Khodadadi and Saeidi *Comput Soc Netw* (2021) 8:6
https://doi.org/10.1186/s40649-021-00087-y

Computational Social Networks

RESEARCH Open Access

Discovering the maximum k-clique on social networks using bat optimization algorithm

Akram Khodadadi and Shahram Saeidi*

*Correspondence:
sh_saeidi@iaut.ac.ir
Department of Industrial Engineering, Islamic Azad University, Tabriz Branch, Tabriz, Iran

Abstract

The k-clique problem is identifying the largest complete subgraph of size *k* on a network, and it has many applications in Social Network Analysis (SNA), coding theory, geometry, etc. Due to the NP-Complete nature of the problem, the meta-heuristic approaches have raised the interest of the researchers and some algorithms are developed. In this paper, a new algorithm based on the Bat optimization approach is developed for finding the maximum k-clique on a social network to increase the convergence speed and evaluation criteria such as *Precision*, *Recall,* and *F1-score*. The proposed algorithm is simulated in Matlab® software over Dolphin social network and DIMACS dataset for $k = 3, 4, 5$. The computational results show that the convergence speed on the former dataset is increased in comparison with the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) approaches. Besides, the evaluation criteria are also modified on the latter dataset and the F1-score is obtained as 100% for $k = 5$.

Keywords: K-clique problem, Bat optimization algorithm, Genetic algorithm, Ant colony optimization

Introduction

Any social structure of individuals that is created based on a social relationship is called a Social Network (SN). A social network includes a set of people and the social relationships among them. Therefore, a social network is composed of two elements: the participating entities in the relationship between these entities. Social networks are divided into two types, offline and online. Offline networks include a network of friends, a network of colleagues, or classmates. Online networks include social networks such as Facebook, Twitter, and Google+ [24].

Nowadays, online Social Networks (SN) are Internet-based services on a platform where people can share their photos, moods, events, schedules, ideas, find-outs, favorites, and any information with each other [1]. The SNs generally propose creating virtual groups of public or private types. In public groups, every kind of users may exist with different attributes, specialties, aims, and motivations. The number of users in such a group like *Facebook* or *Myspace* often reaches a hundred million people. The private groups are restricted on topic, the number of the members and they are formed around a specific subject, the *GoodReaders* on book study, and *Flicker* on photography are some famous examples.

With the rapid development of social networks, determining the prospects for future needs for analysis and forecasting in terms of reducing risks and increasing decision-making accuracy requires scalable modeling and the selection of fast and accurate algorithms. To ensure understanding of network interactions, it is very difficult to select the appropriate algorithm to use network structural knowledge instead of behavioral knowledge. This eliminates the need for the process of learning network changes at consecutive times and allows for rapid prediction of two consecutive decision times for large data sets.

The online social network analysis consists of several different issues like security, determining the leaders, centrality, prestige, cliques, trust, influence, and so on. The social network structure is usually modeled as a graph where the nodes and edges represent the users and the relations, respectively. Since the users of social networks are not necessarily connected all together, the corresponding graph would not be complete. A clique is defined as a subgraph in which all the nodes (users) are connected; in other words, it is a complete subgraph [2].

A clique is a set of users having a common favorite, attributes, or goals [20]. Identifying the cliques is one of the important aspects of the social networks and can increase the efficiency of marketing, advertising, discovering unofficial organizations, or even uncovering criminal or terrorist groups [3, 20]. Krebs [16] in his research has studied the terrorist and maleficent teams in the United States based on the available data before the September 11, 2001 attacks on the World Trade Towers, extracted a subgraph of size 37 perpetrators of the attack which later their role in the attack was confirmed [16].

Identifying the largest clique of size k in a social network is known as the k-clique problem and many researchers have proposed deterministic algorithms having computational complexity greater than or equal to $O(n^2)$ that are not sufficiently rapid, or heuristic methods that are not accurate enough. The cons column of Table 1 identifies some issues.

This issue is schematically depicted in Fig. 1 for $k=3$ and 4.

In this paper, finding the maximal k-clique problem is studied, and considering the NP-Complete nature of the problem [19], a meta-heuristic algorithm based on the bat optimization approach is developed. The proposed algorithm is simulated in Matlab® over the standard dataset adopted from the literature. The convergence speed of the proposed algorithm is compared with previous methods based on GA and ACO algorithms. The evaluation criteria are also calculated and discussed.

The rest of the paper is organized as follows: the literature review and related work are discussed in "Literature Review"; research methodology and proposed algorithm are described in "The Proposed Model"; findings and results are revealed in section IV; conclusions and future works are stated in "Conclusions".

Literature review

Considering the simulation of the proposed algorithm using datasets, numerical assessments, and evaluation of the output results, the research methodology used in this paper can be classified as quantitative, analytical, conclusive, and applied research.

Table 1 The summary of related work

Reference	Method	Pros	Cons
[27]	Genetic Algorithm	Reducing process time and cost	Small pattern tree and using more memory
[6]	Ant Colony Optimization	Searching large patterns without generating small or medium patterns	Hard to analyze and understand the algorithm
[19]	Heuristic Algorithm External Optimization	Good solutions and convergence speed	Low prediction confidence
[12]	Clustering	Usable in Internet marketing	Can be used only in IoT
[8, 8]	Edge centrality, optimization	Usable on large graphs	Need for expert intrusion
[4]	Statistical Methods	High precision and low classification cost	Hard to identify in unsupervised mode
[4]	Greedy Search	Low error	Computational Complexity
[5]	Basic Element Analysis	Better supervised modeling	Less effort on developing the model
[22]	Bit Pattern Search	Capability for using online	Use of pattern creation and testing
[26]	Decomposing Network to subgraphs	Identifying central or isolated users	Ignoring the time and computational complexity
[10]	Formal Analysis	Improving the F1 metric	Difficult to understand the logic of the proposed algorithm
[15]	Mathematical Analysis	Considering the time period	High computational time
[23]	Local Strategies	Reducing computational complexity	Ignoring performance metrics for comparison

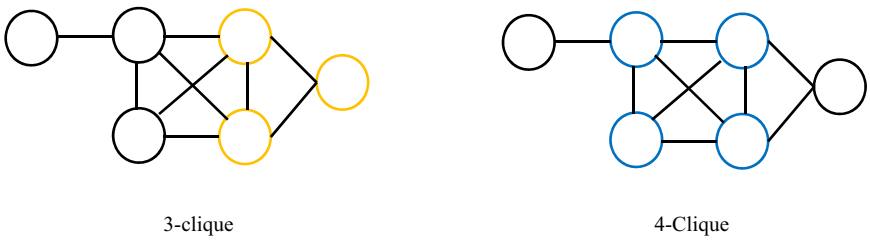

Fig. 1 K-clique examples

The clique definition

There is not a precise definition of the clique concept, because its usage is different for various applications. The clique is a structural and topological attribute of the graphs. Simply, a clique is a set of nodes, such that the number of edges inside this set is far more than the edges of that set with the other vertices of the graph. The cliques of the graph may be overlapped, which means that some nodes of the graph are simultaneously a member of two or more different cliques. Many algorithms are proposed for the detection of the cliques in weighted or unweighted, directed or undirected graphs which are briefly studied and stated in the section "Related Work".

An abstract definition of the clique is proposed by Thai and Pardalos [24] as follows. Considering a simple undirected graph $G=(V, E)$ in which $V=\{1, 2, \ldots, n\}$ is the set of vertices (nodes), and $E \subseteq V \times V$ is the set of edges where $||V||=n$ and $||E||=m$. The graph G is complete if $\forall\ u,v \in V$ where $u \neq v$, the edge $(u, v) \in E$. The clique C is a subset of E if

the extracted subgraph *G[C]* is complete. The clique is called maximal if it is not within a clique larger than itself, and it is called maximum if there is not ant another larger clique in the graph. The size of the maximum clique inside of the graph *G* which is known as the clique number is shown as *ω(G)* symbol [24].

Another formal definition for clique is proposed by Hao et al. [13] and Hao et al. [14]. A clique is a subset of $S \subset V$ where for any two members of S like v_i and v_j, the edge (v_i, v_j) exists in *E*. A clique *c* is called maximal if there is not another clique c' in the same graph G, where $c \subset c'$.

Related work

Several studies are done by the researchers and many algorithms based on deterministic, heuristic, and meta-heuristic approaches are proposed which most of which are time-consuming, sensitive to initial conditions, need to adjust the parameters, low-efficiency in sparse graphs, and not-scalability are the most common disadvantage.

The first review survey was published in 1994. Wu and Hao [27] made a comprehensive review and published a survey including mathematical models, heuristic, and meta-heuristic approaches developed by researchers for solving the maximum clique problem. Considering the NP-Complete nature of the problem, most of the recent studies are focused on developing meta-heuristic algorithms. The Genetic Algorithm (GA) and the Ant Colony Optimization (ACO) approaches are two of the oldest methods that can be mentioned for solving the maximum clique problem. Fenet and Solnon [6] proposed an ACO-based algorithm called Ant-Clique which distinguishes the maximal clique by repeatedly adding new nodes to the partial clique ever found. The ACO algorithm as a basic approach is combined with the Simulated Annealing (SA) algorithm by Xu et al. [28] and Tabu-Search (TS) by Rezvanian and Meybodi [21] which have reported acceptable results; nevertheless, both methods are complicated and the execution times are high.

Hao et al. [11] used the basic component analysis method for finding maximum k-clique on SNs. They first created an official context for the SN using a modified adjacency matrix and next defined new concept names *k-equiconcept* on a network and claimed that the k-clique problem is equivalent to the k-equiconcept problem. Hao et al. [8] 8] developed a method for detecting k-cliques on the Dolphin dataset. In this iterative process, each node tracks its clique—through rate by averaging its neighbors' information at each step. There is also a maximum amount for the number of cliques a graph can be a member of. This approach was evaluated using the Triadic Formal Concept Analysis (TFCA) after transforming a weightless social network. Experimental results show that this algorithm can be effective in identifying reliable bands.

Duan et al. [4] solved the maximum clique problem based on clustering and analyzed it on two datasets, ENRON and DBLP. In this model, a link is created for both users if they participate in a discussion about one or more topics or stories. In this case, they both have similar interests. Therefore, network edges are updated using the information compatibility attitude between each pair of users. Each pair of users *i* and *j* may exchange some topics or identities with each other, but the implicit orientations or attitudes of the two users on different topics may not be the same. Using simple statistical methods, the attitude compatibility between both users is calculated.

The value of this criterion is between 0 and 1, and the higher the value, the greater the degree of consistency of attitudes of both pairs in a subject. In the second part, a fast parallel optimization algorithm is used that performs greedy optimization to find cliques. The experimental results show that the clustering algorithm for k-clique on both datasets has less error to solve the problem.

Patrick and Östergård [19] proposed a fast algorithm for solving the maximum clique problem. The results of the proposed method on DIMACS graphs show favorable responses in terms of clique size and convergence speed compared to other methods. Finally, their proposed method and the results are validated using convergence diagrams. This approach is based on the drop in density between each pair of parent and child nodes. The lower the density, the more likely the child is to make an independent clique. Therefore, based on the maximum flow of the minimum cut theorem, a new algorithm that can automatically find an optimal set of local cliques is proposed.

Hao et al. [8, 9] proposed a strategy to improve the graphs for finding the maximum graph by adding a preprocessing step in which the edges are weighted according to their centrality in the network topology. In this method, the centrality of the edge indicates its cooperation in graph weighting. This strategy effectively tries to complete the information about the network topology and can be used as an additional tool to maximize the graph. The calculation of the center of the ridge is performed by performing several random walks with limited length on the network, which calculates the center of the edge possible in large-scale networks.

The purpose of [12] research was to investigate the method of extracting k-clique information in social network analysis. Initially, using a clustering algorithm, it groups all social topics into a set of topics. Network members are then divided into clusters of social issues in which they are involved. Due to the difference in the strength of the connections between the nodes, link analysis is performed in each local cluster to find cliques. In this study, by applying the principles of social networks to the Internet of Things (IoT) and considering marketing in human social networks, they showed that the impact of the Internet of Things on Internet marketing and making it smart is very important.

In the research of [18], a meta-heuristic algorithm has been proposed that removes low-value vertices with greater probability from the current largest clique. Thus more valuable vertices have more chance (probability) for presence. With each small move, many changes are made to the current clique, and step by step it moves toward finding the largest clique. The results of their proposed method on DIMACS graphs show good results in terms of clique size and convergence speed compared to other methods.

Traag and Bruggeman [25] proposed a new ant colony algorithm to solve the clique problem. In recent years, the ant colony optimization algorithm has achieved successful results in solving various discrete optimization problems, but in the clique problem, the standard ant colony optimization algorithm has low convergence. Therefore, to solve the maximum clique problem, the researchers suggested changes in the way the pheromone is updated to select the appropriate alternative path. Their algorithm, while maintaining the initial successful properties, has low computational complexity and rapid convergence.

Mirghorbani and Krokhmal [17] proposed a method for finding a k-sized clique in k-segmented graphs. This method is based on the bit pattern used to find cliques. Varun and Ravikumar [26] went from network analysis to several associations for community recognition in telecommunications networks. They also identified central users and network isolators by identifying common members between associations.

Himmel et al. [15] examined the society detection problem in time graphs and, by modifying the Bron–Kerbosch algorithm, proposed a method for identifying time cliques. Sun et al. [23] studied the society detection problem in dynamic graphs and showed that their proposed method is less complex in time than the previous methods.

Identifying the largest clique can also be used as a tool in image processing. By defining a threshold for the size of the largest association between the pixels of two consecutive frames of an image, [30] provided a way to predict the direction of motion of a video camera. Saeidi [7] proposed an Artificial Bee Colony optimization algorithm for finding the maximal clique and compared their proposed method with ACO and PS-ACO based on some DIMACS benchmarks. Table 1 summarizes the previous methods and related pros and cons.

The proposed model

In this section, the proposed Bat algorithm and the mapping to the maximum clique problem are discussed.

The bat optimization algorithm

Group intelligence is one of the most powerful optimization techniques based on group behaviors. In this paper, a new algorithm based on the Bat optimization method is proposed to find the largest k-clique in social networks. The Bat Algorithm which is inspired by the collective behavior of bats in the natural environment is developed by [29]. This algorithm is based on the use of sound reflection by bats where they find the exact path and location of their prey by sending sound waves and receiving their reflections. When the sound waves return to the bat transmitter, the bird can draw an acoustic image of the obstacles in front of it and see the surroundings well. Using this system, bats can detect moving objects such as insects and immobile objects such as trees.

The audio location feature enables bats to find their prey. Bats produce a very loud sound pulse and listen to its return from the surrounding objects. Pulses have different characteristics that are determined by considering the strategy of hunting bats and the type of creature which they intend to hunt. Bats can detect the distance and direction of the target and even the speed of their prey. The logic used in the simulated algorithm is as follows.

Each virtual bat flies randomly at a speed equal to v_i, and its location $\mathrm{x_i}$ will be the final solution of the algorithm. A bat changes its sound wavelength A_i and pulses emission rate r_i while searching for prey. The search for the local solution is also enhanced by a random step. This algorithm uses the following three rules.

Rule 1: All bats use voice positioning to detect distances and can distinguish the difference between food and obstacles.

Rule 2: Although the sound amplitude can be changed in different ways, it is assumed that the volume changes from a large positive constant value A_0 to a smaller value A_{min}.

Rule 3: Bats fly randomly at the speed of v_i at position x_i with constant frequency f_{min} and variable wavelength λ to a height of A_0 to find their prey.

According to the stated rules, the location and velocity for each virtual *i*th bat (solution) at iteration t as well as the frequency f_i are calculated using the Eqs. (1), (2), and (3) [29]:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{3}$$

where $\beta \in [0.0.1]$ is a uniform random vector and x* is the best location so far which is updated at every iteration considering the location of all virtual bats. At each iteration, in the local search phase, one of the solutions is selected as the best solution and the new position of each bat is updated locally in a random step using Eq. (4):

$$x_{new} = x_{old} + \varepsilon A^t, \tag{4}$$

where $\varepsilon \in [-\ 1.0.1]$ is a random number and A^t is the average amplitude of the bats at iteration t. Besides, the loudness of A_i and the transmitted pulse rate of r_i are updated using relations (5) and (6):

$$A_i^{t+1} = \alpha A_i^t \tag{5}$$

$$r_i^{t+1} = r_i^0[1 - e^{-\gamma t}]. \tag{6}$$

α and γ are constant values. In fact, α is similar to the cooling coefficient in the Simulated Annealing (SA) algorithm. In the SA method, every point in the search space is similar to a state of a physical system, and the objective function is similar to the internal energy of the system in that state. In this method, the goal is to transfer the system from the desired initial state to the state in which the system has the least energy. For $0 < \alpha < 1$ and $\gamma > 0$, the Eq. (7) will be true:

$$A_i^t \rightarrow 0\ ,\ \ r_i^t \rightarrow r_i^0. \tag{7}$$

The algorithm starts with a randomly generated initial population of artificial bats. Each bat demonstrates a possible solution, a complete subgraph of the problem. The input graph of a social network is implemented by an adjacency binary $n \times n$ matrix. The nodes having a higher degree would have a higher chance to be a member of a complete subgraph. Therefore, a bat is demonstrated as a binary vector of length n. Assuming $n = 10$, Fig. 2 shows a sample bat where nodes 3, 5, 6, and 8 are the members of the subgraph (Clique) as the related vector values are equal to 1.

A randomly generated bat may be infeasible, i.e., some of the proposed nodes of the subgraph are not connected in the main graph. To prevent generating infeasible solutions, a heuristic is applied in the proposed method. By assessing the adjacency matrix, one of the nodes with the highest connections (say node *j*) is randomly selected and the

1	2	3	4	5	6	7	8	9	10
0	0	1	0	1	1	0	1	0	0

Fig. 2 Demonstration of a bat in proposed algorithm

Pseudo code of proposed algorithm
Input: The In-graph array. *//The adjacency matrix of the graph*
Output: Near-optimal (Maximum) K-clique. *// The node numbers of the largest detected k-clique*
1: **Start**
2: Initialize the velocity v_i for each position x_i $i = (1 \ldots n)$ *//Set initial values for velocity and position of the bats*
3: Generate the bat population using the proposed generation heuristic *//Create initial clique and set the In-graph Array*
4: Define pulse frequency f_i at x_i $i = (1 \ldots n)$ *//Initialize the bat frequency variables at x_i position*
5: Initialize pulse rates r_i and the loudness A_i for each position x_i . $i = (1 \ldots n)$
6: **while** stop condition not met do *//The main loop repeats for a certain number of iterations (e.g. 300)*
7: Generate new solutions by adjusting frequency; *// New k-clique is generated by changing f_i variables*
8: Updating velocities and locations/solutions; *//Apply equations (1), (2) and (3)*
Calculate the total fitness and set Clique = In-graph , MaxClique = Size(In-graph) ; *// Calculate equation (8)*
9: **if** ($rand() > r_i$) **then** *// Start a local search for probability r_i*
10: Select a solution among the best solutions; *// One of the best ever solutions is selected*
11: Generate a local solution around the selected best solution; *// Apply a local search and set the new position using equation (4)*
12: **end if**
13: Generate a new solution by flying randomly *// Move bats to new position and find new solutions*
14: **if** ($rand() < A_i$ and $f(x_i) < f(x^*)$) **then** *// Accept new solutions if it is better then the best ever found solution*
15: Accept the new solution;
16: Increase r_i and reduce A_i; *// Update the altitude and pulse rate of the bats using equations (5) and (6)*
17: **end if**
18: Rank the bats and find the current best x^* *// Find the best solution among all calculates solutions*
19: **end while**
20: Decode the optimal solution and output the k-Cliques. *// Show the final solution in a user friendly manner*
21: **End**.

Fig. 3 The pseudocode of the proposed algorithm

j^{th} element of the ith bat is set to 1. Next, the other highest node is selected randomly, and it is accepted if and only if it is connected to node j according to the adjacency matrix. This procedure may be a bit time-consuming, but increases the algorithm's convergence speed and prevents the elimination of the infeasible solutions to keep the population size stable. The flowchart of the Bat optimization algorithm is depicted in Fig. 3.

The body of the algorithm consists of local and global searches. In the local phase, the algorithm uses an initial solution and then moves to neighboring solutions in an iterative loop. If the neighbor solution is better than the current one, the algorithm sets it as the current solution; otherwise, the algorithm will likely accept this solution as the current solution. This procedure is similar to accepting the neighbor points of the search space in the Simulated Annealing method.

The fitness function

The goodness of the solutions generated in an evolutionary algorithm should be evaluated by a fitness (objective) function. In the proposed algorithm, the fitness of the nodes of a solution (subgraph, clique) obtained by a bat is calculated using Eq. (8):

$$\lambda(v_i) = \begin{cases} 1 - \frac{C-1}{adj(v_i)} & v_i \in \text{Clique} \\ 1 & \text{otherwise} \end{cases}, \tag{8}$$

where $\lambda(v_i)$ is the fitness of node v_i, *Clique* is the subgraph generated by a bat, and C is the size of the Clique. After calculating the fitness of the nodes of a clique, the nodes are decreasingly sorted by fitness values. The nodes having the least (the worst) fitness values are randomly substituted with another node that does not belong to the current clique, and it is connected to the node being eliminated. In the proposed method, first, the proximity of the random vertex with all the vertices of the clique is checked, and if confirmed, the same procedure is repeated for the adjacencies of this random vertex and the approved vertices are added one by one to the vertices of the current clique at this stage.

Initializing

In the initializing step of the proposed method, for each $\mathrm{Bat_b}$ (b = 1, 2, …, B), a random solution is generated in which B indicates the number of bats (population size). A possible solution, which is the k-clique, is represented by an array of length n, the number stored in the i^{th} index of the array indicates the ID of the node that is a member of the clique. Therefore, during the initialization step, the initial population of solutions will be a B × n matrix. The pseudocode of the proposed method is presented in Fig. 3.

The algorithm gets the adjacency matrix of the graph as input. The initial population of the bats is generated randomly. The lines (4) and (5) of the pseudocode defines the frequency of the bats at x_i position, and the pulse rates consequently. These are the main parameters of the search method used by bats during hunting. They fly randomly at velocity v_i in position x_i with variable wavelength (frequency) and sound altitude. The man loop of the algorithm starts at the line (6) and ends at line (19). The number of iterations depends and the program size and varies between 100 and 300. The lines (7) and (8) are used to change the frequency and altitude of bats in the main loop. At each iteration, some solutions are generated by changing the frequency and one of them is selected by random to perform a local search around with probability r_i. Line (9) demonstrates this issue. A new solution is also generated by changing the position of the bat in line (10) and its fitness is calculated. If the new solution is better than the best solution found so far, it will be replaced; the pulse rate and altitude parameters will be updated with probability A_i (lines 14–17). At the end of the loop, the bats are sorted and the best position is identified (line 18).

Simulation computational results

The proposed algorithm is implemented in Matlab using a PC with a 4 GHz processor and running Windows 10 and the simulation results are compared with that of GA and ACO.

Evaluation criteria

Hao et al. [11] defined three evaluation metrics as *Precision*, *Recall*, and *F1-Score*. The precision means a ratio of positives that the test correctly marks as positive. In this research, it is defined as the number of k-cliques detected by the algorithm on the total number of real k-cliques. A recall is the ratio of the number of relevant k-cliques in the detection results to the total number of existing relevant k-cliques. The F1-Score

Table 2 The precision metric of the first experiment

k t	3	4	5
1	0.397	0.727	1
2	0.366	0.785	1
3	0.318	0.666	1
4	0.500	0.800	1
5	0.348	0.708	1

Table 3 The recall metric of the first experiment

k t	3	4	5
1	0.659	1	1
2	0.634	1	1
3	0.608	0.941	1
4	0.760	0.800	1
5	0.704	0.944	1

combines the precision and metrics to evaluate the ability of the algorithm in finding k-cliques using Eq. (9):

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{9}$$

Dataset

The proposed algorithm is simulated on the Dolphin dataset containing 62 nodes and 159 edges available at https://networkdata.ics.uci.edu. The related graph is a dynamic graph in which the relations change over time. The structure of the network is dynamic and changes over time. This property is implemented by defining a monthly timestamp on network relations. It means that dolphin D_i is not connected to dolphin D_j in April, but they may establish a relationship in June and terminate again in September.

Besides, to compare the efficiency of the proposed algorithm with similar methods like GA and ACO, standard DIMACS graphs.[1]

Experiments and simulation results

The first experiment is performed on the Dolphin dataset for $k=3, 4, 5$; and time intervals $t=1,2,0.5$ months, and the evaluation metrics are calculated. The simulation is executed on 25 graphs for 30 iterations, and the best-obtained values for precision, recall, and F1-score metrics are reported in Tables 2, 3, and 4 respectively.

[1] Available at http://www.dcs.gla.ac.uk/~pat/maxClique/distribution/DIMACS_cliques.

Table 4 The F1-score of the first experiment

k t	3	4	5
1	0.495	0.841	1
2	0.464	0.879	1
3	0.417	0.779	1
4	0.603	0.800	1
5	0.465	0.809	1

Table 5 The specification of the simulated DIMACS graphs

No	Graph Name	# of Nodes	# of Edged	Largest *k*	*Obtained *k*
1	c-fat200-1	200	1534	12	12
2	c-fat200-2	200	3235	24	24
3	c-fat200-5	200	8473	58	58
4	c-fat500-1	500	4453	14	14
5	c-fat500-2	500	9139	26	26
6	c-fat500-5	500	23,191	64	64
7	Johnson8-4–4	70	1200	14	14
8	Johnson16-2–4	120	5460	8	8
9	Johnson32-2–4	496	107,880	16	16
10	Keller4	171	9435	11	11
11	Keller5	776	225,990	27	27
12	hamming6-2	64	1824	32	32
13	hamming8-2	256	31,616	128	128
14	hamming10-2	1024	518,646	512	512
15	Sanr200-0.7–1	200	13,868	18	17
16	Sanr400-0.5	400	39,984	13	13
17	Sanr1000	1000	250,500	15	14

* The best-obtained values among the individual runs of the proposed algorithm are reported

The results show that the value of the precision and recall metrics increase as the value of *k* increases and the proposed algorithm reaches a better performance for larger cliques.

The second experiment is performed over a standard DIMACS dataset on 17 graphs. The specifications of the graphs are listed in Table 5.

According to Table 5, the proposed algorithm was successful in finding the best-known solutions for *c*, *Johnson*, *Keller*, and *hamming* family graphs. Besides, it has obtained the best or near-optimal solution in *Sanr* family graphs. Therefore, the largest k-clique is identified in 15 out of 17 test cases.

Comparing with other metaheuristics

The Genetic Algorithm (GA) is a population-based approach and is made up of some chromosomes each of which representing a solution to the problem. A chromosome is a group of genes that are the basic elements of the genetic. Generally, the first generation of the chromosomes is created randomly, and during the iterations, the next

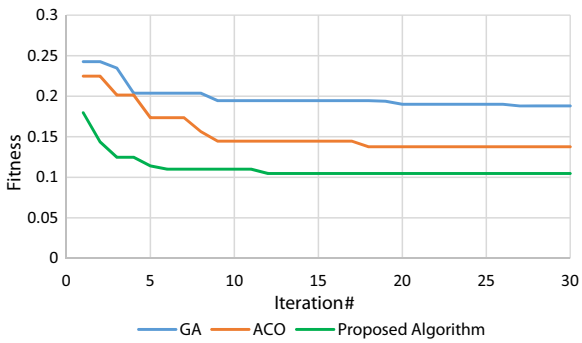

Fig. 4 The convergence diagram of the proposed algorithm, GA, and ACO on Keller4 graph

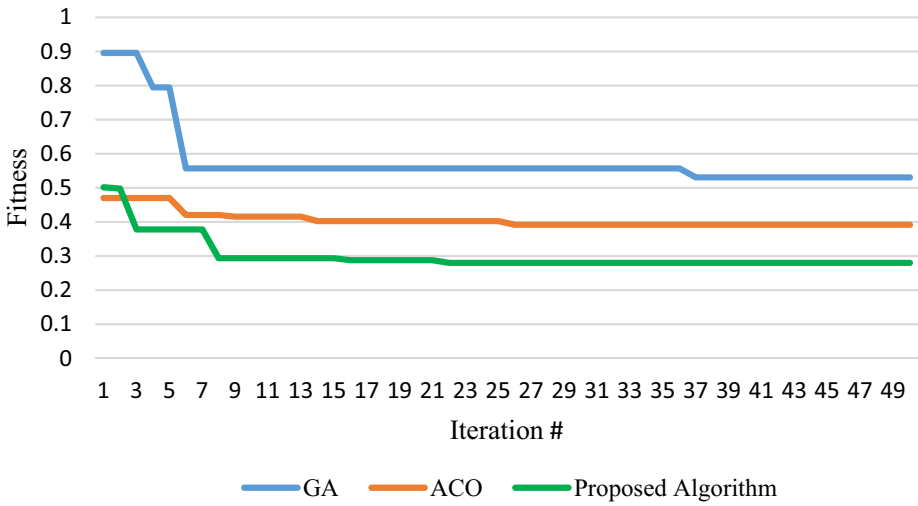

Fig. 5 The convergence diagram of the proposed algorithm, GA, and ACO on Johnson32 graph

generations are created by applying crossover and mutation operators. It mimics the evolution process in nature happening on the chromosomes of the livings during the time.

The ACO approach that was first introduced by Marco Dorigo in 1992 is a population-based search method in which the behavior of the ants and their ability to find the shortest path between their nest and a food source is inspired. Initially, ants randomly search for food sources. Upon finding a food source, they return to the nest while laying down pheromone trails. The rate of pheromone density on the shorter paths is higher. Over time, evaporation reduces the amount of pheromone deposited, and thus, the attractiveness (probability to choose) of the path is reduced.

To evaluate the convergence speed of the proposed method, it is also compared with GA and ACO approaches on *Keller4* and *Johnson32-2–4* graphs, and the results are depicted in Figs. 4 and 5. The horizontal and vertical axes demonstrate the iteration number and the fitness value calculated by Eq. (8), respectively.

As the figures show, the convergence speed of the proposed algorithm is higher than that of GA and ACO, i.e., on the *Keller4* graph Fig. 4, it has reached the final solution at

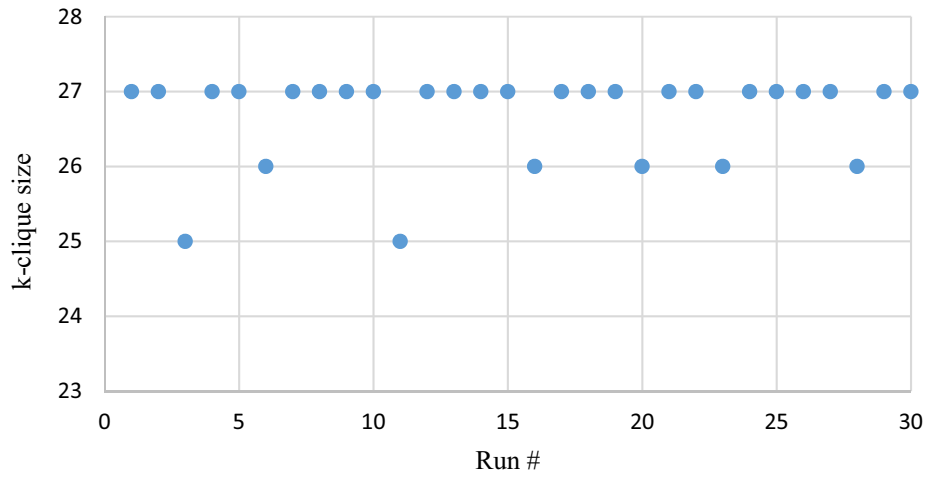

Fig. 6 The stability diagram of the proposed algorithm on Keller5 graph

Table 6 The Comparison of precision, recall, and F1-score

Reference	$k=3$			$k=4$		
	Recall	Precision	F1-score	Recall	Precision	F1-score
Proposed method	0.348	0.704	0.4657	0.708	0.944	0.8091
[23]	0.402	0.677	0.5044	0.655	0.8634	0.7448
[26]	0.312	0.713	0.4340	0.6203	0.8218	0.7069
[15]	0.329	0.684	0.4442	0.6812	0.8819	0.7686

iteration 12, where the GA and ACO have obtained a worse solution at iteration 27 and 18 consequently.

Considering the random nature of the meta-heuristic approaches, the stability test should also be performed to analyze the variation of the obtained solutions during different runs. Figure 6 demonstrates the stability of the proposed algorithm on the *Keller5* dataset obtained in 30 different runs. As the figure shows, the proposed algorithm has obtained the maximum k-clique ($k=27$) in 23 out of 30 runs, and near-optimal solutions $k=26$ and $k=25$ in 5 and 2 runs consequently. Therefore, considering the low variation of the final solutions, it can be concluded that the proposed algorithm consists of good stability.

The comparison of the proposed method in terms of recall, precision, and F1-score along with the methods propped by [23], 26, 15] is also performed on Dolphin dataset for $k=3,4$ and $t=5$ months. The results are reported in Table 6.

As shown in Table 6, considering the F1-score values, the proposed method has obtained a slightly better performance except for [23] method for $k=3$.

Conclusions

In this paper, a new algorithm based on the bat optimization approach is developed for finding the maximum k-clique in social networks. The proposed algorithm is simulated in Matlab over 17 DIMACS standard graphs adopted from the literature and the

Precision, *Recall,* and *F1-score* metrics are calculated. The computational results revealed the high performance of the proposed algorithm finding the largest k-clique in 15 out of 17 test cases.

Besides, to evaluate the efficiency of the proposed method, the Genetic algorithm and Ant Colony Optimization approaches are also simulated in Matlab, and convergence speed is compared. The simulation results show that the bat optimization algorithm performs faster than the other two approaches in finding the final solution.

Finally, to analyze the stability of the proposed method, all test cases are executed at least 30 different runs and a low value of the standard deviation of the final obtained solutions revealed the stability of the proposed algorithm.

The future work will be extended in two directions: first, developing the algorithm to work efficiently on large or huge graphs containing thousands or millions of nodes and edges; second, the current model can be modified to deal with graphs having dynamic structures rather than static and constant relations.

Acknowledgements
Not applicable.

Authors' contributions
This paper is adopted from the M.Sc. dissertation performed by the first author under the supervision of the second author. The subject and its contribution are defined by the second author. All authors read and approved the final manuscript.

Funding
The authors have not received any funding for performing this research.

Availability of data and materials
The dataset used in this research is adopted from: http://www.dcs.gla.ac.uk/~pat/maxClique/distribution/DIMACS_cliques

Competing interests
The authors declare that they have no competing interests.

Received: 12 September 2020 Accepted: 19 January 2021
Published online: 02 February 2021

References

1. Backstrom L, Huttenlocher D, Kleinberg J, Lan X. Group formation in large social networks: membership, growth, and evolution. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06). New York, NY, USA: Association for Computing Machinery; 2016. p. 44–54. https://doi.org/10.1145/1150402.1150412
2. Boyd DM, Ellison NB. Social network sites: definition, history, and scholarship. J Comput Mediat Commun . 2007;13(1):210–30.
3. Can F, Ozyer T, Polat F. State of the art applications of social network analysis. Cham: Springer; 2014.
4. Duan D, Li Y, Li R, Lu Z. Incremental K-clique clustering in dynamic social networks. Artif Intell Rev. 2012;38:129–47. https://doi.org/10.1007/s10462-011-9250-x.
5. Falkowski T. Community analysis in dynamic social networks. PhD. Mag-deburg: Otto-von-Guericke-University; 2009. https://d-nb.info/995040419/34.
6. Fenet S, Solnon C. Searching for maximum cliques with ant colony optimization applications of evolutionary computing. LNCS. 2003;2611:236–45.
7. Fotoohi S, Saeidi S. Discovering the maximum clique in social networks using artificial bee colony optimization method . Info Technol Comput Sci. 2019;10:1–11.
8. Hao F, Yau SS, Min G, Yang LT. Detecting k-balanced trusted cliques in signed social networks. IEEE Internet Comput. 2014;18(2):24–31.
9. Hao F, Stephen S, Yau S, Geyong M, Laurence T. Detecting k-balanced trusted cliques in signed social networks. IEEE Internet Comput. 2014;18(2):159–78.
10. Hao F, Min G, Pei Z, Park DS, Yang LT. K-clique community detection in social networks based on formal concept analysis. IEEE Syst. 2015;54:965–86.
11. Hao F-S, Min P, Young-Sik G, J. Jong-Hyuk P. . k-Cliques mining in dynamic social networks based on triadic formal concept analysis. Neurocomputing. 2016;209:57–66.
12. Hao F, Stephen S, Yau S, Geyong M, Yang LT. K-Clique community detection in social networks based on formal concept analysis. IEEE Syst J. 2016;99:268–88.

13. Hao F, Min G, Pei Z, Park D, Yang LT. k-clique communities detection in social networks based on formal concept analysis. IEEE Syst J. 2017;11(1):250–9.
14. Hao F, Pei Z, Yang LT. Diversified Top-k maximal clique detection in social internet of things. Future GenerComput Syst. 2020;107:408–17.
15. Himmel A, Molter H, Niedermeier R, Sorge M. Enumerating maximal cliques in temporal graphs. In: 2016 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). San Francisco, CA: IEEE; 2016. p. 337–344. https://doi.org/10.1109/ASONAM.2016.7752255.
16. Krebs V. Mapping network of terrorist cells. Connections. 2002;24(3):43–52.
17. Mirghorbani M, Krokhmal P. On finding *k*-cliques in *k*-partite graphs. Optim Lett. 2013;7:1155–65.
18. Palla G, Derenyi I, Farksa I, Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. Nature. 2005;435:814–8.
19. Östergård PR. A fast algorithm for the maximum clique problem. Discret Appl Math. 2002;120(1–3 15):197–207.
20. Pempek TA, Yermolayeva YA, Calvert SL, S.L. . College students' social networking experiences on Facebook. J Appl Dev Psychol. 2009;30(3):227–38.
21. Rezvanian A, Meybodi MR. Finding Maximum Clique in Stochastic Graphs Using Distributed Learning Automata. Int J Unc Fuzz Knowl Based Syst. 2015;23(1):1–31.
22. Shahrivari S, Jalili S. High-performance parallel frequent subgraph discovery. J Supercomput. 2015;71:1–21.
23. Sun S, Wang Y, Liao W, Wang W. Mining maximal cliques on dynamic graphs efficiently by local strategies. In: 2017 IEEE 33rd international conference on data engineering (ICDE). San Diego, CA: IEEE; 2017. p. 115-118. https://doi.org/10.1109/ICDE.2017.53.
24. Thai MT, Pardalos P. Handbook of Optimization in Complex Networks. New York: Springer; 2012.
25. Traag VA, Bruggeman J. Community detection in networks with positive and negative links. Phys Rev. 2009;80:1–6.
26. Varun E, Ravikumar P. Telecommunication community detection by decomposing network into n-cliques. In: 2016 international conference on emerging technological trends (ICETT). Kollam: IEEE; 2016. p. 1–5. https://doi.org/10.1109/ICETT.2016.7873770.
27. Wu Q, Hao J. A review on algorithms for maximum clique problems". Eur J Operat Res. 2015;242(3):693–709.
28. Xu, X., Ma, J. Lei, J. (2007). An Improved Ant Colony Optimization for the Maximum Clique Problem, Proc. Third International Conference on Natural Computation (ICNC 2007), IEEE Press, Aug. 2007, 766–770, DOI: 10. 1109/ ICNC. 2007. 205.
29. Yang XS. A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DavidAlejandro, Cruz C, Terrazas G, Krasnogor N, editors. Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Berlin: Springer; 2010. p. 65–74.
30. Zhang Y, Hou Z, Yang J, Kong H. Maximum clique based RGB-D visual odometry. In: 2016 23rd International conference on pattern recognition (ICPR). Cancun: IEEE; 2016. p. 2764-2769. https://doi.org/10.1109/ICPR.2016.7900054.

Publisher's Note