Simulated Raindrop Algorithm for Global Optimization

Amin Ibrahim, IEEE Member
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology
Oshawa, Canada
amin.ibrahim@uoit.ca

Shahryar Rahnamayan, SMIEEE
Department of Electrical, Computer, and Software Engineering
University of Ontario Institute of Technology
Oshawa, Canada
shahryar.rahnamayan@uoit.ca

Miguel Vargas Martin, IEEE Member
Faculty of Business and IT
University of Ontario Institute of Technology
Oshawa, Canada
miguel.vargasmartin@uoit.ca

***Abstract*— In this paper, we propose a novel single-solution based metaheuristic algorithm called Simulated Raindrop (SRD). The SRD algorithm is inspired by the principles of raindrops. When rain falls on the land, it normally flows from higher altitude to a lower due to gravity, while choosing the optimum path towards the lowest point on the landscape. We compared the performance of simulated annealing (SA) against the proposed SRD method on 8 commonly utilized benchmark functions. Experimental results confirm that SRD outperforms SA on all test problems in terms of variant performance measures, such as convergence speed, accuracy of the solution, and robustness.**

Keywords—Nature-inspired algorithms; S-metaheuristic; raindrop; global optimization; simulated annealing.

I. Introduction

Nowadays, real-world applications are increasingly complex and more encompassing, in the sense that more decision variables are used to model complex situations and more input data and parameters are available to capture the complexity of the problems themselves. As a result, most real-world optimization problems cannot be solved using polynomial-time algorithms (i.e., they are NP-hard problems). Since finding exact or approximate solutions in NP-complete and NP-hard problems still poses a real challenge despite the impact of recent advances in computer technology, there are numerous approximation methods capable of finding "good" solutions in a "reasonable" time. Due to the inherent complexities and dynamics we have in nature, and its ability to approach its own problems, nature is the main source of inspiration for solving our complex problems [1].

Most nature-inspired algorithms are population based, in which multiple agents interact to solve or accomplish a given task. Though arguably nature-inspired algorithms are still at their early stages, many have shown a great potential in solving very complicated problems with diverse applications in engineering, business, economics, and communication networks. For example, evolutionary algorithms (EAs) are nature-inspired population-based methods taken from the biological evolution of living organisms to adapt to their ecosystem. The main genetic-based operations of EAs are selection (the fittest organisms replace the weakest for the next generation) [2], mutation (a subset of genes is chosen randomly and the allele value of the chosen genes is changed), and crossover (replacing some of the genes in one parent by corresponding genes of the other). There has been numerous biological evolution inspired algorithms since the early 1990s and among all EAs, genetic algorithms (GAs) are especially popular [3].

GA is the first evolutionary-based stochastic optimization algorithm in which organisms evolve by rearranging genetic material to survive in hostile environments challenging them. GA was proposed by Holland [4] and it has shown outstanding achievements in solving many economics, engineering and science real-world applications. Differential Evolution (DE) is also one of the most successful evolutionary algorithms; developed by Rainer Storn and Kenneth Price [5], it has solved numerous global optimization problems effectively. The main components of DE and GA are similar except that in DE mutation is the result of arithmetic combinations of individuals whereas in GA mutation is the result of small perturbations to the genes of an individual.

Another nature-inspired population-based algorithm is swarm intelligence. It is inspired from the social behaviour of species such as ants, bees, wasps, termite, fish, and birds which cooperate or compete for food. Among the most successful swarm-based optimization algorithms are particle swarm optimization (PSO) and ant colony optimization (ACO). PSO was introduced by Kennedy and Eberhart [6] in 1995. PSO simulate the behaviour of a flock of birds. In

PSO, each solution is a "particle" and each particle has two values: fitness value which is calculated by fitness function, and velocity which indicates the direction of particles. ACO was proposed by Dorigo [7] as his PhD thesis. ACO mimics the collective behavior of ants to perform complex tasks such as transportation of food and finding food sources. It was observed that ants communicate using a chemical trail called pheromone that is left on the ground during their journey to and from food sources and their nest. Remarkably an ant colony is able to find the shortest path between two points using the smelt quantity of pheromone.

The organization of the paper is as follows: Section II discusses related work, mainly an intelligent water drop algorithm. Section III provides the technical description of the proposed algorithm, called simulated raindrop (SRD). Section IV briefly discusses simulated annealing (SA) and defines the benchmark continuous optimization functions utilized in our experiments. Section V provides the experimental settings and corresponding results. Finally, Section VI concludes with a summary and future work.

II. Related Work

In 2007, Shah-Hosseini [8] proposed a population-based heuristic algorithm - called intelligent water drops algorithm (IWD) - for solving the traveling salesman problem. IWD algorithm tries to simulate the processes that occur in the natural river systems and the interaction among water drops in a river. He observed that a river often chooses an optimum path regarding the conditions of its surroundings before it reaches a lake or sea. He and others later adopted the IWD algorithm to successfully solve a number of known optimization problems, such as: Multidimensional Knapsack Problem (MKP) [9], n-queen puzzle [10], and Robot Path Planning [10], and automatic multilevel thresholding [12].

In 2012, Shah-Hosseini [13] also proposed an IWD for continuous optimization (IWD-CO) where he combined IWD with a mutation-based local search (IWD-CO) to find the optimal values of numerical functions. Although he showed that the IWD algorithm can be modified to handle continuous optimization, his work needs further experimental verification with regard convergence speed and solution accuracy.

While the inspirations for IWD and SRD are similar, they have very different approach in representing/solving optimization problems. First of all, IWD is a population-based heuristic (problem-specific) algorithm whereas SRD is a single-solution based metaheuristic. Second, the mechanics of IWD is different form SRD: IWD updates its current location to the next location/path based on the amount of soil on its beds and updates the velocity by the amount nonlinearly proportional to the inverse of the soil between the two locations. On the other hand SRD updates its current to the next solution based on the splash generated by the rain drop as it hits the ground. In SRD, the use of splashes is discussed in the next section.

III. Proposed SRD Algorithm

The SRD algorithm is a single-solution based metaheuristics (S-metaheuristics) inspired by the principles of raindrops. When rain hits ground, it tends to keep moving towards the lowest point on the landscape due to gravity. Similar to all S-metaheuristics, SRD starts with a single candidate solution and tries to improve it by selecting promising candidate solutions from a set of generated candidates. The analogy between the physical rainfall and SRD are as follows: the terrain represents the objective function; the flow of water from higher altitude to lower altitude is similar to local search on raindrop splashes, and the lowest point on the landscape is the optimal solution.

A. Intialization

Similar to all stochastic heuristics or metaheuristic algorithms, SRD starts by generating an initial candidate solution. This initialization (raindrop) is selected uniform randomly as follows:

Let $x = \{x_1, x_2, x_3, \dots, x_D\}$ represent an initial candidate solution, then

$$x_i = rand(x_{min}, x_{max}) \quad (1)$$

$$N_s = \left\lceil \frac{D}{n} \right\rceil + m \quad (2)$$

$$v_i = rand(x_{\min}, x_{\max}) \quad (3)$$

$$s_i = \begin{cases} x_{min}, & if\ x_i + v_i < x_{min} \\ x_{max}, & if\ x_i + v_i > x_{max} \\ x_j + v_i, & otherwise \end{cases}, \quad (4)$$

where $i = \{1,2,\dots,D\}$, D is the problem dimension, x_{min} and x_{max} are the variable boundaries, N_s number of splashes, v is the splash displacement, s is the splash location and n and m are control parameters. The main goal of Eq. 2 is to establish the minimum number of splashes generated by a given problem to m at the increment of one for every n dimension. In this paper, $m = 3$ and $n = 10$ are selected to be the optimal control parameter values after running some trail experiments. Since this work is in its initial stages, we need to conduct control parameter analysis as we move forward with this research.

B. Splash Generation

In every iteration, SRD generates N_s splashes for the raindrop. The main purpose of the splashes is to "sense" the neighboring landscape so that they guide the raindrop to the next best location. Splashes are generated b according to the following rules.

1) Splash displacment: SRD has a constant splash size (interms of its radius) where the size of splashes are not consistently increasing or decreasing. However the dispalcement of every splash varies. The splash displacement has an essential role in the efficiency of the algorithm. The splash dispacement is defined as follows:

$$v_i = \begin{cases} rand(x_{min}, x_{max}), & imCount \leq 0 \\ rand\left(\frac{x_{min}}{imCount}, \frac{x_{max}}{imCount}\right), & otherwise \end{cases} \quad (5)$$

where x_{min} and x_{max} are the variable boundaries, and $imCount$ is the difference of non-improving moves and improving moves so far and it is defined as:

$$imCount = \begin{cases} imCount + 1, \ if \ \exists s_i : f(s_i) < f(x), \\ \qquad\qquad i \in (1, N_s) \\ imCount - 1, \ otherwise \end{cases} \quad (6)$$

At the earlier stages (iteration) of the algorithm the displacement is large; thus this will encourage the diversification in the search space during the exploration phase. As the raindrop approaches the optimal solution during exploitation, the displacement of the splashes decrease and this will intensify the search in the optimal region of the search space. Steps 9, 11–12 and 21–22 from Table 1 indicate the implementation of splash sizing strategy.

2) Splash replacement strategy: Splashes are generated and replaced according to the folowing rules.

a) Scenario 1 (solution is improved in the previous iteration): If the solution is improved in the previous step (i.e., $\exists s_i : f(s_i) < f(x)$) then at the next iteration the raindrop moves to the location of best improved splash in the search space. Moreover, the number of splashes are reduced by half, and the displacement of one of the splash will be the same as the displacement of the best improved splash in the previous step (see Fig. 1(a)). This is from our observation that water streams tend to move in the same direction unless the landscape changes. Thus, it is not necessary to generate as many splashes as the original number of splashes. As such this feature will result in the reduction of the function calls. However, since the best improving path (displacement) is not guaranteed to be the optimal path, we still generate random splashes to "sense" the landscape in search of a better candidate solution. Steps 6–16 from Table 1 implement the Scenario 1 splash replacement strategy.

TABLE I. PSEUDOCODE FOR SIMULATED RAINDROP (SRD). $\boldsymbol{x_0}$: INITIAL CANDIDATE SOLUTION, $\boldsymbol{N_s}$: MAXIMUM NUMBER OF SPLASHES, $\boldsymbol{D}$: PROBLEM DIMENSION, $\boldsymbol{N_v}$: NUMBER OF SPLASHES IN THE CURRENT ITERATION, $\boldsymbol{v_i}$: DISPLACEMENT OF THE $\boldsymbol{i}$TH SPLASH, $\boldsymbol{v_{best}}$: THE BEST IMPROVED DISPLACEMENT, $\boldsymbol{x_{min}}$: LOWER VARIABLE BOUND, $\boldsymbol{x_{max}}$: UPPER VARIABLE BOUND, $\boldsymbol{imCount}$: THE DIFFERENCE OF IMPROVING AND NON-IMPROVING MOVES, AND $\boldsymbol{f(\cdot)}$:OBJECTIVE FUNCTION.

Simulated Raindrop Algorithm (SRD)

```
1.  $x = x_0$ ; /* Generation of the initial candidate solution/raindrop */
2.  $N_s = \lceil \frac{D}{n} \rceil + m$; /*Number of splashes */
3.  $Impoved$ = False; /* Initialize move info*/
4.  $imCount = 0$ /* Initialize improving move count to zero */
5.  Repeat
6.      If $impoved$ Then
7.          $imCount--$;
8.          $N_v = rand(1, N_s/2)$ /* Generate at most $N_s/2$ splashes */
9.          $s_1 = x + v_{best}$
10.         For $i = 2$ to $N_v$
11.             If $imCount \leq 0$ Then $v_i = rand(x_{min}, x_{max})$
12.             Else $v_i = rand(x_{min}/imCount, x_{max}/imCount)$
13.             End If
14.             $s_i = x + v_i$
15.         End For
16.         $Improved$ = False
17.     Else
18.         $imCount++$;
19.         $N_v = rand(1, N_s)$
20.         For $i = 2$ to $N_v$
21.             If $imCount \leq 0$ Then $v_i = rand(x_{min}, x_{max})$
22.             Else $v_i = rand(x_{min}/imCount, x_{max}/imCount)$
23.             End If
24.             $s_i = x + v_i$
25.         End For
26.     End If
27.     For $i = 1$ to $N_s$
28.         If $f(x) \leq f(s_i)$ Then
29.             $x = s_i$ /* move the raindrop to the best improved location */
30.             $v_{best} = v_i$
31.             $Improved$ = True
32.         End If
33.     End For
34. Until Stopping criteria satisfied
35. Output: Best solution found.
```

b) Scenario 2 (solution is not improved in the previous iteration): If the solution is not improved in the previous step (i.e., $\forall s_i: f(s_i) > f(x)$) then at the next iteration, the raindrop stays at the same location and new random splashes will be generated (see Fig. 1(b)). Steps 17–26 from Table 1 show the implementation of Scenario 2 splash replacement strategy.

C. *Selection Strategy*

In every iteration, the best splash is selected and the raindrop moves to a new location based on best improvement strategy (steepest decent for a minimization problem).

$$x = \begin{cases} s_{best}, \; if \; f(s_{best}) < f(x) \wedge \forall s_i: f(s_{best}) \\ \qquad\qquad < f(s_i), i \in (1, N_s) \\ x, \; otherwise \end{cases} \tag{7}$$

Where s_{best} is the best improving splash. Steps 28–31 from Table 1 show the implementation of splash selection strategy.

As we generate raindrop splashes in every iteration, some of the splashes may go off the upper or lower bounds (x_{min}, x_{max}) of the variable due to the randomness of the system. Thus, every splash generated is bounded by the problem bounds as follows:

Let $s_i(s_{i1}, s_{i2}, \ldots, s_{iD})$ be the i^{th} splash with its displacement $v_i(v_{i1}, v_{i2}, \ldots, v_{iD})$, and $x\,(x_1, x_2, \ldots, x_D)$ be the location of the current drop, then the displacement and the location of the splash is generated according to the following rule:

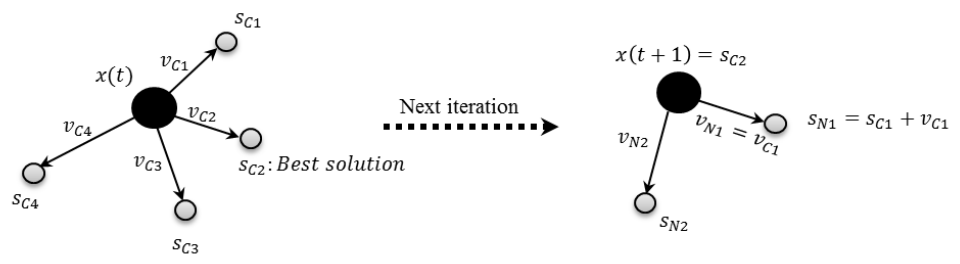
$s_1 = x + v_{best}$ if the solution is improved in the previous step, otherwise, the every splash is generated similar to equation 4.

$$s_{ij} = \begin{cases} x_{min}, & if \; x_j + v_{ij} < x_{min} \\ x_{max}, & if \; x_j + v_{ij} > x_{max} \\ x_j + v_{ij}, & otherwise \end{cases} \tag{8}$$

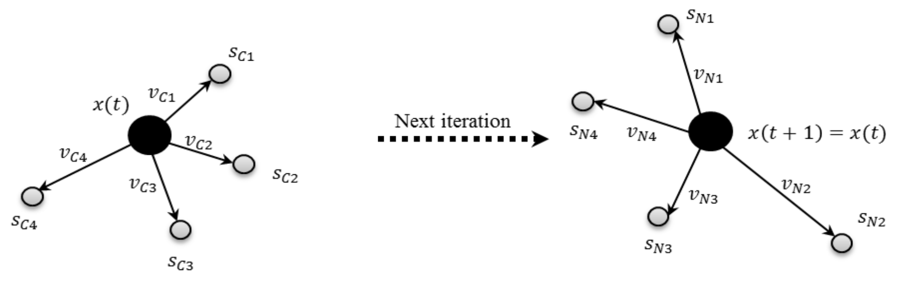
where $v_{best}$is displacement of the best improved splash in the previous step.

IV. EXPERIMENTAL COMPARISON AND SETTINGS

This section describes one of the most widely used S-metaheuristics, simulated annealing (SA), to compare against the performance and quality of the proposed algorithms. Parameter settings for each algorithm, and the benchmark problems used in our experiments are also explained.



(a) Scenario 1: The solution is improved in the previous iteration.



(b) Scenario 2: The solution is not improved in the previous iteration.

Fig. 1. Raindrop with its associated splash displacements and positions. a) Scenario 1: $\exists s_i: f(s_i) < f(x)$; at the next iteration, the raindrop moves to the location of best improved splash in the decision space. Reduce the number of splashes by half and the displacement of the one of the splash is that of the best improved splash. b) Scenario 2: $\forall s_i: f(s_i) > f(x)$; at the next iteration, the raindrop stays at the same location and generates new splashes.

TABLE II. SRD, OSRD, AND BSRD PARAMETER SETTINGS. $\boldsymbol{x_{min}}$: LOWER VARIABLE BOUND, $\boldsymbol{x_{max}}$: UPPER VARIABLE BOUND, $\boldsymbol{D}$: PROBLEM DIMENSION, AND $\boldsymbol{P_{current}}$: CURRENT CANDIDATE SOLUTION.

Initial solution	Uniform random number
Max number of splashes	$N_s = \left\lceil \frac{D}{10} \right\rceil + 3$
New solution	$P_{current} + rand\left(\frac{x_{min}}{imCount}, \frac{x_{max}}{imCount}\right)$

A. Simulated Annealing

Simulated annealing (SA) is a single-solution based probabilistic optimization method proposed by Kirkpatrick and Vecchi [14]. SA is inspired by the physical process whereby materials are treated with heat and slowly cooled to alter their microstructure and as a result they have a strong crystalline structure.

Like all S-metaheuristics, SA starts with a single candidate solution. At each iteration, a random neighbor is generated and compared with the current candidate solution. If the neighbor point improves the current solution then the current solution is replaced by the neighbor candidate solution. Otherwise, the current solution is accepted based on a given probability that depends on the current temperature and the amount of energy difference between the neighbor point and the current point. This characteristic of SA allows non-improving solutions to be accepted and hence avoiding being trapped in local optima.

At each level of temperature, many neighbors are explored until an equilibrium state is reached. Then, the temperature is gradually decreased according to a cooling schedule (usually by a geometric scheduling [14]) so that few and few non-improving solutions are accepted. Finally, SA terminates when the stopping criteria is met (e.g., when the probability of accepting a move is negligible because the temperature is close to 0). Table III reproduces the SA algorithm.

TABLE III. PSEUDOCODE FOR SIMULATED ANNEALING (SA). $\boldsymbol{S_0}$: INITIAL CANDIDATE SOLUTION, $\boldsymbol{T_{max}}$: INITIAL TEMPERATURE, $\boldsymbol{S'}$: NEIGHBOUR SOLUTION, $\boldsymbol{\Delta E}$: ENERGY DIFFERENCE BETWEEN THE CURRENT SOLUTION AND THE NEIGHBOR CANDIDATE SOLUTION, $\boldsymbol{g(T)}$: TEMPERATURE UPDATE FUNCTION, AND $\boldsymbol{f}(\cdot)$: OBJECTIVE FUNCTION.

```
**Simulated Annealing Algorithm**
$s = s_0$ ; /* Generation of the initial solution */
$T = T_{max}$ ; /* Starting temperature */
**Repeat**
    **Repeat** /* At a fixed temperature */
        Generate a random neighbor $s'$ ;
        $\Delta E = f(s') - f(s)$ ;
        **If** $\Delta E \le 0$ **Then** $s = s'$ /* Accept the neighbor solution */
        **Else** Accept s' with a probability $e^{\frac{-\Delta E}{T}}$
    **Until** Equilibrium condition
    $T = g(T)$; /* Temperature update */
**Until** Stopping criteria satisfied
**Output:** Best solution found.
```

The control parameters set for SA in all our experiments are in Table IV.

B. Benchmark Functions

In order to test the quality of the SRD and comparing that with the SA, we have utilized 8 widely used minimization benchmark functions [15-17]. All test functions have an optimum value at zero except f_4; the optimum value is located at one. All test functions are scalable problems, in which functions $f_1 - f_4$ and f_7 are unimodal and functions f_5, f_6 and f_8 are multimodal. Despite the fact f_4 is a unimodal function, it is non-convex and the optimum value is located inside a long, narrow, parabolic shaped flat valley which makes it very challenging for many optimizers. All the test functions used in this paper are to be minimized. Table V shows the numerical benchmark functions utilized in this study.

TABLE IV. SA PARAMETER SETTINGS. $\boldsymbol{x_{min}}$: LOWER FUNCTION BOUND, $\boldsymbol{x_{max}}$: UPPER FUNCTION BOUND, $\boldsymbol{D}$: PROBLEM DIMENSION, AND $\boldsymbol{P_{current}}$: CURRENT CANDIDATE SOLUTION.

Initial solution	Uniform random
New solution	$P_{current} + rand\left(\frac{x_{min}}{15}, \frac{x_{max}}{15}\right)$
Starting temperature	400
Cooling schedule	$T = 0.99 \times T$
Stopping criteria	$NFC = 1000 \times D$

V. EXPERIMENTAL SETTINGS AND VERIFICATION

Three series of experiments have been conducted for the comparison of the proposed method versus the traditional SA using the eight selected scalable benchmark functions. The main difference among these three experiments consisted in the dimension of the problems. The experiments consisted in having a respective dimensionality of 30, 50, and 100. The stopping criteria for all the algorithms are $1000 \times D$, where D is the problem dimension.

The results did not vary much for $D = 30$ and $D = 50$; therefore, only experiments with $D = 30$ and $D = 100$ will be discussed. However, the results for $D = 50$ can be found in Table VII and Figure 3.

As explained earlier, the proposed method is compared with SA. Due to the stochastic nature of the SRD and SA, each method is executed independently 50 times. The mean and standard deviations are compiled for comparison purposes. t-tests were performed with the null hypothesis that two means are equal with 0.95 confidence value.

A. Experimental Series 1: Low Dimension (D=30)

The first experiment is conducted in running the two optimization methods to solve the selected benchmark problems. The number of variables is fixed to 30.

As illustrated in Table VI, the results show that the proposed method outperforms SA on all the eight benchmark problems. Moreover, the results found by the SRD had lower standard deviation (i.e., the results found by SRD are more consistent because of a low fluctuation).

TABLE V. NUMERICAL BENCHMARK FUNCTIONS USED IN THIS STUDY

Problem	Objective Function	Variable Bounds	Global Optimum
De Jong	$f_1(X)=\sum_{i=1}^{n} x_i^2$	$-5.12 \le x_i \le 5.12$	$f_1(0,\dots,0)=0$
Axis Parallel Hyper-Ellipsoid	$f_2(X)=\sum_{i=1}^{n} i x_i^2$	$-5.12 \le x_i \le 5.12$	$f_2(0,\dots,0)=0$
Schwefel's Problem 1.2	$f_3(X)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	$-65 \le x_i \le 65$	$f_3(0,\dots,0)=0$
Rosenbrock's Valley	$f_4(X)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(1-x_i)^2]$	$-2 \le x_i \le 2$	$f_4(1,\dots,1)=0$
Rastrigin's Function	$f_5(X)=10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	$-5.12 \le x_i \le 5.12$	$f_5(0,\dots,0)=0$
Griewangk's Function	$f_6(X)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$-600 \le x_i \le 600$	$f_6(0,\dots,0)=0$
Sum of Different Power	$f_7(X)=\sum_{i=1}^{n}\lvert x_i\rvert^{(i+1)}$	$-1 \le x_i \le 1$	$f_7(0,\dots,0)=0$
Ackley's Problem	$f_8(X)=-20exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n}x_i^2}{n}}\right)-exp\left(\sum_{i=1}^{n}\frac{\cos(2\pi x_i)}{n}\right)+20+e$	$-32 \le x_i \le 32$	$f_8(0,\dots,0)=0$

It can be seen that the proposed method performed very well for problems f_1, f_2, f_3, f_6 and f_7 by finding near to optimal solutions. The performance SRD for problems f_4, and $f_8$was modest. And, it had more difficulties with the problem f_5. In overall, f_5 can be considered as the most difficult problem.

TABLE VI. COMPARISON OF SA AND SRD (D = 30). MEAN BEST AND STANDARD DEVIATION (STD DEV) OF 50 RUNS AND 30,000 FUNCTION CALLS ARE REPORTED. THE BEST ALGORITHM IS EMPHASIZED IN EMPHASIZED IN BOLDFACE. "*" INDICATES THE TWO-TAILED T-VALUE AT 0.05 LEVEL OF SIGNIFICANCE.

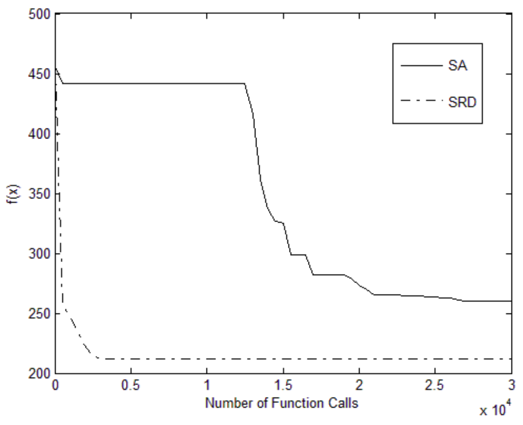
	SA	SRD
	Mean (Std Dev)	Mean (Std Dev)
f_1	1.537765 (0.7493272)	**4.192E-09** (1.696E-09)*
f_2	1.1505374 (0.3939867)	**2.345E-05** (9.468E-06)*
f_3	17.57048 (9.3081783)	**0.0010048** (0.0003012)*
f_4	26.451223 (2.9499834)	**22.8139** (1.838198)
f_5	295.25987 (80.046491)	**262.1036** (51.27188)*
f_6	1.7401887(0.228839)	**0.01437** (0.0104781)*
f_7	0.1616829 (0.077932)	**3.201E-08** (7.458E-09)*
f_8	21.198157(0.1448411)	**19.488083** (0.1719674)*

The proposed method outperformed significantly SA for the problem f_3. It offered better precision for the problems f_1, f_2, f_5, f_6, and f_7. Finally, it offers better precision than SA for the problems f_4 and f_8; however, the difference was not as large as the other previous problems. According to the t-test results, all except for problem $f_5$are statistically significant. Moreover, SRD exhibited significantly lower standard deviation in all problems, except for problems f_4 and f_8, indicating more robust results.

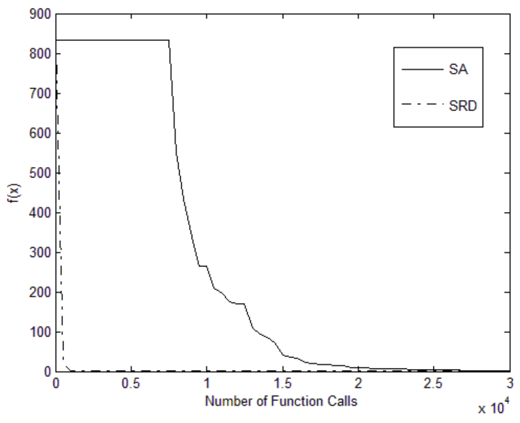
When comparing the convergence of SRD and SA, SRD exhibited a rapid convergence in all test problems except in $f_4$and f_8. However, SRD achieved better final result in these test problems. Figure 2 show the convergence SRD and SA. For each convergence related experiments, the initial solution was kept the same for both algorithms.

TABLE VII. COMPARISON OF SA AND SRD (D = 50). MEAN BEST AND STANDARD DEVIATION (STD DEV) OF 50 RUNS AND 50,000 FUNCTION CALLS ARE REPORTED. THE BEST ALGORITHM IS EMPHASIZED IN EMPHASIZED IN BOLDFACE. "*" INDICATES THE TWO-TAILED T-VALUE AT 0.05 LEVEL OF SIGNIFICANCE.

	SA	SRD
	Mean (Std Dev)	Mean (Std Dev)
f_1	0.4026678 (0.1439543)	**7.28E-09** (1.834E-09)*
f_2	1.0377076 (0.8782136)	**0.0006575** (0.0004058)*
f_3	232.8676 (57.337249)	**0.0419695** (0.0114963)*
f_4	48.59713 (12.653562)	**45.0707** (10.167395)
f_5	709.02457 (144.91634)	**428.55917** (52.153313)*
f_6	0.8623445 (0.0574953)	**0.0093709** (0.0106182)*
f_7	0.0004592 (0.0001943)	**2.11E-08** (4.977E-09)*
f_8	20.115243 (0.0408853)	**19.471323** (0.1429701)*

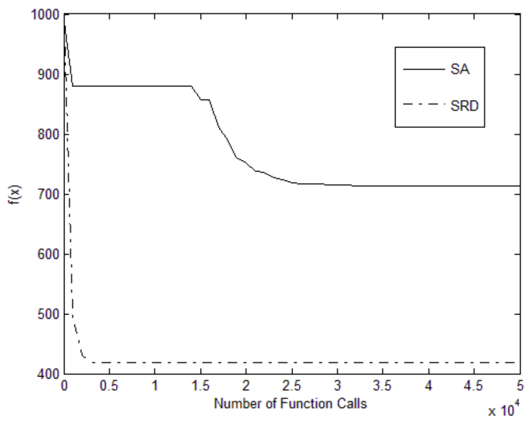


(a) f_5

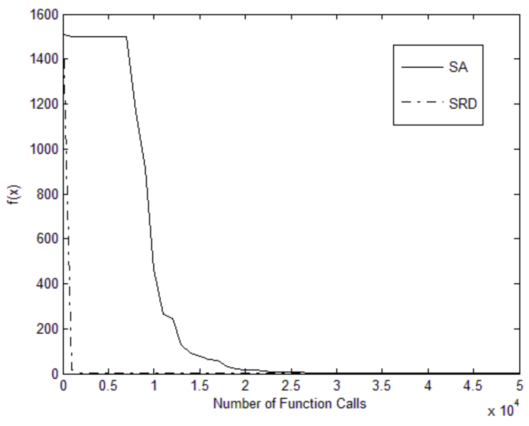


(b) f_6

Fig. 2. Sample graphs (best solution versus NFCs) for performance comparison between SA and SRD for D = 30.

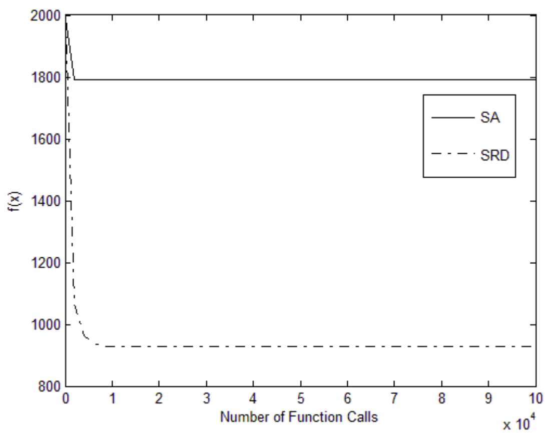


(a) f_5

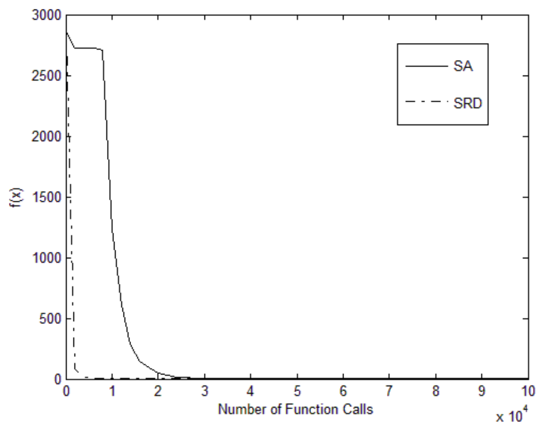


(b) f_6

Fig. 3. Sample graphs (best solution versus NFCs) for performance comparison between SA and SRD for $D = 50$.



(a) f_5



(b) f_6

Fig. 4. Sample graphs (best solution versus NFCs) for performance comparison between SA and SRD for $D = 100$.

TABLE VIII. COMPARISON OF SA AND SRD (D = 100). MEAN BEST AND STANDARD DEVIATION (STD DEV) OF 50 RUNS AND 100,000 FUNCTION CALLS ARE REPORTED. THE BEST ALGORITHM IS EMPHASIZED IN EMPHASIZED IN BOLDFACE. "*" INDICATES THE TWO-TAILED T-VALUE AT 0.05 LEVEL OF SIGNIFICANCE.

	SA	**SRD**
	Mean (Std Dev)	Mean (Std Dev)
f_1	0.6968165 (0.1178876)	**2.763E-08** (6.453E-09)*
f_2	35.375615 (33.543331)	**0.1051867** (0.1060622)*
f_3	6924.7173 (1265.7867)	**25.772213** (7.279176)*
f_4	104.29411 (19.064743)	**102.62772** (18.440902)
f_5	2144.934 (289.38708)	**875.06103** (69.284618)*
f_6	0.6414077 (0.0430748)	**0.0046485** (0.0071604)*
f_7	2.346E-07 (7.023E-08)	**9.558E-09** (1.591E-09)*
f_8	20.05514 (0.0120609)	**19.510647** (0.1054413)*

B. *Experimental Series 2: High Dimension (D=100)*

The results of higher dimension ($D = 100$) are shown in Table VIII. Similar to lower dimesion results the proposed method outperformed SA in terms of the quality of the result and covergence speed. However, similar to the previous experiments, the t-test did not pass for the test problem f_5. Figures 2 to 4 show that as the problem dimension incresases, the gap between SA and SRD also increases (interms candidate solution accuracy). Moreover, as compared to SA, the proposed method show significant improvement in the consistency of the solution found except for problem f_8 (i.e., the standard deviation was significantly lower than that of SA).

VI. CONCLUSION

This paper proposed the simulated raindrop (SRD) algorithm, a single-solution based optimization metaheuristic method. SRD optimization is inspired by raindrops travel from a higher altitude to a lower point on the landscape due to gravity. The performance of SRD was compared against the well-known single-solution based metaheuristic, Simulated Annealing (SA). Eight benchmark problems have been used for comparison purposes with dimensionalities of 30, 50, and 100.

In all test problems the proposed method outperformed SA in terms of solution accuracy. The SRD was statistically better than SA on seven test problems out of eight. Almost in all test problems, regardless of their dimensionality, SRD had better convergence speed and robustness. Moreover, as we increased the problem dimension from $D = 30$ to $D = 50$, and $D = 100$, SRD better results compare to SA in terms of the consistency of the solution found.

For future work, we would like to extend SRD to a population based metaheuristic and compare the performance against other population-based methods, such as GA, DE and PSO.

REFERENCES

[1] S. Binitha & S. S. Sathya, "A Survey of Bio inspired Optimization Algorithms," International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2, Issue-2, 2012.

[2] C. R. Reeves, , & J. E. Rowe, . "Genetic algorithms-principles and perspectives: a guide to GA theory," (Vol. 20). Springer, 2002.

[3] E.Talbi, "Metaheuristics from design to implementation," John Wiley and Sons, ISBN: 978-0470-27858-1, 2009.

[4] J. H. Holland, "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence," MIT press, 1992.

[5] R. Storn and K. Price, "Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical report. International Computer Science Institute, Berkley, 1995.

[6] J. Kennedy and R. C. Eberhart,"Particle swarm optimization," In Proceedings of the 1995 IEEE International Conference on Neural Network, Vol. 4. pp. 1942-1948. IEEE Press, 1995.

[7] M. Dorigo, "Optimization, learning and natural algorithms," PhD thesis, Politecnico di Milano, Italy, 1992.

[8] H. Shah-Hosseini,, "Problem solving by intelligent water drops," Evolutionary Computation, 2007. CEC 2007. IEEE Congress on , vol., no., pp.3226-3231, 25-28 Sept. 2007.

[9] H. Shah-Hosseini, "Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem," Int. Journal of Intelligent Computing and Cybernetics, Vol. 1, No. 2, pp. 193-212, 2008a.

[10] H. Shah-Hosseini, "The Intelligent Water Drops algorithm: A nature-inspired swarm-based optimization algorithm," Int. J. Bio-Inspired Computation, Vol. 1, Nos.1/2, pp. 71–79, 2008b.

[11] H. Duan, , S. Liu & X. Lei, "Air robot path planning based on Intelligent Water Drops optimization," IEEE IJCNN 2008, pp. 1397 – 1401, 2008.

[12] H. Shah-Hosseini, "Optimization with the nature-inspired intelligent water drops algorithm," Evolutionary computation. Tech, Vienna, Austria, pp. 297-320, 2009.

[13] H. Shah-Hosseini, "An approach to continuous optimization by the Intelligent Water Drops algorithm," Procedia - Social and Behavioral Sciences, Volume 32, pp.224-229, 2012.

[14] S. Kirkpatrick, & M. P. Vecchi, "Optimization by simmulated annealing,". science, 220(4598), pp. 671-680, 1983.

[15] X. Yao, Y. Liu and G. Lin, "Evolutionary Programing Made Faster," IEEE Transacations on Evolutionary Computation, vol. 3, pp.82-102, July 1999.

[16] E. Ali, and S. Rahnamayan. "Center-point-based Simulated Annealing," Electrical & Computer Engineering (CCECE), 25th IEEE Canadian Conference on. IEEE, 2012.

[17] J. Brest, S. Greiner, B. Boskovic, M. Mernik, & V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems,". Evolutionary Computation, IEEE Transactions on, 10(6), pp. 646-657, 2006.