ISSN: 0067-2904

An Improved Meerkat Clan Algorithm for Solving 0-1 Knapsack Problem

Samer Alaa Hussein*, Ahmed Yacoub Yousif
Information Technology Center, University of Technology, Baghdad, Iraq

Received: 5/12/2020 Accepted: 13/4/2021

Abstract

Meerkat Clan Algorithm (MCA) is a nature-based metaheuristic algorithm which imitates the intelligent behavior of the meerkat animal. This paper presents an improvement on the MCA based on a chaotic map and crossover strategy (MCA-CC). These two strategies increase the diversification and intensification of the proposed algorithm and boost the searching ability to find more quality solutions. The 0-1 knapsack problem was solved by the basic MCA and the improved version of this algorithm (MCA-CC). The performance of these algorithms was tested on low and high dimensional problems. The experimental results demonstrate that the proposed algorithm had overcome the basic algorithm in terms of solution quality, speed and gained optimality with low dimensional problems. Furthermore, in high dimensional problems, it has competitive results in comparison with the other algorithms.

Keywords: Optimization; Metaheuristic; Meerkat Clan Algorithm; Knapsack Problem.

خوارزمية عشيرة السرقاط المحسنة لحل مشكلة حقيبة الظهر 0-1

سامر علاء حسين*, أحمد يعقوب يوسف
مركز تكنولوجيا المعلومات, الجامعة التكنولوجية, بغداد, العراق

الخلاصة

خوارزمية عشيرة السرقاط (MCA) هي خوارزمية (metaheuristic) قائمة على الطبيعة والتي تحاكي السلوك الذكي لحيوان السرقاط. تقدم هذه الورقة تحسينًا على خوارزمية عشيرة ميركات بناءً على خريطة فوضوية واستراتيجية التقاطع (MCA–CC). تعمل هاتان الاستراتيجيتان على زيادة تنويع وتكثيف الخوارزمية المقترحة وتعزيز القدرة على البحث لإيجاد المزيد من الحلول عالية الجودة. تم حل مشكلة حقيبة الظهر 0–1 بواسطة MCA الأساسي والنسخة المحسنة من هذه الخوارزمية المحسّنة MCA–CC. تم اختبار أداء هذه الخوارزميات على مسائل ذات أبعاد منخفضة وعالية. أظهرت النتائج التجريبية أن الخوارزمية المقترحة تغلبت على الخوارزمية الأساسية من حيث جودة الحل وسرعته واكتسبت الأمثلية مع مشاكل الأبعاد المنخفضة. علاوة على ذلك ، في مشاكل الأبعاد العالية ، لها نتائج تنافسية مقارنة بالخوارزميات الأخرى.

1. Introduction

Several problems springing up in applications are NP-hard, which means getting optimality in polynomial time is troublesome. Therefore, to practically solve this kind of problem, metaheuristic algorithms are utilized for getting approximate solutions in an acceptable time

*Email: samer.a.hussein@uotechnology.edu.iq

[1]. Some examples of the most widespread metaheuristic algorithms are Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO). Metaheuristic algorithms have proven their superiority in solving various optimization problems. However, these algorithms may be trapped into local optima due to their stochastic behavior. Consequently, many improvements for metaheuristic algorithms were applied to overcome their limitations [2].

Recently, in numerous studies, chaos and metaheuristics were blended to exhibit the chaotic behaviors within the metaheuristic algorithms [3]. Chaos, which is defined as the simulation of the dynamic behavior of nonlinear systems, was utilized to overcome the barriers of metaheuristics and boost its performance [2].

The knapsack problem was classified as an NP-hard optimization problem. It has a variety of applications such as production planning problems, project selection, cutting stock problems, and resource allocation. 0-1 knapsack problem (0-1 KP) is one of the knapsack problems which consists of many items with its profits and weight to be involved in a knapsack. It aims to maximize the selected profit of the items without exceeding the knapsack capability [4]. Meerkat Clan Algorithm (MCA) is a nature-stimulated metaheuristic algorithm that was recently developed by Al-Obaidi, Abdullah, and Ahmed [5] for optimization problems. This novel algorithm mimics the intelligent behavior of the Meerkat (Suricata suricatta) animal in desert traversal to seek nourishment. This paper presents an improved version of the MCA to solve the knapsack problem.

The rest of the paper is organized as follows: Section 2 presents some related work. Section 3 offers a brief description of basic MCA. In Section 4, the proposed MCA-CC is presented. The application of the 0-1 KP is described in Section 5. The experimental results are detailed and compared with other optimization techniques in Section 6. In the last section, the conclusions of the paper are drawn.

2. Related Works

The problem of 0–1 KP has been considered by several researchers on different topologies to evaluate its performance and validate its effectiveness in solving this type of problem. Below are some of these recent studies. In [6], parallel simulated annealing on CPU with GPU by using the CUDA platform for handling 0–1 KP was introduced. The parallel method is faster than the single ones and has the capability of delivering a good quality solution in both low and medium dimensional problems. The experiments demonstrated the parallel method has superiority over the sequential one.

In [7], a hybridized PSO and greedy strategy that depended on the penalty factor was presented for tackling 0-1 KPs. The results obtained by this approach were competitive in comparison with other algorithms on large scale instances. In [8], a hybrid approach called cuckoo search algorithm with a global harmony search was suggested to solve 0-1 knapsack problems. This hybridization method overcomes the shortcomings of the cuckoo search algorithm by adapting the mutation operation to prop the search ability and population diversity. The efficiency and the effectiveness of the method have been demonstrated in tackling the low and high dimensional problem and it has the distinction compared with different optimization algorithms. In [9], a binary monkey algorithm was presented to solve the 0-1 knapsack problems. The local search ability of the proposed algorithm was reinforced by using the greedy strategy. The modified somersault process and adapted cooperation process were introduced to escape from the local optimum and speed up the convergence rate. The proposed algorithm showed efficiency in solving various data instances of the problem.

In [10], the complex-valued encoding and the greedy algorithm with the wind-driven optimization were considered for enhancing its performance for solving 0-1 KPs in terms of exploration and the exploitation. The computational results show that the proposed algorithm is convenient for the different scales of the problem's instances. In [11], an improved binary

chicken swarm optimization algorithm was proposed for solving the 0-1 knapsack problem. Greed and mutated strategies were applied to promote the quality and feasibility and increase the possibility of preventing the local minima. The comparison results proved that the proposed algorithm has sublimity in terms of quality of solution, stability, and convergence rate.

In [12], the monarch butterfly optimization with Opposition-Based Learning (OBL) strategy and Gaussian perturbation was presented. The Gaussian perturbation was conducted on a portion of the individuals with deficient fitness in each generation to take out from the local optimum. Besides, the OBL method was performed on half of the population to speed up the convergence rate. The experimental results demonstrated the effectiveness of the proposed method in solving large-scale 0-1 KP problems. In [13], ten approaches based on the binary moth search algorithm were proposed for tackling a discounted 0-1 knapsack problem. In addition to the basic moth search algorithm, nine algorithms with diverse mutation operators based on the global harmony search were suggested. The experimental results demonstrated the prominent performance of the nine proposed methods in both solution quality and computational efficiency compared to the basic moth search.

In [14], Fitness switching genetic algorithm was applied for solving the 0-1 knapsack problem by employing three distinctive methods; fitness leveling, fitness switching, and simple local search. The developed algorithm investigated the benefit of the fitness switching genetic algorithm for solving problems with many feasible solutions as well as problems with rare ones. The experiments demonstrated that the algorithm has an effective search ability, and it is tailored to solve these kinds of problems. In [15], a Binary version of the Flower Pollination Algorithm (BFPA) was proposed for solving the 0–1 knapsack problem. Since standard FPA is used for the continuous optimization, BFPA used a sigmoid function to convert the real-valued generated from FPA into one or zero. The computational results show that BFPA has better performance in large-scale KPs compared to other algorithms.

3. Meerkat Clan Algorithm

Meerkat clan algorithm (MCA) is a recent nature-based metaheuristic algorithm presented in [5]. It mimics the Meerkat animals' behavior in their coexistence and nutrition. Meerkats are animals that live in social groups from five to thirty members. These animals share both lavatories and parental care responsibility due to their sociability. Each gang has a leading alpha male and leading alpha female. Each gang has its region, and they move from one place to another in the absence of food or when forced by a more violent gang. In case they are forced to leave, the less violent gang will have to extend by another tactic or remain until they become more vigorous and retrieve their lost burrow. Each gang has someone who guards over it called a 'sentry'. This sentry guards the gang if there is a danger and warns them to avoid the risks.

Given the meerkat behavior, the MCA can be outlined as the pseudocode shown in algorithm-1 [5]. MCA assumes that there is a clan with size n of meerkat(s) (solution) where each meerkat in the clan at the initialization phase is generated randomly. The best meerkat selected from the generated clan is defined as 'sentry', and the remainder of meerkats in the clan are separated into two groups, foraging group with size m, and care group with size $(n-m-1)$. In the foraging group, each meerkat acquires the best neighbor to be its next position. Then, a fraction Fr of the worst meerkats is swapped with the best ones in the care group. A fraction Cr of the worst meerkats in the care group is discarded and replaced by random ones. The best meerkat in the foraging group becomes the sentry if it has superiority over the sentry obtained so far.

Algorithm 1- The pseudocode of meerkat clan algorithm

Meerkat Clan Algorithm

Input: Clan size (n), foraging size (m), care size (c), worst foraging rate (Fr), worst care rate (Cr), neighbor solution (k).
Output: Best Solution.
Begin
Generate random clan of solutions clan (n)
Compute fitness for clan solutions
$Sentry = best\ solution\ of\ the\ clan$
Divide the clan into two groups (foraging & care).
While not termination condition Do
For $i = 1$ to m
Generate k neighbors from foraging set
Foraging (i) = best one from k neighbor
End for
Swap the worst for Fr solution in foraging group with best ones solution in care group;
Drop the worst Cr solution from care group and generate ones solution randomly;
Select the best one of foraging call it $best\_forg$
If $best\_forg <= Sentry$ then
$Sentry = best\_forg$
End if
End while
End

4. The Proposed Algorithm

Meerkat Clan Algorithm based on chaotic map and crossover operation (MCA-CC) is an improved version of the basic MCA algorithm, which is proposed in this paper to enrich the searching behavior of the basic MCA algorithm for finding the optimal solution. The pseudo-code of the proposed MCA-CC algorithm is shown in algorithm-2. In MCA-CC, the initial solutions were generated using chaos instead of a random generation in the basic algorithm. This modification is proposed based on the characteristic of chaos which has been utilized to overcome the limitations of randomness. Furthermore, it considerably enhances the exploration ability of the algorithm. In MCA-CC, the well-known logistic map is employed to generate the initial solution via iterating one step of the equation (1) starting from a random initial value at the first iteration. The logistic map equation is described in equation 1 [16]:

$$c(t+1) = \mu\, c(t)(1 - c(t)) \quad (1)$$

where (μ) is between 3.57 and 4 for fulfilling the chaotic behavior.
Additionally, at every generation, after each meerkat in the foraging group updates its position with a neighbouring search using (2-opt) a number of the best meerkats are selected as an elite. These meerkats are used with the sentry for producing a new solution by 2-point crossover operation.
The 2-point crossover operation is performed on the two selected parents in the proposed algorithm; it takes sentry solution as parent 1 and one solution in the elite as parent 2 to create one child ($Cross\_Sol$). This operation selects two crossover points randomly and combines the parents at the crossover points to create the child or offspring solution. Thereby, the genes between the two points are copied from the parents 1 into the child and other genes are taken

from parent 2. Figure 1 shows an example of this operation. After performing the crossover operation and producing $Cross\_Sol$, one of the foraging group solutions is selected randomly ($Foraging(i), 1 \leq i \leq m$), and it is compared with the $Cross\_Sol$, if the $Cross\_Sol$ is better than $Foraging(i)$, replace $Foraging(i)$ by $Cross\_Sol$. This method is suggested to provide proposed algorithm with an ability for using the experiences of the Clan members for creating a promising solution. Furthermore, the crossover operator used is to make the offspring inherit its parent attributes and promote the search ability of the proposed algorithm for finding a global optimum.

Algorithm 2- The pseudocode of the proposed algorithm

Improved Meerkat Clan Algorithm

Input: Clan size (n), foraging size (m), care size (c), worst foraging rate (Fr), worst care rate (Cr), neighbor solution (k), elite rate (Er).

Output: Best Solution.

Begin

Generate clan of solutions clan (n) using Equation 1
Compute fitness for clan solutions
$Sentry = best\ solution\ of\ the\ clan$
Divide the clan into two groups (foraging & care).
While ($iter < Max\_iter$) Do
For $i = 1$ to m
Generate k neighbors from foraging set
Foraging (i) = best one from k neighbor
End for
Find E of the best solutions in foraging group m where ($E = Er \times n$)
For $j = 1$ to E
$Cross\_Sol = Crossover\ (Sentry, E(j))$
Select a Meerkat among Foraging (m) (say, p) randomly
If $F(Cross\_Sol) > F(p)$ then
Replace p by the $Cross\_Sol$
End for
Swap the worst for Fr solution in foraging group with best ones solution in care group;
Drop the worst Cr solution from care group and generate ones solution randomly;
Select the best one of foraging call it $best\_forg$
If $F(best\_forg) \leq F(Sentry)$ then
$Sentry = best\_forg$
End if
$Iter = iter + 1$
End while

End

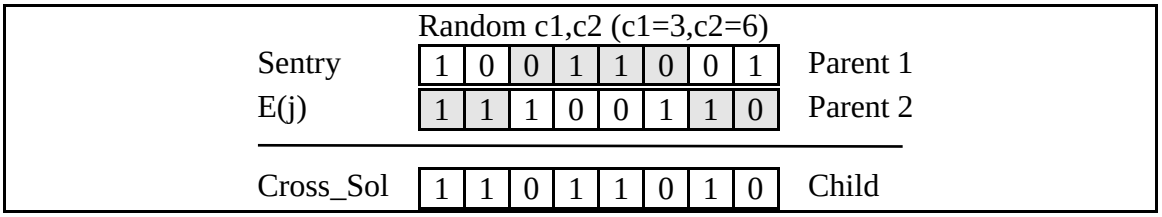

Figure -1 The crossover operation

For the proposed algorithm, the chaotic map, on one hand, is utilized to produce uniformly distributed meerkats to improve the quality of the initial population rather than random production of the initial population in the basic MCA to overcome the lack of randomness. On the other hand, the local best regions of the solution space that is found in each generation, which is not taking into consideration in the basic MCA, is employed by crossover strategy. This can make the MCA-CC algorithm proceed with searching in promising areas, thereby raising the probability of finding global optimal or near to optimal solution and speed up the convergence rate.

5. MCA-CC for Solving the Knapsack Problem

In The 0-1 knapsack problem, there are n of items to be included in the knapsack, where each item has a profit p_i and a weight w_i, and maximum weight capability W. x_i clarifies if the item is included or not in the Knapsack. The objective of this problem is to maximize the items profit so that total weights do not exceed the limited weight capability of the knapsack [15]. 0–1 knapsack problem may be described as in equations 2 and 3:

$$Max \ \sum_{i=1}^{n} p_i x_i \tag{2}$$

$$Subject\ to \ \sum_{i=1}^{n} w_i x_i \leq W, x = 0 \ or \ 1 \tag{3}$$

Consequently, to solve the 0–1 knapsack Problem by the MCA-CC algorithm, first, an initial clan of n binary solutions (meerkats) S_i is generated using equation (1). Each meerkat has a dimension m which represents the number of items available in the Knapsack Problem. The basic steps of the initialization phase can be described as follows:

- For each meerkat in the clan, generate m-dimensional random numbers ∈ [0,1].
- Each random number is used as an initial value in Equation (logistic) to produce chaotic numbers.
- Each chaotic number is converted to zero or one using equation 4:

$$c = \begin{cases} 0, & \text{if } c < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

Furthermore, Solution quality is evaluated by the fitness function, which is calculated as the gathering of the profits for all items selected in the knapsack. To retain the feasibility of the solution, the total weights of the packed items must not exceed the capacity of the knapsack so that the knapsack constraint is not contradicted. If the solution contradicts the knapsack constraint it means this is not a feasible solution. It is repaired by using the proposed repair algorithm which is illustrated in algorithm-3 to be a feasible solution.

Algorithm 3- The pseudocode of the proposed repair algorithm

Repair Algorithm

Input: infeasible solution $S(j)$.
Output: feasible solution $S(j)$
Begin
  For $i = 1$ to n (dimension of each meerkat)
    Calculate $RO = \frac{p_i}{w_i}$
  End for
  Calculate total_weight of Solution $S(j)$ via $\sum_{j=1}^{n} w_j p_j$
  While ($total\_weight > knapsack\_capacity$)
    Remove items with lowest RO
    Calculate $total\_weight$
  End while
End

6. Experimental Results

The 0-1 knapsack problem was used to measure the performance of the basic and improved version of the meerkat clan algorithm (MCA and MCA-CC respectively) and these algorithms were conducted with MATLAB R2018b and run using Intel(R) Core i7 CPU Q740, 1.73GHz with 4GB memory capacity computer. The parameter settings for MCA and MCA-CC algorithms are presented in Table 1. The datasets used for the 0-1 Knapsack problem were obtained from [17]. These datasets were classified into two categories (low and high) depending on the dimension of the problems. The data types were classified into four categories; Low Dimensional Uncorrelated Data Instances (LD-UD), High Dimensional Uncorrelated Data Instances (HD-UD), High dimensional Weakly Correlated Data Instances (HD-WC), and High Dimensional Strongly Correlated Data Instances (HD-SC) due to their variance within the correlation of the data instances. To test the algorithms performance, 10 independent runs were made for the MCA and MCA-CC algorithms on the Low-dimensional and high-dimensional 0-1 Knapsack problem. The best-obtained solution and the error rate (ER) of each algorithm for the variant dimension of the problem are presented in Table 2 and Table 3. The error rate is calculated using equation 5:

$$ER = \frac{(opt - BS)}{opt} \quad (5)$$

where (opt) is the optimal value and (BS) is the best value of the algorithm.

Table 1- Parameter setting for the algorithms

MCA		**MCA-CC**	
Parameter	**Value**	**Parameter**	**Value**
Max Iteration	500	Max Iteration	500
No. of Meerkats	75	No. of Meerkats	75
Foraging size	50	Foraging size	50
Care size	24	Care size	24
Worst foraging rate	20%	Worst foraging rate	20%
Worst Care rate	30%	Worst Care rate	30%
Neighbor solution	20	Neighbor solution	20
N/A	-	μ	3.8282
N/A	-	Elite rate	20%

Table 2- Best results of MCA and MCA-CC on the low dimension sizes with its characteristics

Problem	***Data type***	***No. of items (n)***	***Max. weight (w)***	***Optimal***	***MCA***		***MCA-CC***	
					BS	***ER***	***BS***	***ER***
f1_1-d_kp_10_269	LD-UD	10	269	295	295	0	295	0
f2_l-d_kp_20_878	LD-UD	20	878	1024	1024	0	1024	0
f3_l-d_kp_4_20	LD-UD	4	20	35	35	0	35	0
f4_l-d_kp_4_11	LD-UD	4	11	23	23	0	23	0
f5_l-d_kp_15_375	LD-UD	15	375	481.0694	481.0694	0	481.0694	0
f6_l-d_kp_10_60	LD-UD	10	60	52	52	0	52	0
f7_l-d_kp_7_50	LD-UD	7	50	107	107	0	107	0
f8_l-d_kp_23_10000	LD-UD	23	10000	9767	9767	0	9767	0
f9_l-d_kp_5_80	LD-UD	5	80	130	130	0	130	0

f10_l-d_kp_20_879	LD-UD	20	879	1025	1025	0	1025	0
Average error rate						***0***		***0***

The experimental results showed, in Table 2, that in the low dimensional problems for the basic MCA and the proposed MCA-CC have obtained the optimal solution. Whereas, in high dimensional problems, the MCA algorithm outperforms the basic MCA algorithm in terms of solutions quality for all instances. Thereby, it produced a more quality solution with less error rate. Otherwise, in Table 4 and Table 5, the iteration number (iter) of the proposed algorithm is less than the basic algorithm. It has been observed that, in most of the low-dimensional datasets, the MCA-CC finds the optimum solution at the initialization phase of algorithm 2 due to the effectiveness of the chaos in the initialization phase. Furthermore, the MCA-CC algorithm has superiority over the basic MCA algorithm in finding a more quality solution with lower number of iterations in most of the high dimensional datasets.

Table 3- Best results of MCA and MCA-CC on the high dimension sizes with its characteristics

Problem	***Data type***	***No. of items (n)***	***Max. weight (w)***	***Optimal***	***MCA***		***MCA-CC***	
					BS	***ER***	***BS***	***ER***
knapPI_1_100_1000	HD-UD	100	995	9147	9147	0	9147	0
knapPI_1_200_1000	HD-UD	200	1008	11238	11238	0	11238	0
knapPI_1_500_1000	HD-UD	500	2543	28857	28005	0.0295	28857	0
knapPI_1_1000_1000	HD-UD	1000	5002	54503	49222	0.0968	54273	0.0042
knapPI_1_2000_1000	HD-UD	2000	10011	110625	94387	0.1467	107171	0.0312
knapPI_1_5000_1000	HD-UD	5000	25016	276457	219857	0.2047	259514	0.0612
knapPI_1_10000_1000	HD-UD	10000	49877	563647	429428	0.2381	512076	0.0914
knapPI_2_100_1000	HD-WC	100	995	1514	1512	0.0013	1512	0.0013
knapPI_2_200_1000	HD-WC	200	1008	1634	1634	0	1634	0
knapPI_2_500_1000	HD-WC	500	2543	4566	4474	0.0201	4566	0
knapPI_2_1000_1000	HD-WC	1000	5002	9052	8713	0.0374	9046	0.0006
knapPI_2_2000_1000	HD-WC	2000	10011	18051	16781	0.0703	17891	0.0088
knapPI_2_5000_1000	HD-WC	5000	25016	44356	40622	0.0841	43356	0.0225
knapPI_2_10000_1000	HD-WC	10000	49877	90204	80889	0.1032	86979	0.0357
knapPI_3_100_1000	HD-SC	100	997	2397	2397	0	2397	0
knapPI_3_200_1000	HD-SC	200	997	2697	2697	0	2697	0
knapPI_3_500_1000	HD-SC	500	2517	7117	6717	0.0562	7117	0
knapPI_3_1000_1000	HD-SC	1000	4990	14390	13090	0.0903	14190	0.0138
knapPI_3_2000_1000	HD-SC	2000	9819	28919	25119	0.1314	28019	0.0311
knapPI_3_5000_1000	HD-SC	5000	24805	72505	61905	0.1461	68994	0.0484
knapPI_3_10000_1000	HD-SC	10000	49519	146919	122719	0.1647	136717	0.0694
Average error rate						***0.0771***		***0.0199***

Table 4- Comparison of iterations number of MCA and MCA-CC on the low dimensional problems.

Problem	*Data type*	*No. of items (n)*	*Max. weight (w)*	*MCA iterations no.*	*MCA-CC iterations no.*
knapPI_1_100_1000	LD-UD	10	269	0	0
knapPI_1_200_1000	LD-UD	20	878	149	2
knapPI_1_500_1000	LD-UD	4	20	0	0
knapPI_1_1000_1000	LD-UD	4	11	0	0
knapPI_1_2000_1000	LD-UD	15	375	4	0
knapPI_1_5000_1000	LD-UD	10	60	0	0
knapPI_1_10000_1000	LD-UD	7	50	0	0
knapPI_2_100_1000	LD-UD	23	10000	1	0
knapPI_2_200_1000	LD-UD	5	80	0	0
knapPI_2_500_1000	LD-UD	20	879	22	3

Moreover, the error rate obtained by the proposed algorithm (MCA-CC) was compared with different algorithms (Simulated Annealing (SA), Greedy Search Algorithm (GSA), Dynamic Programming (DP), Branch and Bound (BB) [17], and Cohort Intelligence (CI) [18]) on low dimension test datasets. This comparison is illustrated in Table 6. Additionally, on high dimension test datasets, the comparison between the results gained by the MCA-CC algorithm and other algorithms (SA, GSA, and BB [17]) are shown in Table 7. Furthermore, from the comparison results, it is observable that MCA-CC has the minimum error rate for all tested datasets means the proposed algorithm has the superiority over all tested data on the low dimension. Whereas, on the high dimension, MCA-CC has superiority in most of the tested data. To decide the statistical contrasts between the MCA-CC and the comparison algorithms with the low and the high problems' scale, the Friedman test was set. The results are presented in Table 8 and Table 9. It can be seen from the Friedman test results that the distinctions among the calculations of the algorithm are statistically significant with 99% assurance on the low and the high problems scale. The MCA-CC obtains the best overall rank.

Table 5- Comparison of iterations number of MCA and MCA-CC on the high dimensional problems.

Problem	*Data type*	*No. of items (n)*	*Max. weight (w)*	*MCA iterations no.*	*MCA-CC iterations no.*
knapPI_1_100_1000	HD-UD	100	995	18	0
knapPI_1_200_1000	HD-UD	200	1008	87	11
knapPI_1_500_1000	HD-UD	500	2543	327	235
knapPI_1_1000_1000	HD-UD	1000	5002	408	425
knapPI_1_2000_1000	HD-UD	2000	10011	463	424
knapPI_1_5000_1000	HD-UD	5000	25016	498	448
knapPI_1_10000_1000	HD-UD	10000	49877	488	478
knapPI_2_100_1000	HD-WC	100	995	1	0

knapPI_2_200_1000	HD-WC	200	1008	27	0
knapPI_2_500_1000	HD-WC	500	2543	364	73
knapPI_2_1000_1000	HD-WC	1000	5002	312	230
knapPI_2_2000_1000	HD-WC	2000	10011	233	434
knapPI_2_5000_1000	HD-WC	5000	25016	435	366
knapPI_2_10000_1000	HD-WC	10000	49877	488	376
knapPI_3_100_1000	HD-SC	100	997	56	13
knapPI_3_200_1000	HD-SC	200	997	58	11
knapPI_3_500_1000	HD-SC	500	2517	373	86
knapPI_3_1000_1000	HD-SC	1000	4990	165	106
knapPI_3_2000_1000	HD-SC	2000	9819	125	167
knapPI_3_5000_1000	HD-SC	5000	24805	320	497
knapPI_3_10000_1000	HD-SC	10000	49519	141	483

Table 6- The comparison among algorithms on the low dimensional problems based on best error rate.

Problem	***MCA-CC***	***SA***	***GA***	***GSA***	***DP***	***BB***	***CI***
f1_l-d_kp_10_269	**0**	0.0033	0	0.0237	0	0.0508	0
f2_l-d_kp_20_878	**0**	0.0507	0	0	0	0.0507	0
f3_l-d_kp_4_20	**0**	0	0	0.2	0	0.3142	0
f4_l-d_kp_4_11	**0**	0.0434	0	0.1739	0	0.0434	0
f5_l-d_kp_15_375	**0**	0	0.0084	0	0.0084	0.0247	0
f6_l-d_kp_10_60	**0**	0	0	0.1730	0	0.0576	0.0192
f7_l-d_kp_7_50	**0**	0.0467	0	0.0654	0	0.1028	0.0186
Average error rate	***0***	***0.0206***	***0.0012***	***0.0908***	***0.0012***	***0.0920***	***0.0054***

The data bold stylistic means the best results among the comparison algorithms

Table 7- The comparison among algorithms on the high dimensional problems based on best error rate.

Problem	***Data type***	***MC-CC***	***SA***	***GSA***	***BB***
knapPI_1_100_1000	HD-UD	**0**	**0**	0.6738	0.1225
knapPI_1_200_1000	HD-UD	**0**	0.0956	0.5956	0.0713
knapPI_1_500_1000	HD-UD	**0**	0.2587	0.6581	0.0282
knapPI_1_1000_1000	HD-UD	**0.0042**	0.3262	0.7261	0.0202
knapPI_1_2000_1000	HD-UD	0.0312	0.4052	0.7687	**0.0085**
knapPI_1_5000_1000	HD-UD	0.0612	0.4547	0.8216	**0.0026**
knapPI_1_10000_1000	HD-UD	0.0914	**0**	0.4814	0.9997
knapPI_2_100_1000	HD-WC	**0.0013**	0.0184	0.3124	0.0488
knapPI_2_200_1000	HD-WC	**0**	0.0593	0.3433	0.0189
knapPI_2_500_1000	HD-WC	**0**	0.1800	0.3537	0.0179

knapPI_2_1000_1000	HD-WC	**0.0006**	0.2453	0.3730	0.0050
knapPI_2_2000_1000	HD-WC	**0.0088**	0.2920	0.3870	0.0142
knapPI_2_5000_1000	HD-WC	0.0225	0.3412	0.3825	**0.0035**
knapPI_2_10000_1000	HD-WC	0.0357	0.3586	0.3964	**0.0014**
knapPI_3_100_1000	HD-SC	**0**	0.0421	0.5431	0.0538
knapPI_3_200_1000	HD-SC	**0**	0.0381	0.5939	0.0574
knapPI_3_500_1000	HD-SC	**0**	0.1424	0.5902	0.0171
knapPI_3_1000_1000	HD-SC	0.0138	0.1807	0.6116	**0.0082**
knapPI_3_2000_1000	HD-SC	0.0311	0.2225	0.6259	**0.0066**
knapPI_3_5000_1000	HD-SC	0.0484	0.2597	0.6234	**0.0022**
knapPI_3_10000_1000	HD-SC	0.0694	0.2777	0.6289	**0.0008**
Average error rate		***0.0199***	***0.1999***	***0.5471***	***0.0718***

The data bold style means the best results among the compared algorithms

Table 8- Friedman test of different algorithms for low dimensional problems.

Algorithm	Rank	1-p value	χ 2	Diff.?
MCA-CC	2.57	0.99	25.436	Yes
GA	3.00			
DP	3.00			
CI	3.21			
SA	4.29			
GSA	5.36			
BB	6.57			

Table 9- Friedman test of different algorithms for high dimensional problems.

Algorithm	Rank	1-p value	χ 2	Diff.?
MCA-CC	1.45	0.99	45.545	Yes
BB	1.9			
SA	2.69			
GSA	3.95			

7. Conclusions

The meerkat clan algorithm is a promising metaheuristic algorithm for solving optimization problems. In this paper, we presented an improved version of basic MCA based on a chaotic map and crossover (MCA-CC) for solving the 0-1 Knapsack problem. In MCA-CC, the initialization of the solutions is done using a chaotic map instead of randomization. This enhancement provides the proposed algorithm with more diversity in the clan (population). Moreover, the crossover operation is performed on the best solution attained so far which is called 'sentry' and the set of the best solution found in the foraging group which is called elite. These solutions are suggested to be a promising solution for producing better solutions that increase the ability of the proposed algorithm for finding the global optimum solution and speed up the convergence rate. The experimental results demonstrate that the basic MCA and MCA-CC have reached the optimal solution; however, the MCA-CC algorithm finds the optimal solution in less iterations than basic MCA in the low-dimensional tested problems. In the large-dimensional tested problems, the MCA-CC outperforms the basic MCA in terms of solution quality in all tested problems and finds better solutions in less iterations in most of them. Furthermore, in comparison with other algorithms, the proposed algorithm gained competitive solutions in all tested problems.

References

[1] Hussein, S.A. and Al-Obaidi, A.T.S., **2019**. An Improved Crow Search Algorithm for Solving the Traveling Salesman Problem. *Journal of Computational and Theoretical Nanoscience*, **16**(3): 978-981.

[2] Sayed, G.I., Khoriba, G. and Haggag, M.H., **2018**. A novel chaotic salp swarm algorithm for global optimization and feature selection. *Applied Intelligence*, **48**(10): 3462-3481.

[3] Kaveh, A., **2017**. Chaos embedded metaheuristic algorithms. In *Advances in metaheuristic algorithms for optimal design of structures* (pp. 375-398). Springer, Cham.

[4] Abdel-Basset, M., El-Shahat, D. and Sangaiah, A.K., **2019**. A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem. *International Journal of Machine Learning and Cybernetics*, **10**(3): 495-514.

[5] Al-Obaidi, A.T.S., Abdullah, H.S. and Ahmed, Z.O., **2018**. Meerkat clan algorithm: A new swarm intelligence algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, **10**(1): 354-360.

[6] Sonuc, E., Sen, B. and Bayir, S., **2016**. A parallel approach for solving 0/1 knapsack problem using simulated annealing algorithm on CUDA platform. *International Journal of Computer Science and Information Security*, **14**(12): 1096.

[7] Nguyen, P.H., Wang, D. and Truong, T.K., **2016**. A new hybrid particle swarm optimization and greedy for 0-1 knapsack problem. *Indones J Electr Eng Comput Sci*, **1**(3): 411-418.

[8] Feng, Y., Wang, G.G. and Gao, X.Z., **2016**. A novel hybrid cuckoo search algorithm with global harmony search for 0-1 knapsack problems. *International Journal of Computational Intelligence Systems*, **9**(6): 1174-1190.

[9] Zhou, Y., Chen, X. and Zhou, G., 2016. An improved monkey algorithm for a 0-1 knapsack problem. *Applied Soft Computing*, **38**: 817-830.

[10] Zhou, Y., Bao, Z., Luo, Q. and Zhang, S., **2017**. A complex-valued encoding wind driven optimization for the 0-1 knapsack problem. *Applied Intelligence*, **46**(3): 684-702.

[11] Han, M. and Liu, S., **2017**, December. An improved binary chicken swarm optimization algorithm for solving 0-1 knapsack problem. In *2017 13th International Conference on Computational Intelligence and Security (CIS)* (pp. 207-210). IEEE.

[12] Feng, Y., Wang, G.G., Dong, J. and Wang, L., **2018**. Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0-1 knapsack problem. *Computers & Electrical Engineering*, **67**: 454-468.

[13] Feng, Y.H. and Wang, G.G., **2018**. Binary moth search algorithm for discounted {0-1} knapsack problem. *IEEE Access*, **6:** 10708-10719.

[14] WOO, K., KWAK, J., JUNG, A.C., MYOUNG, S., LEE, H.K., WOONG, P., AHN, H., MYUNG, J.J., KWANG, O., EUN SOOK, L.E.E. and KIM, H.J., **2018**. APPLICATION OF FITNESS SWITCHING GENETIC ALGORITHM FOR SOLVING 0-1 KNAPSACK PROBLEM. *Journal of Theoretical and Applied Information Technology*, **96**(22): 7339-7348.

[15] Abdel-Basset, M., El-Shahat, D. and El-Henawy, I., **2019**. Solving 0–1 knapsack problem by binary flower pollination algorithm. *Neural Computing and Applications*, **31**(9): 5477-5495.

[16] Wu, G.C. and Baleanu, D., **2014**. Discrete fractional logistic map and its chaos. *Nonlinear Dynamics*, **75**(1-2): 283-287.

[17] Ezugwu, A.E., Pillay, V., Hirasen, D., Sivanarain, K. and Govender, M., **2019**. A Comparative study of meta-heuristic optimization algorithms for 0–1 knapsack problem: Some initial results. *IEEE Access*, **7:** 43979-44001.

[18] Kulkarni, A.J., Krishnasamy, G. and Abraham, A., **2017**. Solution to 0–1 knapsack problem using cohort intelligence algorithm. In *Cohort Intelligence: A Socio-inspired Optimization Method* (pp. 55-74). Springer, Cham.