD}_{\text{sol}}}. \qquad (22)$$

Finally, this stage is used to update the global-best solution found up to that point. In addition, at this stage, the temperature is reduced by 50°C. The lowering and raising of the temperature helps to prevent the water drops from sticking with the same solution every iteration.

3.6. Precipitation Stage. This stage is considered the termination stage of the cycle, as the algorithm has to check whether the termination condition is met. If the condition has been met, the algorithm stops with the last global-best solution. Otherwise, this stage is responsible for reinitializing all the dynamic variables, such as the amount of the soil on each edge, depth of paths, the velocity of each water drop, and the amount of soil it holds. The reinitialization of the parameters helps the algorithm to avoid being trapped in local optima, which may affect the algorithm's performance in the next cycle. Moreover, this stage is considered as a reinforcement stage, which is used to place emphasis on the best water drop (the collector). This is achieved by reducing the amount of soil on the edges that belong to the best water drop solution:

$$\text{Soil}\,(i,j) = 0.9 * \text{soil}\,(i,j), \quad \forall\,(i,j) \in \text{Best}^{\text{WD}}. \qquad (23)$$

The idea behind that is to favor these edges over the other edges in the next cycle.

3.7. Summarization of HCA Characteristics. The HCA can be distinguished from other swarm algorithms in the following ways:

(1) HCA involves a number of cycles and number of iterations (multiple iterations per cycle). This gives the advantage of performing certain tasks (e.g., solution improvements methods) every cycle instead of every iteration to reduce the computational effort. In addition, the cyclic nature of the algorithm contributes to self-organization as the system converges to the solution through feedback from the environment and internal self-evaluation. The temperature variable controls the cycle, and its value is updated according to the quality of the solution obtained in every iteration.

(2) The flow of the water drops is controlled by paths depth and soil amount heuristics (indirect communication). Paths depth factor affects the movement of water drops and helps to avoid choosing the same paths by all water drops.

(3) Water drops are able to gain or lose portion of their velocity. This idea supports the competition between water drops and prevents the sweep of one drop for the rest of the drops because a high-velocity water drop can carve more paths (i.e., remove more soils) than slower one.

(4) Soil can be deposited and removed, which helps to avoid a premature convergence and stimulates the spread of drops in different directions (more exploration).

(5) Condensation stage is utilized to perform information sharing (direct communication) among the water drops. Moreover, selecting a collector water drop represents an elitism technique, and that helps to perform exploitation for the good solutions. Further, this stage can be used to improve the quality of the obtained solutions to reduce the number of iterations.

(6) The evaporation process is a selection technique and helps the algorithm to escape from local optima solutions. The evaporation happens when there are no improvements in the quality of the obtained solutions.

(7) Precipitation stage is used to reinitialize variables and redistribute WDs in the search space to generate new solutions.

```
HCA procedure
(i) Problem input
(ii) Initialization of parameters
(iii) While (termination condition is not met)
    Cycle = Cycle + 1
    % Flow stage
    Repeat
            (a) Iteration = iteration + 1
            (b) For each water drop
                (1) Choose next node (Eqs. (5)–(8))
                (2) Update velocity (Eq. (9))
                (3) Update soil and depth (Eqs. (10)-(11))
                (4) Update carrying soil (Eq. (13))
            (c) End for
            (d) Calculate the solution fitness
            (e) Update local optimal solution
            (f) Update Temperature (Eqs. (14)–(17))
    Until (Temperature = Evaporation_Temperature)
    Evaporation (WDs)
    Condensation (WDs)
    Precipitation (WDs)
(iv) End while
```

Pseudocode 1: HCA pseudocode.

Condensation, evaporation, and precipitation ((5), (6), and (7)) above distinguish HCA from other water-based algorithms, including IWD and WCA. These characteristics are important and help make a proper balance between exploration exploitation, which helps in convergence towards the global solution. In addition, the stages of the water cycle are complementary to each other and help improve the overall performance of the HCA.

3.8. The HCA Procedure. Pseudocodes 1 and 2 show the HCA pseudocode.

```
Evaporation procedure (WDs)
    Evaluate the solution quality
    Identify Evaporation Rate (Eq. (18))
    Select WDs to evaporate
End
Condensation procedure (WDs)
    Information exchange (WDs) (Eqs. (20)–(22))
    Identify the collector (WD)
    Update global solution
    Update the temperature
End
Precipitation procedure (WDs)
    Evaluate the solution quality
    Reinitialize dynamic parameters
    Global soil update (belong to best WDs) (Eq. (23))
    Generate a new WDs population
    Distribute the WDs randomly
End
```

Pseudocode 2: The Evaporation, Condensation, and Precipitation pseudocode.

3.9. Similarities and Differences in Comparison to Other Water-Inspired Algorithms. In this section, we clarify the major similarities and differences between the HCA and other algorithms such as WCA and IWD. These algorithms share the same source of inspiration—water movement in nature. However, they are inspired by different aspects of the water processes and their corresponding events. In WCA, entities are represented by a set of streams. These streams keep moving from one point to another, which simulates the flow process of the water cycle. When a better point is found, the position of the stream is swapped with that of a river or the sea according to their fitness. The swap process simulates the effects of evaporation and rainfall rather than modelling these processes. Moreover, these effects only occur when the positions of the streams/rivers are very close to that of the sea. In contrast, the HCA has a population of artificial water drops that keep moving from one point to another, which represents the flow stage. While moving, water drops can carry/drop soil that change the topography of the ground by making some paths deeper or fading others. This represents the erosion and deposition processes. In WCA, no consideration is made for soil removal from the paths, which is considered a critical operation in the formation of streams and rivers. In HCA, evaporation occurs after certain iterations when the temperature reaches a specific value. The temperature is updated according to the performance of the water drops. In WCA there is no temperature and the evaporation rate is based on a ratio based on quality of solution. Another major difference is that there is no consideration for the condensation stage in WCA, which is one of the crucial stages in the water cycle. By contrast, in HCA, this stage is utilized to implement the information sharing concept between the water drops. Finally, the parameters, operations, exploration techniques, the formalization of each stage, and solution construction differ in both algorithms.

Despite the fact that the IWD algorithm is used with major modifications as a subpart of HCA, there are major differences between IWD and HCA. In IWD, the probability of selecting the next node is based on the amount of soil. In contrast, in HCA, the probability of selecting the next node is an association between the soil and the path depth, which enables the construction of a variety of solutions. This association is useful when the same amount of soil exists on different paths; therefore, the paths' depths are used to differentiate between these edges. Moreover, this association helps in creating a balance between the exploitation and exploration of the search process. The edge with less depth is chosen, and this favors the exploration. Alternatively, choosing the edge with less soil will favor exploitation. Further, in HCA, additional factors such as amount of soil in comparison to depth are considered in velocity updating, which allows the water drops to gain or lose velocity. This gives other water drops a chance to compete, as the velocity of water drops plays an important role in guiding the algorithm to discover new paths. Moreover, the soil update equation enables soil removal and deposition. The underlying idea is to help the algorithm to improve its exploration, avoid premature convergence, and avoid being trapped in local optima. In IWD only indirect communication is considered, while direct and indirect communications are considered in HCA. Furthermore, in HCA, the carrying soil is encoded with the solution quality; more carrying soil indicates a better solution. Finally, in HCA, three critical stages (i.e., evaporation, condensation, and precipitation) are considered to improve the performance of the algorithm. Table 1 summarizes the major differences between IWD, WCA, and HCA.

4. Experimental Setup

In this section, we explain how the HCA is applied to continuous problems.

4.1. Problem Representation. Typically, the input of the HCA is represented as a graph in which water drops can travel between nodes using the edges. In order to apply the HCA over the COPs, we developed a directed graph representation that exploits a topographical approach for continuous number representation. For a function with N variables and precision P for each variable, the graph is composed of ($N \times P$) layers. In each layer, there are ten nodes labelled from zero to nine. Therefore, there are ($10 \times N \times P$) nodes in the graph, as shown in Figure 8.

This graph can be seen as a topographical mountain with different layers or contour lines. There is no connection between the nodes in the same layer; this prevents a water drop from selecting two nodes from the same layer. A connection exists only from one layer to the next lower layer, in which a node in a layer is connected to all the nodes in the next layer. The value of a node in layer i is higher than that of a node in layer $i + 1$. This can be interpreted as the selected number from the first layer being multiplied by 0.1. The selected number from the second layer is found by multiplying by 0.01, and so on. This can be done easily using

$$X_{\text{value}} = \sum_{L=1}^{m} n \times 10^{-L}, \qquad (24)$$

Table 1: A summary of the main differences between IWD, WCA, and HCA.

Criteria	IWD	WCA	HCA
Flow stage	Moving water drops	Changing streams' positions	Moving water drops
Choosing the next node	Based on the soil	Based on river and sea positions	Based on soil and path depth
Velocity	Always increases	N/A	Increases and decreases
Soil update	Removal	N/A	Removal and deposition
Carrying soil	Equal to the amount of soil removed from a path	N/A	Is encoded with the solution quality
Evaporation	N/A	When the river position is very close to the sea position	Based on the temperature value, which is affected by the percentage of improvements in the last set of iterations
Evaporation rate	N/A	If the distance between a river and the sea is less than the threshold	Random number
Condensation	N/A	N/A	Enable information sharing (direct communication). Update the global-best solution, which is used to identify the collector
Precipitation	N/A	Randomly distributes a number of streams	Reinitializes dynamic parameters such as soil, velocity, and carrying soil

where L is the layer number, m is the maximum layer number for each variable (the precision digit's number), and n is the node in that layer. The water drops will generate values between zero and one for each variable. For example, if a water drop moves between nodes 2, 7, 5, 6, 2, and 4, respectively, then the variable value is 0.275624. The main drawback of this representation is that an increase in the number of high-precision variables leads to an increase in search space.

4.2. Variables Domain and Precision. As stated above, a function has at least one variable or up to n variables. The variables' values should be within the function domain, which defines a set of possible values for a variable. In some functions, all the variables may have the same domain range, whereas, in others, variables may have different ranges. For simplicity, the obtained values by a water drop for each variable can be scaled to have values based on the domain of each variable:

$$V_{\text{value}} = (U_B - L_B) \times \text{Value} + L_B, \tag{25}$$

where V_{value} is the real value of the variable, U_B is upper bound, and L_B is lower bound. For example, if the obtained value is 0.658752 and the variable's domain is $[-10, 10]$, then the real value will be equal to 3.17504. The values of continuous variables are expressed as real numbers. Therefore, the precision (P) of the variables' values has to be identified. This precision identifies the number of digits after the decimal point. Dealing with real numbers makes the algorithm faster than is possible by simply considering a graph of binary numbers as there is no need to encode/decode the variables' values to and from binary values.

4.3. Solution Generator. To find a solution for a function, a number of water drops are placed on the peak of this continuous variable mountain (graph). These drops flow down along the mountain layers. A water drop chooses only one number (node) from each layer and moves to the next layer below. This process continues until all the water drops reach the last layer (ground level). Each water drop may generate a different solution during its flow. However, as the algorithm iterates, some water drops start to converge towards the best water drop solution by selecting the highest node's probability. The candidate solutions for a function can be represented as a matrix, in which each row consists of a

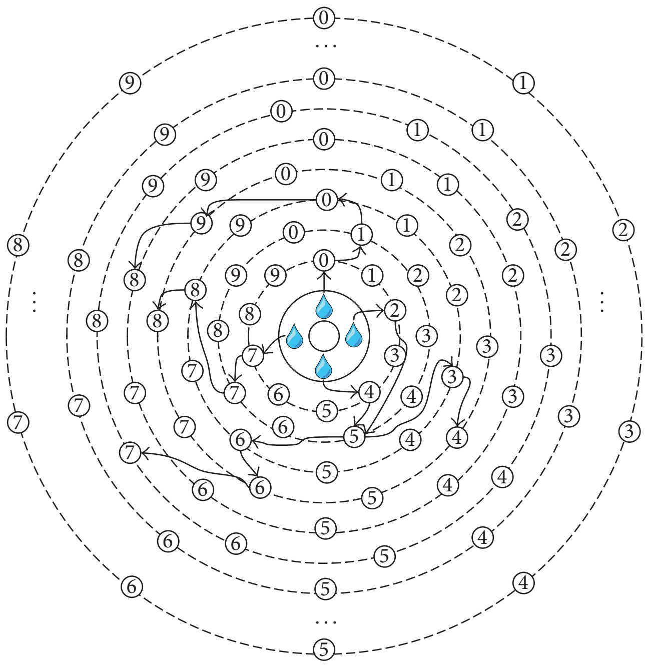


Figure 8: Graph representation of continuous variables.

Table 2: HCA parameters and their values.

Parameter	Value
Number of water drops	10
Maximum number of iterations	500/1000
Initial soil on each edge	10000
Initial velocity	100
Initial depth	Edge length/soil on that edge
Initial carrying soil	1
Velocity updating	$\alpha = 2$
Soil updating	$PN = 0.99$
Initial temperature	$50, \beta = 10$
Maximum temperature	100

water drop solution. Each water drop will generate a solution to all the variables of the function. Therefore, the water drop solution is composed of a set of visited nodes based on the number of variables and the precision of each variable:

$$\text{Solutions} = \begin{bmatrix} \mathrm{WD}_s^1 & 1 & 2 & \cdots & m_{N\times P} \\ \mathrm{WD}_s^2 & 1 & 2 & \cdots & m_{N\times P} \\ \mathrm{WD}_s^i & 1 & 2 & \cdots & m_{N\times P} \\ & \vdots & & \cdots & \\ \mathrm{WD}_s^k & 1 & 2 & \cdots & m_{N\times P} \end{bmatrix}. \tag{26}$$

An initial solution is generated randomly for each water drop based on the function dimensionality. Then, these solutions are evaluated, and the best combination is selected. The selected solution is favored with less soil on the edges belonging to this solution. Subsequently, the water drops start flowing and generating solutions.

4.4. Local Mutation Improvement. The algorithm can be augmented with mutation operation to change the solution of the water drops during collision at the condensation stage. The mutation is applied only on the selected water drops. Moreover, the mutation is applied randomly on selected variables from a water drop solution. For real numbers, a mutation operation can be implemented as follows:

$$V_{\text{new}} = 1 - (\beta \times V_{\text{old}}), \tag{27}$$

where β is a uniform random number in the range $[0, 1]$, V_{new} is the new value after the mutation, and V_{old} is the original value. This mutation helps to flip the value of the variable; if the value is large, then it becomes small, and vice versa.

4.5. HCA Parameters and Assumption. In the HCA, a number of parameters are considered to control the flow of the algorithm and to help to generate a solution. Some of these parameters need to be adjusted according to the problem domain. Table 2 lists the parameters and their values used in this algorithm after initial experiments.

The distance between the nodes in the same layer is not specified (i.e., no connection) because a water drop cannot choose two nodes from the same layer. The distance between the nodes in different layers is the same and equal to one unit. Therefore, the depth is adjusted to equal the inverse of the number of times a node is selected. Under this setting, a path between (x, y) will deepen with increasing number of selections of node y after node x. This adjustment is required because the internodal distance is constant as stipulated in the problem representation. Hence, considering the depth to equal the distance over the soil (see (8)) will not affect the outcome as supposed.

4.6. Experimental Results and Analysis. A number of benchmark functions were selected from the literature; see [4] for a full description of these functions. The mathematical formulations of the used functions are listed in the Appendix. The functions have a variety of properties with different types. In this paper, only unconstrained functions are considered. The experiments were conducted on a PC with a Core i5 CPU and 8 GB RAM using MATLAB on Microsoft Windows 7. The characteristics of these functions are summarized in Table 3.

For all the functions, the precision is assumed to be six significant figures for each variable. The HCA was executed twenty times with a maximum number of iterations of 500 for all functions, except for those with more than three variables, for which the maximum was 1000. The algorithm was tested without mutation (HCA) and with mutation (MHCA) for all the functions.

The results are provided in Table 4. The algorithm numbers in Table 4 (the column named "Number") maps to the same column in Table 3. For each function, the worst, average, and best values are listed. The symbol "#" represents the average number of iterations needed to reach the global solution. The results show that the algorithm gave satisfactory results for all of the functions tested, and it was able to find the global minimum solution within a small number of iterations for some functions. In order to evaluate the algorithm performance, we calculated the success rate for each function using

$$\text{Success rate}\% = \left(\frac{\text{Number of successful runs}}{\text{Total number of runs}}\right) \times 100. \tag{28}$$

The solution is accepted if the difference between the obtained solution and the optimum value is less than 0.001. The algorithm achieved 100% for all of the functions, except for Powell-4v [HCA: (60%), MHCA: (90%)], Sphere-6v [HCA: (60%), MHCA: (40%)], Rosenbrock-4v [HCA: (70%), MHCA: (100%)], and Wood-4v [HCA: (50%), MHCA: (80%)]. The numbers highlighted in boldface text in the "best" column represent the best value achieved in the cases where HCA or MHCA performed best; in all other cases HCA and MHCA were found to give the same "best" performance.

The results of HCA and MHCA were compared using the Wilcoxon Signed-Rank test. The P values obtained in the test were 0.2460, 0.1389, 0.8572, and 0.2543 for the *worst*, *average*, *best*, and *average number of iterations*, respectively. According to the P values, there is no significant difference between the

Table 3: Functions' characteristics.

Number	Function name	Lower bound	Upper bound	D	$f(x^*)$	Type
(1)	Ackley	−32.768	32.768	2	0	Many local minima
(2)	Cross-In-Tray	−10	10	2	−2.06261	Many local minima
(3)	Drop-Wave	−5.12	5.12	2	−1	Many local minima
(4)	Gramacy & Lee (2012)	0.5	2.5	1	−0.86901	Many local minima
(5)	Griewank	−600	600	2	0	Many local minima
(6)	Holder Table	−10	10	2	−19.2085	Many local minima
(7)	Levy	−10	10	2	0	Many local minima
(8)	Levy N. 13	−10	10	2	0	Many local minima
(9)	Rastrigin	−5.12	5.12	2	0	Many local minima
(10)	Schaffer N. 2	−100	100	2	0	Many local minima
(11)	Schaffer N. 4	−100	100	2	0.292579	Many local minima
(12)	Schwefel	−500	500	2	0	Many local minima
(13)	Shubert	−5.12	5.12	2	−186.7309	Many local minima
(14)	Bohachevsky	−100	100	2	0	Bowl-shaped
(15)	Rotated Hyper-Ellipsoid	−65.536	65.536	2	0	Bowl-shaped
(16)	Sphere (Hyper)	−5.12	5.12	2	0	Bowl-shaped
(17)	Sum Squares	−10	10	2	0	Bowl-shaped
(18)	Booth	−10	10	2	0	Plate-shaped
(19)	Matyas	−10	10	2	0	Plate-shaped
(20)	McCormick	$[-1.5, 4]$	$[-3, 4]$	2	−1.9133	Plate-shaped
(21)	Zakharov	−5	10	2	0	Plate-shaped
(22)	Three-Hump Camel	−5	5	2	0	Valley-shaped
(23)	Six-Hump Camel	$[-3, 3]$	$[-2, 2]$	2	−1.0316	Valley-shaped
(24)	Dixon-Price	−10	10	2	0	Valley-shaped
(25)	Rosenbrock	−5	10	2	0	Valley-shaped
(26)	Shekel's Foxholes (De Jong N. 5)	−65.536	65.536	2	0.99800	Steep ridges/drops
(27)	Easom	−100	100	2	−1	Steep ridges/drops
(28)	Michalewicz	0	3.14	2	−1.8013	Steep ridges/drops
(29)	Beale	−4.5	4.5	2	0	Other
(30)	Branin	$[-5, 10]$	$[0, 15]$	2	0.39788	Other
(31)	Goldstein-Price I	−2	2	2	3	Other
(32)	Goldstein-Price II	−5	−5	2	1	Other
(33)	Styblinski-Tang	−5	−5	2	−78.33198	Other
(34)	Gramacy & Lee (2008)	−2	6	2	−0.4288819	Other
(35)	Martin & Gaddy	0	10	2	0	Other
(36)	Easton and Fenton	0	10	2	1.744152	Other
(37)	Rosenbrock D1.2	−1.2	1.2	2	0	Other
(38)	Sphere	−5.12	5.12	3	0	Other
(39)	Hartmann 3-D	0	1	3	−3.86278	Other
(40)	Rosenbrock	−1.2	1.2	4	0	Other
(41)	Powell	−4	5	4	0	Other
(42)	Wood	−5	5	4	0	Other
(43)	Sphere	−5.12	5.12	6	0	Other
(44)	DeJong	−2.048	2.048	2	3905.93	Maximize function

Table 4: The obtained results with no mutation (HCA) and with mutation (MHCA).

Number	HCA				MHCA			
	Worst	Average	Best	#	Worst	Average	Best	#
(1)	8.88E − 16	8.88E − 16	8.88E − 16	220.2	8.88E − 16	8.88E − 16	8.88E − 16	279.35
(2)	−2.06E + 00	−2.06E + 00	−2.06E + 00	362	−2.06E + 00	−2.06E + 00	−2.06E + 00	196.1
(3)	−1.00E + 00	−1.00E + 00	−1.00E + 00	330.8	−1.00E + 00	−1.00E + 00	−1.00E + 00	220.4
(4)	−8.69E − 01	−8.69E − 01	−8.69E − 01	297.65	−8.69E − 01	−8.69E − 01	−8.69E − 01	345.95
(5)	0.00E + 00	0.00E + 00	0.00E + 00	212.25	0.00E + 00	0.00E + 00	0.00E + 00	284.8
(6)	−1.92E + 01	−1.92E + 01	−1.92E + 01	321.7	−1.92E + 01	−1.92E + 01	−1.92E + 01	302.75
(7)	4.23E − 07	1.31E − 07	1.50E − 32	399.5	3.60E − 07	8.65E − 08	1.50E − 32	394.8
(8)	1.63E − 05	4.70E − 06	**1.35E − 31**	399.45	9.97E − 06	3.06E − 06	1.60E − 09	366.3
(9)	0.00E + 00	0.00E + 00	0.00E + 00	289.4	0.00E + 00	0.00E + 00	0.00E + 00	352.9
(10)	0.00E + 00	0.00E + 00	0.00E + 00	284.1	0.00E + 00	0.00E + 00	0.00E + 00	333.85
(11)	2.93E − 01	2.93E − 01	2.93E − 01	358.6	2.93E − 01	2.93E − 01	2.93E − 01	336.25
(12)	6.82E − 04	4.19E − 04	2.08E − 04	337.6	1.28E − 03	6.42E − 04	**2.02E − 04**	323.75
(13)	−1.87E + 02	−1.87E + 02	−1.87E + 02	368	−1.87E + 02	−1.87E + 02	−1.87E + 02	391.45
(14)	0.00E + 00	0.00E + 00	0.00E + 00	264.9	0.00E + 00	0.00E + 00	0.00E + 00	288.25
(15)	0.00E + 00	0.00E + 00	0.00E + 00	309.9	0.00E + 00	0.00E + 00	0.00E + 00	291
(16)	0.00E + 00	0.00E + 00	0.00E + 00	283.15	0.00E + 00	0.00E + 00	0.00E + 00	293.6
(17)	0.00E + 00	0.00E + 00	0.00E + 00	305.95	0.00E + 00	0.00E + 00	0.00E + 00	271.6
(18)	2.03E − 05	6.33E − 06	**0.00E + 00**	433.5	1.97E − 05	5.78E − 06	2.96E − 08	363.5
(19)	0.00E + 00	0.00E + 00	0.00E + 00	258.35	0.00E + 00	0.00E + 00	0.00E + 00	287.55
(20)	−1.91E + 00	−1.91E + 00	−1.91E + 00	282.65	−1.91E + 00	−1.91E + 00	−1.91E + 00	225.8
(21)	1.12E − 04	5.07E − 05	4.73E − 06	328.15	3.94E − 04	1.30E − 04	**4.28E − 07**	242.05
(22)	0.00E + 00	0.00E + 00	0.00E + 00	238.8	0.00E + 00	0.00E + 00	0.00E + 00	288.9
(23)	−1.03E + 00	−1.03E + 00	−1.03E + 00	348.15	−1.03E + 00	−1.03E + 00	−1.03E + 00	332.4
(24)	3.08E − 04	9.69E − 05	3.89E − 06	394.25	5.46E − 04	2.67E − 04	**4.10E − 08**	380.55
(25)	2.03E − 09	2.14E − 10	0.00E + 00	350.55	2.25E − 10	3.21E − 11	0.00E + 00	282.55
(26)	9.98E − 01	9.98E − 01	9.98E − 01	354.6	9.98E − 01	9.98E − 01	9.98E − 01	370.35
(27)	−9.97E − 01	−9.98E − 01	−1.00E + 00	354.15	−9.97E − 01	−9.99E − 01	−1.00E + 00	344.6
(28)	−1.80E + 00	−1.80E + 00	−1.80E + 00	428	−1.80E + 00	−1.80E + 00	−1.80E + 00	398.1
(29)	9.25E − 05	5.25E − 05	**3.41E − 06**	379.9	1.33E − 04	7.37E − 05	2.56E − 05	393.75
(30)	3.98E − 01	3.98E − 01	3.98E − 01	345.8	3.98E − 01	3.98E − 01	3.98E − 01	409.9
(31)	3.00E + 00	3.00E + 00	3.00E + 00	255.5	3.00E + 00	3.00E + 00	3.00E + 00	310.8
(32)	1.00E + 00	1.00E + 00	1.00E + 00	410.7	1.00E + 00	1.00E + 00	1.00E + 00	379.95
(33)	−7.83E + 01	−7.83E + 01	−7.83E + 01	351	−7.83E + 01	−7.83E + 01	−7.83E + 01	341
(34)	−4.29E − 01	−4.29E − 01	−4.29E − 01	262.05	−4.29E − 01	−4.29E − 01	−4.29E − 01	286.65
(35)	0.00E + 00	0.00E + 00	0.00E + 00	370.9	0.00E + 00	0.00E + 00	0.00E + 00	347.1
(36)	1.74E + 00	1.74E + 00	1.74E + 00	385.75	1.74E + 00	1.74E + 00	1.74E + 00	408.8
(37)	9.22E − 06	3.67E − 06	**6.63E − 08**	382.2	4.66E − 06	2.60E − 06	2.79E − 07	351.6
(38)	4.43E − 07	8.27E − 08	0.00E + 00	371.3	2.13E − 08	4.50E − 09	0.00E + 00	330.45
(39)	−3.86E + 00	−3.86E + 00	−3.86E + 00	392.05	−3.86E + 00	−3.86E + 00	−3.86E + 00	327.5
(40)	1.94E − 01	7.02E − 02	**1.64E − 03**	816.5	8.75E − 02	3.04E − 02	2.79E − 03	806
(41)	2.42E − 01	9.13E − 02	3.91E − 03	863.95	1.12E − 01	4.26E − 02	**1.96E − 03**	840.85
(42)	1.12E + 00	2.91E − 01	7.62E − 08	753.95	2.67E − 01	6.10E − 02	**1.00E − 08**	694.4
(43)	1.05E + 00	3.88E − 01	**1.47E − 05**	879	8.96E − 01	3.10E − 01	6.51E − 03	505.2
(44)	3.91E + 03	3.91E + 03	3.91E + 03	**314.8**	3.91E + 03	3.91E + 03	3.91E + 03	373.85
		Mean		378.45				361.30

Table 5: Average execution time per number of variables.

Hypersphere function	2 variables (500 iterations)	3 variables (500 iterations)	4 variables (1000 iterations)	6 variables (1000 iterations)
Average execution time (second)	2.8186	4.0832	10.716	15.962

results. This indicates that the HCA algorithm is able to obtain optimal results without the mutation operation. However, it should be noted that using the mutation operation had the advantage of reducing the average number of iterations required to converge on a solution by 61%.

The success rate was lower for functions with more than four variables because of the problem representation, where the number of layers in the representation increases with the number of variables. Consequently, the HCA performance was limited to functions with few variables. The experimental results revealed a notable advantage of the proposed problem representation, namely, the small effect of the function domain on the solution quality and algorithm performance. In contrast, the performances of some algorithms are highly affected by the function domain, because their working mechanism depends on the distribution of the entities in a multidimensional search space. If an entity wanders outside the function boundaries, its position must be reset by the algorithm.

The effectiveness of the algorithm is validated by analyzing the calculated average values for each function (i.e., the average value of each function is close to the optimal value). This demonstrates the stability of the algorithm since it manages to find the global-optimal solution in each execution. Since the maximum number of iterations is fixed to a specific value, the average execution time was recorded for Hypersphere function with different number of variables. Consequently, the execution time is approximately the same for the other functions with the same number of variables. There is a slight increase in execution time with increasing number of variables. The results are reported in Table 5.

Based on the literature, the most commonly used criteria for evaluating algorithms are number of iterations (function evaluations), success rate, and solution quality (accuracy). In solving continuous problems, the performance of an algorithm can be evaluated using the number of iterations rather than execution time or solution accuracy. The aim of evaluating the number of iterations is to infer the convergence speed of the entities of an algorithm towards the global-optimal solution. Another aim is to remove hardware aspects from consideration. Generally speaking, premature or slow convergence is not preferred because it may lead to trapping in local optima solutions.

The performance of the HCA was also compared with that of the WCA on specific functions, where the WCA results were taken from [29]. The obtained results for both algorithms are displayed in Table 6. The symbol "#" represents the average number of iterations.

The results presented in Table 6 show that HCA and WCA produced optimal results with differing accuracies. Significance values of 0.75 *(worst)*, 0.97 *(average)*, and 0.86 *(best)* were found using Wilcoxon Signed Ranks Test, indicating no significant difference in performance between WCA and HCA.

In addition, the *average number of iterations* is also compared, and the P value (0.03078) indicates a significant difference, which confirms that the HCA was able to reach the global-optimal solution with fewer iterations than WCA (in 10 of 15 cases). However, the solutions' accuracy of the HCA over functions with more than two variables was lower than WCA.

HCA is also compared with other optimization algorithms in terms of average number of iterations. The compared algorithms were continuous ACO based on Discrete Encoding CACO-DE [44], Ant Colony System (ANTS), GA, BA, and GEM—using results from [25]. WCA and ER-WCA were also compared—with results taken from [29]. The success rate was 100% for all the algorithms, including HCA, except for Sphere-6v [HCA: (60%)] and Rosenbrock-4v [HCA: (70%)]. Table 7 shows the comparison results.

As can be seen in Table 7, the HCA results are very competitive. We applied Wilcoxon test to identify the significance of differences between the results. We also applied T-test to obtain P values along with the W-values. The results of the P/W-values are reported in Table 8.

Based on Table 8, there are significant differences between the results of ANTS, GA, BA, and HCA, where HCA reached the optimal solutions in fewer iterations than the ANTS, GA, and BA. Although the P value for "BA versus HCA" indicates that there is no significant difference, the W-value shows there is a significant difference between the two algorithms. Further, the performance of HCA CACO-DE, GEM, WCA, and ER-WCA is very competitive. Overall, the results also indicate that HCA is highly efficient for function optimization.

HCA, MHCA, Continuous GA (CGA), and Enhanced Continuous Tabu Search (ECTS) were also compared on some other functions in terms of average number of iterations required to reach an optimal solution. The results for CGA and ECTS were taken from [16]. The paper reported only the average number of iterations and the success rate of their algorithms. The success rate was 100% for all the algorithms. The results are listed in Table 9, where numbers in boldface indicate the lowest value.

From Table 9, it can be noted that both HCA and MHCA achieved optimal solutions in fewer iterations than the CGA. This is confirmed using Wilcoxon test and T-test on the obtained results; see Table 10.

Despite there being no significant difference between the results of HCA, MHCA, and ECTS, the means of HCA and MHCA results are less than ECTS, indicating competitive performance.

Table 6: Comparison between WCA and HCA in terms of results and iterations.

Function name	D	Bond	Best result	WCA				HCA			
				Worst	Avg	Best	#	Worst	Avg	Best	#
De Jong	2	±2.048	3905.93	3906.121	3905.940	3905.93	684	3905.93	3905.93	3905.93	**314.8**
Goldstein & Price I	2	±2	3	3.000968	3.000561	3.00002	980	$3.00E+00$	$3.00E+00$	**3.00E + 00**	**345.8**
Branin	2	±2	0.3977	0.398717	0.398272	0.397731	**377**	$3.98E-01$	$3.98E-01$	$3.98E-01$	379.9
Martin & Gaddy	2	[0, 10]	0	$9.29E-04$	$4.16E-04$	$5.01E-06$	**57**	$0.00E+00$	$0.00E+00$	**0.00E + 00**	262.1
Rosenbrock	2	±1.2	0	$1.46E-02$	$1.34E-03$	$2.81E-05$	**174**	$9.22E-06$	$3.67E-06$	**6.63E−08**	370.9
Rosenbrock	2	±10	0	$9.86E-04$	$4.32E-04$	$1.12E-06$	623	$2.03E-09$	$2.14E-10$	**0.00E + 00**	**350.6**
Rosenbrock	4	±1.2	0	$7.98E-04$	$2.12E-04$	**3.23E − 07**	**266**	$1.94E-01$	$7.02E-02$	$1.64E-03$	816.5
Hypersphere	6	±5.12	0	$9.22E-03$	$6.01E-03$	**1.34E − 07**	**101**	$1.05E+00$	$3.88E-01$	$1.47E-05$	879
Shaffer	2	±100	0	$9.71E-03$	$1.16E-03$	$2.61E-05$	8942	$0.00E+00$	$0.00E+00$	**0.00E + 00**	**284.1**
Goldstein & Price I	2	±5	3	3	3.00003	3	2400	$3.00E+00$	$3.00E+00$	$3.00E+00$	**345.8**
Goldstein & Price II	2	±5	1	1.1291	1.0118	1	47,500	$1.00E+00$	$1.00E+00$	$1.00E+00$	**255.5**
Six-Hump Came back	2	±10	−1.0316	−1.0316	−1.0316	−1.0316	3105	$-1.03E+00$	$-1.03E+00$	$-1.03E+00$	**348.2**
Easton & Fenton	2	[0, 10]	1.74	1.7441	1.7441	1.7441	650	$1.74E+00$	$1.74E+00$	$1.74E+00$	**385.6**
Wood	4	±5	0	$3.81E-05$	$1.58E-06$	**1.30E − 10**	15,650	$1.12E+00$	$2.91E-01$	$7.62E-08$	**754**
Powell quartic	4	±5	0	$2.87E-09$	$6.09E-10$	**1.12E − 11**	23,500	$2.42E-01$	$9.13E-02$	$3.91E-03$	**864**
			Mean				7000.6				463.8

Table 7: Comparison between HCA and other algorithms in terms of average number of iterations.

	Function name	*D*	*B*	CACO-DE	ANTS	GA	BA	GEM	WCA	ER-WCA	HCA
(1)	De Jong	2	±2.048	1872	6000	10,160	868	746	684	1220	**314.8**
(2)	Goldstein & Price I	2	±2	666	5330	5662	999	701	980	480	**345.8**
(3)	Branin	2	±2	-	1936	7325	1657	689	377	**160**	379.9
(4)	Martin & Gaddy	2	[0, 10]	340	1688	2488	526	258	**57**	100	262.05
(5a)	Rosenbrock	2	±1.2	-	6842	10,212	631	572	174	**73**	370.9
(5b)	Rosenbrock	2	±10	1313	7505	-	2306	2289	623	**94**	350.55
(6)	Rosenbrock	4	±1.2	624	8471	-	28,529	82,188	**266**	300	816.5
(7)	Hypersphere	6	±5.12	270	22,050	15,468	7113	423	101	**91**	879
(8)	Shaffer	2		-	-	-	-	-	8942	1110	**284.1**
	Mean			847.5	7477.8	8552.5	5328.6	10983.3	1356.0	403.1	444.8

Table 8: Comparison between P/W-values for HCA and other algorithms (based on Table 7).

Algorithm versus HCA	Using T-test	Using Wilcoxon test		Is the result significant at
	P value	Z-value	W-value (at critical value of w)	$P \leq 0.05$
CACO-DE versus HCA	0.324033	−0.9435	6 (at 0)	No
ANTS versus HCA	0.014953	−2.5205	0 (at 3)	Yes
GA versus HCA	0.005598	−2.2014	0 (at 0)	Yes
BA versus HCA	0.188434	−2.5205	0 (at 3)	Yes
GEM versus HCA	0.333414	−1.5403	7 (at 3)	No
WCA versus HCA	0.379505	−0.2962	20 (at 5)	No
ER-WCA versus HCA	0.832326	−0.5331	18 (at 5)	No

Table 9: Comparison of CGA, ECTS, HCA, and MHCA.

Number	Function name	D	CGA	ECTS	HCA	MHCA
(1)	Shubert	2	575	-	**368**	391.45
(2)	Bohachevsky	2	370	-	**264.9**	288.25
(3)	Zakharov	2	620	**195**	328.15	242.05
(4)	Rosenbrock	2	960	480	350.55	**282.55**
(5)	Easom	2	1504	1284	**354.6**	370.35
(6)	Branin	2	620	**245**	379.9	393.75
(7)	Goldstein-Price 1	2	410	**231**	345.8	409.9
(8)	Hartmann	3	582	548	392.05	**327.5**
(9)	Sphere	3	750	338	371.3	**330.45**
	Mean		710.1	474.4	350.6	337.4

Table 10: Comparison between P/W-values for CGA, ECTS, HCA, and MHCA.

Algorithm versus HCA	Using T-test	Using Wilcoxon test		Is the result significant at $P \leq 0.05$
	P value	Z-value	W-value (at critical value of w)	
CGA versus HCA	0.012635	−2.6656	0 (at 5)	Yes
CGA versus MHCA	0.012361	−2.6656	0 (at 5)	Yes
ECTS versus HCA	0.456634	−0.3381	12 (at 2)	No
ECTS versus MHCA	0.369119	−0.8452	9 (at 2)	No
HCA versus MHCA	0.470531	−0.7701	16 (at 5)	No

Figure 9 illustrates the convergence of global and local solutions for the Ackley function according to the number of iterations. The red line "Local" represents the quality of the solution that was obtained at the end of each iteration. This shows that the algorithm was able to avoid stagnation and keep generating different solutions in each iteration reflecting the proper design of the flow stage. We postulate that such solution diversity was maintained by the depth factor and fluctuations of the water drop velocities. The HCA minimized the Ackley function after 383 iterations.

Figure 10 shows a 2D graph for Easom function. The function has two variables, and the global minimum value is equal to −1 when ($X_1 = \pi$, $X_2 = \pi$). Solving this function is difficult as the flatness of the landscape does not give any indication of the direction towards global minimum.

Figure 11 shows the convergence of the HCA solving Easom function.

As can be seen, the HCA kept generating different solutions on each iteration while converging towards the global minimum value. Figure 12 shows the convergence of the HCA solving Rosenbrock function.

Figure 13 illustrates the convergence speed for HCA on the Hypersphere function with six variables. The HCA minimized the function after 919 iterations.

As shown in Figure 13, the HCA required more iterations to minimize functions with more than two variables because of the increase in the solution search space.

5. Conclusion and Future Work

In this paper, a new nature-inspired algorithm called the Hydrological Cycle Algorithm (HCA) is presented. In HCA, a collection of water drops pass through different phases such as flow (runoff), evaporation, condensation, and precipitation to generate a solution. The HCA has been shown to solve continuous optimization problems and provide high-quality solutions. In addition, its ability to escape local optima and find the global optima has been demonstrated,

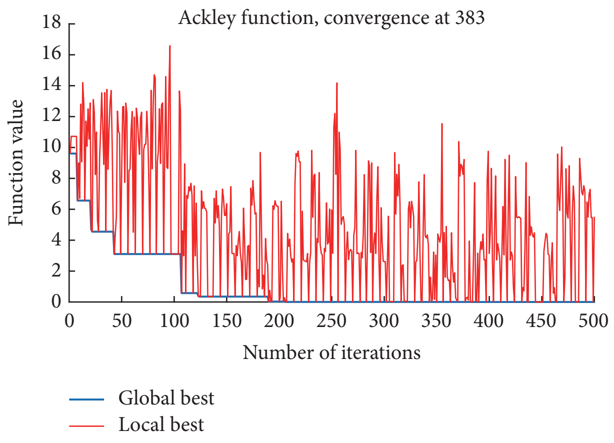


Figure 9: The global and local convergence of the HCA versus iterations number.

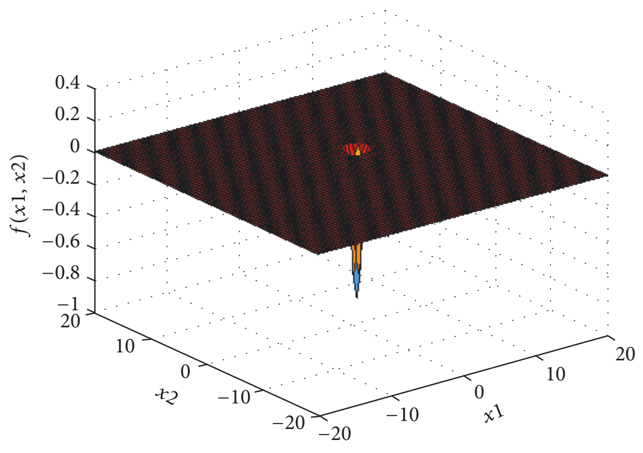


Figure 10: Easom function graph (from [4]).

which validates the convergence and the effectiveness of the exploration and exploitation process of the algorithm. One of the reasons for this improved performance is that both direct and indirect communication take place to share information among the water drops. The proposed representation of the continuous problems can easily be adapted to other problems. For instance, approaches involving gravity can regard the representation as a topology, with swarm particles starting at the highest point. Approaches involving placing weights on graph vertices can regard the representation as a form of optimized route finding.

The results of the benchmarking experiments show that HCA provides results in some cases better and in others not significantly worse than other optimization algorithms. But there is room for further improvement. Further work is required to optimize the algorithm's performance when dealing with problems involving two or more variables. In terms of solution accuracy, the algorithm implementation and the problem representation could be improved, including dynamic fine-tuning of the parameters. Possible further improvements to the HCA could include other techniques to dynamically control evaporation and condensation. Studying

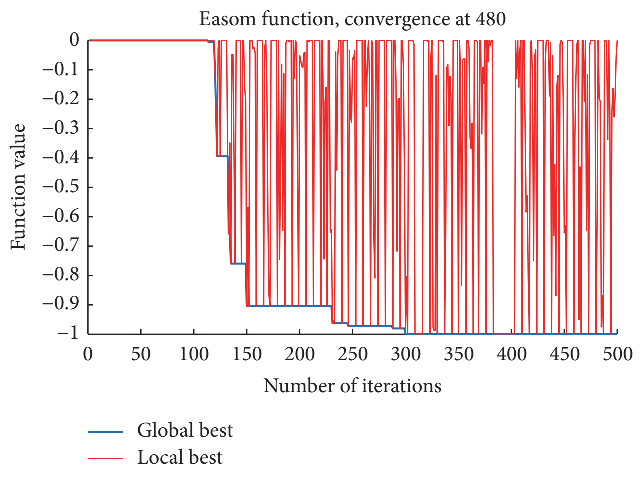


Figure 11: The global and local convergence of the HCA on Easom function.

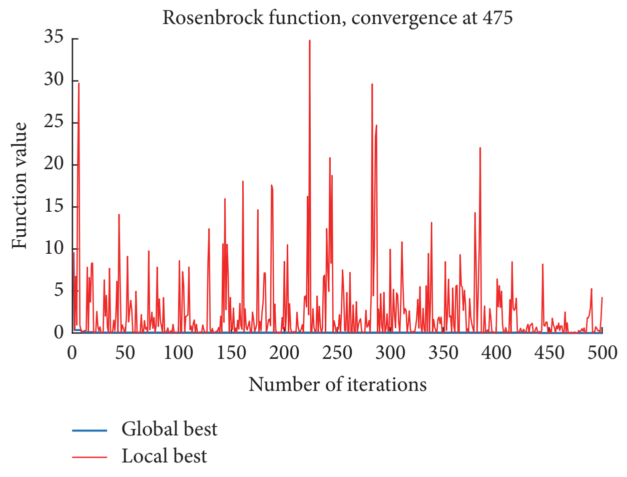


Figure 12: The global and local convergence of the HCA on Rosenbrock function.

the impact of the number of water drops on the quality of the solution is also worth investigating.

In conclusion, if a natural computing approach is to be considered a novel addition to the already large field of overlapping methods and techniques, it needs to embed the two critical concepts of self-organization and emergentism at its core, with intuitively based (i.e., nature-based) information sharing mechanisms for effective exploration and exploitation, as well as demonstrate performance which is at least on par with related algorithms in the field. In particular, it should be shown to offer something a bit different from what is available already. We contend that HCA satisfies these criteria and, while further development is required to test its full potential, can be used by researchers dealing with continuous optimization problems with confidence.

Appendix

Table 11 shows the mathematical formulations for the used test functions, which have been taken from [4, 24].

Table 11: The mathematical formulations.

Function name	Mathematical formulations
Ackley	$f(x) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1) \quad a = 20,\ b = 0.2 \text{ and } c = 2\pi$
Cross-In-Tray	$f(x) = -0.0001\left(\left\|\sin(x_1)\sin(x_2)\exp\left(\left\|100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right\|\right)\right\| + 1\right)^{0.1}$
Drop-Wave	$f(x) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5\left(x_1^2 + x_2^2\right) + 2}$
Gramacy & Lee (2012)	$f(x) = \frac{\sin(10\pi x)}{2x} + (x-1)^4$
Griewank	$f(x) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Holder Table	$f(x) = -\left\|\sin(x_1)\cos(x_2)\exp\left(\left\|1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right\|\right)\right\|$
Levy	$f(x) = \sin^2(3\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2\left[1 + \sin^2(2\pi w_d)\right]$ where, $w_i = 1 + \frac{x_i - 1}{4},\ \forall i = 1,\ldots,d$
Levy N. 13	$f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2[1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + \sin^2(3\pi x_2)]$
Rastrigin	$f(x) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right]$
Schaffer N. 2	$f(x) = 0.5 + \frac{\sin^2\left(x_1^2 + x_2^2\right) - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}$
Schaffer N. 4	$f(x) = 0.5 + \frac{\cos\left(\sin\left(\left\|x_1^2 - x_2^2\right\|\right)\right) - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}$
Schwefel	$f(x) = 418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{\left\|x_i\right\|}\right)$
Shubert	$f(x) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$
Bohachevsky	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
Rotated Hyper-Ellipsoid	$f(x) = \sum_{i=1}^{d}\sum_{j=1}^{i} x_j^2$
Sphere (Hyper)	$f(x) = \sum_{i=1}^{d} x_i^2$
Sum Squares	$f(x) = \sum_{i=1}^{d} i x_i^2$
Booth	$f(x) = \left(x_1 + 2x_2 - 7\right)^2 - \left(2x_1 + x_2 - 5\right)^2$
Matyas	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$
McCormick	$f(x) = \sin\left(x_1 + x_2\right) + \left(x_1 + x_2\right)^2 - 1.5x_1 + 2.5x_2 + 1$
Zakharov	$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$
Three-Hump Camel	$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$

Table 11: Continued.

Function name	Mathematical formulations
Six-Hump Camel	$f(x)=\left(4-2.1x_1^2+\frac{x_1^4}{3}\right)x_1^2+x_1x_2+\left(-4+4x_2^2\right)x_2^2$
Dixon-Price	$f(x)=(x_1-1)^2+\sum_{i=1}^{d} i\left(2x_i^2-x_{i-1}\right)^2$
Rosenbrock	$f(x)=\sum_{i=1}^{d-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$
Shekel's Foxholes (De Jong N. 5)	$f(x)=\left(0.002+\sum_{i=1}^{25}\frac{1}{i+(x_1-a_{1i})^6+(x_2-a_{2i})^6}\right)^{-1}$, where $a=\begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32\end{pmatrix}$
Easom	$f(x)=-\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$
Michalewicz	$f(x)=-\sum_{i=1}^{d}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$, where $m=10$
Beale	$f(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$
Branin	$f(x)=a\left(x_2-bx_1^2+cx_1-r\right)^2+s(1-t)\cos(x_1)+s$ $a=1,\ b=\frac{5.1}{(4\pi^2)},\ c=\frac{5}{\pi},\ r=6,\ s=10,\ \text{and } t=\frac{1}{8\pi}$
Goldstein-Price I	$f(x)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]$ $\times\left[30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$
Goldstein-Price II	$f(x)=\exp\left\{\frac{1}{2}\left(x_1^2+x_2^2-25\right)\right\}^2+\sin^4(4x_1-3x_2)+\frac{1}{2}(2x_1+x_2-10)^2$
Styblinski-Tang	$f(x)=\frac{1}{2}\sum_{i=1}^{d}\left(x_i^4-16x_i^2+5x_i\right)$
Gramacy & Lee (2008)	$f(x)=x_1\exp\left(-x_1^2-x_2^2\right)$
Martin & Gaddy	$f(x)=(x_1-x_2)^2+\left(\frac{(x_1+x_2-10)}{3}\right)^2$
Easton and Fenton	$f(x)=\left(12+x_1^2+\frac{1+x_2^2}{x_1^2}+\frac{x_1^2x_2^2+100}{(x_1x_2)^4}\right).\left(\frac{1}{10}\right)$
Hartmann 3-D	$f(x)=-\sum_{i=1}^{4}\alpha_i\exp\left(-\sum_{j=1}^{3}A_{ij}\left(x_j-p_{ij}\right)^2\right)$, where $\alpha=(1.0,1.2,3.0,3.2)^T,\ A=\begin{pmatrix}3.0 & 10 & 30\\ 0.1 & 10 & 35\\ 3.0 & 10 & 30\\ 0.1 & 10 & 35\end{pmatrix},\ P=10^{-4}\begin{pmatrix}3689 & 1170 & 2673\\ 4699 & 4387 & 7470\\ 1091 & 8732 & 5547\\ 381 & 5743 & 8828\end{pmatrix}$
Powell	$f(x)=\sum_{i=1}^{d/4}\left[\left(x_{4i-3}+10x_{4i-2}\right)^2+5\left(x_{4i-1}-x_{4i}\right)^2+\left(x_{4i-2}-2x_{4i-1}\right)^4+10\left(x_{4i-3}-x_{4i}\right)^4\right]$
Wood	$f(x_1,x_2,x_3,x_4)=100(x_1-x_2)^2+(x_1-1)^2+(x_3-1)^2+90\left(x_3^2-x_4\right)^2$ $+\ 10.1\left((x_2-1)^2+(x_4-1)^2\right)+19.8(x_2-1)(x_4-1)$
DeJong	$\max\quad f(x)=(3905.93)-100\left(x_1^2-x_2\right)^2-(1-x_1)^2$

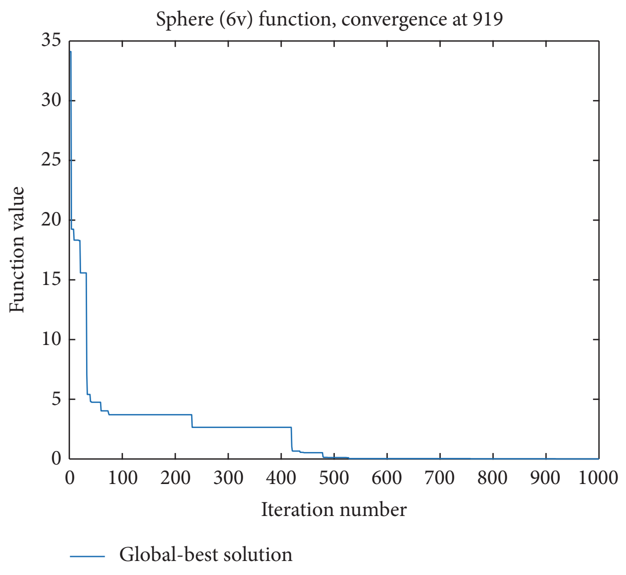


Figure 13: Convergence of HCA on the Hypersphere function with six variables.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] F. Rothlauf, "Optimization problems," in *Design of Modern Heuristics: Principles and Application*, pp. 7–44, Springer, Berlin, Germany, 2011.

[2] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*, vol. 76, John & Wiley Sons, 2013.

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

[4] S. Surjanovic and D. Bingham, *Virtual library of simulation experiments: test functions and datasets*, Simon Fraser University, 2013, http://www.sfu.ca/~ssurjano/index.html.

[5] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[6] M. Mathur, S. B. Karale, S. Priye, V. K. Jayaraman, and B. D. Kulkarni, "Ant colony approach to continuous function optimization," *Industrial & Engineering Chemistry Research*, vol. 39, no. 10, pp. 3814–3822, 2000.

[7] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[8] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 993, pp. 25–39, 1995.

[9] N. Monmarché, G. Venturini, and M. Slimane, "On how Pachycondyla apicalis ants suggest a new search algorithm," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 937–946, 2000.

[10] J. Dréo and P. Siarry, "Continuous interacting ant colony algorithm based on dense heterarchy," *Future Generation Computer Systems*, vol. 20, no. 5, pp. 841–856, 2004.

[11] L. Liu, Y. Dai, and J. Gao, "Ant colony optimization algorithm for continuous domains based on position distribution model of ant colony foraging," *The Scientific World Journal*, vol. 2014, Article ID 428539, 9 pages, 2014.

[12] V. K. Ojha, A. Abraham, and V. Snášel, "ACO for continuous function optimization: A performance analysis," in *Proceedings of the 2014 14th International Conference on Intelligent Systems Design and Applications, ISDA 2014*, pp. 145–150, Japan, November 2014.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass, USA, 1992.

[14] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, USA, 1998.

[15] H. Maaranen, K. Miettinen, and A. Penttinen, "On initial populations of a genetic algorithm for continuous optimization problems," *Journal of Global Optimization*, vol. 37, no. 3, pp. 405–436, 2007.

[16] R. Chelouah and P. Siarry, "Continuous genetic algorithm designed for the global optimization of multimodal functions," *Journal of Heuristics*, vol. 6, no. 2, pp. 191–213, 2000.

[17] Y.-T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing*, vol. 8, no. 2, pp. 849–857, 2008.

[18] K. James and E. Russell, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, IEEE, Perth, WA, Australia, 1942.

[19] W. Chen, J. Zhang, Y. Lin et al., "Particle swarm optimization with an aging leader and challengers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241–258, 2013.

[20] J. F. Schutte and A. A. Groenwold, "A study of global optimization using particle swarms," *Journal of Global Optimization*, vol. 31, no. 1, pp. 93–108, 2005.

[21] P. S. Shelokar, P. Siarry, V. K. Jayaraman, and B. D. Kulkarni, "Particle swarm and ant colony algorithms hybridized for improved continuous optimization," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 129–142, 2007.

[22] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 2, pp. 1980–1987, IEEE, Portland, OR, USA, 2004.

[23] S. C. Esquivel and C. A. C. Coello, "On the use of particle swarm optimization with multimodal functions," in *Proceedings of the Congress on Evolutionary Computation (CEC '03)*, pp. 1130–1136, IEEE, Canberra, Australia, December 2003.

[24] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm—a novel tool for complex optimisation," in *Proceedings of the Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference*, Elsevier, July 2006.

[25] A. Ahrari and A. A. Atai, "Grenade Explosion Method—a novel tool for optimization of multimodal functions," *Applied Soft Computing*, vol. 10, no. 4, pp. 1132–1140, 2010.

[26] H. Shah-Hosseini, "Problem solving by intelligent water drops," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pp. 3226–3231, sgp, September 2007.

[27] H. Shah-Hosseini, "An approach to continuous optimization by the intelligent water drops algorithm," in *Proceedings of the 4th International Conference of Cognitive Science, ICCS 2011*, pp. 224–229, Iran, May 2011.

[28] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, "Water cycle algorithm - A novel metaheuristic optimization method for solving constrained engineering optimization problems," *Computers & Structures*, vol. 110-111, pp. 151–166, 2012.

[29] A. Sadollah, H. Eskandar, A. Bahreininejad, and J. H. Kim, "Water cycle algorithm with evaporation rate for solving constrained and unconstrained optimization problems," *Applied Soft Computing*, vol. 30, pp. 58–71, 2015.

[30] Q. Luo, C. Wen, S. Qiao, and Y. Zhou, "Dual-system water cycle algorithm for constrained engineering optimization problems," in *Proceedings of the Intelligent Computing Theories and Application: 12th International Conference, ICIC 2016*, D.-S. Huang, V. Bevilacqua, and P. Premaratne, Eds., pp. 730–741, Springer International Publishing, Lanzhou, China, August 2016.

[31] A. A. Heidari, R. Ali Abbaspour, and A. Rezaee Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks," *Neural Computing and Applications*, vol. 28, no. 1, pp. 57–85, 2017.

[32] M. M. al-Rifaie, J. M. Bishop, and T. Blackwell, "Information sharing impact of stochastic diffusion search on differential evolution algorithm," *Memetic Computing*, vol. 4, no. 4, pp. 327–338, 2012.

[33] T. Hogg and C. P. Williams, "Solving the really hard problems with cooperative search," in *Proceedings of the National Conference on Artificial Intelligence*, p. 231, 1993.

[34] C. Ding, L. Lu, Y. Liu, and W. Peng, "Swarm intelligence optimization and its applications," in *Proceedings of the Advanced Research on Electronic Commerce, Web Application, and Communication: International Conference, ECWAC 2011*, G. Shen and X. Huang, Eds., pp. 458–464, Springer, Guangzhou, China, April 2011.

[35] L. Goel and V. K. Panchal, "Feature extraction through information sharing in swarm intelligence techniques," *Knowledge-Based Processes in Software Development*, pp. 151–175, 2013.

[36] Y. Li, Z.-H. Zhan, S. Lin, J. Zhang, and X. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Information Sciences*, vol. 293, pp. 370–382, 2015.

[37] M. Mavrovouniotis and S. Yang, "Ant colony optimization with direct communication for the traveling salesman problem," in *Proceedings of the 2010 UK Workshop on Computational Intelligence, UKCI 2010*, UK, September 2010.

[38] T. Davie, *Fundamentals of Hydrology*, Taylor & Francis, 2008.

[39] J. Southard, *An Introduction to Fluid Motions, Sediment Transport, and Current-Generated Sedimentary Structures [Ph. D. thesis]*, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006.

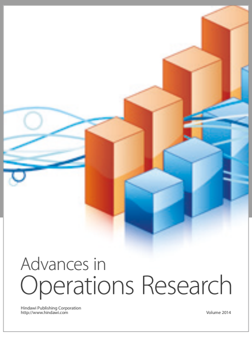
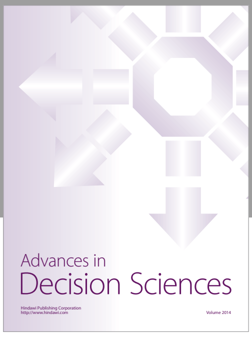
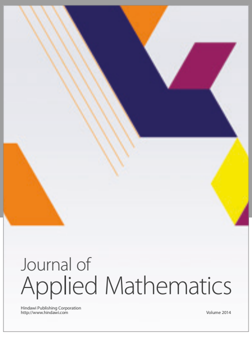
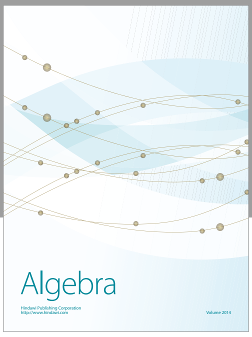
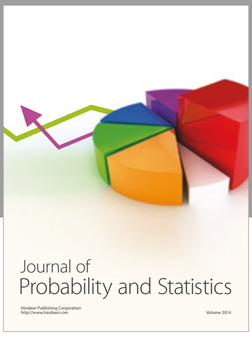
[40] B. R. Colby, *Effect of Depth of Flow on Discharge of Bed Material*, US Geological Survey, 1961.

[41] M. Bishop and G. H. Locket, *Introduction to Chemistry*, Benjamin Cummings, San Francisco, Calif, USA, 2002.

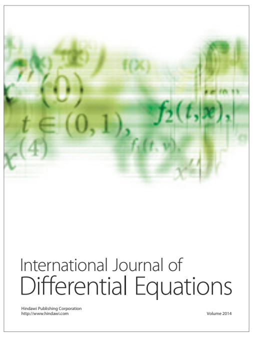
[42] J. M. Wallace and P. V. Hobbs, *Atmospheric Science: An Introductory Survey*, vol. 92, Academic Press, 2006.

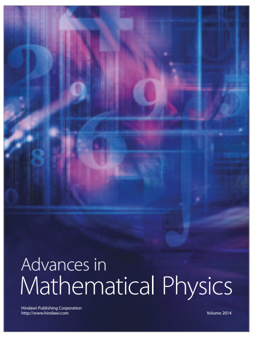
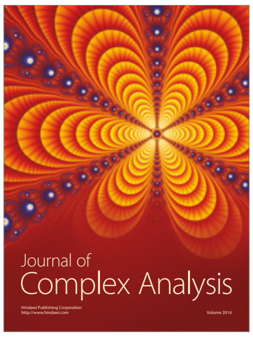
[43] R. Hill, *A First Course in Coding Theory*, Clarendon Press, 1986.

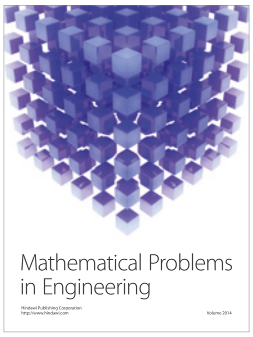
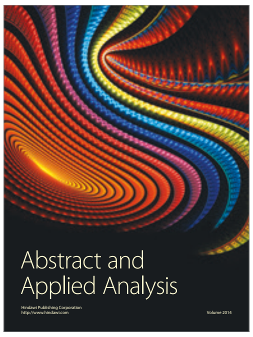
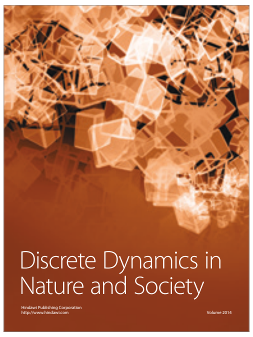
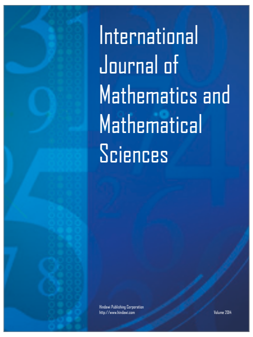
[44] H. Huang and Z. Hao, "ACO for continuous optimization based on discrete encoding," in *Proceedings of the International Workshop on Ant Colony Optimization and Swarm Intelligence*, pp. 504-505, 2006.


Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

Algebra

Journal of
Probability and Statistics

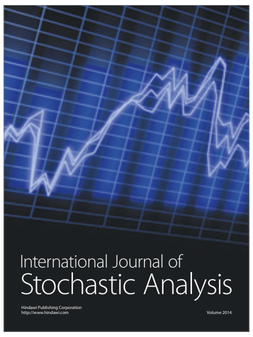
The Scientific
World Journal


International Journal of
Differential Equations

International Journal of
Combinatorics

Advances in
Mathematical Physics

Journal of
Complex Analysis

Journal of
Mathematics

Mathematical Problems
in Engineering

Abstract and
Applied Analysis

Discrete Dynamics in
Nature and Society

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces

International Journal of
Stochastic Analysis

Journal of
Optimization