Solving The Routing Problem Using The Improved Camel Herds Algorithm

Ahad Khaleghi Ardabili, Zied Othman Ahmed, Ali Layth Abbood
Faculty of Engineering and Natural Sciences, Altinbas University, Istanbul, Turkey
Computer Sciences Department Mustansiriyah University Baghdad, Iraq
Computer Engineering Department, Altinbas University, Istanbul, Turkey
ahad.ardabili@altinbas.edu.tr, zied_othman@uomustansiriyah.edu.iq, ali.la.abbood@gmail.com

Abstract— This paper introduces a new adaptive distributed routing algorithm based on the Camel Herds Algorithm (CHA). It is an intelligent, multi-agent optimization algorithm that is inspired by the behavior of camels and how they search for food in their desert environment. We examine its ability to solve the routing problem in switched networks: finding the shortest path in the process of transferring data packets between networks. The proposed approach is compared with three well-known meta-heuristic algorithms (ACO, GA, PSO) on ten datasets (weighted, integer, and not negative graphs) with various sizes of nodes (from 10 nodes to 297 nodes). Three performance criteria were used to evaluate the performance of the algorithms (mean relative error, standard deviation, and number of function evaluations). The results proved that the performance of the proposed algorithm is both promising and competitive with other algorithms.

Keywords— Camel Herds Algorithm, Routing Algorithms, SP Problem, Swarm Intelligence.

I. Introduction

The rapid development of networks and communications has resulted in networks being the backbone of many applications and various fields; with this continuous acceleration and progress, the challenges facing the process of transferring data and packages also increases, especially in packet-switched networks. Routing is the process of calculating the route between two nodes in networks—the source and the destination—to enable the intermediate network between them to be exploited effectively [1]. This routing process occurs in the network layer via routing protocols. each protocol depends on a specific routing strategy that uses an algorithm to build the routing table. There are two types of routing, static and dynamic.

A non-adaptive or static routing technique is simple in implementation, and the path is determined between any two nodes in a static routing table. It builds the route without taking into consideration current traffic conditions [1]. On the contrary, an adaptive routing technique helps find an alternative route when the routing node senses that the previously determined path is now more difficult to route the packet through due to changing network conditions. There are many algorithms being used to solve the routing problem. each one has a different technique to calculate the route. These techniques are based on both static and variable network metrics, such as hop count, bandwidth, cost, load or congestion, reliability, and routing delay [2].

Metaheuristic algorithms can be categorized according to the fields that inspired them. There are bio-inspired (genetic algorithm), swarm intelligence-inspired (ANT colony), and physics-inspired (simulated annealing), as well as the new promising field of game-inspired (Battle Royale optimization algorithm [3]), etc. The algorithm proposed in this paper falls under the category of swarm intelligence (SI) algorithms, which use an intelligent, multi-agent-based system that is inspired by the behaviors of animal groups, such as ants, bees, termites, and wasps, as well as flocks of birds and schools of fish [4][5]. SI-based algorithms possess three important features that greatly contribute to supporting the routing process in networks: adaptability, robustness, and scalability [6]. Our algorithm is an improvement on the Camel Herds Algorithm (CHA). Which is derived from the way camel herds search for food in their environment and from their ability to sense the humidity in the air to reach water [7].

Various algorithms and approaches have been proposed to find the shortest-path problems in networks routing. Ali and Kamoun [8] proposed a method to solve the routing problem in packet-switched networks by applying computational tools using an improved version of the neural networks. their goal was to minimize the average delay time with adaptability to changes in costs or change network topology.

Gianni Di Caro [9] produced a mobile agents routing algorithm, AntNet. It worked in two stages (forward and backward) and was based on the Ant Colony (ACO) algorithm.

S.Sugden et al.[10] tackled the shortest-path problem by choosing a certain portion of pre-calculated paths—those restricted to a specific number of hops or capacity—and then altering these paths using the simulated annealing algorithm.

Nagib and Ali 11] developed a genetic algorithm to solve the network routing protocol problem in a random ten node network topology. They compared it with the Dijkstra algorithm, which is considered one of the most popular algorithms to solve the routing problem. they produced similar results.

Ahn and Ramakrishna [12] introduced variable-length genes into the genetic algorithm to solve the routing problem. They proposed an equation for population-sizing based on the gambler's ruin model.

Casali et al. [13] proposed using Tabu Search as a metaheuristic algorithm to solve the routing problem. Instead of traditional algorithms that require exhaustive computational operations, especially in networks with high nodes. Still, it is not optimal for finding the shortest path.

A. W. Mohemmed et al. [14] developed a Particle Swarm Optimization (PSO) algorithm that utilized a changed priority-based encoding with a heuristic operator. It would minimize the probability of looping in route construction. The algorithm surpasses the results of the genetic algorithm in [12].

Verma et al. [15] developed the Omicron Ant Colony Algorithm, based on the ACO framework, to provide a solution to the routing problem. Their algorithm was built on the principle of not updating the value of the pheromone per cycle but instead updating it after a specified number of iterations to reduce the time required to update the pheromone value. The minimum and maximum levels of the pheromone is also limited.

Abdalla et al. [16] utilized neural networks to solve the routing problem in computer networks. They proposed two strategies: having a neural network with forward-feeding at each point for local decision-making, and having one with a central point to define the complete path between a pair of points (source, target). They included a supervisory component in the pathfinding system.

II. CAMEL HERDS ALGORITHM: A BRIEF OVERVIEW

The camel herds algorithm (CHA) is a metaheuristic SI optimization algorithm and intelligent multi-agent system proposed by Ahmed [7]. The CHA is inspired by the behavior of camels and how they search for food in their desert environment. It was tested for the first time to solve the Flexible Job Shop Scheduling Problem (FJSP) to try to reduce makespan value. Its results indicate that it is an effective approach that depends on exploring neighborhoods for the current solution in the problem space.

Camels are divided into herds, and each one has a leader. The leader searches for water in the desert by detecting the humidity level in the air. Wherever they find water in the desert, they also find food. These actions form the core of CHA's ability to find optimal solutions. It needs parameters that make it adaptive and scalable to solve many problems. The parameters are *n* (number of camels), *Hc* (number of herds), *d* (number of neighbors for the current solution), and *Hum* (the rate of humidity). The *Hum* is the essential factor, is set randomly for every herd. This rate directs the search and offers an intelligent way of guessing which node neighbor leads to a target. It is a decreased/increased value that is needed for each step.

MIN problems are defined using a search area that is a graph with vertices and weighted edges. The weight of the edge is the operator's cost. A MIN problem is finding a route from an origin point to a target point with a total minimum edge weight. MAX problems are specified similarly, except that operators have rewards rather than costs, and the aim is to find a way from source to the target state with a maximum amount of boundary weights.

The first step in CHA identifies the *Hc* parameter (number of herds). Each *Hc* represents one solution and has a specified number of camels *n*, and one of them is selected by the algorithm as the leader for that herd (*LHc*). Each *LHc* stores an initial starting state. This starting state is randomly chosen and allocated to leaders. The leader, *LHc*, directs the other herd members to investigate the search space to find the solution. Each herd is separate starting from a different point. This strategy provides a diversity in finding solutions. The algorithm begins with each herd, *Hc*, producing a neighbor's *d* (*d* is the number of the herd camels except the leader) for the *LHc*. All neighbors are checked by the equation (1) to achieve the best result and insert it to the *LHc* list.

III. SHORTEST PATH PROBLEM IN NETWORK ROUTING

Solving the shortest path (SP) problem means finding the optimal route between two pre-defined points. This optimum could be the shortest distance, the least time spent, or the lowest value of the path between those two points. There are many applications that require a solution to the shortest path problem, particularly operational ones, like the navigation system of vehicles [17], Travelling Salesman's Problem (TSP) [18], robotic systems path planning [19], and routing in telecommunication networks (MANETs)[20]. This research focuses on the routing problem in switched networks. It is the process of transferring data packets between networks requires finding the shortest path, which may include the shortest delivery time, bandwidth link, or less cost. Many well-known algorithms have investigated the SP problem, such as the Dijkstra algorithm, the Floyd-Warshall algorithm, and the

Bellmen-Ford algorithm. Each algorithm has its own technique for finding the shortest path. Shaveta Bhatia [21] classifies these algorithms into two types based on the complexity of the calculations they perform to optimize the path. The first is non-intelligent techniques (hard computing), such as Dijkstra's Algorithm, A* search Algorithm, etc. These techniques are useful for deterministic and certainty metrics, optimization with fixed costs, distances, and specified restrictions. The second type is intelligent techniques (soft computing) such as the ANN algorithm, GA algorithm, and ACO algorithm, which are useful for dynamic network conditions and situations. This type of algorithms uses deferent search strategies to find solutions; some of them are population-based, while others are based on local searches. The main difference is that the large search space in a population-based (it may encompass the entire network) algorithm uses a collection of candidate solutions rather than a single search point. Therefore, population-based approaches require more calculations than simple local approaches [22].

One of the most straightforward structures to represent the SP problem is a graph. It consisting of multiple vertices and arcs. Here, the vertices of the graph represent routers (where decisions of packet routing are taken) and the arcs (edges in terms of graph theory) that link these nodes are the physical connections between these. A connection also has a cost value for sending a packet over the link. The cost may represent one of the network metrics (time delay, bandwidth, etc.).

IV. PROPOSED ROUTING MODEL

This paper outlines a new novel strategy to solve the routing problem that expands the existing CHA algorithm and improves on its approach. It makes a comparison between two deference types of metaheuristic optimization algorithms (local search and population-based) in performance to demonstrate the effect of the size of the search space on the accuracy and the speed of finding the shortest path. The CHA approach depends on the exploration of neighbors for the current position of herd (a single search point) to build the path from source to destination. This approach can be classified as a local-search algorithm. It is different the population-based search algorithms that start with a collection of candidate paths and operate between them to get the best path [21].

A. Parameter Initialization

The CHA algorithm begins with the initial value of the parameters, which are *Hum*, *n*, *Hc*, *source node*, and *destination node*, to solve the SP problem.

One of the most important parameters in the CHA algorithm is the Humidity Rate (*Hum*). This rate is determined according to the problem to be solved. It is a random value assigned to each herd before starting the search in the problem space. After several tests and experiments, it was found that a rate between ("0.4 to 0.44") is the ideal value for finding the best path for the SP problem.

The second important parameter is the number of herds (*Hc*). The algorithm includes a list which stores the path for the herds during and after the search process. After running the algorithm, the final number of all the solutions is the same as the number of herds that were launched from the starting point.

As discussed above, each herd is a group of camels *n*, one of which is assigned as the leader of the herd. The rest of the camels explore and determine the best neighbor for the leader by applying the cost function to the neighboring nodes up to (n -1) times (the number of camels except for the leader). In this model, the maximum number of camels (n = 10) was used.

B. Camel Herds Strategy for Routing

After initializing the parameters, the *Hum* value is randomly determined according to the number of herds (a random value between 0.4 and 0.44 for each herd), and then the leader of each herd is identified.

The CHA starts by having each herd move from different nodes (initial node) of the neighbor's source node Fig. 1.a to ensure the provision of diversity in solutions. It starts from different points in the graph. And then pushes the source node and that initial state to the leader list (*LHc*) for that herd.

The members of the herd start to test the neighbors of the current *LHc* node by spreading camels to these nodes Fig. 1.b and testing them via the cost function (equation 1). The minimum value of cost function between all camels determines the best neighbor that will be appended to the *LHc* list for that herd.

After determining the best neighbor, the herd moves to that neighbor Fig. 1.c. The *Hum* updates, and the herd start over again from the new position, spreading camels and testing the neighbors again until the destination node is reached. These steps are repeated for each herd, and we get various solutions and then pick the best.

In summarize each herd goes through four basic steps to find the shortest path in CHA algorithm:

- Determine the camel leader in each herd (*LHc*).
- Determine the initial node for each herd (*Hc*).

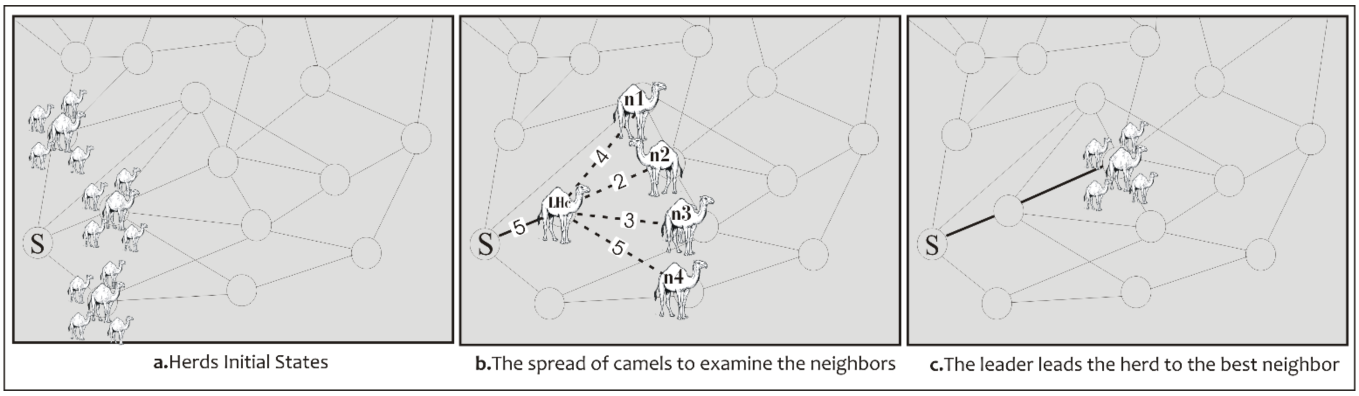
S
n1
n2
n3
n4
LHc
5
4
2
3
5
a.Herds Initial States
b.The spread of camels to examine the neighbors
c.The leader leads the herd to the best neighbor

Fig. 1 An example of artificial herds camel movement

- Spread camels *n* to test the neighbors.
- Select the best neighbor and insert it into the path.

We have improved upon this initial CHA algorithm by allowing camels to test two consecutive nodes in the network. This improvement can provide a more expansive search space to reach the destination. The results show that the algorithm is able to achieve the optimal solution, which will be discussed below.

C. Cost Function

The herd's routes for finding the shortest path depend on the output of the equation (1) [7] value for each *n* in the herd (the leader's neighbors).

$$\mathbf{Xi' = Xi * (1/Hum)}$$
$$\mathbf{|Xi+1| = (XLed - Xi') / squirt(XLed + Xi')} \qquad (1)$$

The equation was adapted to suit the SP problem, and it became as follows:

Xi: The cost of the arc from *LHc* position to each camel (*n*) after spread (neighbors nodes of the leader).

Hum: The humidity rate for current *Hc*.

XLed: The total path cost from the *source node* to the current node of the leader *LHc*.

For example, we assume the *Hum* = 0.44 for the herd in fig 1.b, the **Xi** value according to equation (1) for camels will be (**n1**=9.09, **n2**=4.54, **n3**=6.81, **n4**=11.36), and the|**Xi+1**| will be (|**Xi+1**|**n1**=1.08, |**Xi+1**|**n2**=0.14, |**Xi+1**|**n3**=0.52, |**Xi+1**|**n4**=1.57); according to the calculation of the equation above for each camel, the herd moves to the node of camel (**n2**) that gives the minimum value of (|**Xi+1**|).

The algorithm is useful, fast, and adaptable to the size of the problem. One of the most critical features is its ability to control the search space by launching more herds or even by increasing the number of camels. These critical features provides a broader exploration of the neighbors of the current state to ensure more access to the best solutions. Fig 2 shows the pseudocode of the CHA for the routing problem.

```
Input: n camels number
       Hc herds number
       min_H, max_H Humidity Rate
       source_node, destination_node
Output: The shortest path found by LHc tabu list
Begin
    For each herd (Hck ) Do
        select one of camels as a leader LHc and
insert source_node to its tabu list.
        insert starting state (Random neighbor of the
source_node) to LHc tabu list.
    End for
    For each herd Hck Do */k=1 to no.of Hc/*
        Initialize(Hum);
        Repeat
            Generate neighbors (d) randomly for LHck ;
            Generate neighbors for each d
            For z: =1 to d Do
                NCz=NCz* 1/Hum
                NCz=LHck -NCz\dis (LHck ,NCz)
            End for
            Insert the best neighbor to the path LHck
            Update Hum
        Until reach destination_node
    End for
    Print the best Hc solution
End
```

Fig.2 CHA pseudocode

V. EXPERIMENTAL RESULTS

To demonstrate the performance of the expanded camel herds algorithm, we built it using the Python 3.8 environment. It is powerful in dealing with lists and has a large number of libraries and functions that make building the code simple. Ten well-known graphs (weighted, integer, not negative) of various sizes (from 10 nodes to 297 nodes) with a number of edges (up to19900 edges) were used to test the proposed routing algorithm. Three random paths for each graph were used.

We compared the results with the Ant Colony algorithm as a non-population based approach (the same improved CHA approach). And with two other population-based approaches (Genetic algorithm, particle swarm optimization algorithm). It is important to illustration the difference between the two approaches in solving the SP problem. We especially looked at the effect of search space of these two approaches on CPU consumption time and on the quality of the solution provided by the four algorithms.

The CHA parameters are prepared with the same settings for the ACO (number of herds = number of ants, with the same number of iteration in both) to provide a fair comparison as shown in the (CHA) column in Table I; it is the first test. In the second test, the settings were reduced to 50% (for herd and iteration), as shown in the (CHA reduced) column in Table I, since the CHA algorithm does not require a high herd count or high iteration.

The first column in Table I, denotes the name and size of the graph. The source and destination for paths and its optimal costs are in columns 2 and 3. And the rest of the columns display the best cost and execution time for each algorithm. The Fig. 3 show the difference between the proposed CHA and the others (ACO, GA, PSO algorithms) in execution time for each path.

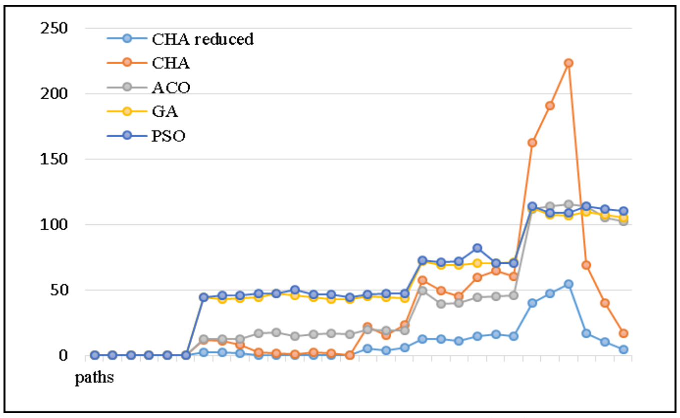

Fig. 3 Execution Times for All Paths in (sec)

Three criteria were used to check the performance of the proposed algorithm in this field: mean relative error (MRE), standard deviation (SD), and number of function evaluations (NFEs). The MRE was counted according to Equation (2), as follows:

Table I Performance compression between CHA and (ACO,GA,PSO) Shows the costs, CPU consumed time (sec)

Dataset	paths	Optimal cost	Methods									
			CHA (reduced)		CHA		ACO		GA		PSO	
			cost	time	cost	time	cost	time	cost	time	cost	time
1- Random (10-nodes)	(4 , 3)	9	**9**	**0.016**	9	0.037	9	0.062	9	0.327	9	0.243
	(8,1)	12	13	0.019	13	0.047	**12**	**0.074**	12	0.321	12	0.245
	(3,9)	9	12	0.016	12	0.047	**9**	**0.062**	9	0.319	9	0.241
2- NSFnet (14-nodes)	(0 , 13)	27	**27**	**0.015**	27	0.019	27	0.095	27	0.456	27	0.417
	(10,4)	22	**22**	**0.015**	22	0.016	22	0.092	22	0.459	22	0.425
	(3,9)	32	**32**	**0.015**	32	0.016	32	0.095	36	0.471	32	0.414
3- st70 (70-nodes)[23]	(3 , 44)	65	66	2.877	66	12.017	66	12.83	66	44.449	**65**	**44.662**
	(69,10)	78	79	2.892	**78**	**11.184**	79	13.02	86	43.466	79	46.174
	(48,49)	82	**82**	**2.108**	82	8.493	83	12.835	83	44.03	83	46.135
4- lesmis (77-nodes)[24]	(10 , 75)	3	**3**	**0.637**	3	2.481	11	17.151	9	44.815	3	47.417
	(26,70)	2	**2**	**0.563**	2	2.026	9	17.597	10	47.516	3	47.749
	(44,55)	4	**4**	**0.313**	4	1.28	4	14.75	16	46.197	4	50.122
5- sandi_authors (86-nodes)[25]	(29,76)	5	**5**	**0.564**	5	2.396	5	16.565	7	44.869	5	46.886
	(54,84)	5	**5**	**0.57**	5	2.162	5	16.868	5	42.837	7	46.637
	(32,2)	3	**3**	**0.101**	3	0.392	3	16.184	5	43.158	3	44.816
6- kroa100 (100-nodes)[23]	(8,88)	1827	1828	5.435	**1827**	**22.247**	1828	19.973	1828	45.409	1827	46.744
	(21,29)	3473	3474	3.946	**3474**	**15.856**	3474	19.204	3474	44.357	3474	47.162
	(99,98)	3759	3760	5.812	**3759**	**23.343**	3760	19.366	3769	43.775	3762	47.226
7- ch130 (130-nodes)[23]	(2 , 123)	368	**369**	**12.743**	369	57.858	369	49.451	369	72.281	369	72.775
	(65,8)	839	**840**	**12.356**	840	49.921	840	39.568	841	69.124	841	71.507
	(96,114)	421	422	11.116	**421**	**45.36**	422	39.965	432	69.449	427	72.381
8- Kroa150 (150-nodes)[23]	(1 , 135)	1843	**1843**	**14.943**	1843	59.89	1844	44.832	1844	70.46	1843	82.069
	(25,70)	4080	**4081**	**16.24**	4081	65.091	4081	45.519	4081	70.738	4081	70.403
	(143,142)	1877	1878	14.651	**1877**	**60.415**	1878	45.984	1998	71.509	1922	70.313
9- Kroa200 (200-nodes)[23]	(42 , 176)	2014	2015	39.924	2014	162.8792	2017	112.19	**2014**	**112.862**	2026	114.449
	(195,0)	2607	2608	47.766	**2607**	**190.804**	2608	114.468	2608	107.541	2614	109.345
	(153,77)	2800	2801	54.358	2801	223.587	2808	115.503	2804	106.818	**2801**	**108.891**
10- celegansneural (297-nodes)[24]	(100,279)	7	8	17.319	**7**	**69.002**	7	114.355	8	109.546	12	114.186
	(0,250)	4	**4**	**10.239**	4	39.955	4	105.639	5	107.445	4	111.921
	(260,6)	4	**4**	**4.431**	4	17.341	4	102.389	5	105.735	4	110.637

MRE = (algorithm cost - optimal cost) / optimal cost (2)

The results show in Table II and Fig. 4 that the CHA outperformed the others with a lower average error rate to find the optimal cost; Even in the reduced CHA the average is better than that of the others.

Table II Mean Relative Error for costs

Datasets	paths	Methods				
		CHA reduced	CHA	ACO	GA	PSO
1	(4,3)	0	0	0	0	0
	(8,1)	0.08333	0.08333	0	0	0
	(3,9)	0.33333	0.33333	0	0	0
2	(0,13)	0	0	0	0	0
	(10,4)	0	0	0	0	0
	(3,9)	0	0	0	0.125	0
3	(3,44)	0.01538	0.01538	0.01538	0.01538	0
	(69,10)	0.01282	0	0.01282	0.10256	0.01282
	(48,49)	0	0	0.01220	0.01220	0.01220
4	(10,75)	0	0	2.66667	2	0
	(26,70)	0	0	3.5	4	0.5
	(44,55)	0	0	0	3	0
5	(29,76)	0	0	0	0.4	0
	(54,84)	0	0	0	0	0.4
	(32,2)	0	0	0	0.66667	0
6	(8,88)	0.00055	0	0.00055	0.00055	0
	(21,29)	0.00029	0.00029	0.00029	0.00029	0.00029
	(99,98)	0.00027	0	0.00027	0.00266	0.00080
7	(2,123)	0.00272	0.00272	0.00272	0.00272	0.00272
	(65,8)	0.00119	0.00119	0.00119	0.00238	0.00238
	(96,114)	0.00238	0	0.00238	0.02613	0.01425
8	(1,135)	0	0	0.00054	0.00054	0
	(25,70)	0.00025	0.00025	0.00025	0.00025	0.00025
	(143,142)	0.00053	0	0.00053	0.06446	0.02397
9	(42,176)	0.00050	0	0.00149	0	0.00596
	(195,0)	0.00038	0	0.00038	0.00038	0.00269
	(153,77)	0.00036	0.00036	0.00286	0.00143	0.00036
10	(100,279)	0.14286	0	0	0.14286	0.14286
	(0,250)	0	0	0	0.25	0.25
	(260,6)	0	0	0	0.25	0.25
Average MRE		0.01990	0.0145	0.20735	0.3688	0.05405

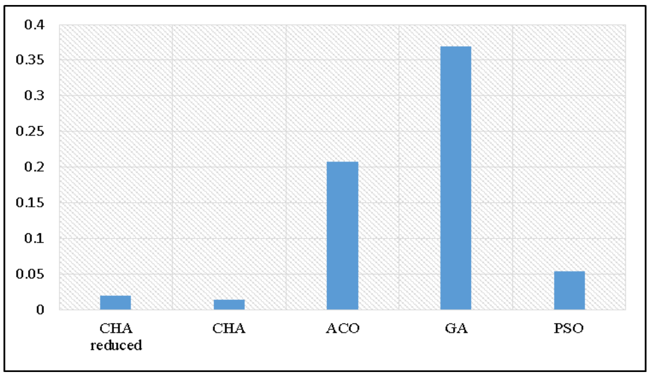

Fig. 4 Average Mean Relative Error

For standard deviation, in Table III the first and second columns denote the graphs and its paths, while the rest of columns depict the standard deviation values. Fig. 5 shows the average, and the results show that the values are close to zero (sometimes zero) for both tests CHA and (CHA reduced).

Table III Standard Deviation for Costs

Datasets	paths	Methods				
		CHA reduced	CHA	ACO	GA	PSO
1	(4,3)	0.223	0	0.444	0	2.459
	(8,1)	0	0	0	0	0
	(3,9)	0	0	0	0	0
2	(0,13)	0	0	3.332	0	6.626
	(10,4)	0	0	0	0	4.021
	(3,9)	0	0	1.231	3.071	3.360
3	(3,44)	0	0	2.087	4.701	0
	(69,10)	0.223	0.410	2.125	1.231	0.315
	(48,49)	0.444	0	7.090	0	2.593
4	(10,75)	0	0	0	4.363	0
	(26,70)	0	0	0	13.180	1.551
	(44,55)	0	0	0	6.830	0.638
5	(29,76)	0	0	0	2.190	0.816
	(54,84)	0	0	0.410	0	1.032
	(32,2)	0	0	0.447	0	2.133
6	(8,88)	0.502	0.366	48.03	0	230.29
	(21,29)	0	0	151.97	0	91.001
	(99,98)	0.223	0.410	93.996	0	55.982
7	(2,123)	0.223	0.366	28.009	38.070	47.854
	(65,8)	0	0	8.801	83.042	23.958
	(96,114)	0.512	0.366	12.330	79.593	68.122
8	(1,135)	0.366	0.444	27.831	0	207.193
	(25,70)	0.502	0.223	60.305	0	48.389
	(143,142)	0.410	0.410	30.203	0	297.896
9	(42,176)	0.307	0.366	50.793	0	51.240
	(195,0)	0.470	0.307	131.137	0	77.292
	(153,77)	0	0	71.428	0.223	135.198
10	(100,279)	0.510	0.223	0.615	0.75	1.974
	(0,250)	0.223	0	0.510	0	7.932
	(260,6)	0	0	0	0	2.229
Average SD		0.1715	0.129	24.437	7.908	45.736

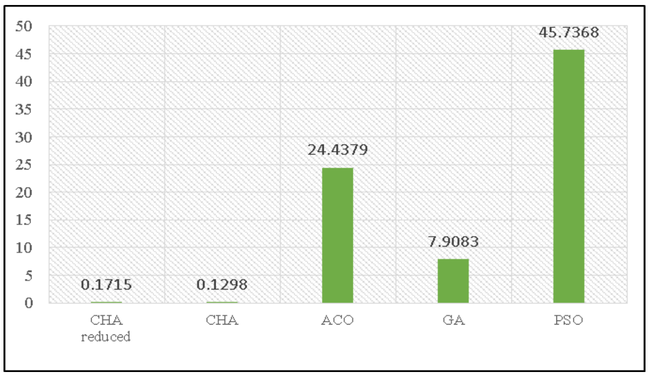

Fig. 5 Average Standard Deviation

As discussed above, the herd moves according to the lowest value determined by the camels after examining the leader's neighbors, so the number of function evaluations of the cost function depends directly on the number of camels, which raises the CHA's NFEs, as shown in Fig. 6. However, the number of camels offers the advantage of a wider exploration of the nodes. (CHA reduced) still provides promising results compared to other algorithms.

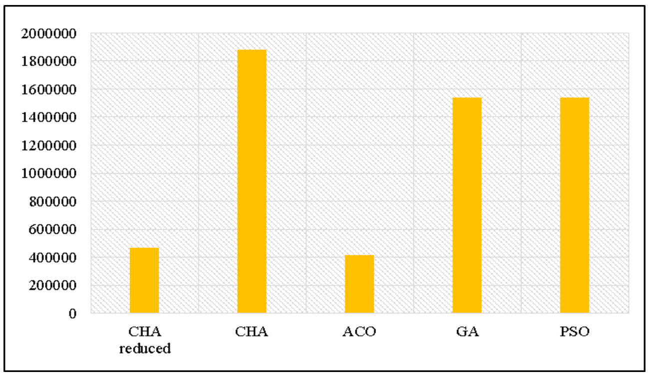

Fig. 6 Number of Function Evaluations

VI. CONCLUSION

Many routing algorithms have been developed to reach the best path. In this paper, a new routing algorithm was proposed that expands the camel herds algorithm, which relies on examining the best neighbors of the current solution. The algorithm can be easily adapted to the problem space, as the search space can be controlled according to the size and type of the network. In contrast to population-based algorithms, in which the search space is large and time-consuming. The CHA is a fast algorithm and offers more diversity because each herd can provide an independent path. The algorithm was run to solve the shortest path problem for ten networks of different sizes. The obtained results show that the algorithm is efficient compared to other algorithms (ACO, GA, PSO). In the future, the algorithm will be used to solve a k-shortest path problem.

VII. REFERENCES

[1] D. Medhi and K. Ramasamy, *Network routing: algorithms, protocols, and architectures*. 2017.

[2] R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel, "A survey on routing metrics," *TIK Rep.*, p. 53, 2007.

[3] T. Rahkar Farshi, "Battle royale optimization algorithm," *Neural Comput. Appl.*, Jun. 2020.

[4] X.-S. Yang and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation Theory and Applications*. 2013.

[5] C. Blum and X. Li, "Swarm Intelligence in Optimization," *Swarm Intell.*, pp. 43–85, 2008.

[6] F. Ducatelle, G. A. Di Caro, and L. M. Gambardella, "Principles and applications of swarm intelligence for adaptive routing in telecommunications networks," *Swarm Intell.*, vol. 4, no. 3, pp. 173–198, 2010.

[7] A. T. Sadiq Al-Obaidi, H. S. Abdullah, and Z. O. Ahmed, "Camel Herds Algorithm: a New Swarm Intelligent Algorithm to Solve Optimization Problems," *Int. J. Perceptive Cogn. Comput.*, vol. 3, no. 1, pp. 6–10, 2017.

[8] M. K. Mehmet Ali and F. Kamoun, "Neural Networks for Shortest Path Computation and Routing in Computer Networks," *IEEE Trans. Neural Networks*, vol. 4, no. 6, pp. 941–954, 1993.

[9] G. Di Caro, "Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks," *Intelligence*, vol. 41, p. 374, 2004.

[10] S. J. Sugden, M. Randall, G. Mcmahon, and S. Sugden, "A Simulated Annealing Approach to Communication Network Design Learningn spaces project View project Water management project View project A Simulated Annealing Approach to Communication Network Design Recommended Citation A Simulated Annealing Approach to Communication Network Design," *Artic. J. Comb. Optim.*, vol. 6, no. 1, pp. 55–65, Mar. 2002.

[11] G. Nagib and W. G. Ali, "Network routing protocol using Genetic Algorithms," *Int. J. Electr. Comput. Sci. IJECS-IJENS*, vol. 10, no. 02, pp. 40–44, 2010.

[12] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 566–579, 2002.

[13] D. Casali, G. Costantini, and M. Carota, "A tabu search based routing optimization algorithm for packet switching networks," *Proc. 11th Conf. Proc. 11th WSEAS Int. Conf. Circuits*, no. January 2007, pp. 180–184, 2007.

[14] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Appl. Soft Comput. J.*, vol. 8, no. 4, pp. 1643–1653, 2008.

[15] O. P. Verma, N. Gupta, M. Sharma, P. Nanda, and S. Chawla, "A new approach to dynamic network routing using Omicron Ant Colony algorithm," in *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, 2011, vol. 5, pp. 177–181.

[16] A. Y. Abdalla, P. T. Y. Abdalla, and K. A. Nasar, "Computer Network Routing Using Neural Networks," vol. 30, no. 1, pp. 1–14, 2012.

[17] Y. Tan, "A New Shortest Path Algorithm Generalized on Dynamic Graph for Commercial Intelligent Navigation for Transportation Management," 2019.

[18] Z. O. Ahmed, A. T. Sadiq, and H. S. Abdullah, "Solving the Traveling Salesman's Problem Using Camels Herd Algorithm," *SCCS 2019 - 2019 2nd Sci. Conf. Comput. Sci.*, pp. 1–5, 2019.

[19] K. Akka and F. Khaber, "Mobile robot path planning using an improved ant colony optimization," *Int. J. Adv. Robot. Syst.*, vol. 15, no. 3, pp. 1–7, 2018.

[20] S. Liu *et al.*, "Dynamic Analysis for the Average Shortest Path Length of Mobile Ad Hoc Networks under Random Failure Scenarios," *IEEE Access*, vol. 7, pp. 21343–21358, 2019.

[21] D. S. Bhatia, "Survey of shortest Path Algorithms," *Int. D. S. Bhatia, "Survey shortest Path Algorithms," Int. J. Comput. Sci. Eng., vol. 6, no. 11, pp. 33–39, 2019.al J. Comput. Sci. Eng.*, vol. 6, no. 11, pp. 33–39, 2019.

[22] D. A. Perez Ruiz, " Modern Optimization with R ," *J. Stat. Softw.*, vol. 70, no. Book Review 3, pp. 10–12, 2016.

[23] G. Reinelt, "Tsplib 95," *Interdiszip. Zent. für Wissenschaftliches Rechn. (IWR), Heidelb.*, vol. 338, pp. 1–16, 1995.

[24] M. Newman, "Newman Datasets," 2013.

[25] V. Batagelj and A. Mrvar, "Pajek datasets (2006)," 2009.