Research paper

Kidney-inspired algorithm for optimization problems

Najmeh Sadat Jaddi [a],[*], Jafar Alvankarian [b], Salwani Abdullah [a]

[a] Data Mining and Optimization Research Group, Centre for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia
[b] Institute of Microengineering and Nanoelectronics, Universiti Kebangsaan Malaysia, Bangi 43600, Malaysia

ARTICLE INFO

ABSTRACT

In this paper, a population-based algorithm inspired by the kidney process in the human body is proposed. In this algorithm the solutions are filtered in a rate that is calculated based on the mean of objective functions of all solutions in the current population of each iteration. The filtered solutions as the better solutions are moved to filtered blood and the rest are transferred to waste representing the worse solutions. This is a simulation of the glomerular filtration process in the kidney. The waste solutions are reconsidered in the iterations if after applying a defined movement operator they satisfy the filtration rate, otherwise it is expelled from the waste solutions, simulating the reabsorption and excretion functions of the kidney. In addition, a solution assigned as better solution is secreted if it is not better than the worst solutions simulating the secreting process of blood in the kidney. After placement of all the solutions in the population, the best of them is ranked, the waste and filtered blood are merged to become a new population and the filtration rate is updated. Filtration provides the required exploitation while generating a new solution and reabsorption gives the necessary exploration for the algorithm. The algorithm is assessed by applying it on eight well-known benchmark test functions and compares the results with other algorithms in the literature. The performance of the proposed algorithm is better on seven out of eight test functions when it is compared with the most recent researches in literature. The proposed kidney-inspired algorithm is able to find the global optimum with less function evaluations on six out of eight test functions. A statistical analysis further confirms the ability of this algorithm to produce good-quality results.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, the researchers are trying to imitate nature in technology due to the nature is the best trainer for technology and its designs and capabilities are enormous. In addition, these two fields have much stronger connection since a lot of new problems in computer science are categorized as nature problems. Therefore, easy mapping is possible between nature and technology in real world. Although, there are many nature-inspired algorithms in the literature in recent years, we still believe that there is room to improve this mapping between nature and computer science by both enhancement of the existing nature- inspired algorithms and introducing new ones. This can be as motivation of introducing kidney- inspired algorithm in this study.

* Corresponding author.
E-mail addresses: najmehjaddi@gmail.com (N.S. Jaddi), alvankarian@gmail.com (J. Alvankarian), salwani@ukm.edu.my (S. Abdullah).

Table 1
Examples of nature-inspired algorithms in the literature.

Nature-inspired algorithm	Imitation	Reference
Simulated annealing (SA)	simulates the steel annealing process	[2]
Great deluge (GD)	imitates the process whereby a person in a great deluge climbing a hill	[3]
Gravitational search algorithm (GSA)	Imitates the law of gravity and the idea of mass interactions	[4]
Genetic algorithm (GA)	Imitates natural selection operators and natural genetic variation	[5]
Ant colony optimization (ACO)	stimulates finding shortest path to the food sources of ants	[6]
Intelligent water drops (IWDs)	imitates destination finding behavior of natural rivers	[7]
Particle swarm optimization (PSO)	Imitates the swarm behavior	[8]
Bat algorithm (BA)	Simulates echolocation behavior of bats	[9]
Honey bee optimization (HBO)	Mimics the food-foraging behavior of honey bee colonies	[10]
Honey bee mating optimization (HBMO)	Imitates self-explanatory honey bee mating	[11]
Firefly algorithm (FA)	imitates the flashing light emitted by fireflies in the natural world	[12]
Cuckoo search (CS)	inspires reproduction behavior of the cuckoo	[13]
Harmony search (HS)	simulates finding the harmony in music	[14]
Black hole (BH)	Imitates the black hole phenomenon	[15]
Imperialist competitive (IC)	Simulates competition among empires	[16]

In computer science, a meta-heuristic is a higher level heuristic that may offer a suitably high-quality solution for a given optimization problem. Among all the meta-heuristic algorithms in the literature, nature-inspired algorithms have become particularly attractive over the past few decades for solving various optimization problems [1]. The nature-inspired algorithms are derived from the activities of physical or biological systems in the natural world. Some examples of nature-inspired algorithms in the literature are shown in Table 1.

Nature-inspired algorithms have found application in many fields such as data mining [17-26], timetabling and scheduling [27-29], pattern recognition [30-33], industry [34-36], engineering [37-39] and economics [40, 41]. The balance between exploration and exploitation has a considerable effect on the efficiency of a meta-heuristic [42]. This paper proposes a new optimization algorithm inspired by the kidney process in the human body exploration and exploitation capabilities.

In the kidney process, there are four steps in urine formation. The first step, filtration, involves the transfer of solutes and water from the blood to the tubules in the kidneys. The second step, reabsorption, is the movement of useful water and solutes from the tubules back into the blood. In the third step, secretion, the tubules continue to secrete extra and harmful substances into the tubular. The end result of the above three steps leaves the body via urine in the fourth step, excretion. These four steps of the kidney process are simulated in the proposed kidney-inspired algorithm (KA). The KA starts with an initial population of water and solutes particles (solutions). At each iteration, the solutes are filtered depending on a filtration rate that is calculated based on mean of objective functions (MOF) of all solutes. The filtered solutes are moved to filtered blood (FB) and the rest are transferred to waste (W). These steps imitate the glomerular filtration process in the kidneys. Reabsorption, secretion and excretion are the other three functions of a kidney that are considered and imitated in the KA. A solute allocated to W is reabsorbed if it can become part of FB after applying the reabsorption operator, otherwise it is excreted from W. In addition, a solution in FB is secreted if it is not better than the worst solution in FB. After treating all the solutions in the population, all the solutions are ranked, W and FB are merged to be the new population and the filtration rate is updated. In this algorithm, the generation of a new solution and reabsoption operator is calculated based on the quality of the current solution and the best solution found so far. In this algorithm, exploitation is achieved by the filtration process and exploration is provided by the new solution generation and reabsorption process.

The rest of the paper is structured as follows: an overview of the overall kidney process in the human body is provided in Section 2. The details of the proposed KA are specified and explained in Section 3. The four simulated components of the KA and its pseudocode are also given in this section. The experimental results of the assessment of the proposed algorithm based on eight standard test functions and a statistical analysis are presented in Section 4. The conclusion is provided in Section 5 in which potential extensions are also discussed.

2. The biological kidney process

The kidneys are the main biological structure of the urinary system in the human body. They generally filter blood in order to eliminate excess water and wastes through the urine. They also handle the amount of ions in the blood.

The kidney's primary functional component is called the nephron. In each kidney there are about one million nephrons. Each nephron has a filtering system called a glomerulus, and a tubule, through which the filtered fluid passes. Urine formation starts in the glomerular capillaries. In glomerular capillaries, the dissolved substances are passed into the tubule as an effect of the force of blood pressure and the pressure in Bowman's capsule. The tubule in the kidney is responsible for reabsorption and secretion. Reabsorption is the procedure of transferring solutes from the tubules and returning them to the bloodstream. Some substances such as sodium and glucose are reabsorbed when the plasma level reaches a specific threshold. The process of moving solutes toward the renal tubule so that they can be excreted in the urine is defined as secretion. Secretion permits substances such as hydrogen ions to be eliminated. The kidney function has effects on the general state of health, including blood pressure, blood chemistry and fluid balance. The kidney process can be summarized as follows:
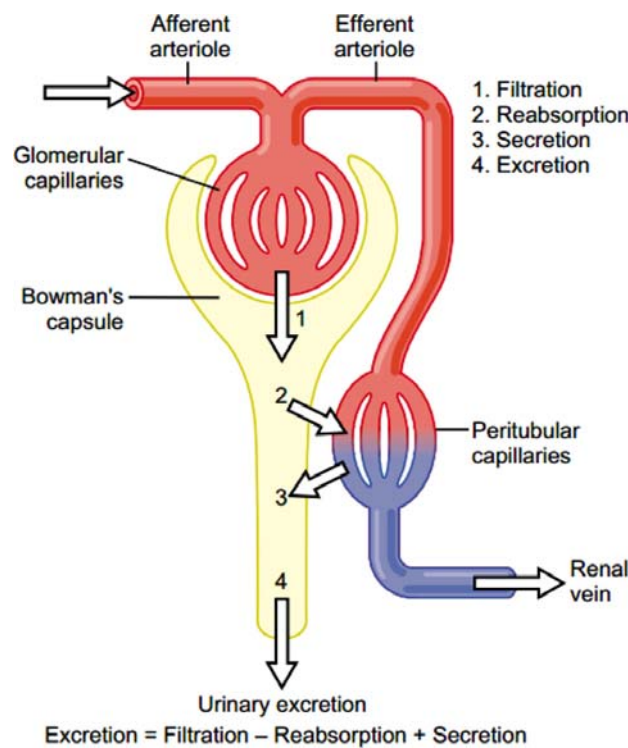
Fig. 1. Basic physiologic mechanism of the kidney [43].

1. Filtration: moving solutes and water from blood to the tubule.
2. Reabsorption: transporting useful water and solutes from the tubule back into the bloodstream.
3. Secretion: transferring extra and harmful substances from the blood into the tubular.
4. Excretion: passing the waste results of the earlier three procedures into the urine.

The basic physiologic mechanism of the kidney taken from [43] is shown in Fig. 1. As noted at the bottom of Fig. 1, the amount of excretion is equal to filtration minus reabsorption plus secretion. The systematic loop in the mechanism of the kidney results in the continual movement of water and solutes into the nephron and the transfer of useful substances into filtered blood that moves toward the bloodstream and heart. In this repeated process, the waste and harmful substances, which form the urine, are removed from the bloodstream.

3. Kidney-inspired algorithm

The proposed KA is a population-based method, so it has some characteristics in common with other population-based algorithms. Furthermore, as the name suggests, it imitates some the procedures in the kidney biological system. The four major components of the kidney process described above are used in this simulation.

In the initial stage of the KA, a random population of candidate solutions is created and the objective function is calculated for all of them. At each iteration, a new solution is generated for all candidate solutions by moving toward the best solution found so far. Then, by applying the filtration operator the candidate solutions with high quality in the population are filtered into FB and the rest are moved into W. The reabsorption, secretion and excretion mechanisms of the biological kidney process are simulated in this part of search process by checking some conditions embedded in the algorithm. If a candidate solution is assigned to W, the algorithm gives this solution another opportunity to improve itself so that it can then be moved into FB. If this opportunity is unsatisfied, the solution is excreted from W and a random solution is added to W. On the other hand, if after filtration, a candidate solution is assigned to FB and the quality of this solution is not better than the worst solution in FB this solution is secreted, however, if the solution is better than the worst, the worst solution is secreted from FB. Finally, the solutions in FB are ranked and the best solution is updated. The filtration rate is updated and FB and W are merged. This iterative process is continued until the termination criterion is met. The flowchart of the KA is provided in Fig. 2.

An analogy between the KA and the kidney biological system is given in Table 2. The details of the components involved in the KA are explained in following subsections.
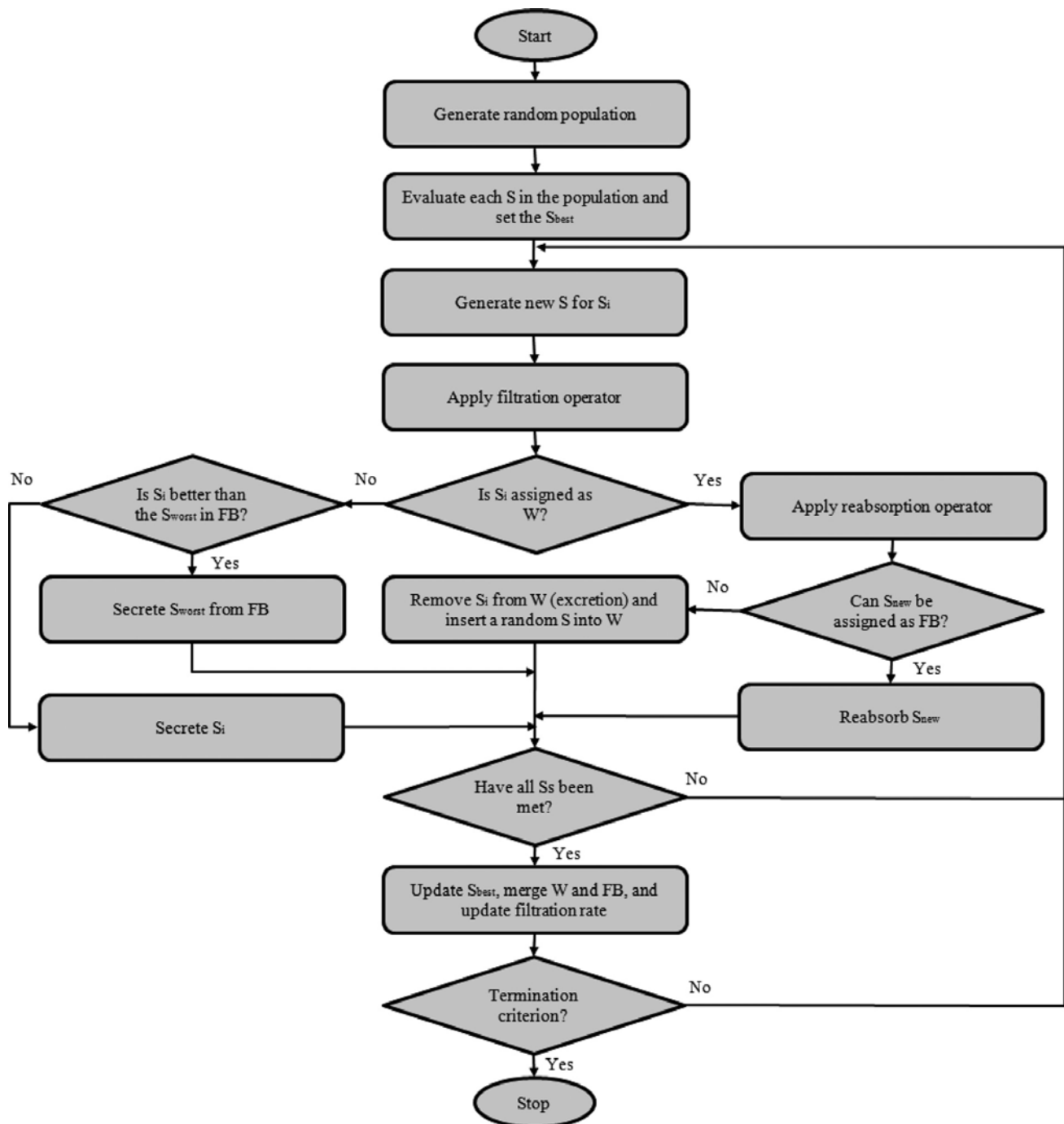
Fig. 2. Flowchart of KA.

Table 2
Analogy of kidney-inspired algorithm with kidney biological system.

Kidney biological system	Kidney-inspired algorithm
Particles of water and solutes in plasma	Solutions
Continuous flow of fluids into kidney	Generating new solution or inserting random solution
Glomerular filtration rate	Algorithm filtration rate
Filtering useful particles from harmful particles	Separating better solutions (FB) from worse solutions (W)
Reabsorbing useful particle	Sending useful solution from W to FB
Secreting harmful particle	Sending worst solution from FB to W
Excretion	Removing solution from W

3.1. Movement of virtual solutes

Each solution in the population of the KA represents a solute in the biological kidney. In the KA, a new solution is generated by moving the solution from the previous iteration toward the best solution found by the algorithm so far. In other words, the new solution is generated by trying to improve a current solution through the best solution. This movement is formulated as follows:

$$S_{i+1} = S_i + rand(S_{best} - S_i) \tag{1}$$

In Eq. 1, S represents a solution in the population of the KA (or a solute in the biological kidney). S_i is the solution in the ith iteration. The value of *rand* is a random number between zero and a given number and S_{best} is the best solution found by the algorithm in past iterations. This equation provides the algorithm with a good diversity of solutions which it means the solutions are varied based on current solution and the best solution. Besides, moving the solutions toward the best solution intensifies the local convergence ability of the algorithm.

3.2. Filtration

The solutions in the population are filtered by using a filtration rate that is calculated by a filtration function at each iteration. The filtration rate (*fr*) is calculated as follows:

$$fr = \alpha \times \frac{\sum\limits_{i=1}^{p} f(x_i)}{p} \tag{2}$$

The value of α is a constant value in range of (0, 1], which is adjusted in advance. P is the population size. $f(x_i)$ is the objective function of solution x at iteration i. It can be seen in this formula that the filtration rate for each iteration is dependent on the values of objective functions of all solutions in the population. This equation provides a ratio of MOF of all solutions in the population through the role of α. If α is equal to 0, fr is equal to 0, which means there is no filtration procedure for the algorithm. The value of α must be greater than 0 and equal to or less than 1. If this value is set to 1, the mean of all objective functions are used as fr. The filtration rate is varied to aid the convergence of the algorithm. At each iteration, the values of the objective function get closer to the global optimum, therefore calculating the filtration rate based on those solutions provides the algorithm with better solutions in FB. This process proves exploration.

3.3. Reabsorption

The reabsorption operator is a process that gives a solution that has been assigned to W a chance to become part of FB. A solution assigned to W can be moved to FB if, after applying the movement operator (Eq. 1) again, it satisfies the filtration rate and can be assigned to FB. This is a simulation of the reabsorption of solutes in the biological kidney. The reabsorption is useful for exploration.

3.4. Secretion

Secretion is an operator for the solutions that have been assigned to FB. If a solution that has been assigned to FB is not better than the worst solution in FB, it is secreted and is moved to W; otherwise this solution remains in FB and the worst solution in FB is secreted and is transferred to W.

3.5. Excretion

The solutions in W are excreted if, after giving them a chance of reabsorption, they cannot satisfy the filtration rate to become part of FB. These solutions are excreted if they do not have the ability to be FB after two times moving. In this case, such a solution in W is replaced with a random solution. The insertion of random solutions imitates the continuous insertion of solutes and water into the glomerular capillaries of the kidney.

In this algorithm, the filtration strategy and moving toward the best solution in new solution generation provides the algorithm with high exploitation or intensification. The filtration in this algorithm works by always focusing on the search space around the best solution. Conversely, the reabsorption operator offers good exploration or diversification to avoid local optima. The pseudocode and the schematic process of the KA are shown in Figs. 3 and 4, respectively.

As can be seen in Fig. 4, the solutes in the initial population (Fig. 4(a)) are separated into two, FB and W, by evaluation of the solutions and filtration rate measurement in the filtration stage (Fig. 4(b)). After that, each solute (depending on whether it is FB or W) and its value of objective functions uses the reabsorption, secretion or excretion operators (Fig. 4(c)). Within this process some of the solutes that are strong enough and have been assigned to FB in the filtration stage remain as members of FB. Some of them that are assigned to W and that after having the chance of reabsorption still cannot be assigned to FB are excreted, but some are successfully reabsorbed. Secretion is performed on the worst solute among all those that have been assigned to FB. After placement of each solute (Fig. 4(d)), the search is continued using the merged

set the population
evaluate the solute in the population
set the best solute, S_{best}
set filtration rate, fr, Eq. 2
set waste, W
set filtered blood, FB
set number of iterations, numofite
do while *(ite<numofite)*
          for *all s_i*
                    generate new S_i, Eq. 1
                    check the S_i using fr
                    if *S_i assigned to W*
                              apply reabsorption and generate S_{new}, Eq. 1
                              if *reabsorption is not satisfied (S_{new} cannot be a part of FB)*
                                        remove S_i from W (excretion)
                                        insert a random S into W to replace S_i

                              endif
                              *S_{new} is **r**eabsorbed*
                    else
                              if *it is better than the S_{worst} in FB*
                                        S_{worst} is secreted
                              else
                                        S_i is secreted
                              endif
                    endif
          endfor
          rank the Ss from FB and update the S_{best}
          merge W and FB
          update filtration rate, Eq. 2
end while
return S_{best}

Fig. 3. Pseudocode of KA.

FB and W as a new population (Fig. 4(e)). The solutes in W are included in the new population with hope of achieving an improvement in the next iteration.

4. Experimental results

In our experiments, the solution representation can be either numbers or vectors depend on the problem, where we used numbers. The KA was implemented in Java. The KA was tested on eight diverse and well-known test functions. The number of iterations for the algorithm was set to 100 and the population size was also set to 100. If the number of iterations and the population size is higher the speed of the algorithm is lower and it may lead the algorithm toward a better results. The details of the global optimum and dimension of these test functions are shown in Table 3. The value of d provided in parenthesis is a constant value that was set in the formula of the test function.

Due to space constraints, two examples are discussed here in; Figs. 5 and 6 show the performance of the KA in solving the Eggcrate and Easom test functions, respectively. The 3D plot of the functions and snapshots are provided. The 3D plot of the Eggcrate function in Fig. 5(a) provides an overview of the search space in this problem. The initial location of each solution in the initial population is shown in Fig. 5(b), while Fig. 5(c) shows the location of the solutions when the algorithm meets the global optimum at (0, 0) and $f(x) = 0$.

As can be seen from Fig. 5, the solutions in the initial population are located in the whole area of the search space. After several iterations (around 65) the solutions move towards the global optimum which is at (0, 0). The successful convergence in the algorithm is due to the exploration and exploitation procedure. The generation of a new solution together with reabsorption provides exploration while the filtration mechanism facilitates exploitation. The trade-off between exploration and exploitation results in a good quality of convergence. In a similar manner, the Easom function presented in Fig. 6 shows the successful convergence of algorithm. It is clear from Fig. 6 that, although the global minimum has a small region relative to the search space in the Easom test function (Fig. 6(a)), by starting with a random initial population (Fig. 6(b)) and using a good balance of exploration and exploitation, the algorithm is able to reach to the global optimum in $x = (\pi, \pi)$ and $f(x) = -1$ (Fig. 6(c)).

The results of the eight test functions are presented in Table 4. The program was run 100 times and the average of 100 results and the best result for each test function are reported in this table. Table 4 shows that in all the test functions, with the exception of the Michalewicz and Schwefel functions, the algorithm meets the global best in all 100 runs. This
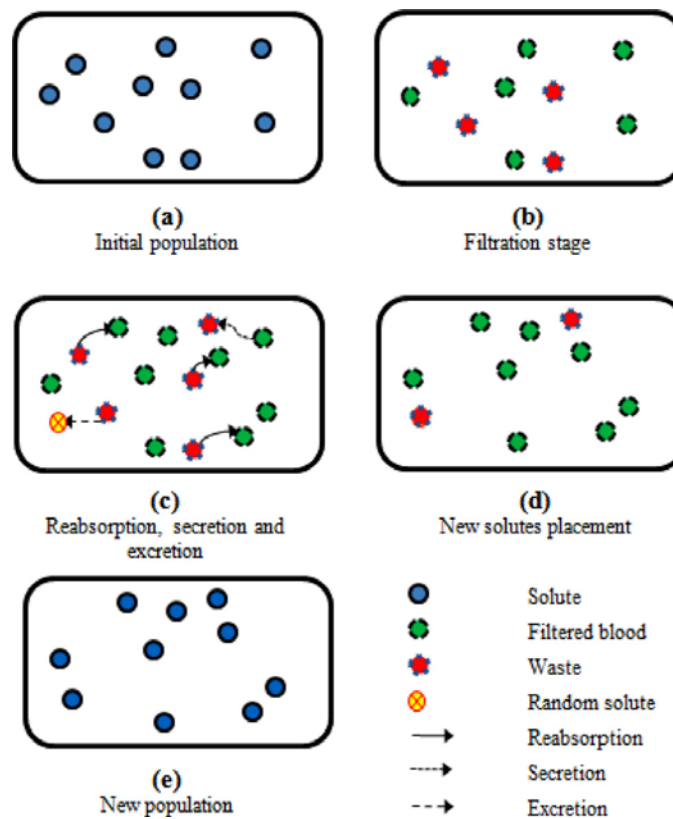
Fig. 4. Schematic of KA process: (a) initial population, (b) filtration stage, (c) reabsorption, secretion and excretion, (d) new solutes placement, (e) new population.

Table 3
Details of used test functions.

Test function	Equation of test function	Global optimum	Dimension
Michalewicz ($d=2$)	$f(x) = -\sum\limits_{i=1}^{d} \sin(x_i)\sin^{2m}(\frac{ix_i^2}{\pi})$	$f(x) = -1.8013\ x = (2.20,\ 1.57)$	$0 < x < \pi$
Rosenbrock ($d=16$)	$f(x) = \sum\limits_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$f(x) = 0\ x = (1,\ 1)$	$-2.048 < x < 2.048$
De Jong ($d=256$)	$f(x) = \sum\limits_{i=1}^{d} x_i^2$	$f(x) = 0\ x = (0,\ 0)$	$-5.12 < x < 5.12$
Schwefel ($d=128$)	$f(x) = 418.9829d - \sum\limits_{i=1}^{d} x_i \sin(\sqrt{|x_i|})$	$f(x) = 0\ x = (420.9687,\ 420.9687)$	$-500 < x < 500$
Ackley ($d=128$)	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{d}\sum\limits_{i=1}^{d} x_i^2}) - \exp(\frac{1}{d}\sum\limits_{i=1}^{d}\cos(2\pi x_i)) + 20 + e$	$f(x) = 0\ x = (0,\ 0)$	$-32.768 < x < 32.768$
Rastrigin	$f(x) = 10d + \sum\limits_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]$	$f(x) = 0\ x = (0,\ 0)$	$-5.12 < x < 5.12$
Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	$f(x) = -1\ x = (\pi,\ \pi)$	$-100 < x < 100$
Griewangk	$f(x) = \sum\limits_{i=1}^{d} \frac{x_i^2}{4000} - \prod\limits_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$f(x) = 0\ x = (0,\ 0)$	$-600 < x < 600$
Eggcrate	$f(x) = x_1^2 + x_2^2 + 25 \times [\sin^2(x_1) + \sin^2(x_2)]$	$f(x) = 0\ x = (0,\ 0)$	$-5 < x < 5$

Table 4.
Results of test functions and comparison against state-of-the-art algorithms.

| Test function	Optimum	PSOLF[44]		ELPSO[45]		ALO[46]		ODFA[47]		KA	
		Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Michalewicz	−1.8013	–	–	–	–	–	–	**−1.8013**	–	−1.8142	3.2E-04
Rosenbrock	0	3.55	2.41	0.0952	0.1988	0.3467	0.1095	0	–	**0**	**0**
De Jong	0	–	–	–	–	2.5E-10	1.6E-10	4.6E-11	–	**0**	**0**
Schwefel	0	7.0E+03	7.1E+02	–	–	–	–	2.5E-5	–	**2.0E-5**	**2.3E-6**
Ackley	0	1.3E+00	9.3E-01	1.6E-7	2.5E−8	3.7E-15	1.5E-15	5.7E-5	–	**0**	**0**
Rastrigin	0	4.0E+01	1.1E+01	0.4261	0.1931	–	–	6.9E-9	–	**0**	**0**
Easom	−1	–	–	–	–	–	–	−1	–	**−1**	**0**
Griewangk	0	1.8E-02	2.6E-02	0.0114	3.9E-3	0.0186	9.5E-3	4.1E-11	–	**0**	**0**
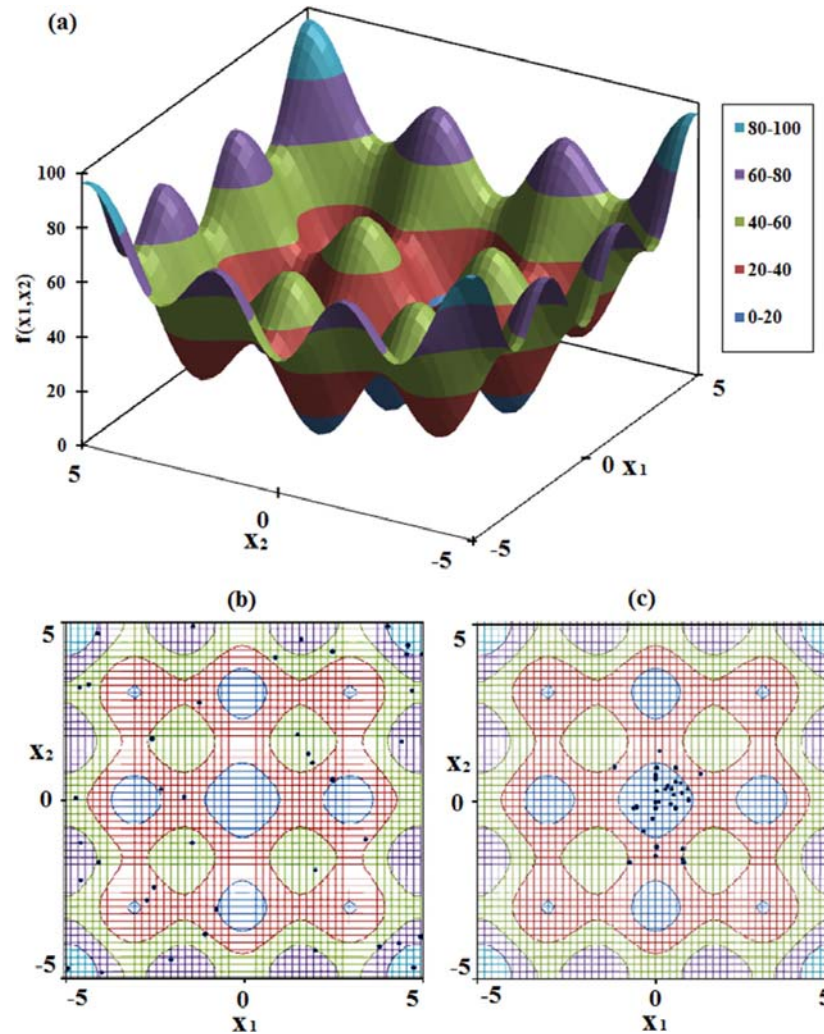
Fig. 5. Minimization of Eggcrate function: (a) search space, (b) initial population, (c) location of solutions in the global best and its surroundings.

shows the robustness of the algorithm in finding the global optimum. Even in the case of the two excepted functions, the algorithm meets the global optimum in 99% of runs. The algorithm could not achieve the global optimum in just one run. We believe this is because we have run the experiment for only 100 times, if the algorithm is run for more than 100 times it can achieve the global optimum for both Michalewicz and Schwefel functions. In this table, the results are compared against most recent algorithms in the literature. The algorithms in comparison are PSOLF [44], ELPSO [45], ALO [46] and ODFA [47]. The best results are shown in bold. KA algorithm outperforms state-of-the-art algorithms on seven out of eight test functions that shows the better ability of KA compared to others.

In order to compare the KA speed with state-of-the-art, the number of function evaluations when the algorithm reaches the global optimum was calculated for 100 runs. The average of number of function evaluations of 100 results for each test function is reported in Table 5. In this table, the results of the KA are compared with the available results of genetic algorithm (GA), particle swarm optimization (PSO), and bat algorithm (BA) that are previously reported in [9], firefly algorithm (FA) that is taken from [12], and the cuckoo search (CS) algorithm (taken from [13]). In Table 5, the best result is shown in bold.

In order to have fair comparison, the other nature-inspired algorithms from oldest (GA) to newest (CS) from literature are chosen. The same benchmark test functions compared in these studies in the literature are selected due to having fair comparison. Furthermore, the number of function evaluations as measurement of the comparison provided in Table 5 is another reason for having fair comparison. As comparing the running time in different machines and systems may lead the researchers to have unfair comparison, the number of function evaluations is commonly used. Since the function evaluation is the time consuming part for the optimization algorithms, if we have more number of function evaluations when the algorithm reaches to optimal solution so we have more running time and vice versa.

In the KA, the function evaluation is performed for each solution in the population to generate a new solution in each iteration. Furthermore, the function evaluation is performed if reabsorption is carried out. In the KA process there are two
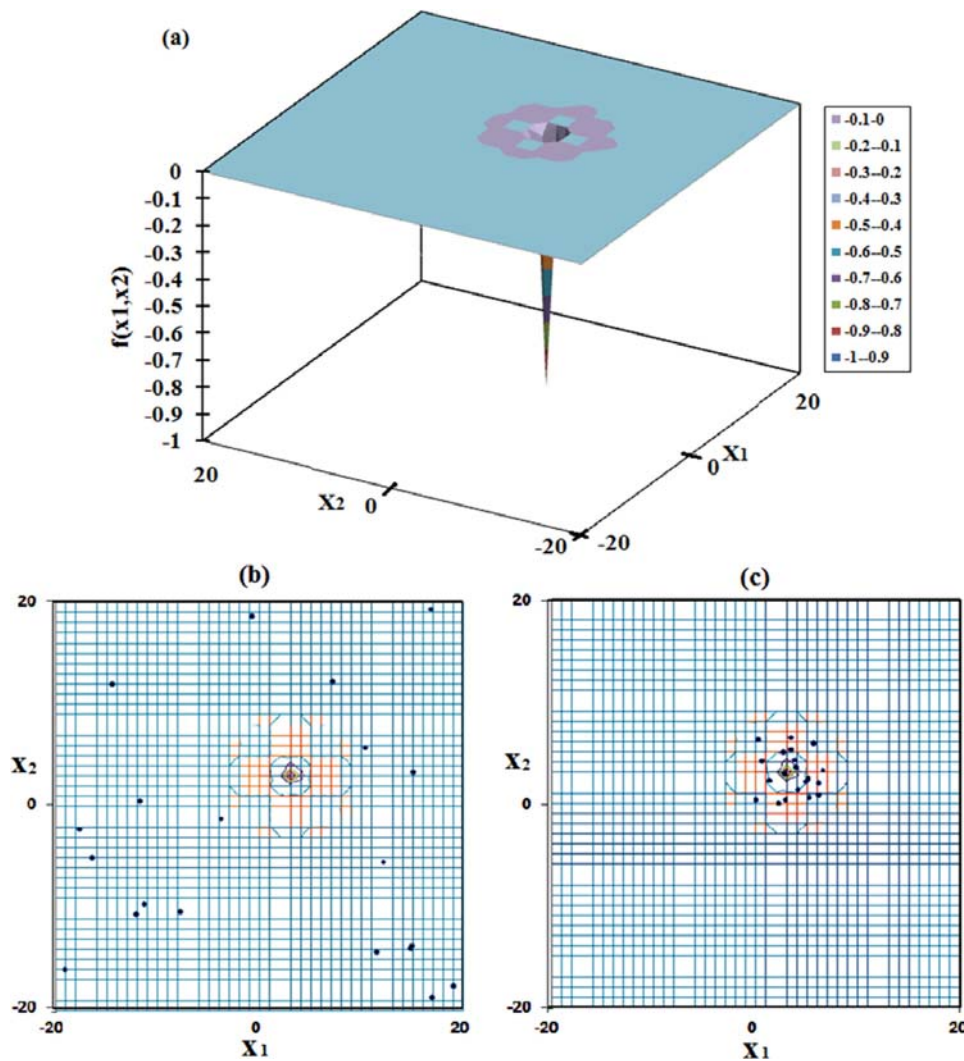
Fig. 6. Minimization of Easom function: (a) search space, (b) initial population, (c) location of solutions in the global best and its surroundings.

Table 5.
Comparison of number of function evaluations for KA and other algorithms.

Test function	GA		PSO		BA		FA		CS		KA	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Michalewicz	89,325	7914	6922	537	4752	753	3752	725	**3221**	519	3301	428
Rosenbrock	55,723	8901	32,756	5325	7923	3293	7792	2923	5923	1937	**5663**	1283
De Jong	25,412	1237	17,040	1123	5273	490	7217	730	**4971**	754	5829	542
Schwefel	227,329	7572	14,522	1275	8929	729	9902	592	8829	625	**8810**	623
Ackley	32,720	3327	23,407	4325	6933	2317	5293	4920	4936	903	**4736**	769
Rastrigin	110,523	5199	79,491	3715	12,573	3372	15,573	4399	10,354	3755	**9785**	3211
Easom	19,239	3307	17,273	2929	7532	1702	7925	1799	6751	1902	**5673**	892
Griewangk	70,925	7652	55,970	4223	9792	4732	12,592	3715	10,912	4050	**8239**	3846

function evaluations, so it would seem that the number of function evaluations for the KA should be more than that of other algorithms such as the GA, PSO and BA that evaluate the function only once when they generate a new solution. However, the results of the KA show superior performance compared to the other algorithms. This is because of fewer iterations are required to find the optimal solution. This is achieved by focusing on the best area of the search space with the aid of the filtration process in the KA. Filtration provides the algorithm with good exploitation and fast convergence. On the other hand, reabsorption gives the algorithm a good diversity of solutions and thus superior exploration. The balance between exploration and exploitation in this algorithm results in enhanced convergence. The standard deviation (STD) of the KA is also compared with that of the other algorithms in Table 5. The STD of the KA is lower and proves the robustness of this algorithm.

Table 6
Comparison of average ranks of KA and other algorithms.

Algorithm	Rank
KA	1.375
CS	1.875
BA	3.125
FA	3.625
PSO	5
GA	6

Table 7
Nemenyi test for number of function evaluations of KA and other algorithms.

	Mean	GA 78,899.5	PSO 30,922.63	BA 7963.375	FA 8755.75	CS 6987.125	KA 6504.5
GA	78,899.5	–	47,976.88	70,936.13	70,143.75	71,912.38	72,395
PSO	30,922.63	–	–	22,959.25	22,166.88	23,935.5	24,418.13
BA	7963.375	–	–	–	792.375	976.25	1458.875
FA	8755.75	–	–	–	–	1768.625	2251.25
CS	6987.125	–	–	–	–	–	482.625
KA	6504.5	–	–	–	–	–	–

Moreover, an average ranking test for the different algorithms with respect to the mean of function evaluation was performed to discover the first ranked algorithm. The ranks were computed by employing the RANK function in Microsoft Excel and the results are shown in Table 6. From this table it can be observed that the KA is ranked as first algorithm in terms of the number of function evaluations.

The optimization progress of the GA, PSO, BA, FA, CS and KA for all eight test functions is shown in Fig. 7. In order to generate the graphs in this figure the algorithms were re-implemented to be compared with KA. In contrast, the fast convergence of the KA is clearly evident in most of the test functions in this figure. The fast convergence is the result of the good exploitation provided by the filtration stage of the KA.

In order to further improve the comprehensiveness of the assessment, the Friedman test and Nemenyi test were performed to find out whether there are significant differences between the number of function evaluations of the proposed KA and of other algorithms. For this, the p-value computed by the Friedman test for the algorithms was 36.14286, which is greater than 10.71 (critical value) for the mean of functions evaluations. This assessment showed that there is a significant difference between the number of function evaluations of the KA and of the other algorithms. A Nemenyi test was carried out as a post-hoc test to find the group of algorithms that differ from the others. After calculating the standard error and its posterior computing the minimum significant difference (MSD), there is a need to observe where any differences in means went further than the MSD. The MSD for our assessment was equal to 8.845921 and the result showed that there is a significant difference between all the algorithms for the number of function evaluations which exceed the MSD. Table 7 shows the details of the Nemenyi test for the number of function evaluations.

5. Conclusion

In this paper, we proposed a new kidney-inspired algorithm for the optimization problem. This algorithm imitated four components in the biological structure of the urinary system of kidneys, namely filtration, reabsorption, secretion, and excretion. In this simulation, the solutions in the population are divided into filtered blood and waste based on their objective functions. The reabsorption and secretion steps are applied for some conditions in this algorithm. Reabsorption gives another chance to the solutions that have been assigned to waste to become part of filtered blood. The solutions that have already had this chance but cannot be assigned to filtered blood are excreted. The implementation of the algorithm and its results for eight standard test functions showed that the algorithm is effective when compared with other meta-heuristic algorithms available in the literature. The balance between exploration, which is provided by new solution generation and reabsorption, and exploitation, which is offered by filtration that focuses on the better area of the search space is the main advantage of this algorithm. This advantage enables the algorithm to achieve better convergence. The performance of the proposed algorithm is better on seven out of eight test functions when compared with the most recent researches in literature. The proposed kidney-inspired algorithm is able to find the global optimum with less function evaluations compared to other algorithms on six out of eight test functions. The statistical analysis proved the ability of the algorithm as it had fewer function evaluations compared to other algorithms.

These promising results suggest that there is potential to extend this study to undertake a wider comparison with a greater range of algorithms using test functions in higher dimensions. Besides, since we investigated the continuous op-
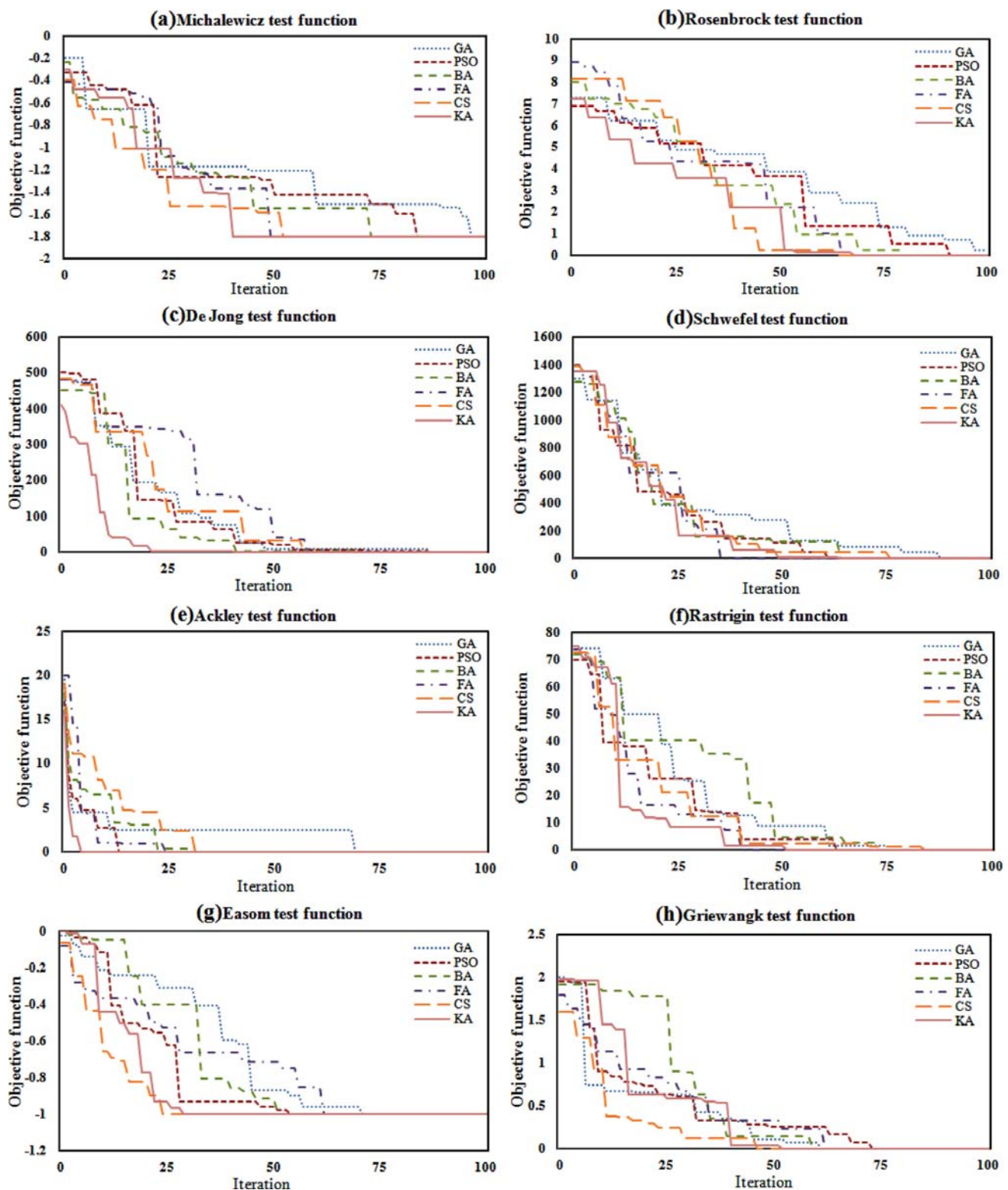
Fig. 7. Optimization progress in: (a) Michalewicz, (b) Rosenbrock, (c) De Jong, (d) Schwefel, (e) Ackley, (f) Rastrigin, (g) Easom and (h) Griewangk test functions.

timization problem, another interesting extension of this study would be to apply the kidney-inspired algorithm to the discrete optimization problem. In addition, further studies could be performed to apply the algorithm to some real-world problems such as data mining, image processing, timetabling and scheduling, as well as engineering problems to assess its performance in solving such problems.

Acknowledgments

References

[1] Dressler F, Akan OB. A survey on bio-inspired networking. Comput Netw 2010;54:881–900.
[2] Johnson DS, Aragon CR, McGeoch LA, Schevon C. Optimization by simulated annealing: an experimental evaluation. Part I, graph partitioning. Oper Res 1989;37:865–92.
[3] Dueck G. New optimization heuristics: the great deluge algorithm and the record-to-record travel. J Comput Phys 1993;104:86–92.
[4] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179:2232–48.
[5] Holland JH. Adaptation in natural and artificial systems. MIT Press; 1992.
[6] Bianchi L, Gambardella LM, Dorigo M. An ant colony optimization approach to the probabilistic traveling salesman problem. In: Proceedings of the 7th international conference on parallel problem solving from nature. New York: Springer-Verlag; 2002. p. 883–92.
[7] Shah-Hosseini H. Problem solving by intelligent water drops. CEC 2007 IEEE Congress Evol Comput 2007;3226–31.
[8] Kennedy J, Eberhart R. Particle swarm optimization. In: Neural networks, 1995 proceedings, IEEE international conference on, vol.4; 1995. p. 1942–8.
[9] Yang X-S. A new metaheuristic bat-inspired algorithm. In: González J, Pelta D, Cruz C, Terrazas G, Krasnogor N, editors. Nature inspired cooperative strategies for optimization (NICSO 2010). Springer Berlin Heidelberg; 2010. p. 65–74.
[10] Pham DT, Ghanbarzadeh A, Koç E, Otri S, Rahim S, Zaidi M. The bees algorithm. Cardiff University, UK, Technical Note, Manufacturing Engineering Centre, Cardiff; 2005.
[11] Fathian M, Amiri B, Maroosi A. Application of honey-bee mating optimization algorithm on clustering. Appl Math Comput 2007;190:1502–13.
[12] Yang X-S. Firefly algorithm, lévy flights and global optimization. In: Bramer M, Ellis R, Petridis M, editors. Research and development in intelligent systems. Springer London; 2010. p. 209–18.
[13] Xin-She Y, Deb S. Cuckoo search via lévy flights. Nat Biol Inspired Comput 2009 NaBIC 2009 World Cong. Coimbatore in India; 2009. p. 210–14.
[14] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. SIMULATION 2001;76(2):60–8.
[15] Hatamlou A. Black hole: a new heuristic optimization approach for data clustering. Inf Sci 2013;222:175–84.
[16] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. Evol Comput 2007 CEC 2007 IEEE Cong 2007:4661–7.
[17] Jaddi NS, Abdullah S, Hamdan AR. Multi-population cooperative bat algorithm-based optimization of artificial neural network model. Inf Sci 2015;294:628–44.
[18] Jaddi NS, Abdullah S, Hamdan AR. Taguchi-based parameter designing of genetic algorithm for artificial neural network training. Inf Creat Multim (ICICM), 2013 Int Conf IEEE 2013:278–81.
[19] Jaddi NS, Abdullah S. Nonlinear great deluge algorithm for rough set attribute reduction. J Inf Sci Eng 2013;29(1):49–62.
[20] Jaddi NS, Abdullah S. Hybrid of genetic algorithm and great deluge algorithm for rough set attribute reduction. Turk J Electric Eng Comput Sci 2013;21:1737–50.
[21] Shreem SS, Abdullah S, Nazri MZA. Hybrid feature selection algorithm using symmetrical uncertainty and a harmony search algorithm. Int J Syst Sci 2014:1–18.
[22] Mafarja M, Abdullah S. A fuzzy record-to-record travel algorithm for solving rough set attribute reduction. Int J Syst Sci 2013;46(3):503–12.
[23] Abdullah S, Sabar NR, Ahmad Nazri MZ, Ayob M. An Exponential Monte-Carlo algorithm for feature selection problems. Comput Ind Eng 2014;67:160–7.
[24] Jaddi NS, Abdullah S, Hamdan AR. A solution representation of genetic algorithm for neural network weights and structure. Inf Process Lett 2016;116:22–5.
[25] Jaddi NS, Abdullah S, Hamdan AR. Optimization of neural network model using modified bat-inspired algorithm. Appl Soft Comput 2015;37:71–86.
[26] Majdi M, Abdullah S, Jaddi NS. Fuzzy population-based meta-heuristic approaches for attribute reduction in rough set theory, world academy of science, engineering and technology. Int J Comput Electric Autom Control Inf Eng 2015;9:2065–73.
[27] Alzaqebah M, Abdullah S. Hybrid bee colony optimization for examination timetabling problems. Comput Oper Res 2015;54:142–54.
[28] Abdullah S, Abdolrazzagh-Nezhad M. Fuzzy job-shop scheduling problems: a review. Inf Sci 2014;278:380–407.
[29] Alzaqebah M, Abdullah S. An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. J Sched 2014;17:249–62.
[30] Connolly J-F, Granger E, Sabourin R. An adaptive classification system for video-based face recognition. Inf Sci 2012;192:50–70.
[31] Cuevas E, Oliva D, Zaldivar D, Pérez M, Rojas R. Circle detection algorithm based on electromagnetism-like optimization. In: Zelinka I, Snášel V, Abraham A, editors. Handbook of optimization. Springer Berlin Heidelberg; 2013. p. 907–34.
[32] Cuevas E, Oliva D, Diaz M, Zaldivar D, perez-Cisneros M, Pajares G. White blood cell segmentation by circle detection using electromagnetism-like optimization. Comput Math Methods Med 2013;2013:1–15.
[33] Picard D, Revel A, Cord M. An application of swarm intelligence to distributed image retrieval. Inf Sci 2012;192:71–81.
[34] Fox B, Xiang W, Lee H. Industrial applications of the ant colony optimization algorithm. Int J Adv Manuf Technol 2007;31:805–14.
[35] Patil B, Nataraj PSV. An improved bernstein global optimization algorithm for minlp problems with application in process industry. Math Comput Sci 2014;8:357–77.
[36] Besiashvili G, Khachidze M, Chokhonelidze D. Application of adaptive genetic algorithm in mining industry. In: Problems of cybernetics and informatics (PCI), 2012 IV international conference; 2012. p. 1–3.
[37] Yang X-S. 1 - Optimization and metaheuristic algorithms in engineering. In: Alavi X-SYHGTH, editor. Metaheuristics in water, geotechnical and transport engineering. Oxford: Elsevier; 2013. p. 1–23.
[38] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 2005;194:3902–33.
[39] Alvankarian J, Jaddi NS, Vakilian M, Barantalab F, Jamal R, Majlis BY. Consideration of nonuniformity in elongation of microstructures in a mechanically tunable microfluidic device for size-based isolation of microparticles. J. Microelectromech. Syst. 2015;24(2):309–18.
[40] Liang Y-C, Cuevas Juarez JR. A normalization method for solving the combined economic and emission dispatch problem with meta-heuristic algorithms. Int J Electric Power Energy Syst 2014;54:163–86.
[41] Chamba MS, Ano O. Economic dispatch of energy and reserve in competitive markets using meta-heuristic algorithms, Latin America transactions. IEEE (Revista IEEE Am Latina) 2013;11:473–8.
[42] Yang X-S. Nature-inspired metaheuristic algorithms. Luniver Press; 2010.
[43] Hall JE. Guyton and hall textbook of medical physiology: enhanced e-book. Elsevier: Elsevier Health Sciences; 2010.
[44] Jensi R, Jiji GW. An enhanced particle swarm optimization with levy flight for global optimization. Appl Soft Comput 2016;43:248–61.
[45] Jordehi AR. Enhanced leader PSO (ELPSO): a new PSO variant for solving global optimisation problems. Appl Soft Comput 2015;26:401–17.
[46] Mirjalili S. The ant lion optimizer. Adv Eng Softw 2015;83:80–98.
[47] Verma OP, Aggarwal D, Patodi T. Opposition and dimensional based modified firefly algorithm. Expert Syst Appl 2016;44:168–76.