Information Sciences 670 (2024) 120631

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An enhanced discrete particle swarm optimization for structural k-Anonymity in social networks

Navid Yazdanjue [a], Hossein Yazdanjouei [b], Ramin Karimianghadim [c], Amir H. Gandomi [a,d,*]

[a] Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia
[b] Microelectronics Research Laboratory, Urmia University, Urmia, Iran
[c] School of Engineering, University of British Columbia, Kelowna, BC V1V 1V7, Canada
[d] University Research and Innovation Center (EKIK), Óbuda University, 1034 Budapest, Hungary

ARTICLE INFO

Keywords:
Social networks
K-anonymity
Structural Information Loss
Discrete particle swarm optimization algorithm
Generalization approach

ABSTRACT

Recently, social network usage has exhibited explosive growth, leading to a huge amount of users' private data available. The main challenge in releasing social network data publicly is the protection of the users' privacy while preserving its utility for third parties. Accordingly, several social network privacy-preserving methods have been introduced, where anonymization is the most common approach. Structural k-anonymity is a widely used anonymization model to mask the structure of social networks by clustering the edges and nodes into super-edges and super-nodes. However, it comes at the cost of losing structural information, which is measured by a criterion called structural information loss (SIL). This study introduces an enhanced discrete particle swarm optimization (EDPSO) algorithm, which effectively minimizes the SIL within the clustering process of the structural k-anonymity model, leading to a high-utility anonymized network. In this regard, we propose a vector-based solution representation that can be efficiently exploited by the EDPSO. Moreover, a novel position updating heuristic is suggested for the EDPSO, which adaptively tunes the operators' selection probabilities. This happens based on each operator's performance both in the current iteration and their history regarding the number and the average amount of fitness improvements in the previous iterations. We also propose two fortified versions of the EDPSO algorithm (EDPSOVNS and EDPSOSA) by employing two new network-specific local search strategies to enhance the exploration, exploitation, and convergence rate of the process. Simulation results on the nine real-world networks demonstrate the superiority of the suggested algorithms in terms of the fitness value, reliability, and convergence rate over other analyzed approaches found in the literature.

* Corresponding author at: Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia.
E-mail address: gandomi@uts.edu.au (A.H. Gandomi).

https://doi.org/10.1016/j.ins.2024.120631
Received 29 May 2022; Received in revised form 23 January 2024; Accepted 10 April 2024
Available online 14 April 2024

Nomenclature.

Sets	
V	Set of nodes in the original graph
E	Set of edges in the original graph
$\overline{V}$	Set of nodes in the anonymized graph
$\overline{E}$	Set of edges in the anonymized graph
$CL\_Set = [Cl_1, \cdots, Cl_{s\_id}, \cdots, Cl_s]$	Set of clusters
C_Nodes	Set of candidate nodes whose velocity $= 1$ in $\overrightarrow{V_i}$
Sub_Nodes	Set of nodes in a subgraph
$Subgraph^h_{n\_id}$	The subgraph of graph G, including the nodes with the shortest path of h-hops ($h = [1, \cdots, n-1]$) from the node with n_id label
N_Sols	Set of new solutions obtained from applying neighborhood operator
Vectors	
$\overrightarrow{X_i^t} = [x_1, \cdots, x_{n\_id}, \cdots, x_n]$	The position vector of i th particle at time t
$\overrightarrow{Vel_i^t} = [vel_1, \cdots, vel_{n\_id}, \cdots, vel_n]$	The velocity vector of i th particle at time t
$\overrightarrow{Pbest_i}$	The personal best position vector of i th particle
$\overrightarrow{Gbest}$	The global best position vector in the swarm
$\overrightarrow{FV^t}$	Fitness value vector at time t
$\overrightarrow{TFV^t}$	Transformed fitness value vector at time t
$\overrightarrow{NFV^t}$	Normalized fitness value vector at time t
$\overrightarrow{NoI^t}$	Number of improvement vector at time t
$\overrightarrow{NNoI^t}$	Normalized number of improvement vector at time t
$\overrightarrow{AoI^t}$	Amount of improvement vector at time t
$\overrightarrow{AvgAoI^t}$	The average amount of improvement vector at time t
$\overrightarrow{NAvgAoI^t}$	The normalized average amount of improvement vector at time t
$\overrightarrow{OSP^t}$	Operator selection probability vector at time t
Parameters	
$\|V\| = n$	Number of nodes
$\|E\| = m$	Number of edges
$v_{n\_id} \in V$	A node (vertex) in the social network graph
$n\_id \in [1, \cdots, n]$	Label of the node in the network
$e_{v_i, v_j} \in E$	An edge between nodes v_i and v_j
$adm_{i,j} \in [0, 1]$	An element of the adjacency matrix
k	Structural k-anonymity constraint
$s\_id \in [1, \cdots, s]$	Label of the super-node
Cl_{s_id}	A cluster with the label s_id
$\|Cl_{s\_id}\|$	Number of nodes in the cluster with label s_id
$\left\|E_{Cl_{s\_id}}\right\|$	Number of intra-edges in the cluster with label s_id
$\left\|E_{Cl_1, Cl_2}\right\|$	Number of inter-edges between cluster 1 and cluster 2
$P_i : i \in [1, \cdots, p]$	The i th particle
$x_{n\_id} \in \overrightarrow{X_i^t}$	The position value of the node with n_id label in the position vector
$vel_{n\_id} \in \overrightarrow{Vel_i^t}$	The velocity value of the node with n_id label in the velocity vector
it	Iteration number
it_{Max}	Maximum iteration number
sub_it	Sub-iteration number
sub_it_{Max}	Maximum sub-iteration number
$temp$	The temperature in the simulated annealing algorithm
n_{Pop}	Swarm size (population size) of the EDPSO algorithm
$P_i.Ft$	The fitness value of i th particle
c_node	A candidate node whose velocity is 1 in $\overrightarrow{V_i}$
r_node	A random node
sub_node	A node in a subgraph
rn_id	Label of the random node in the network
cn_id	Label of the candidate node in the network
op_id	The index of the operator
S_op	The selected operator
n_sol	A new solution obtained from applying neighborhood operator
w_1	Importance weight of the $\overrightarrow{NFV^t}$
w_2	Importance weight of the $\overrightarrow{NNoI^t}$
w_3	Importance weight of the $\overrightarrow{NAvgAoI^t}$
Ft_{Max}	The maximum fitness value found so far
r, r_1, r_2	The random numbers in the range of $[0, 1]$
Constants	
IW	Inertia weight
c_1, c_2	Acceleration constants

(*continued on next page*)

(*continued*)

Sets	
V	Set of nodes in the original graph
T_0	Initial temperature
α	Temperature reduction rate
Other Indices	
G	Original social network graph
$\overline{G}$	Anonymized social network graph
ADM	The adjacency matrix of the graph
intraSIL	Intra structural information loss
interSIL	Inter structural information loss
SIL	Structural information loss
MAXSIL	Maximum structural information loss
NSIL	Normalized structural information loss

1. Introduction

Over the past century, there has been a remarkable surge in the popularity of social networking applications and websites, such as Facebook, Twitter, Instagram, and Google + . With millions of users engaging in various communities worldwide, these online social networks (OSNs) contain a massive amount of users' personal and private data as the users share a lot of personal details of their lives with millions of other users. The structure of these social data is in a graph form, wherein each node represents a user within the network. These nodes can have various attributes, such as name, social security number, zip code, contact details, and more. Also, each edge within the graph indicates a connection between two users, delineating their relationships.

In these OSNs, every user should have the right to decide what information is shared with others and under what circumstances, which is called an individual's privacy. Privacy disclosure risks arise when online platforms decide to publish the OSN data publicly or share it with third-party companies for research or business-related purposes. In other words, the major challenge facing social network platforms is to strike a balance between publishing a version of the data that is beneficial for analyses conducted by academic researchers and business managers while simultaneously protecting the user's personal and private information. As a result, to address this challenge, it is urgent for OSN data owners to adopt reliable privacy-preserving methods, such as anonymization techniques. Recently, several anonymization techniques have been proposed to transform a dataset into its anonymous version. These techniques involve removing or obfuscating personally identifiable information (PII) from data so that the individuals' identities are indeterminable [1]. One of the simplest anonymization methods is Naïve anonymization, in which synthetic attributes replace the original ones [2]. Although Naïve anonymization is a common practice, it is possible for an adversary to reidentify an individual by collecting external information on the individual's relationships. To prevent this issue, various anonymization techniques have the ability to change the graph's structure, resulting in an obfuscated graph that preserves users' relationships. However, these anonymization processes may reduce data utility since changing the graph's structure may lead to information loss. It is evident that, as the number of changes in the graph structure increases, users' privacy is enhanced, but the utility of the anonymized graph decreases. In other words, a trade-off between utility and privacy in the anonymization process necessitates a balance between these two criteria. Hence, while the results of analyses performed on the anonymized graph should be similar to those of the original graph, the reconstruction process of the original OSN structure should be as arduous as possible, if not impossible.

In recent decades, graph theory has emerged as a prominent research field for proposing efficient social network anonymization techniques. These techniques can be categorized into three main groups involving: Edge/Vertex Modification, Uncertain Graph, and Generalization/Clustering approaches [3,4]. The Edge/Vertex Modification approach anonymizes a graph by adding and/or deleting edges or vertices in the graph [2]. The Uncertain Graph approach anonymizes a graph by generating an uncertain graph in which a probability in the range of $[0, 1]$ is assigned to every possible edge in the graph [5]. Generalization/Clustering is a well-known approach that clusters nodes and edges into so-called super-nodes and super-edges for the proper anonymization of the network [6–8]. One of the most used techniques in the Generalization/Clustering approach is structural k-anonymity, which protects the structural information of an OSN by clustering the nodes and edges in a way that the structural properties of each node are indistinguishable from at least k-1 other nodes [6,8–10].

The process of finding optimum clusters in the structural k-anonymity method is proved to be an NP-hard optimization problem that can be solved by heuristic methods or by guided random search techniques (e.g., *meta*-heuristics) [6,9]. Numerous studies have proposed greedy or heuristic algorithms for social network anonymization based on the structural k-anonymity method. Although these methods try to find the optimal clusters with respect to the privacy and utility criteria, they often suffer from the drawback of trapping in local optima, not being able to guarantee to obtain the best possible solution. Therefore, some researchers have recently used *meta*-heuristics and evolutionary algorithms in their suggested models to tackle the mentioned issues, yielding better solutions concerning the balance between privacy and utility.

Accordingly, in the current research study, a practical OSN anonymization approach based on the structural k-anonymity model is presented, which utilizes a novel enhanced discrete particle swarm optimization (EDPSO) algorithm to mask the social network. The main objective of the proposed approach is to generate an anonymized network that preserves the users' privacy with respect to their relationships while minimizing the structural information loss (SIL) to maintain utility. To meet this goal, we first formulate the structural k-anonymity problem using an efficient solution representation based on arrays rather than complex matrix-based solution representations used in the previous studies [9,10]. Next, we enhance the discrete particle swarm optimization (DPSO) algorithm

introduced in [11] by proposing a novel heuristic to update the position vector of each particle effectively. The suggested heuristic modifies the operator selection probability in an adaptive manner based on the fitness values obtained from each operator in the current iteration, as well as their historical performance. This historical performance considers the number and the average amount of fitness improvements achieved by each operator in the previous iterations. By incorporating this information, the heuristic effectively adjusts the operator selection probability, enabling a balance between exploration and exploitation capabilities while avoiding premature convergence. Eventually, to enhance the suggested algorithm's performance in terms of fitness values and convergence rate, we combined the EDPSO with two local search algorithms, which are based on the simulated annealing (SA) [12] and variable neighborhood search (VNS) [13] algorithms, called EDPSOSA and EDPSOVNS, respectively.

The rest of the paper is organized as follows. The literature review of the existing graph anonymization approaches is surveyed in Section 2. In Section 3, we describe the structural k-anonymity problem. The solution representation, the definition of the objective function, and the proposed methods, comprising EDPSO, EDPSOSA, and EDPSOVNS, are explained in Section 4. Subsequently, in Section 5, we present experimental results, including an evaluation of information loss and a comparison with other state-of-the-art methods. Finally, Section 6 concludes the paper and points to potential future work directions.

2. Literature review

In recent years, a large and growing body of literature has investigated OSN privacy-preserving using graph anonymization techniques. As mentioned earlier, these techniques can be categorized into three classes: Edge/Vertex Modification, Uncertain Graph, and Generalization/Clustering. The subsequent sub-sections will delve into further detail on each of these categories.

a. Edge/Vertex Modification Approach

This approach anonymizes a graph by adding or deleting edges or vertices in the graph. The existing literature on this approach can be broadly categorized into two main groups (i.e., random perturbations and constraint perturbations) depending on the way that the modifications are applied to the graph.

The random perturbation techniques anonymize a graph by introducing random noise to the original data by removing actual nodes/edges and adding the fake ones at the same time. One of the pioneering studies in this category was published by Hay et al. in 2007, in which a graph is anonymized by removing p edges and then randomly adding p fake edges to the perturbed graph [2]. While this method has the advantages of low complexity and easy understanding, the original graph can easily be reidentified since this method does not protect the network hubs. Afterwards, Casas-Roma employed different graph modification techniques in the Edge/Vertex Modification approach, and the obtained anonymized graphs were analyzed regarding the privacy level and data utility criteria to identify the most efficient technique [14]. Most recently, in 2021, Kumar et al. proposed an OSN privacy-preserving approach based on the fuzzy sets and rewiring algorithm, focusing on the randomization of the social network while simultaneously preserving the degree of each node [15].

A great deal of research has also been conducted on the constrained perturbation approach, which involves the application of edge addition and deletion techniques to fulfil specific constraints. The most well-known method of this approach is the k-anonymity model introduced by Samarati and Sweeney in a study published in 1998 [16]. The primary aim of their model was to anonymize a group of individuals while ensuring that the anonymized network satisfies the k-anonymity constraint. Generally, the k-anonymity constraint stipulates that each user should not be distinguished from at least k-1 other users in the released anonymized network [17,18]. In other words, when attackers strive to reidentify the users' identities in a k-anonymous network, they should not be able to do so with a probability higher than 1/k [19].

In recent years, there has been an interest in extending the application of the k-anonymity model to OSN anonymization. For instance, in 2020, Rajabzadeh et al. proposed a method based on the Edge/Vertex Modification approach and were able to satisfy the k-degree anonymity model using the Genetic Algorithm (GA) [20]. More recently, in 2021, Kiabod et al. suggested an algorithm to increase the k-degree anonymization speed and improve the network utility using the Edge/Vertex Modification approach and the

Table 1
An Overview of the Studies on the Edge/Vertex Modification Approach.

Year	Authors	Key Contribution
1998	Samarati and Sweeney [16]	Introduced the k-anonymity model for group anonymization.
2007	Hay et al. [2]	Proposed a graph anonymization method by randomly removing and adding edges.
2020	Rajabzadeh et al. [20]	Proposed a method for reducing the utility loss on the graph using Genetic Algorithm (GA).
2021	Kumar et al. [15]	Proposed a Fuzzy Sets and Rewiring Algorithm for OSN privacy-preserving, focusing on randomization while preserving node degrees.
2021	Kiabod et al. [21]	Suggested an algorithm to increase k-degree anonymization speed using the firefly optimization algorithm (FA).
2023	Kaur et al. [22]	The average path length of the graph was preserved, and noisy nodes/edges addition was reduced in k-degree-anonymization on social networks.
2023	Kiabod et al. [23]	Efficiently finds an optimal k value for each social network graph, striking a balance between privacy and information loss.
2023	Medková et al. [24]	Proposed HAkAu algorithm, which enhances the k-automorphism method using a GA for efficient social network anonymization.

firefly optimization algorithm (FA) [21]. In 2023, Kaur et al. proposed an enhanced version of k-degree-anonymization for social network privacy preservation [22]. This version incorporates a hybrid of Artificial Neural Network (ANN) and Support Vector Machine (SVM), known as k-NeuroSVM, to maintain the average path length of the network while substantially minimizing the noisy nodes and edge addition. During the same year, Kiabod et al. presented a novel technique called FSopt_k that effectively identifies the optimal value of k for each social network to address the increasing information loss and often unnecessary high privacy levels in previous studies [23]. Their method considers the graph's structural features to determine an optimal privacy level, balancing the preservation of privacy and the minimization of information loss. Table 1 provides a summary of the discussed studies, focusing on the edge/vertex modification approach. Besides, in a recent study, Medková et al. introduced the novel HAkAu algorithm, enhancing k-automorphism methods using GA for improved social network anonymization [24]. The authors proposed a unique chromosome representation for k-automorphism that inherently ensures k-anonymity and employs a divide-and-conquer strategy for selecting vertex-disjoint subgraphs, making it highly efficient.

b. Uncertain Graphs Approach

Several recent studies have explored the background concepts of uncertain graphs to achieve anonymized networks. In this approach, the anonymization process generates an uncertain graph by assigning a probability in the range of $[0, 1]$ to every possible edge in the graph. The uncertain graph approach is relatively new compared to the two other approaches, resulting in fewer studies focused on it. The earliest method of this approach was proposed by Boldi et al., in which the authors injected uncertainty in social graphs and published the resulting uncertain ones [5]. However, their method only considered a subset of vertex pairs in the graph while ignoring the rest. Afterwards, in 2018, Yan et al. presented a new method to make a trade-off between privacy and utility through the utilization of the uncertain graph approach while modifying the structure of the influential nodes [25]. Also, in the same year, Parchas et al. proposed an uncertain graph-based method in which they reduced and redistributed the number of edges while preserving the underlying structure [26]. In a recent study, Qu et al. developed a privacy preservation technique for publishing a homogeneous network, employing differential privacy uncertainty. Their method involves dividing the network into the community, bridging subgraphs, and applying differential privacy and random perturbation techniques for encoding rearrangements and decoding the uncertain subgraph [27]. The discussed studies on the uncertain graphs approach are summarized in Table 2.

c. Generalization/Clustering Approach

Generalization/Clustering is a well-known approach, which groups the nodes and edges into super-nodes and super-edges to properly anonymize users' information. In recent times, this approach has attracted significant attention within the realm of OSN anonymization methods. An early attempt in this category was undertaken by Zheleva et al., who introduced five clustering-based social network anonymization techniques, along with the information loss criteria for the problem of sensitive links disclosure [8]. A year later, an influential study based on the Generalization/Clustering approach was carried out by Campan et al., in which the authors proposed a greedy anonymization algorithm called the social network greedy anonymization algorithm (SANGreea) [6]. This method tries to protect social networks' structural information (neighborhood) by satisfying the structural k-anonymity. The methodology involves transforming a social network graph into an anonymous version in a way that the structural property of each node remains unidentifiable with at least k-1 other nodes through clustering nodes and edges into super-nodes and super-edges. In addition, the authors tried to simultaneously preserve the utility of the anonymous social network by maintaining the users' attributes and structural information by reducing the generalization information loss (GIL) and SIL criteria. It is important to note that SANGreea relies on greedy algorithms, which means it may not consistently achieve the optimal global solution, as greedy algorithms tend to make confident choices early on without considering the potential superior choices that may arise in the future. In another study, Campan et al. tried to improve the structural k-anonymity model by proposing a greedy clustering-based algorithm that implements the p-sensitive k-anonymity model in social networks [28]. In 2011, Tassa et al. introduced Sequential Clustering (SC) as a method to improve the clustering phase of SaNGreeA, and it offers superior optimization compared to SaNGreeA due to its ability to modify clusters [29]. Later, in 2012, Sihag developed a clustering-based method and applied it to undirected and unlabeled graphs, intending to maximize the preservation of privacy in social networks [9]. The author used the GA to achieve the structural k-anonymity model by grouping the nodes into clusters with at least k nodes. A notable drawback of this study was its disregard for the utility of the anonymized social network, as the primary focus was solely on increasing the privacy level by maximizing the NSIL of the anonymized network. Also, this method proved impractical for large-scale networks due to the reliance on a matrix-based solution representation,

Table 2
An Overview of the Studies on Uncertain Graphs Approach.

Approach	Year	Authors
2012	Boldi et al. [5]	Introduced the concept of injecting uncertainty in social graphs.
2018	Yan et al. [25]	Presented a new method to make a trade-off between privacy and utility through the utilization of the uncertain graph approach while modifying the structure of the influential nodes.
2018	Parchas et al. [26]	Proposed an uncertain graph-based method in which they reduced and redistributed the number of edges while preserving the underlying structure.
2023	Qu et al. [27]	Introduced a method for homogeneous network publishing using differential privacy uncertainty.

which negatively affected the algorithm's process time and required hardware resources. In 2015, Campan et al. published a paper that compared the SANGreea with the k-degree method regarding the community preserving criterion [30]. They indicated that the k-degree method performs better in preserving the communities, but SANGreea implements a much stronger privacy model than the k-degree method. During the same year, Casas-Roma et al. developed the k-shell method based on the Generalization/Clustering approach for the undirected and unlabeled networks [31]. In 2019, Ros-Martín et al. presented a graph anonymization method named scalable non-deterministic clustering-based k-anonymization (SCAN), which uses a greedy and non-deterministic algorithm to achieve k-anonymity on labeled and undirected networks [7]. Their proposed method is inspired by the well-known SANGreea, improving both SIL and GIL criteria by introducing a new information loss score for the relationships (ILR) and an information loss score for the attributes (ILA). During the same year, Mohapatra et al. presented a new clustering-based method called Modified Anatomy-based Clustering (MAC). Unlike previous works such as SaNGreeA and SC, which heavily rely on generalization-based clustering, MAC focuses on preserving data originality. As a result, this technique achieves lower information loss and higher utility than previous methods [32]. In a 2020 study, Yazdanjue et al. proposed four OSN privacy-preserving methods based on the structural k-anonymity using GA, binary particle swarm optimization (BPSO), and two hybrid methods named GAPSO and PSOGA *meta*-heuristic algorithms [10]. The aim was to optimize the clustering process in the structural k-anonymity model by finding the optimal clusters with the minimum SIL and ILR values, leading to a high-utility anonymized social network. The authors developed a modification technique to ensure that the matrix-based solution representation aligns with the structural k-anonymity constraint, which requires each super-node to contain at least k nodes. However, this modification technique may potentially impact the algorithm's performance and increase its processing time. During the same year, Langari et al. introduced an anonymization approach for social networks, combining the k-member Fuzzy Clustering and Firefly Algorithm (KFCFA) to protect the users' attributes and network structural data from disclosure while minimizing information loss [33]. In 2021, Gangarde et al. proposed a method for anonymizing social networks using a multi-graph-properties-based clustering approach [34]. Their method aims to ensure the privacy of edges, nodes, and user attributes within social networks while fulfilling k-anonymity, l-diversity, and t-closeness models in each cluster. It implements a data normalization algorithm to improve the quality of raw OSN data, followed by k-means clustering based on multiple graph properties to achieve an optimized anonymous network. A year later, in 2022, Kadhiwala et al. introduced an anatomy-based clustering approach inspired by the MAC method [35]. Their method centered around addressing attribute disclosure in clustering formation as well as protecting against identity disclosure during the publishing phase of collaborative social network data through the utilization of the m-privacy model. It is worth noting that the authors used the SIL criterion to measure the information loss resulting from their clustering approach. Also, in 2023, Wang et al. suggetsed the Graph-Clustering Anonymity with Fused Distance-Attributes (GCA-DA) algorithm, a novel method to enhance social network data privacy by clustering nodes based on integrated distance and attribute similarities [36]. This technique anonymizes clusters' nodes to prevent attacks using background knowledge, while preserves data utility. Their results demonstrated the effectiveness in improving clustering quality and reducing information loss. In a recent study inspired by their previous work in 2021, Gangarde et al. proposed an improved clustering mechanism to protect the privacy of link, node, and user attributes while enhancing utility [37]. In this regard, they employed a threshold technique to modify the existing fixed k-means clustering used in their previous study. This modification enabled successful k-anonymization of clusters while optimizing them to satisfy the requirements of k-anonymity, t-closeness, and l-diversity. Table 3 summarizes the mentioned studies in the generalization/clustering category.

In summary, while the aforementioned studies have brought insight into OSN privacy-preserving, few of them have explored the use of evolutionary algorithms and *meta*-heuristics in their models to achieve a balance between both privacy and utility criteria. As mentioned, the clustering step of the Generalization/Clustering approach, used to satisfy the structural k-anonymity model, is an NP-

Table 3
An Overview of the Studies on the Generalization/Clustering Approach.

Year	Authors	Key Contribution
2007	Zheleva et al. [8]	Introduced five clustering-based social network anonymization techniques.
2008	Campan et al. [6]	Proposed SANGreea as a greedy anonymization algorithm satisfying structural k-anonymity.
2010	Campan et al. [28]	Improved the structural k-anonymity model using a greedy clustering-based algorithm to implement the p-sensitive k-anonymity model.
2011	Tassa et al. [29]	Introduced Sequential Clustering (SC). It offers better optimization compared to SaNGreeA due to its modifiable cluster capability.
2012	Sihag [9]	Proposed a GA-based clustering method for undirected and unlabeled graphs to achieve maximum privacy.
2015	Campan et al. [30]	Compared SANGreea with the k-degree method in terms of community preservation.
2015	Casas-Roma et al. [31]	Developed the k-shell method based on the Generalization/Clustering approach.
2019	Ros-Martín et al. [7]	Presented a method improving both SIL and GIL criteria using SCAN, a greedy and non-deterministic algorithm.
2019	Mohapatra et al. [32]	Introduced MAC, focusing on preserving data originality through Modified Anatomy-based Clustering.
2020	Yazdanjue et al. [10]	Proposed methods based on structural k-anonymity using GA, BPSO, GAPSO, and PSOGA algorithms.
2020	Langari et al. [33]	Introduced KFCFA, combining k-member Fuzzy Clustering and Firefly Algorithm.
2021	Gangarde et al. [34]	Developed a multi-graph-properties-based clustering approach using the k-means algorithm for anonymizing social networks.
2022	Kadhiwala et al. [35]	Presented a new anatomy-based clustering approach focusing on attribute and identity disclosure protection.
2023	Wang et al. [36]	Presented the GCA-DA algorithm, effectively balancing network data privacy with minimal information loss in clustering.
2023	Gangarde et al. [37]	Proposed a method with an improved k-means clustering mechanism to protect link, node, and user attributes privacy while enhancing utility.

hard problem [6,9,10]; hence, various state-of-the-art *meta*-heuristics can be used to find optimal clusters with minimum SIL, resulting in an improved structural k-anonymous OSN. Nonetheless, many studies in the literature have relied on heuristic and greedy algorithms to find the structural k-anonymous social network. However, these methods have limitations that may lead to suboptimal solutions. For instance, greedy algorithms tend to make early choices and can get stuck in local optima, failing to find the best possible solution. Likewise, although heuristics can obtain near-optimal solutions in a relatively reasonable time, they cannot guarantee the quality of the solution. In contrast, *meta*-heuristic algorithms are formally defined as iterative processes that intelligently explore and exploit the search space using a small initial population size and a feasible number of iterations. Therefore, proposing an efficient *meta*-heuristic algorithm for the structural k-anonymity problem that can simultaneously preserve network privacy and maintain the structural information of the OSN is an active research subject. Therefore, this article presents an enhanced *meta*-heuristic algorithm to solve the structural k-anonymity problem effectively and find high-utility anonymous OSNs. The main contributions of this work can be summarized as follows:

1) As the main contribution of this study, we propose an enhanced DPSO algorithm, called EDPSO, to solve the structural k-anonymity problem effectively and efficiently. Our approach involves the intelligent and dynamic application of effective operators, including Swap, Reversion, Rotate to Left, Rotate to Right, and Insertion. This increases the algorithm's exploration and exploitation capabilities, leading to the discovery of an optimal structural k-anonymous network with a minimum SIL value. To achieve this, we have designed a novel heuristic algorithm that significantly improves the update process of each particle's position vector. This algorithm adaptively tunes the selection probabilities of the operators based on the obtained fitness values of each operator in the current iteration and also the number and the average amount of fitness improvements made by them in previous iterations. This results in a tradeoff between exploration and exploitation capabilities and avoids premature convergence.
2) In addition, two network-specific local search strategies based on the SA and VNS algorithms have been developed to assist the proposed EDPSO algorithm in achieving higher performance. These local searches further improve both the exploration and exploitation of the proposed EDPSO algorithm by preventing it from being trapped in local optima and increase the convergence rate of the EDPSO. As a result, the integration of these local searches assists the EDPSO algorithm in achieving near-optimal solutions with minimum SIL in a reasonable number of iterations.
3) We also utilize a vector-based solution representation for solving the structural k-anonymity problem using *meta*-heuristic algorithms, which mitigates the shortcomings of the matrix-based solution representation used in [9,10]. Unlike the matrix-based approach, our vector-based representation inherently satisfies the structural k-anonymity constraint. Consequently, it eliminates the need to demand validity checks or modification steps as required in the previous methods [9,10]. Hence, by using this vector-based solution representation, the proposed algorithms demonstrate superior exploration and exploitation capabilities compared to the previous studies, and they can reach higher-quality solutions. Furthermore, this representation reduces the number of required numeric values, leading to a relative decrease in the complexity of the algorithm and the requirement for hardware resources during the execution process.
4) Furthermore, a comprehensive set of experiments was conducted on real-world networks to evaluate the performance of the proposed algorithms. They were compared with previous *meta*-heuristic methods (DPSO, PSOGA, and GA) based on fitness value (1-NSIL) for different k values. The results showed that the proposed EDPSO, EDPSOSA, and EDPSOVNS algorithms achieved higher 1-NSIL values, indicating a balance between exploration and exploitation and avoiding premature convergence. Also, comparisons with traditional matrix-based methods demonstrated that the use of vector-based solution representation in EDPSO led to superior exploration, exploitation, higher-quality solutions, and reduced process time for larger and more complex datasets. The effectiveness of SA and VNS-based local searches in improving convergence rates and attaining near-optimal solutions was also assessed, showing that EDPSOVNS and EDPSOSA algorithms achieved higher convergence rates and 1-NSIL values compared to other algorithms. Eventually, the proposed algorithms were compared with Anatomy-based clustering, Enhanced k-means clustering, SCAN, and SANGreea methods, considering 1-NSIL and 1-ILR criteria. The results indicated that the proposed algorithms achieved lower structural information loss while preserving users' privacy.

3. Problem statement

Building upon the principles of k-anonymity, structural k-anonymity is a specialized technique often used in the Generalization/Clustering approach to anonymize the structure of social networks. It primarily focuses on protecting the structural information, which includes the neighborhood relationships and connections of a network's nodes. In a structurally k-anonymous network, the arrangement and properties of connections for each user are generalized or clustered to make them indistinguishable from at least k-1 other users [6,8–10]. The goal of structural k-anonymity is to preserve the privacy of users against attempts to reidentify them based on their structural characteristics within the network. Consequently, an attacker trying to reveal the identity of a user in the network will face significant uncertainty. More specifically, the primary challenge in the Generalization/Clustering approach utilizing structural k-anonymity is to cluster the nodes and edges of the network into super-nodes and super-edges while minimizing the disruption to the original network structure, which is measured through the SIL metric. By keeping the SIL as low as possible, the approach tries to preserve the data utility to the greatest extent while still achieving the desired level of anonymity. However, designing an efficient algorithm for creating a structurally k-anonymous network through Generalization/Clustering remains a challenging and active research area in OSN anonymization. This is because the process of finding optimum clusters in the structural k-anonymity method is proved to be an NP-hard optimization problem, which is often solved using heuristic methods or guided random search techniques like *meta*-heuristic algorithms [6,9,10].

Hence, in the current study, we propose three *meta*-heuristic algorithms (EDPSO, EDPSOVSN, and EDPSOSA) to minimize the SIL criterion while satisfying the specified structural k-anonymity constraint. The structural k-anonymity constraint requires that each super-node contains a minimum of $k \leq n$ nodes, where n represents the total number of nodes in the network. Therefore, the number of super-nodes can be calculated using Equation (1).

$$s = \lfloor n/k \rfloor \tag{1}$$

Here we consider the social network as a simple undirected graph $G = (V,E)$, where V is the set of nodes (representing the users in the network), and $E \subseteq V \times V$ is the set of edges (representing relationships between the users). The set V consists of n nodes ($|V| = n$) and each node (v_{n_id}) is labeled by $n\_id \in [1,\cdots,n]$. In addition, the set E consists of m edges ($|E| = m$) and $e_{v_i,v_j} \subseteq E$ denotes an edge between nodes v_i and v_j. It is important to note that only binary relationships are permitted in this context, and all relationships are unsigned and undirected. Finally, the anonymized version of the original graph is referred to as $\overline{G} = (\overline{V},\overline{E})$. Also, the adjacency matrix (ADM) of a social network is defined by Equation (2), which is a $n \times n$ binary matrix, where the ones indicate the edges between nodes. The elements on the main diagonal of this matrix are zero since the graph is a simple one with no cycles.

$$adm_{i,j} = \begin{cases} 1, if\, i \neq j\, And\, e_{v_i,v_j} \in E \\ 0, Otherwise \end{cases} \tag{2}$$

As previously mentioned, in the Generalization/Clustering approach, the nodes/users in the network will be grouped into distinct clusters called super-nodes. In this study, each of these super-nodes (Cl_{s_id}) is labeled by $s\_id \in [1,\cdots,s]$, and the set of clusters (CL_Set) is defined as in Equation (3).

$$CL\_Set = \bigcup_{s\_id=1}^{s} Cl_{s\_id} = \{Cl_1,\cdots,Cl_s\}; Cl_i \cap Cl_j = \emptyset; i,j \in [1s]; i \neq j \tag{3}$$

4. Proposed method

The suggested *meta*-heuristic method in this study is based on the PSO algorithm, which is a swarm-based optimization algorithm inspired by the behavior of insect swarms, bird flocks, and fish schools in nature [38,39]. In PSO, the optimization process starts with the generation of an initial swarm, and each particle of the swarm has a position vector and a velocity vector. The position vector denotes the current solution of the corresponding particle, and the velocity vector is used to adjust the particle's position in the direction of the optimal solution's position. Although the PSO algorithm was initially proposed to solve continuous optimization problems, several versions of this algorithm have been suggested to solve discrete and binary problems. The advantages and disadvantages of the PSO algorithm are discussed in the literature. One notable advantage of PSO is its straightforward implementation since it uses simple mathematical operators and has a low number of parameters to adjust. Besides, this algorithm converges faster than other population-based optimization algorithms, such as GA. However, its main drawback is its tendency for premature convergence, which is a result of getting stuck in local optima.

In 2015, Cai et al. recognized the need to develop a discrete PSO (DPSO) algorithm, addressing the drawbacks of original PSO in discrete optimization problems [11]. However, their proposed algorithm still suffers from premature convergence and inadequate exploration and exploitation capabilities, particularly in large-scale real-world social networks. Therefore, as the main contribution of the current study, EDPSO is proposed to overcome the mentioned issues. To this end, we have developed a novel heuristic algorithm that intelligently applies problem-specific operators involving Swap, Reversion, Insertion, Rotate to Left, and Rotate to Right to the particles. This algorithm updates the position vector of each particle more effectively by dynamically applying the mentioned operators. More specifically, the proposed algorithm adaptively tunes the selection probabilities of the operators based on the obtained fitness value in the current iteration and the performance history of each operator (i.e., the number and the average amount of fitness improvements of them in the previous iterations). This heuristic approach achieves a balanced tradeoff between exploration and exploitation capabilities while avoiding premature convergence.

Additionally, we have reinforced the proposed EDPSO with two auxiliary network-specific local search strategies based on the SA and VNS algorithms, called EDPSOSA and EDPSOVNS, respectively. These local search algorithms serve auxiliary roles and enhance the exploration and exploitation of the proposed EDPSO algorithm. They prevent the algorithm from being trapped in a local optimum and enable it to achieve a structural k-anonymous network with the minimum NSIL. Moreover, these strategies increase the convergence rate of the EDPSO, leading it toward the best possible solution in fewer iterations. A comprehensive discussion of the proposed EDPSO algorithm and its combinations with the local search strategies (EDPSOSA and EDPSOVNS) will be provided in sections 4.1 and 4.2.

4.1. EDPSO Framework

Algorithm 1 presents the pseudocode of the EDPSO algorithm, which serves as the global search component of our suggested social network privacy-preserving method, aimed at maximizing the 1-NSIL value. The details of the initialization process for each particle (*Random_Based_Initialization*), velocity updating rule, novel position updating heuristic (*Adaptive_Heuristic_Position_Updating*), and

developed local search strategies (*Local_Search*) are discussed in this section.

Algorithm 1. *"Framework of EDPSO Algorithm"*

1. Inputs:
   The adjacency matrix of the graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$: $\boldsymbol{ADM}$
   Swarm Size: $\boldsymbol{n_{Pop}}$
   Current Iteration:$\boldsymbol{it}$
   Maximum Number of Iterations:$\boldsymbol{it_{Max}}$
   Inertia Weight:$\boldsymbol{IW}$
   Learning Factors: $\boldsymbol{c_1}$, $\boldsymbol{c_2}$
   Minimum Cluster Size:$\boldsymbol{k}$
Procedure:
2. Apply Random_Based_Initialization ($\boldsymbol{n_{Pop}}$) to initialize $\boldsymbol{Swarm}$ with random values
3. For each Particle $\boldsymbol{P_i}$ in $\boldsymbol{Swarm}$ do
4. Calculate Particle Fitness: $\boldsymbol{P_i.Ft} \leftarrow 1 - \text{NSIL}\ (\overrightarrow{X_i^{t-1}})$
5. $\overrightarrow{\boldsymbol{Pbest_i}} \leftarrow \overrightarrow{X_i^{t-1}}$
6. End For
7. $\overrightarrow{\boldsymbol{Gbest}} \leftarrow$ Find the best $\overrightarrow{\boldsymbol{Pbest_i}}$
8. While $\boldsymbol{it} \leq \boldsymbol{it_{Max}}$ do
9. For each Particle $\boldsymbol{P_i}$ in $\boldsymbol{Swarm}$ do
10. Update Velocity: $\overrightarrow{\boldsymbol{Vel_i^t}} \leftarrow$ Update $\overrightarrow{\boldsymbol{Vel_i^{t-1}}}$
11. Update Position: $\overrightarrow{X_i^t} \leftarrow$ Adaptive_Heuristic_Position_Updating $(\overrightarrow{X_i^{t-1}})$
12 $\boldsymbol{P_i.Ft} \leftarrow 1 - \text{NSIL}\ (\overrightarrow{X_i^t})$
13. If $\boldsymbol{P_i.Ft} > 1 - \text{NSIL}\ (\overrightarrow{\boldsymbol{Pbest_i}})$ then
14. Update $\overrightarrow{\boldsymbol{Pbest_i}}$
15. If $1 - \text{NSIL}\ (\overrightarrow{\boldsymbol{Pbest_i}}) > 1 - \text{NSIL}\ (\overrightarrow{\boldsymbol{Gbest}})$ then
16. Update $\overrightarrow{\boldsymbol{Gbest}}$
17. End If
18. End If
19. End For
20. Update Global best solution using local search: $\overrightarrow{\boldsymbol{Gbest}} \leftarrow$ Local_Search $(\overrightarrow{\boldsymbol{Gbest}}, \overrightarrow{\boldsymbol{Vel_i^t}})$
21. Increment $\boldsymbol{it}$
22. End While
23. Output: The $\overrightarrow{\boldsymbol{Gbest}}$ as the anonymized graph $\overline{\boldsymbol{G}}$, whose utility as well as privacy, is maximum.

4.1.1. Solution representation

In this section, we will outline the solution representation employed by the proposed EDPSO algorithm. Unlike previous matrix-based solution representations, the proposed EDPSO algorithm adopts a vector-based solution representation that inherently satisfies the structural k-anonymity constraint, eliminating the need for exhaustive validity checks or modification steps required in prior approaches [9,10]. This feature allows the proposed EDPSO to achieve superior exploration and exploitation capabilities, resulting in solutions of higher quality and mitigating premature convergence. Also, this representation leads to the use of a smaller number of numeric values compared to matrix-based solution representations, which in turn relatively decreases the hardware resources required for the execution process.

The EDPSO algorithm initiates by generating a population of feasible random solutions, referred to as a swarm, that involves p particles (solutions). Each of these particles ($P_i : i \in [1, \cdots, p]$) comprises position and velocity vectors. As shown in Equation (4), the position vector for the i^{th} particle at time t is represented as $\overrightarrow{X_i^t}$, in which the value of its elements (x_{n_id}) correspond to the label of the super-node (s_id) where the node with the label n_id belongs. Similarly, in Equation (5), the velocity vector is represented as $\overrightarrow{Vel_i^t}$, in which the value of its elements (vel_{n_id}) shows the velocity of the node with n_id label. It is worth noting that the $\overrightarrow{Vel_i^t}$ decides whether the current position of the particle should be updated or not. If $vel_{n\_id} = 1$, the corresponding value in the position vector (x_{n_id}) will be modified; otherwise, if $vel_{n\_id} = 0$, the corresponding position value remains unchanged.

For instance, if the network size is $n = 7$ and the constraint of the structural k-anonymity is $k = 2$, then based on Equation (1), the number of clusters will be $s = 3$. Subsequently, Equation (6) shows a presumed position vector for the first particle of the swarm ($\overrightarrow{X_1^t}$), such that the nodes with labels $n\_id = 2, n\_id = 5$ are clustered in the first super-node with $s\_id = 1$ (i.e., $x_2 = 1, x_5 = 1$), the nodes with labels $n\_id = 4, n\_id = 7$ are clustered in the second super-node with $s\_id = 2$ (i.e., $x_4 = 2, x_7 = 2$), and finally the nodes with labels $n\_id = 1, n\_id = 3, n\_id = 6$ are clustered in the third super-node with $s\_id = 3$ (i.e., $x_1 = 3, x_3 = 3, x_6 = 3$) in the targeted social network.

$$\overrightarrow{X_i^t} = [x_1, \cdots, x_{n\_id}, \cdots, x_n] \tag{4}$$

$$\overrightarrow{Vel_i^t} = [vel_1, \cdots, vel_{n\_id}, \cdots, vel_n] \tag{5}$$

$$\overrightarrow{X_1^t} = [3, 1, 3, 2, 1, 3, 2] \tag{6}$$

Fig. 1(a) and 1(b) illustrate the $\overrightarrow{X_1^t}$ solution vector related to Equation (6) and its graphical representation, respectively.

According to the concept of the structural k-anonymity constraint, the number of nodes with similar *s_id* in each position vector (i.e., the number of nodes in each super-node) must be greater than or equal to k. The mentioned condition for $\overrightarrow{X_i^t}$ is expressed in Equation (7).

$$\forall s\_id \in [1, \cdots, s] : \sum_{n\_id=1}^{n} if(x_{n\_id} == s\_id) \geq k \tag{7}$$

To initialize the position vector of each particle ($\overrightarrow{X_i^t}$) in the initial swarm of the EDPSO, the random initialization method is used. This method generates position vectors with random numbers, which results in maintaining the randomness of the algorithm. Simultaneously, the velocity vectors (Vel_i^0) are set to the zeroes vector at the initial phase. As mentioned above, the vector-based representation guarantees that all generated solutions throughout the EDPSO's execution process adhere to the structural k-anonymity constraint. This is because the Swap, Reversion, Insertion, Rotate to Left, and Rotate to Right operators, applied to the vector-based solutions, solely modify the arrangement of nodes among different clusters without altering the number of nodes within the clusters.

4.1.2. Fitness function evaluation

As discussed in the literature review section, many state-of-the-art studies use the well-known SIL criterion to assess the amount of structural information loss in the networks resulting from the application of the Generalization/Clustering approach. The SIL serves as a measure that quantifies the probability of error in reconstructing the original structure from the network, which is anonymized by grouping nodes and edges into super-nodes and super-edges. In the current study, we adopt the same SIL criterion as the objective function to find the best structural k-anonymous network with the minimum information loss. The following sub-section presents the underlying concepts of SIL as the chosen objective function.

a) Structural Information Loss

The primary objective of the clustering method introduced by Campan et al. was to achieve a high level of structural k-anonymity that any two users within each cluster could not be distinguished from each other based on their identities and relationships [6]. To accomplish this, they presented an edge generalization technique consisting of two components: edge intra-cluster and edge inter-cluster generalization. The edge intra-cluster component generalizes the edges within each super-node, while the edge inter-cluster component generalizes the edges between super-nodes.

However, this generalization technique comes with certain consequences, as it results in the loss of some structural information from the original graph. To clarify, some structural information will be lost in each of the edge intra-cluster and edge inter-cluster generalization components. Consequently, the introduced SIL criterion equals the sum of intra-cluster structural information loss (*intraSIL*) and the inter-cluster structural information loss (*interSIL*). The *intraSIL* measures the probability of incorrectly labeling a pair of nodes within a cluster as either connected or unconnected, as defined in Equation (8). On the other hand, the *interSIL*, shown in Equation (9), calculates the probability of wrongly labeling a pair of nodes, where one node belongs to one cluster, and the other belongs to a different cluster, as either connected or unconnected.

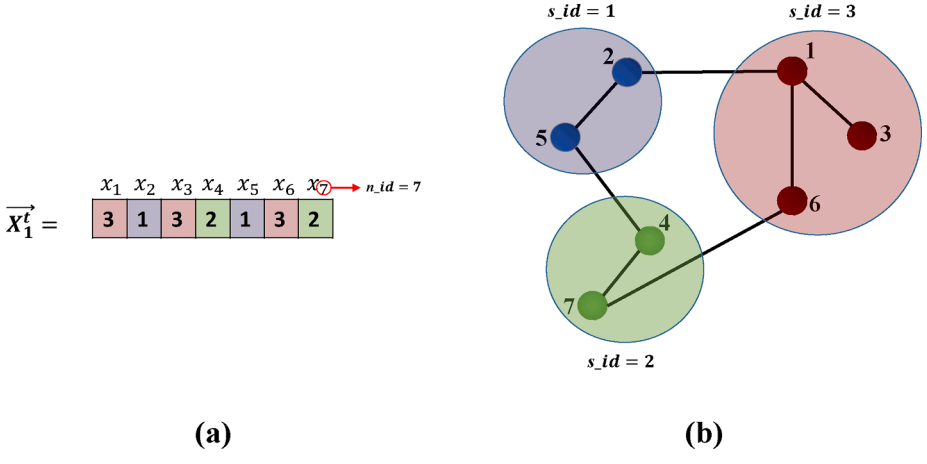

Fig. 1. The Graphical Representation of the $\overrightarrow{X_1^t}$ Solution Vector.

$$intraSIL(Cl_i) = \left(\left(\binom{|Cl_i|}{2} - |E_{Cl_i}| \right) . \frac{|E_{Cl_i}|}{\binom{|Cl_i|}{2}} \right) + \left(|E_{Cl_i}| . \left(1 - |E_{Cl_i}| / \binom{|Cl_i|}{2} \right) \right) = 2.|E_{Cl_i}| . \left(1 - |E_{Cl_i}| / \binom{|Cl_i|}{2} \right); i \in [1, \cdots, s_{id}, \cdots, s] \tag{8}$$

$$interSIL(Cl_i, Cl_j) = \left(\left(|Cl_i| . |Cl_j| - |E_{Cl_i,Cl_j}| \right) . \frac{|E_{Cl_i,Cl_j}|}{|Cl_i| . |Cl_j|} \right) + \left(|E_{Cl_i,Cl_j}| . \left(1 - \frac{|E_{Cl_i,Cl_j}|}{|Cl_i| . |Cl_j|} \right) \right) = 2.|E_{Cl_i,Cl_j}| . \left(1 - \frac{|E_{Cl_i,Cl_j}|}{|Cl_i| . |Cl_j|} \right); i,j \in [1, \cdots, s_{id}, \cdots, s], i \neq j \tag{9}$$

In this sense, if the graph G is converted to an anonymous graph $\overline{G}$ with the cluster set of $CL\_Set = \{Cl_1, \cdots, Cl_{s\_id}, \cdots, Cl_s\}$, the total SIL can be calculated as in Equation (10).

$$SIL(G, CL\_Set) = \sum_{i=1}^{s} (intraSIL(Cl_i)) + \sum_{i=1}^{s} \sum_{j=i+1}^{s} (interSIL(Cl_i, Cl_j)); i,j \in [1, \cdots, s_{id}, \cdots, s], i \neq j \tag{10}$$

The maximum value of the total structural information loss,*MAXSIL*, can be calculated by Equation (11).

$$MAXSIL(G, CL\_Set) = \sum_{i=1}^{s} \left(\frac{|Cl_i| \bullet (|Cl_i| - 1)}{4} \right) + \sum_{i=1}^{s} \sum_{j=i+1}^{s} \left(\frac{|Cl_i| \bullet |Cl_j|}{2} \right) = \frac{n \times (n-1)}{4}; i,j \in [1, \cdots, s_{id}, \cdots, s], i \neq j \tag{11}$$

It is evident that when the SIL value is maximum, the adversary will face the highest error probability during the reconstruction of the original network. However, in this situation, the utility of the anonymized network is minimum since the anonymization algorithm shuffles the structure of the social network as much as possible. Consequently, it is essential to make a trade-off between protecting the structural data against disclosure attacks and maintaining the utility of the anonymized social network for third-party companies.

The value of SIL can be normalized through the division of SIL by the *MAXSIL*, as demonstrated in Equation (12), resulting in NSIL. The NSIL ranges from 0 to 1, with 0 indicating no loss of information (for a graph with no edges or a complete graph) and 1 indicating maximum loss of information. It is important to note that the less NSIL is, the higher the utility of the anonymous network will be. Hence, in this study, we use NSIL to evaluate the total structural information loss of the anonymized network represented by each particle in the EDPSO algorithm. To obtain the anonymous network with the highest possible utility, we aim to maximize 1-NSIL, shown in Equation (13), which serves as the fitness function.

$$NSIL(G, CL\_Set) = \frac{SIL(G, CL\_Set)}{MAXSIL(G, CL\_Set)} = \frac{SIL(G, CL\_Set)}{\frac{n \times (n-1)}{4}} \tag{12}$$

$$Fitness(G, CL\_Set) = 1 - NSIL \tag{13}$$

For example, the fitness value (1-NSIL) of the network shown in Fig. 1 (b) can be calculated based on the structural information presented in Table 4, and it is equal to 0.4127.

4.1.3. Update rules

To achieve an optimal structural k-anonymous social network, the 1-NSIL should be maximized by the suggested EDPSO algorithm. In this regard, each particle's velocity and position vectors in the swarm should be updated in the discrete form, which is explained thoroughly in the following sections.

1) The Velocity Vector Updating Mechanism

The velocity of each particle indicates the distance traveled by the particle at each iteration and is updated in the next iteration based on its personal best experience ($\overrightarrow{Pbest}$), and the best experience of the whole swarm ($\overrightarrow{Gbest}$). In the current study, we used the velocity updating mechanism introduced in [11] to update the velocity vector of each particle, which is defined by Equation (14).

$$\overrightarrow{Vel_i^t} \leftarrow \Phi\left(\left[IW * \overrightarrow{Vel_i^{t-1}} \right] + [c_1 r_1 * \left(\overrightarrow{Pbest_i} \oplus \overrightarrow{X_i^{t-1}} \right)] + [c_2 r_2 * \left(\overrightarrow{Gbest} \oplus \overrightarrow{X_i^{t-1}} \right)] \right) \tag{14}$$

Where IW is the inertia weight, $\overrightarrow{Pbest_i}$ indicates the personal best position of i^{th} particle, and $\overrightarrow{Gbest}$ represents the global best position of the swarm. Besides, parameters c_1 and c_2 are acceleration constants for the cognitive and social components, respectively, and r_1 and r_2 denote the two generated random numbers within the range $[0, 1]$. Furthermore, the operator "$\oplus$" in Equation (14) is defined as the

Table 4
Number of Nodes, Intra-edges, and Inter-edges of the Super-nodes in the Sample Network.

Variable	$\lvert Cl_1 \rvert$	$\lvert Cl_2 \rvert$	$\lvert Cl_3 \rvert$	$\lvert E_{Cl_1} \rvert$	$\lvert E_{Cl_2} \rvert$	$\lvert E_{Cl_3} \rvert$	$\lvert E_{Cl1,Cl2} \rvert$	$\lvert E_{Cl1,Cl3} \rvert$	$\lvert E_{Cl2,Cl3} \rvert$
Value	2	2	3	1	1	2	1	1	1

logical XOR operator. In the case of the structural k-anonymity problem, the mentioned "⊕" operator demonstrates the structural difference between two k-anonymous networks since each solution is considered as an anonymized network with different structural features.

Subsequently, according to Equation (14), the previous velocity vector of the i^{th} particle ($\overrightarrow{Vel_i^{t-1}}$), and the result of two XOR operations will be summed together, and the result of the mentioned summation process will be passed to the Φ function, which is defined subsequently.

Given the $\overrightarrow{Z} = [z_1, z_2, \cdots, z_n]$ vector, the $\Phi\left(\overrightarrow{Z}\right)$ will be represented as $\Phi\left(\overrightarrow{Z}\right) = [\varphi(z_1), \varphi(z_2) \ldots, \varphi(z_n)]$, where $\varphi(z_i)$ $(i = 1, 2, \ldots, n)$ is defined as a threshold function shown in Equation (15).

$$\forall z_i \in \overrightarrow{Z} : \varphi(z_i) = \begin{cases} 0 z_i < 1; \\ 1 z_i \geq 1; \end{cases} \tag{15}$$

Overall, for each particle in the swarm, there will be an updated velocity vector ($\overrightarrow{Vel_i^t}$) including '0' or '1' values associated with the elements (nodes) in the position vector ($\overrightarrow{X_i^{t-1}}$) of that particle. The nodes whose corresponding element in the velocity vector is equal to '0', have a higher potential to move in the direction of the global optimum, and hence their current positions are better to be reserved. On the contrary, the '1' elements in the velocity vector reflect the candidate nodes whose movement directions are deemed unsuitable, and they need to be updated based on the proposed adaptive heuristic updating mechanism discussed in the following section.

2) The Proposed Position Vector Updating Mechanism

The whole framework of the proposed heuristic mechanism for position vector updating is depicted in Fig. 2.

In this stage, the position of the particles will be updated in accordance with the updated velocity vector using Equation (16).

$$\overrightarrow{X_i^t} \leftarrow \overrightarrow{X_i^{t-1}} \circledast \overrightarrow{Vel_i^t} \tag{16}$$

where the operator "⊛" is a heuristic updating mechanism that plays a significant role in guiding the particles to the promising regions.

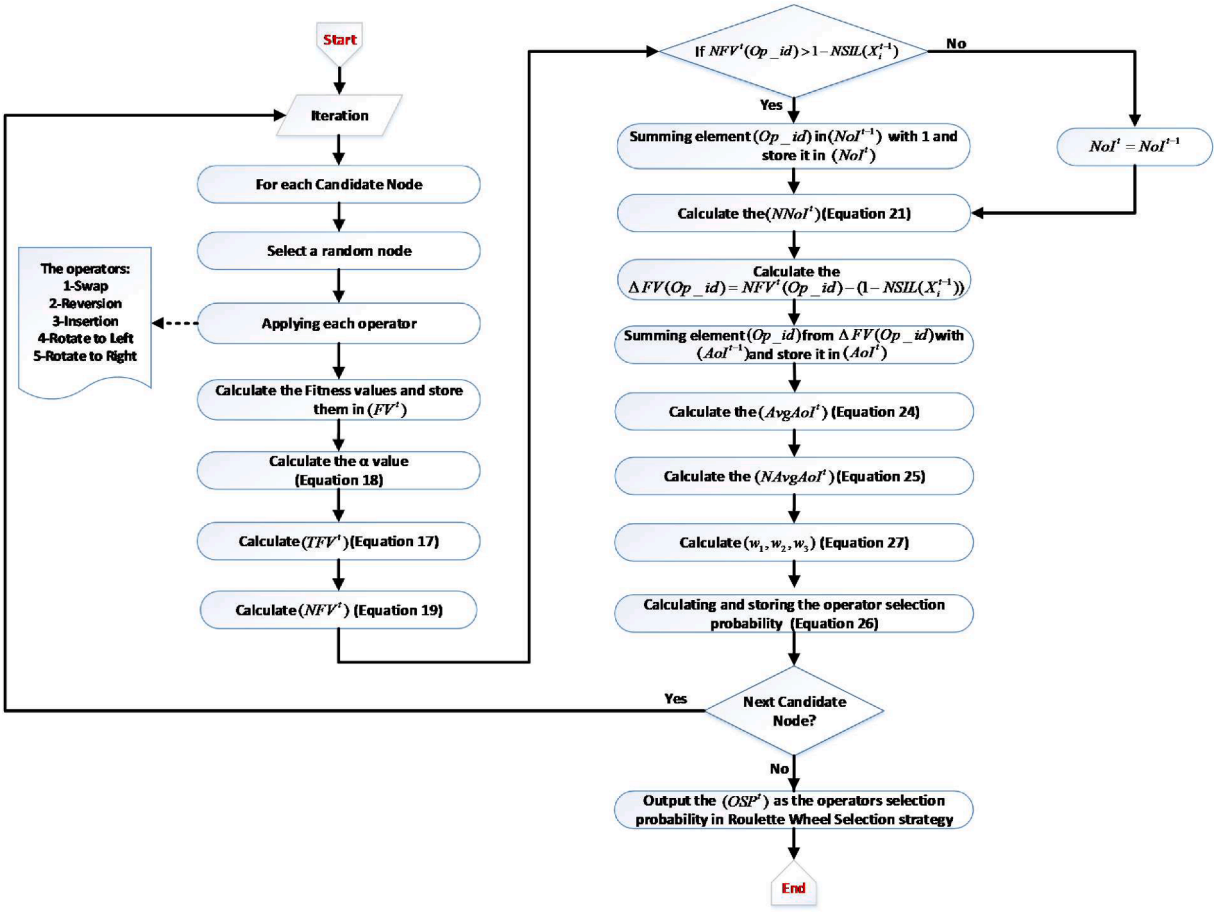

Fig. 2. The Proposed Adaptive Heuristic Position Updating Mechanism.

The process of updating the particle position through the mentioned heuristic is described in the following seven steps.

1- In the first step, for each candidate node *cand_node* (i.e., elements in the particle whose corresponding element in the updated velocity vector ($\overrightarrow{Vel_i^t}$) is 1), another node from the same particle should be selected randomly (*r_node*).

2- Next, five updating operators comprising Swap, Reversion, Insertion, Rotate to Left, and Rotate to Right are applied to the two selected nodes in the position vector of i th particle, specified in the previous step. These operators are defined as follows.

- The $Swap\left(\overrightarrow{X_i^{t-1}}, cn\_id, rn\_id\right)$ operator is a function that swaps the *cand_node* (x_{cn_id}) with the *r_node* (x_{rn_id}) in the position vector of i th particle.
- The $Rev\left(\overrightarrow{X_i^{t-1}}, cn\_id, rn\_id\right)$ operator (Reversion) is a function that reverses the sequence of the elements which are between the *cand_node* (x_{cn_id}) and the *r_node* (x_{rn_id}) in the position vector of i th particle.
- The $Ins\left(\overrightarrow{X_i^{t-1}}, cn\_id, rn\_id\right)$ operator (Insertion) is a function that removes the *r_node* (x_{rn_id}) and inserts it after the *cand_node* (x_{cn_id}) in the position vector of i th particle.
- The $RtoL\left(\overrightarrow{X_i^{t-1}}, cn\_id, rn\_id\right)$ operator (Rotate to Left) rotates the sequence between *cand_node* (x_{cn_id}) and the *r_node* (x_{rn_id}) one step to the left in the position vector of i th particle.
- Similarly, the $RtoR\left(\overrightarrow{X_i^{t-1}}, cn\_id, rn\_id\right)$ operator (Rotate to Right) rotates the sequence between *cand_node* (x_{cn_id}) and the *r_node* (x_{rn_id}) one step to the right in the position vector of i th particle.

It is worth mentioning that these operators satisfy the constraint of k-anonymity, which is mentioned in Equation (7) inherently.

3- In the third step, three input vectors of size 1×5 are defined for the i th particle at time t, comprising the fitness value vector ($\overrightarrow{FV^t}$), number of improvement vector ($\overrightarrow{NOI^t}$), and the amount of improvement vector ($\overrightarrow{AOI^t}$). As depicted in Fig. 3, each element in these vectors is in reference to one of the five operators described in the previous step and is indexed using $op\_id \in [1, 2, 3, 4, 5]$. Moreover, the initial values of these vectors are set to zeros.

4- In this step, the fitness value of the updated position vectors obtained from each operator is calculated based on the 1-NSIL fitness function and is inserted in the index related to that operator in the $\overrightarrow{FV^t}$. Since the calculated fitness values are usually close together, it is preferable to increase the differences among these values so that the operator that leads to a higher fitness value has a greater chance of being selected. Hence, the exponential transformation function shown in Equation (17) is applied to the fitness values stored in $\overrightarrow{FV^t}$ and the outputs are stored in the transformed fitness value vector ($\overrightarrow{TFV^t}$).

$$\overrightarrow{TFV^t}(op\_id) = \left[\overrightarrow{FV^t}(op\_id)\right]^{\alpha} \tag{17}$$

Where α is defined as in Equation (18).

$$\alpha = \frac{\alpha_{max} - \alpha_{min}}{it_{Max} - 1} \times (it - 1) + \alpha_{min} \tag{18}$$

In Equation (18),α_{min}, α_{max}, it, and it_{Max} indicate the minimum value of α, the maximum value of α, the current iteration number, and the maximum iteration number of the EDPSO algorithm, respectively. It is evident that α increases linearly from α_{min} to α_{max}, as the EDPSO proceeds from $it = 1$ to $it = it_{Max}$. In this study, the experimental results show that the best values for α_{min} and α_{max} are 1 and it_{Max}, respectively. The gradual rise of α leads to the adaptive increase in the scale difference between transformed fitness values, which is a strategy to raise the selection probability of the most effective operators as the EDPSO approaches higher iterations.

Eventually, the $\overrightarrow{TFV^t}$ values are normalized using Equation (19), and the output values are stored in the normalized fitness value vector ($\overrightarrow{NFV^t}$).

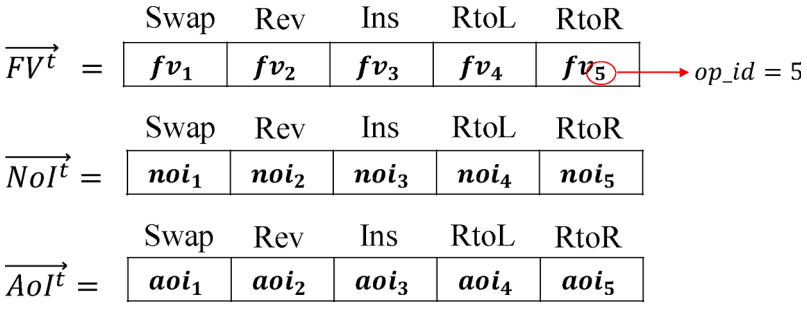

Fig. 3. The Input Vectors for the i th Particle in Time..t

$$\overrightarrow{NFV^t}(op\_id) = \frac{\overrightarrow{TFV^t}(op\_id)}{\sum_{op\_id=1}^{5}\overrightarrow{TFV^t}(op\_id)} \tag{19}$$

5- Next, the fitness values of the updated position vectors ($\overrightarrow{X_i^t}$), stored in $\overrightarrow{FV^t}$, are compared elementwise with the fitness value of the $\overrightarrow{X_i^{t-1}}$. As shown in Equation (20), if the fitness value of the updated position vector is greater than the fitness value of the $\overrightarrow{X_i^{t-1}}$, the corresponding updating operator in the $\overrightarrow{NoI^{t-1}}$ is incremented by '1', and the result is stored in $\overrightarrow{NoI^t}$.

$$\overrightarrow{NoI^t} = \left[\overrightarrow{FV^t}. > \left(1 - NSIL\left(\overrightarrow{X_i^{t-1}}\right)\right)\right]. + \overrightarrow{NoI^{t-1}} \tag{20}$$

Subsequently, the values of the $\overrightarrow{NoI^t}$ are normalized according to Equation (21) and are stored in $\overrightarrow{NNoI^t}$.

$$\overrightarrow{NNoI^t}(op\_id) = \frac{\overrightarrow{NoI^t}(op\_id)}{\sum_{i=1}^{5}\overrightarrow{NoI^t}(op\_id)} \tag{21}$$

6- Following step 5, we subtract each of the fitness values stored in $\overrightarrow{FV^t}$ from the fitness value of $\overrightarrow{X_i^{t-1}}$ and store it in $\overrightarrow{Z}$. Next, the function $\boldsymbol{L}$ is applied to the result of the subtraction so that if the $\overrightarrow{FV^t}(op\_id)$ is greater than $1 - NSIL\left(\overrightarrow{X_i^{t-1}}\right)$, the function will return its input value as the output; otherwise, it will return zero as the output. The general form of function $\boldsymbol{L}$ can be defined as $\boldsymbol{L}\left(\overrightarrow{Z}\right) = [l(z_1), l(z_2)..., l(z_n)]$ where $l(z_i)$ ($i \in \{1, 2, ..., n\}$) is expressed as shown in Equation (22).

$$\forall z_i \in \overrightarrow{Z} : l(z_i) = \begin{cases} 0 z_i \leq 0; \\ z_i z_i > 0; \end{cases} \tag{22}$$

Then, the output values of function $\boldsymbol{L}$ will be added elementwise to the numbers that already exist in the $\overrightarrow{AoI^{t-1}}$, and the results are stored in $\overrightarrow{AoI^t}$. The above-explained procedure can be summarized as represented in Equation (23).

$$\overrightarrow{AoI^t} = \boldsymbol{L}\left(\overrightarrow{FV^t}. - \left(1 - NSIL\left(\overrightarrow{X_i^{t-1}}\right)\right)\right). + \overrightarrow{AoI^{t-1}} \tag{23}$$

Further to this step, as shown in Equation (24), the average amount of improvement ($\overrightarrow{AvgAoI^t}$) is attained by dividing each of the values in $\overrightarrow{AoI^t}$ by its corresponding number of improvements ($\overrightarrow{NoI^t}$) achieved in step 5.

$$\overrightarrow{AvgAoI^t} = \overrightarrow{AoI^t}./\overrightarrow{NoI^t} \tag{24}$$

After all, the values of the $\overrightarrow{AvgAoI^t}$ are normalized according to Equation (25), and results are stored in $\overrightarrow{NAvgAoI^t}$.

$$\overrightarrow{NAvgAoI^t}(op\_id) = \frac{\overrightarrow{AvgAoI^t}(op\_id)}{\sum_{op\_id=1}^{5}\overrightarrow{AvgAoI^t}(op\_id)} \tag{25}$$

7- As the final step in the proposed adaptive heuristic updating mechanism, the selection probability of each operator in the roulette wheel selection strategy is calculated using Equation (26) and is stored in the operator selection probability vector ($\overrightarrow{OSP^t}$).

$$\overrightarrow{OSP^t}(op\_id) = w_1. \times \overrightarrow{NFV^t} + w_2. \times \overrightarrow{NNoI^t} + w_3. \times \overrightarrow{NAvgAoI^t} \tag{26}$$

It is apparent from this Equation that each operator selection probability value depends on the three obtained vectors, which comprise $\overrightarrow{NFV^t}$, $\overrightarrow{NNoI^t}$, and $\overrightarrow{NAvgAoI^t}$. Also, three weights, including w_1, w_2, and w_3 are considered in Equation (26) to determine the importance of $\overrightarrow{NFV^t}$, $\overrightarrow{NNoI^t}$, and $\overrightarrow{NAvgAoI^t}$, respectively. The values of w_1, w_2, and w_3 are set in a way that they change adaptively in each iteration. In this case, in the initial iteration, the amounts of these weights are equal, and in the following iterations, the amount of w_1 increases, and the amounts of w_2 and w_3 decrease gradually. Each of these weights is calculated using Equation (27).

$$w_1 = I + (1 - I)\left[1 - exp^{\left(\frac{-\beta \times it}{it_{Max}}\right)}\right], w_2 = w_3 = \frac{1 - w_1}{2} \tag{27}$$

where parameter I is the initial value, and β is the slope value of the exponential function. In order to tune the I and β parameters, different values were evaluated, and the best values (i.e., $I = 1/3$ and $\beta = 2$) were selected empirically. Consequently, the proper w_1 formula is presented in Equation (28).

$$w_1 = \frac{1}{3} + \left(1 - \frac{1}{3}\right)\left[1 - exp^{\left(\frac{-2 \times it}{it_{Max}}\right)}\right] \tag{28}$$

Generally, in each iteration, the suggested heuristic position updating mechanism is applied to the swarm particles. This mechanism greatly enhances the exploration and exploitation capabilities of the proposed EDPSO algorithm by effectively directing the particles towards the promising regions of the search space. In the following, in order to further enhance the EDPSO algorithm's exploration and exploitation capabilities and increase its convergence rate, two local search strategies are developed, which are thoroughly explained in the next section.

4.2. Local search frameworks

Although the proposed EDPSO algorithm generally performs well in most situations and yields a favorable solution using the suggested position updating rule, it may experience slow convergence in real-world large-scale OSNs with numerous users and relations. Hence, we have devised two network-specific local search strategies based on the SA and VNS algorithms, which serve as auxiliary methods for the proposed EDPSO, enhancing its exploration and exploitation capabilities and increasing its convergence rate by escaping from the local optima. Note that these local searches only apply to the leader particle ($\overrightarrow{Gbest}$) with the best 1-NSIL value in each iteration. The pseudo-codes of the proposed local searches are described in Algorithms 2 and 3, respectively.

a) VNS-based Local Search

The VNS was introduced by Mladenović et al. as a single-solution *meta*-heuristic algorithm, usually used as a local search strategy for solving NP-hard problems [13]. This algorithm uses the idea of neighborhood change until the local optimum is attained. In this regard, VNS applies systematic techniques to switch to different neighborhoods, allowing further search progress. In this study, we use the core idea behind VNS in a straightforward way to introduce a VNS-based local search for the structural k-anonymity problem. This local search strategy, combined with the proposed EDPSO algorithm, is called EDPSOVNS. The pseudo-code of the proposed VNS-based local search is shown in Algorithm 2.

At first, the velocity and position vectors of the global best solution, the operator selection probability vector ($\overrightarrow{OSP^t}$), and the network's subgraphs ($Subgraph^h_{n\_id}$) are passed to the algorithm as the inputs. These subgraphs include the nodes that can be reached within h-hops ($h = [1, \cdots, n-1]$) from the node with label *n_id*. For example, the $Subgraph^2_1$ represents the subset of the graph G that consists of the nodes which are reachable with 2-hops from the node with label $n\_id = 1$. Given that an OSN graph can be a large-scale network with numerous nodes and edges, employing the VNS may be time-consuming to find the proper local optimum solution. To mitigate this issue, we divide the original graph into subgraphs to reduce the search space size, leading to a significant decline in the algorithm's process time.

In the next step, the $\overrightarrow{OSP^t}$ is sorted in descending order to prioritize the application of neighborhood operators based on their effectiveness thus far (step 2). That is, the operator with a higher selection probability has priority over the one with a lower selection probability, as they have demonstrated a greater impact on performance improvement up to that point. Afterwards, the nodes whose vel_{n_id} equals 1 are selected as candidate nodes (step 3). For each candidate node, the algorithm tries to find a better solution by searching the subgraphs associated with that node, including the subgraph of 1-hop nodes up to the h_{max}-hop nodes (steps 4 to 25). To this end, the algorithm starts from the $Subgraph^1_{cn\_id}$ and applies the selected operator (*S_op*) based on the order specified in step 2 to the global best solution. Specifically, the algorithm uses the first selected operator whose selection probability is higher than others. This operator is applied to the solution using the candidate node and the adjacent nodes reachable within 1-hop as inputs, generating new solutions (*N_Sols*) called neighborhood solutions (steps 10 to 13).

Subsequently, the best new solution (*n_sol*) is selected, and if it is better than the global best solution, the global best will be updated, and the algorithm will proceed to the next candidate node (steps 14 to 18). Otherwise, the neighborhood operator with the next best selection probability will be selected (step 20), and all the described steps 10 to 18 will be repeated. The algorithm will continue the search process in the current subgraph with different operators until an improvement occurs in the solution. If the algorithm reaches the last operator and does not discover any improved solution, the subgraph is updated to include nodes that are $h+1$-hop away from the candidate node (step 23). Subsequently, the algorithm tries to find a better solution using the aforementioned steps 7 to 22 in the updated subgraph. The algorithm will end when steps 5 to 25 are performed for all the candidate nodes. As a result, an updated global best solution will be obtained as the output of the VNS-based local search algorithm.

Algorithm 2. *"Framework of VNS-based Local Search"*

1. Inputs:
Global best solution: i^{th} particle with vectors $\overrightarrow{X_i}, \overrightarrow{V_i}$
Operator Selection Probability vector: $\overrightarrow{OSP^t}$
Nodes reachable with h −hop, where $h = [1, \cdots, n-1]$: $Subgraph^h$
Procedure:
2. Sort $\overrightarrow{OSP^t}$ in descending order.
3. *C_Nodes* ← Find nodes with velocity = 1 in $\overrightarrow{V_i}$
4. For each *c_node* in *C_Nodes* do
5. $h \leftarrow 1$
6. While $h \leq n-1$ do
7. *op_id* ← 1

(*continued on next page*)

(*continued*)

Algorithm 2. *"Framework of VNS-based Local Search"*

8. ***Sub_Nodes*** ← Retrieve ***Subgraph***^{h}(***c_node***)
9. While ***op_id*** ≤ Length of $\overrightarrow{OSP^{t}}$
10. ***S_op*** ← Select the operator $\overrightarrow{OSP^{t}}$(***op_id***)
11. For each ***sub_node*** in ***Sub_Nodes*** do
12. ***N_Sols*** ← Apply ***S_op***($\overrightarrow{X_i}$, ***c_node***, ***Sub_node***)
13. End For
14. Calculate the fitness of new solutions: 1-NSIL(***N_Sols***)
15. ***n_sol*** ← Select the best solution in ***N_Sols***
16. If 1 – NSIL(***n_sol***) > 1 − NSIL($\overrightarrow{X_i}$) then
17. Update the particle: $\overrightarrow{X_i}$←***n_sol***
18. Reset ***op_id*** and ***h*** to 1
19. Go to step 4 // New iteration with updated solution, ***op_id***, and ***h***
20. Else
21. Increment ***op_id***
22. End If
23. End While
24. Increment ***h***
25. End While
26. End For
27. Output: The $\overrightarrow{X_i}$ as a new Global best solution.

b) SA-based Local Search.

The SA is another single-solution *meta*-heuristic algorithm that mimics the heating and slow cooling processes of a material to modify its physical properties [12]. This algorithm is known for its simplicity, ease of implementation, and flexibility, allowing it to approach the global optimum solution effectively. Hence, in the current study, we proposed another local search strategy utilizing SA. This strategy takes the global best solution's velocity and the updated operator selection probability vectors as inputs and searches the local region around the solution space in order to discover a more optimal solution. This SA-based local search strategy, combined with the proposed EDPSO algorithm, is called EDPSOSA. The pseudo-code for EDPSOSA is provided in Algorithm 3.

In the first step, the SA parameters, the operator selection probability vector ($\overrightarrow{OSP^{t}}$), and the global best solution's velocity and position vectors are given to the algorithm as inputs. Then, in every iteration, the algorithm generates a new solution for each candidate node (*c_node*) with a velocity of 1 ($vel_{n\_id} = 1$). This is done by applying an operator which is selected by a roulette wheel adjusted based on the operator selection probability. The selected operator uses both the candidate node and another randomly selected node (*r_node*) from the set of nodes (*V*), to generate a new solution (*n_sol*) (steps 9 to 11). The current solution will be updated only if the new solution improves 1the −NSIL value compared to the previous solution (steps 12 to 14). Otherwise, the new solution will be accepted with a probability of *Prob* (steps 15 to 21). These steps (i.e., 9 to 22) are repeated until the maximum sub-iteration is met. Afterwards, the temperature will be reduced (step 24), and the next iteration will start. The algorithm will terminate when the above-mentioned processes are performed for all the candidate nodes. Consequently, an updated global best solution will be achieved as the output of the SA-based local search algorithm.

Algorithm 3. *"Framework of SA-based Local Search"*

1. Inputs:
Global best solution: ***i*** th particle with $\overrightarrow{X_i}$, $\overrightarrow{V_i}$ vectors
Operator Selection Probability vector:$\overrightarrow{OSP^{t}}$
Maximum number of iterations:it_{Max}
Maximum number of sub-iterations:sub_it_{Max}
Initial temperature:T_0
Temperature reduction rate: ***α***
Procedure:
2. ***C_Nodes*** ← Find nodes with velocity = 1 in $\overrightarrow{V_i}$
3. ***temp*** ←T_0
4. For each ***c_node*** in ***C_Nodes*** do
5. ***it*** ←1
6. While ***it*** ≤ it_{Max} do
7. ***sub_it*** ←1
8. While ***sub_it*** ≤ sub_it_{Max} do
9. ***S_op*** ← RouletteWheel($\overrightarrow{OSP^{t}}$)
10. ***r_node*** ← A random node from node set ***V***
11. ***n_sol*** ← Apply ***S_op***($\overrightarrow{X_i}$, ***c_node***, ***r_node***)
12. If 1 − NSIL(***n_sol***) > 1 − NSIL($\overrightarrow{X_i}$) then
13. Update the particle: $\overrightarrow{X_i}$←***n_sol***

(*continued on next page*)

(*continued*)

Algorithm 3. *"Framework of SA-based Local Search"*

14. Else
15. ***Delta*** ← 1 − NSIL(***n_sol***) − 1 − NSIL($\overrightarrow{X_i}$)
16. ***r*** ← Generate a random number in [0,1]
17. Calculate Acceptance Probability: ***Prob*** ← $exp^{-Delta/temp}$
18. If ***Prob*** > ***r*** then
19. Update the particle: $\overrightarrow{X_i}$ ← ***n_sol***
20. End If
21. End If
22. Increment ***sub_it***
23. End While
24. ***temp*** ← ***temp*** ×***α***
25. Increment ***it***
26. End While
27. End For
28. Output: The $\overrightarrow{X_i}$ as a new Global best solution.

4.3. Computational complexity analysis

In this section, we analyze the computational complexity of proposed algorithms. The computational cost of the EDPSO algorithm can be analyzed by examining the time complexity of each step in its pseudocode shown in Algorithm 1. The *Random_Based_Initialization* function has a time complexity of O(n_{Pop}), as it involves initializing each particle's position randomly. As the for loop in Steps 3–6 comprises iterating over each particle in the swarm and calculating its fitness using the 1-NSIL function, it has a time complexity of O(n_{Pop}*s^2*n^2), where s is the number of clusters and n is the number of nodes in the social network. Finding the $\overrightarrow{Gbest}$ in Step 7 has a time complexity of O(n_{Pop}). The while loop in Steps 8–22 is the main loop of the algorithm that runs for a maximum of it_{Max} iterations, and within each iteration, it updates the velocity and position of each particle in the swarm and also updates the personal best and global best solutions. According to Equation (14), the Update Velocity function in Step 10 has a time complexity of O(1). Also, the flowchart shown in Fig. 2 demonstrates that the *Adaptive_Heuristic_Position_Updating* function in Step 11 has a time complexity of O(5*|*C_Nodes*|*s^2*n^2), where "5″ represents the number of operators applied to each candidate node, |*C_Nodes*| is the number of candidate nodes with a velocity of 1, and s^2n^2 represents the complexity of calculating 1-NSIL for each updated particle. The if statement in Steps 13–18 has a time complexity of O(1), as it involves comparing the fitness of each particle to its personal best and global best fitness values and updating the $\overrightarrow{Pbest_i}$ and $\overrightarrow{Gbest}$. Finally, Step 20 involves using the *Local_Search* function, whose time complexity is dependent on the complexity of the local search algorithm used. Overall, the complexity of the EDPSO algorithm can be estimated as O($n_{Pop} + n_{Pop}$*s^2*$n^2 + n_{Pop} + it_{Max}$*n_{Pop}*$\left(5^*|C\_Nodes|^*s^2{*}n^2\right)$ + it_{Max}*O(*Local_Search*)). Considering the highest order of growth, the overall time complexity can be simplified to O(it_{Max}*(n_{Pop}*5*|*C_Nodes*|*s^2*n^2 + O(*Local_Search*))). It is worth noting that the actual running time of the algorithm may vary depending on the specific implementation of its functions. For instance, in the *Adaptive_Heuristic_Position_Updating* function, all five operators can be applied to a particle simultaneously using parallel processing techniques, which reduces the algorithm's running time. Additionally, the size of the candidate node set |*C_Nodes*| generally decreases during the course of the algorithm, which again affects the overall running time of the EDPSO algorithm. In order to provide a detailed analysis of the performance of this function, we will examine the time complexities of both the VNS and SA algorithms below.

The computational complexity of the VNS-based Local Search algorithm can be analyzed by examining the time complexity of each step shown in Algorithm 2. Given that there are n nodes in the network and.

$|\overrightarrow{OSP}| = 5$ operators in the operator selection probability vector, step 2, which involves sorting the five operator selection probabilities, has a time complexity of O(5 log 5). Step 3, which involves finding candidate nodes with velocity = 1 in the velocity vector, has a time complexity of O(1). Next, in steps 4–25, the algorithm involves a nested loop structure, where the outer loop iterates over candidate nodes for |*C_Nodes*| times, and the inner loops iterate over the values of hops, operators, and sub-nodes for $n-1$, 5, and |*Sub_Nodes*| times, respectively. Within these loops, the following operations are performed: In step 8, a subgraph of h hops is retrieved for each candidate node with a complexity of O(n). In step 10, an operator is selected, and in step 12, this operator is applied to the $\overrightarrow{X_i}$ which both have a complexity of O(1). The fitness values of new solutions are calculated in step 14 with a complexity of O(|*Sub_Nodes*|*s^2*n^2). The following steps, which include selecting the best solution, updating the particle, resetting and incrementing the *op_id* and h variables, all have a complexity of O(1). Therefore, the time complexity of the entire algorithm can be expressed as O(5log5 + |*C_Nodes*|*n(n*5*|*Sub_Nodes*|$\left(1+s^2{*}n^2\right)$)) which can be simplified to O(5*n^2*|*C_Nodes*|*|*Sub_Nodes*|$\left(1+s^2{*}n^2\right)$).

Finally, we analyze the computational complexity of the SA algorithm by examining the time complexity of each step in Algorithm 3. Steps 2 and 3 involve finding candidate nodes with velocity = 1 and setting the initial temperature, both with the time complexity of O(1). Steps 4–27 involve a nested loop structure, where the outer loop iterates over the candidate nodes, and the inner loops iterate over the values of iterations and sub-iterations, respectively. The time complexity of these loops depends on the values of |*C_Nodes*|, it_{Max} and sub_it_{Max}. The time complexity of selecting an operator by the roulette wheel, generating a random number, and applying an operator are O(1), and the calculation of the fitness value of the new solution has a time complexity of O(s^2*n^2). Considering that the time complexity of the remaining steps, consisting of updating the particle, calculating *Delta*, calculating acceptance probability,

incrementing the *sub_it* and *it* variables, and updating the temperature value, is O(1), the overall time complexity of SA algorithm can be expressed as $O(|C\_Nodes|*it_{Max}*sub\_it_{Max}*s^2*n^2)$.

5. Experimental results

In this section, first, the Taguchi experiment is performed to set the parameters of the EDPSO algorithm. Then, to investigate the effectiveness of the developed EDPSO, EDPSOVNS, and EDPSOSA algorithms, we compare them with the previous related *meta*-heuristic algorithms, such as DPSO [11], PSOGA [10], and GA [9]. In this regard, we first implement the mentioned algorithms on various datasets for different k values, which are selected depending on the dataset's number of nodes. Each algorithm is executed 30 times on each of the k values, and the average 1-NSIL value is computed. Second, for a specific k value on each dataset, we demonstrate the algorithms' performance and compare them regarding the best 1-NSIL value (F_{Best}), average 1-NSIL value ($F_{Average}$), worst 1-NSIL value (F_{Worst}), the standard deviation of the obtained 1-NSIL values (STD), and the average process-time (Process-Time $_{Average}$) criteria. Subsequently, to indicate the convergence rate of these algorithms, their convergence behaviors were analyzed. Eventually, to validate the performance of the suggested algorithms, they are compared with the four well-known non *meta*-heuristic methods (Anatomy-based clustering [35], Enhanced k-means clustering [37], SANGreea [6] and SCAN [7]), in terms of both 1-NSIL and 1-ILR criteria. Finally, the statistical analysis of the obtained results using Friedman is provided to determine if there are any noteworthy distinctions among the outcomes of the proposed algorithms compared to the previous algorithms.

5.1. Experimental networks

All the mentioned algorithms were implemented in MATLAB R2018b on a PC with Intel (R) Core (TM)-i5, CPU, 2.7 GHz, 8 GB RAM, running Windows 10 enterprise. Besides, the experiments were carried out on nine real-world datasets, which are listed in Table 5. Some structural features of the experimental networks, such as the number of nodes and edges, are also presented in Table 5.

5.2. Experiments for setting the parameters

In the early 1980 s, Genichi Taguchi developed a method based on orthogonal array experiments in which an optimum set of parameters in a fewer number of experiments can be reached [48]. To set the optimal parameters in the EDPSO algorithm, including c_1, c_2, IW, n_{Pop}, and it_{Max} parameters, the Taguchi method exploits a statistical measure of performance called signal-to-noise ratio (SNR), which is a logarithmic function of the desired output. Taguchi's SNR is the mean (signal) ratio to the standard deviation (noise), taking both the mean and the variability into account. The standard SNR ratios are categorized into three types, named nominal is best (NB), lower the better (LB), and higher the better (HB). In this study, HB is exploited because the suggested algorithms should achieve the highest 1-NSIL value. The optimal parameter set is always the parameter combination with the highest SNR, regardless of the type of Taguchi design, and in the current study, the L25 Taguchi design was used, as illustrated in Table 6. It is worth noting that each experiment is executed 10 times, and the Taguchi analysis is performed based on the average fitness value (1-NSIL) of these 10 repeats.

The graphical representation of the SNR is shown in Fig. 4. Also, the optimal level of each parameter for parameter tuning of the proposed EDPSO is shown in Table 7.

As mentioned above, to tune the α parameter of Equation (18), different values were evaluated for the α_{min} and α_{max} parameters, and it was concluded that 1 and it_{Max} are the best values for them, respectively.

5.3. Comparison with Meta-Heuristic algorithms

This experiment sought to investigate the performance comparison of the proposed methods with the previously introduced DPSO, PSOGA, and GA algorithms on various datasets and for different k values. As mentioned earlier, each algorithm has been run 30 times on each dataset, and the algorithms are compared with each other in terms of the average 1-NSIL value. Note that, to have a reasonable comparison, the population size and maximum iteration number parameters of the DPSO algorithm are set similarly to the EDPSO, and the parameters of the PSOGA and GA algorithms are set according to the articles presented in [10] and [9], respectively.

As can be seen in Fig. 5, the proposed algorithms outperform the other *meta*-heuristic algorithms in all the networks and can find

Table 5
Experimental Networks.

Dataset	Nodes	Edges
Karate Club [40]	34	78
Tailor Shop [41]	39	158
Prison [42]	67	182
Joint Senate Press Releases [43]	92	477
Collaboration in Jazz [44]	198	2742
Innovation Among Physicians [45]	246	1098
Facebook Pages Food [46]	620	2103
Wikipedia Vote [46]	889	2914
Political Blogs [47]	1490	19,090

Table 6
L25 Taguchi Design.

Factors		Levels				
Code	EDPSOL Parameters	1	2	3	4	5
A	c_1(Personal Learning Coefficient)	1.45	1.5	1.75	1.85	2
B	c_2(Global Learning Coefficient)	1.45	1.5	1.75	1.85	2
C	IW(Inertia Weight)	0.7	0.73	0.75	0.77	0.8
D	n_{Pop}(Population Size)	10	20	50	70	100
E	it_{Max}(Maximum Iteration)	100	150	200	250	300

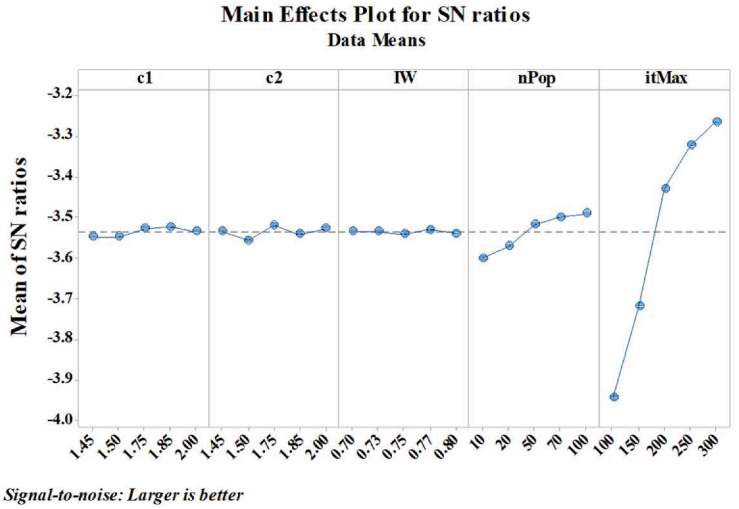

Fig. 4. Graphical Representation of the SNR.

Table 7
Optimal Parameters of EDPSO.

Factor									
c_1		c_2		IW		n_{Pop}		it_{Max}	
Selected Level	Level Value	Selected Level	Level Value	Selected Level	Level Value	Selected Level	Level Value	Selected Level	Level Value
A4	1.85	B3	1.75	C4	0.77	D5	100	E5	300

anonymized networks with higher utility compared to other algorithms. The reason behind this superiority is due to the use of the novel heuristic algorithm for the position vector updating process in the EDPSO. This heuristic algorithm adaptively tunes the selection probabilities of the used operators based on the obtained fitness values from each operator both in the current iteration and the history of each operator's performance in the previous iterations. So, this algorithm increases the effectiveness of the EDPSO by balancing the exploration and exploitation capabilities while preventing it from converging too early. Besides, using the VNS and SA local search strategies provides higher search capabilities and convergence rate to the EDPSO since these local searches change the solution's direction to a more favorable region, preventing it from being trapped in local optima. On the contrary, as illustrated in the results, the GA method is the worst among the algorithms. This happens mainly because of limiting the search space by omitting the solutions that do not satisfy the structural k-anonymity constraint in the crossover step, which in turn downgrades the exploration ability of GA. Figs. 5(a) to 5(i) indicate that the EDPSOVNS and EDPSOSA algorithms have almost similar performances, outperforming the other algorithms. However, the EDPSOVNS's performance is slightly better than the EDPSOSA. Indeed, across all datasets and for each distinct k value, EDPSOVNS consistently outperforms the other algorithms in finding high-utility anonymized networks. It is also noteworthy that the EDPSOSA and EDPSO outperform the DPSO, PSOGA, and GA in all the cases. In what follows, we will analyze the results to highlight how our new algorithms (EDPSOVNS, EDPSOSA, and EDPSO) outperform the previous ones (DPSO, PSOGA, and GA) at various k values. To keep it short and precise, we will be concentrating our discussion on three specific networks: the Karate Club, Collaboration in Jazz, and Political Blogs.

In the Karate Club network (i.e., Fig. 5(a)), the 1-NSIL value of EDPSOVNS in $k = 3$ equals 0.7861, which has superiority over the EDPSOSA, EDPSO, DPSO, PSOGA, and GA algorithms with 1-NSIL values of 0.7841, 0.7767, 0.7373, 0.7119, and 0.6891, respectively. Besides, the EDPSOSA demonstrates a 6.3 %, 10.1 %, and 13.7 % increase in the 1-NSIL value compared to the DPSO, PSOGA, and GA,

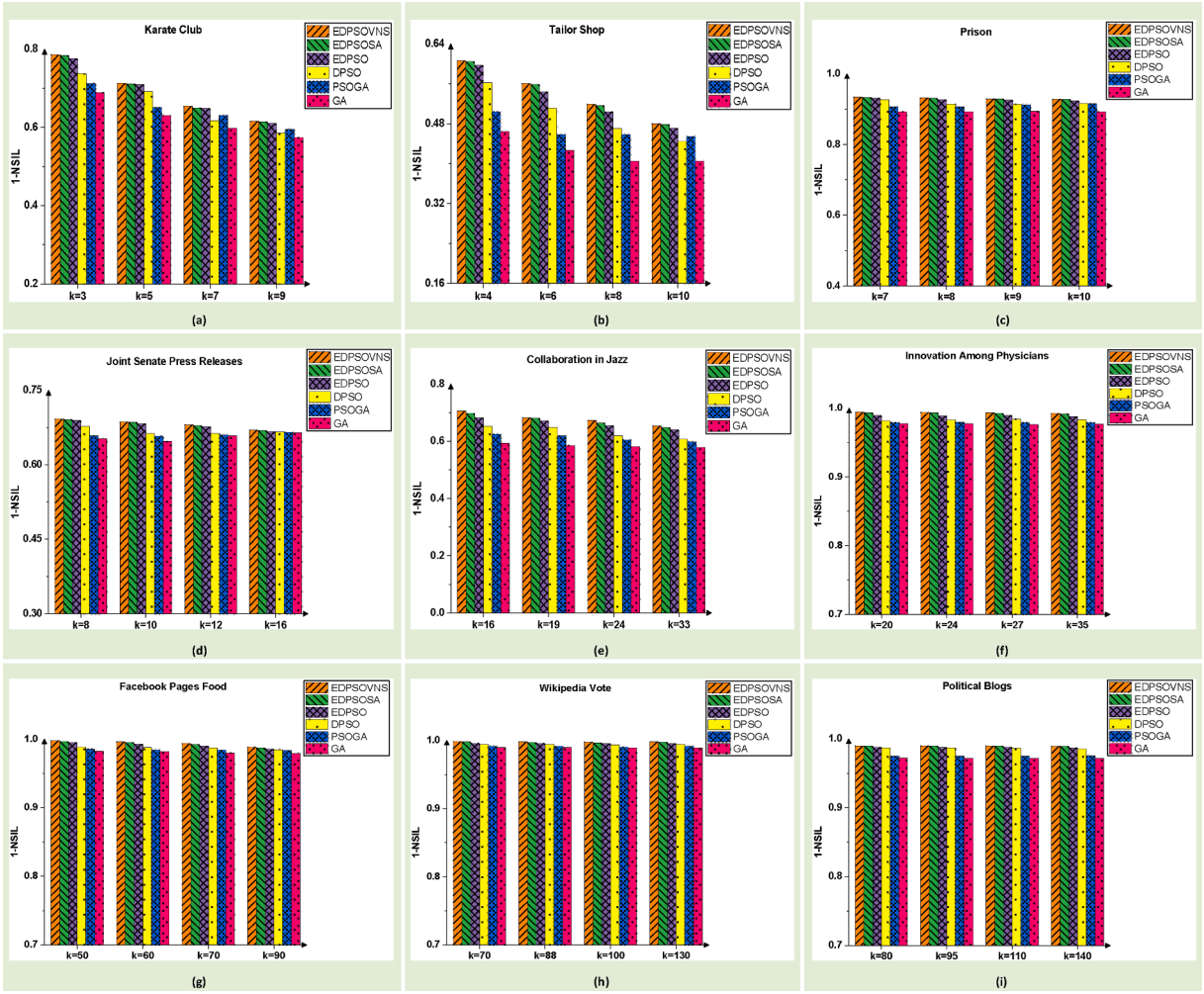

Fig. 5. *Comparing the Proposed Methods with Previous Meta-heuristics on Various Datasets and for Different* k *Values.*

respectively. Likewise, EDPSO shows a 5 %, 9.1 %, and 12.7 % higher 1-NSIL value over the DPSO, PSOGA, and GA. As previously mentioned, the GA algorithm has the worst performance among all the compared methods. Furthermore, in the case of $k = 5$ in the Karate Club network, the 1-NSIL value of EDPSOVNS equals 0.7129, showcasing its superior performance compared to the EDPSOSA, EDPSO, DPSO, PSOGA, and GA algorithms, which scored 1-NSIL values of 0.71128, 0.70955, 0.69206, 0.65079, and 0.6307, respectively. In addition to this, the EDPSOSA algorithm demonstrates improvements of 2.8 %, 9.3 %, and 12.8 % in the 1-NSIL value when compared to the DPSO, PSOGA, and GA algorithms, respectively. Likewise, the EDPSO shows an increase in the 1-NSIL value by 2.5 %, 9 %, and 12.5 % over the DPSO, PSOGA, and GA, respectively. As in the previous case for $k = 3$, the GA algorithm still underperforms in comparison to all other methods, holding the least favorable results.

When observing the algorithms' performance on $k = 7$ of the Karate Club network, the 1-NSIL value of EDPSOVNS stands at 0.6545, maintaining its edge over the other algorithms, EDPSOSA, EDPSO, DPSO, PSOGA, and GA, which reached 1-NSIL values of 0.65038, 0.64867, 0.6175, 0.63069, and 0.59808, respectively. Also, the EDPSOSA demonstrates a relative enhancement in 1-NSIL value by approximately 5.3 %, 3.1 %, and 8.7 % compared to DPSO, PSOGA, and GA algorithms, respectively. Similarly, EDPSO illustrates an increase in 1-NSIL value by roughly 5 %, 2.9 %, and 8.5 % in comparison to DPSO, PSOGA, and GA, respectively. It is worth mentioning that the PSOGA overtook DPSO in terms of 1-NSIL value, which was not the case at $k = 3$ and $k = 5$. This suggests that PSOGA deals better with higher k than DPSO in the context of the Karate Club network. Again, as consistently noted from previous results at $k = 3$ and $k = 5$, the GA algorithm continues to perform least favorably among all compared methods, with the lowest results when $k = 7$. Finally, in the case of $k = 9$ for the Karate Club network, as before, the EDPSOVNS algorithm takes the lead with a 1-NSIL value of 0.61579, outperforming EDPSOSA, EDPSO, DPSO, PSOGA, and GA, which have 1-NSIL values of 0.61464, 0.61029, 0.58572, 0.59537, and 0.57455, respectively. Further, EDPSOSA has an improvement in 1-NSIL value by approximately 4.9 %, 3.2 %, and 7 % when compared to DPSO, PSOGA, and GA, respectively. Similarly, EDPSO indicates a rise in the 1-NSIL value by roughly 4.2 %, 2.5 %, and 6.2 % in relation to DPSO, PSOGA, and GA, respectively. At $k = 9$, we observe that PSOGA continues to outperform DPSO, which reinforces the trend we noticed at $k = 7$. This hints at the adaptability of PSOGA to larger k values in this dataset. Consistent with the

previous findings at k = 3, k = 5, and k = 7, the GA algorithm again has the lowest performance among the compared algorithms.

Regarding the Collaboration in Jazz network (i.e., Fig. 5(e)), the performance of the algorithms is analyzed at the various k values. It is worth noting that, throughout all k values, GA consistently shows the lowest performance, while EDPSOVNS maintains the highest 1-NSIL values, indicating its superior performance in anonymizing this network. It is also notable that both EDPSOSA and EDPSO consistently outperform DPSO, PSOGA, and GA algorithms. At k = 16, EDPSOVNS leads with a 1-NSIL value of 0.70544, outperforming EDPSOSA, EDPSO, DPSO, PSOGA, and GA, which reached 1-NSIL values of 0.69757, 0.68203, 0.65162, 0.62398, and 0.59334, respectively. It is worth noting that EDPSOSA shows an increase in the 1-NSIL value by roughly 7 %, 11.8 %, and 17.6 % compared to DPSO, PSOGA, and GA, respectively. EDPSO also outperforms DPSO, PSOGA, and GA by approximately 4.7 %, 9.3 %, and 14.9 %. For k = 19, EDPSOVNS still leads with a 1-NSIL value of 0.68206, surpassing EDPSOSA, EDPSO, DPSO, PSOGA, and GA, whose 1-NSIL values are 0.68035, 0.67146, 0.64769, 0.61831, and 0.58573, respectively. In this scenario, EDPSOSA shows an improvement in 1-NSIL value by approximately 5 %, 10 %, and 16.2 % compared to DPSO, PSOGA, and GA, respectively. Also, EDPSO shows an increase in 1-NSIL value by roughly 3.7 %, 8.6 %, and 14.6 % over DPSO, PSOGA, and GA, respectively. In the case of k = 24, EDPSOVNS maintains its leading position with a 1-NSIL value of 0.67247, superior to EDPSOSA, EDPSO, DPSO, PSOGA, and GA which have 1-NSIL values of 0.66361, 0.65408, 0.61993, 0.60531, and 0.58128, respectively. EDPSOSA and EDPSO demonstrate relative increases in 1-NSIL value by about 7 %, 9.6 %, 14.2 %, and 5.5 %, 8.1 %, 12.5 % compared to DPSO, PSOGA, and GA, respectively. Finally, at k = 33 of this network, EDPSOVNS yet again leads with a 1-NSIL value of 0.65356, followed by EDPSOSA, EDPSO, DPSO, PSOGA, and GA, which obtained 1-NSIL values of 0.64767, 0.63996, 0.60672, 0.59716, 0.57811, respectively. EDPSOSA shows an improvement in 1-NSIL value by approximately 6.7 %, 8.4 %, and 12 % compared to DPSO, PSOGA, and GA, respectively. Similarly, EDPSO exhibits an increase in the 1-NSIL value by roughly 5.5 %, 7.2 %, and 10.7 % over DPSO, PSOGA, and GA, respectively.

Furthermore, with respect to the Political Blogs network, the performance of the algorithms on various k values is analyzed, shown in Fig. 5(i). In this dataset, EDPSOVNS consistently shows the highest 1-NSIL values at all given k values, illustrating its strong performance in anonymizing this network to preserve privacy while maintaining its utility. However, the differences among the top three algorithms (EDPSOVNS, EDPSOSA, and EDPSO) are quite minimal. On the other hand, GA consistently performs the worst among the six algorithms. For k = 80, EDPSOVNS achieves the highest 1-NSIL value of 0.98982, demonstrating better performance than EDPSOSA, which achieves a 1-NSIL of 0.98976. EDPSO follows closely with a 1-NSIL of 0.98796, ahead of DPSO with a 1-NSIL of 0.98725, PSOGA with a 1-NSIL of 0.97508, and GA with a 1-NSIL of 0.97237. At k = 95, EDPSOVNS once again leads with a 1-NSIL value of 0.98996, marginally superior to EDPSOSA, which scores a 1-NSIL of 0.98972. EDPSO follows with a 1-NSIL of 0.98748, outperforming DPSO with a 1-NSIL of 0.98694, PSOGA with a 1-NSIL of 0.97488, and GA with a 1-NSIL of 0.97199. In the case of k = 110, EDPSOVNS maintains its superior performance with a 1-NSIL value of 0.98984. EDPSOSA follows closely with a 1-NSIL of 0.98958, ahead of EDPSO with a 1-NSIL of 0.98798, DPSO with a 1-NSIL of 0.98665, PSOGA with a 1-NSIL of 0.97508, and GA with a 1-NSIL of 0.97167. Lastly, for k = 140, EDPSOVNS again achieves the highest 1-NSIL value of 0.98941, outperforming EDPSOSA, which obtains a 1-NSIL of 0.98928, EDPSO with a 1-NSIL of 0.98723, DPSO with a 1-NSIL of 0.98555, PSOGA with a 1-NSIL of 0.97581, and GA with a 1-NSIL of 0.97193.

In conclusion, the obtained results demonstrate the effectiveness of the proposed algorithms (EDPSOVNS, EDPSOSA, and EDPSO) as a result of utilizing a heuristic mechanism for position vector updating, leading to a trade-off between exploration and exploitation. Furthermore, this experiment underscores the effectiveness of the vector-based solution representation in addressing the structural k-anonymity problem, outperforming conventional matrix-based approaches like GA and PSOGA. The comparative analysis clearly shows that incorporating a vector-based solution representation in the developed EDPSO, EDPSOVNS, and EDPSOSA enhances both exploration and exploitation, which is evidenced by the achievement of higher-quality solutions.

Subsequently, the reliability of the proposed algorithms is investigated, such that if an algorithm achieves similar results in successive runs, it is considered to be more reliable. Hence, in Tables 8 to 16, for each of the mentioned algorithms, its performance regarding the F_{Best}, $F_{Average}$, F_{Worst}, STD, and Process-Time $_{Average}$ criteria in a sample k value for each dataset is demonstrated. It is evident from these tables that the proposed EDPSOVNS algorithm outperforms other algorithms in terms of the $F_{Average}$ and F_{Worst} criteria. Moreover, although this algorithm has superiority over other algorithms regarding the F_{Best} criterion in the majority of times, in some cases, such as Karate Club (Table 8), Tailor Shop (Table 9), and Collaboration in Jazz (Table 12), this algorithm and the EDPSOSA attain identical F_{Best} value. Therefore, this algorithm can find an anonymized network with the highest utility among the compared algorithms. Besides, it has the lowest STD among the compared algorithms, meaning that it reaches similar 1-NSIL values in different runs, so the EDPSOVNS can be considered the most reliable algorithm. In addition, according to the comparison results, the proposed EDPSOSA and EDPSO algorithms are superior to the DPSO, PSOGA, and GA in terms of the F_{Best}, $F_{Average}$, and F_{Worst} criteria. Regarding the STD criterion, the EDPSOSA achieves a lower STD value than the DPSO, PSOGA, and GA in all cases; hence, it is more

Table 8
Results of the Algorithms on Karate Club Dataset with k = 5.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 9
Results of the Algorithms on Tailor Shop Dataset with k = 6.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 10
Results of the Algorithms on Prison Dataset with k = 7.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 11
Results of the Algorithms on Joint Senate Press Releases Dataset with k = 8.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 12
Results of the Algorithms on Collaboration in Jazz Dataset with k = 16.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 13
Results of the Algorithms on Innovation Among Physicians Dataset with k = 20.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

reliable than others. Also, in the majority of the cases, the EDPSO has a lower STD value than DPSO, PSOGA, and GA; nevertheless, in Tailor Shop (Table 9) and Innovation Among Physicians (Table 13) datasets, the PSOGA's STD value is slightly better than the EDPSO.

Furthermore, the comparative results indicate that the average process time values of the proposed EDPSOVNS and EDPSOSA algorithms are higher than the EDPSO, DPSO, and GA. The primary reason for the high computational time of these algorithms is their hybrid essence, i.e., the combination of the EDPSO with the VNS and SA local searches results in an increase in the processing time of both the EDPSOVNS and EDPSOSA. Note that this increase can be considered acceptable since these two algorithms are able to achieve anonymized social networks whose structural information loss is much lower than the previous algorithms. Also, the results indicate

Table 14
Results of the Algorithms on Facebook Pages Food Dataset with k = 50.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 15
Results of the Algorithms on Wikipedia Vote Dataset with k = 70.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9989	0.9983	0.9972	9.68E-04	20,449
EDPSOSA	0.9988	0.9979	0.9964	1.18E-03	17,342
EDPSO	0.9968	0.9958	0.9941	1.54E-03	12,353
DPSO	0.9961	0.9942	0.9931	1.61E-03	10,589
PSOGA	0.9931	0.9915	0.9899	1.62E-03	25,365
GA	0.9922	0.9897	0.9884	2.16E-03	14,087

Table 16
Results of the Algorithms on Political Blogs Dataset with k = 80.

Algorithm	F_{Best}	$F_{Average}$	F_{Worst}	STD	Process-Time $_{Average}$ (S)
EDPSOVNS	0.9902	0.9898	0.9896	2.05E-04	75,020
EDPSOSA	0.9901	0.9898	0.9896	2.08E-04	63,996
EDPSO	0.9883	0.9880	0.9874	4.01E-04	46,701
DPSO	0.9882	0.9873	0.9864	8.72E-04	37,170
PSOGA	0.9765	0.9751	0.9740	8.92E-04	92,883
GA	0.9735	0.9724	0.9718	9.97E-04	50,219

that the DPSO algorithm has the lowest and the PSOGA has the highest process time. Nonetheless, none of these algorithms could find an anonymized network with a higher 1-NSIL value compared to the proposed algorithms. The reason behind the high processing time of the PSOGA is due to its matrix-based solution representation and also the hybrid utilization of the PSO and GA algorithms in a sequential manner. In addition, as given in the following tables, the average process time of the GA is slightly lower than the EDPSO in the low-dimension networks, such as Karate Club, Tailor Shop, and Prison, but its process time grows rapidly as the dimension of the evaluated networks increases, mainly due to its matrix-based solution representation. Therefore, the EDPSO has a lower process time than the GA in Joint Senate Press Releases, Collaboration in Jazz, and Innovation Among Physicians networks. These findings support the fact that the proposed vector-based solution representation offers lower complexity and greater resource efficiency when compared to matrix-based solution representation.

In the last experiment of the current section, the convergence behavior analysis is performed for the above-mentioned algorithms. This experiment mainly evaluates the impact of incorporating SA and VNS-based local search strategies into the proposed EDPSO algorithm, particularly in enhancing convergence rates and achieving near-optimal solutions, we conducted an in-depth analysis of the convergence behavior of each algorithm across various datasets. In this regard, the average number of iterations and standard deviation of the convergence rates (i.e., AVG and STD) obtained from running each of these algorithms 30 times are demonstrated in Table 17. The STD values indicate how varied the convergence rates are. Lower standard deviation values suggest a higher level of

Table 17
The Average and Standard Deviation of the Convergence Iteration of Different Algorithms.

Algorithms	EDPSOVNS		EDPSOSA		EDPSO		DPSO		PSOGA		GA	
Convergence Rate Criteria	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Karate Club, k = 5	64	11.6	107	12.4	154	12.9	131	13.5	183	14.1	227	14.3
Tailor Shop, k = 6	76	11.4	95	11.9	165	12.8	142	13.2	232	14.2	270	14.5
Prison, k = 7	91	10.8	102	12.1	173	13.1	149	13.9	248	15.3	281	15.7
Joint Senate Press Releases, k = 8	95	12.2	116	12.8	204	13.6	168	14.1	246	14.8	269	15.3
Collaboration in Jazz, k = 16	79	12.7	98	12.9	197	13.3	152	13.8	236	14.6	276	14.9
Innovation Among Physicians, k = 20	75	12.1	93	12.5	173	13.6	142	14.2	239	14.6	268	15.9
Facebook Pages Food, k = 50	83	12.4	112	12.7	162	13.4	145	13.7	235	15.1	266	15.6
Wikipedia Vote, k = 70	87	11.9	117	12.3	171	13.1	154	13.3	243	14.9	274	15.4
Political Blogs, k = 80	81	11.5	110	12.2	159	12.7	138	13.1	231	14.7	253	14.8

consistency and reliability in an algorithm's performance.

It is evident from the comparative results illustrated in Table 17 that the EDPSOVNS algorithm consistently exhibits the fastest convergence rate across all datasets. This is attributed to its VNS-based local search, which efficiently directs the algorithm toward higher-quality solutions in fewer iterations. Not only does EDPSOVNS achieve optimal results, but its rapid convergence underlines its overall efficiency. Furthermore, the EDPSOVNS algorithm demonstrates the most stable performance, as indicated by the lowest standard deviation values. This stability reflects the algorithm's reliability and consistency in performance. In addition, the EDPSOSA algorithm is the second-most efficient in terms of convergence rate. Its relatively fast convergence is due to its SA-based local search, which enhances the algorithm's efficiency. It is apparent that the EDPSO and DPSO algorithms have relatively slower convergence rates compared to EDPSOVNS and EDPSOSA, but both exhibit faster convergence rates compared to PSOGA and GA. However, these two algorithms suffer from a slow convergence rate in real-world and large-scale OSNs with numerous users and relations. On the other end of the spectrum, the GA algorithm is the least efficient among the tested algorithms. It requires the highest average number of iterations to reach convergence, which is a considerable drawback, especially in applications dealing with large-scale data.

As an example, in the Karate Club network, EDPSOVNS only required an average of 64 iterations to reach convergence. In contrast, GA took the longest, with an average of 227 iterations. This suggests that EDPSOVNS is the most efficient algorithm in this context. The standard deviation, which represents the variability of iterations, is also lowest for EDPSOVNS (11.6) and highest for GA (14.3), illustrating that the convergence rate for GA has a higher degree of variability. The similar performance pattern in all the other datasets further reinforces the finding that EDPSOVNS consistently demonstrates the quickest convergence rate and the most consistent performance among the tested algorithms.

Additionally, Fig. 6 shows the convergence diagram of the analyzed algorithms for the run with the fastest convergence rate. These diagrams clearly approve that the proposed EDPSOVNS and EDPSOSA algorithms have the highest convergence rate compared to others. It is also important to note that the EDPSO algorithm uses a proposed heuristic algorithm as the position vector updating mechanism to make an efficient tradeoff between exploration and exploitation capabilities and escape premature convergence. In

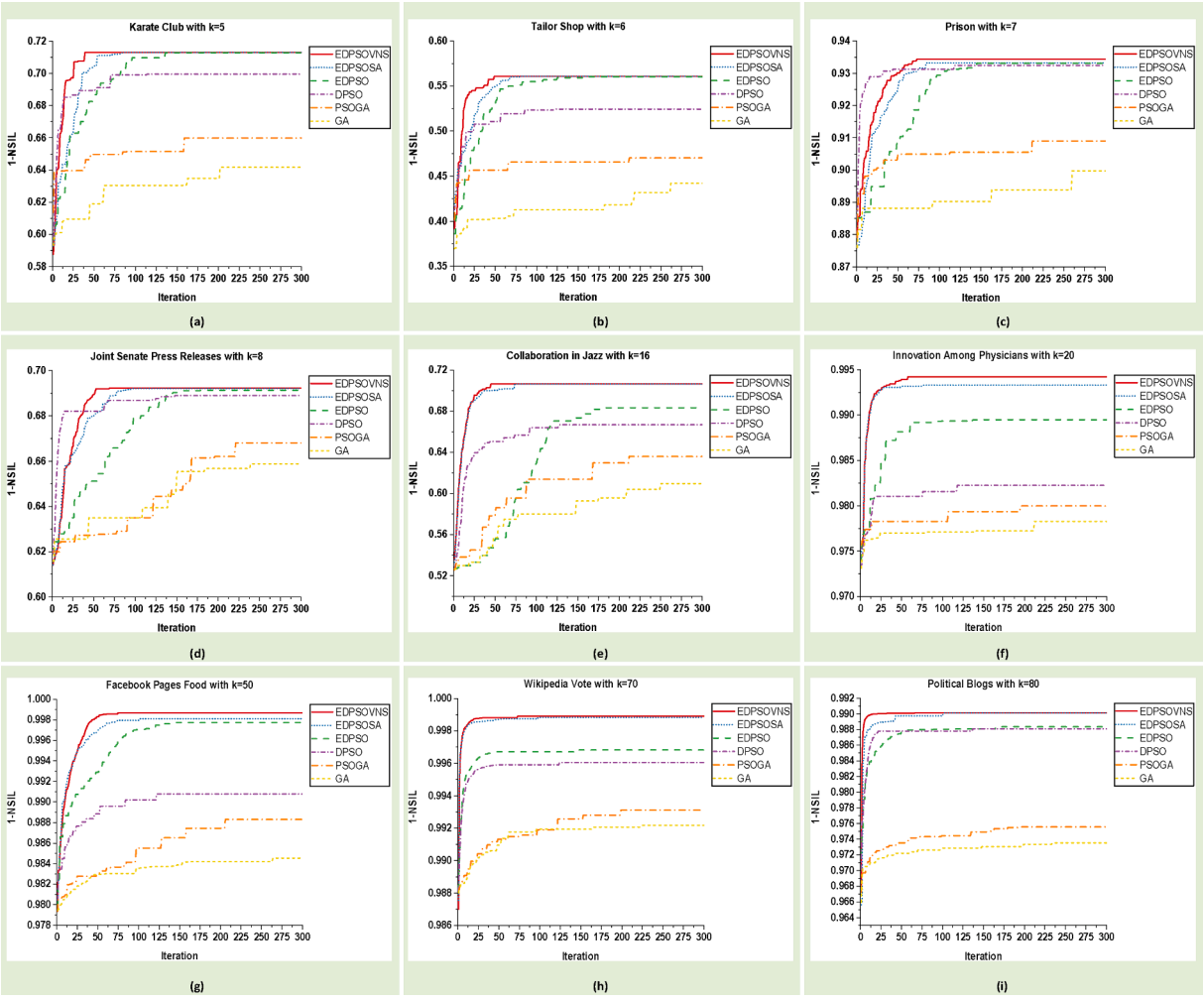

Fig. 6. Convergence Diagrams of the Different Algorithms for the Run with the Fastest Convergence Rate.

other words, the EDPSO employs a powerful, intelligent mechanism to find a more promising solution than the DPSO, which blindly updates its particles. Therefore, it is evident that because of the strong exploration and exploitation capabilities of the EDPSO, it attains a better solution at the cost of a slower convergence rate than the DPSO.

In summary, the results obtained from the experiments demonstrate that the proposed algorithms possess significant advantages over previous *meta*-heuristics in terms of fitness value and reliability. These findings confirm that the EDPSO, EDPSOVNS, and EDPSOSA algorithms exhibit superior exploration and exploitation capabilities with their adaptive heuristic position updating mechanism. Furthermore, the process-time results for the evaluated algorithms indicate that the use of vector-based solution representation not only reduces algorithm complexity and hardware resource requirements but also enables the algorithms to achieve higher-quality solutions. Additionally, the EDPSOVNS and EDPSOSA algorithms outperform other algorithms in terms of convergence rate, indicating that the proposed local search strategies enhance both exploration and exploitation, preventing the algorithms from getting trapped in local optima. Consequently, the suggested algorithms offer a suitable and effective approach for anonymizing social networks while preserving high utility.

5.4. Comparison with non Meta-Heuristic algorithms

Herein, we compare the proposed algorithms with the four well-known greedy and heuristic methods, namely Anatomy-based clustering (Anatomy), Enhanced k-means clustering (Ek-means), SANGreea, and SCAN, regarding both 1-NSIL and 1-ILR information loss criteria. Note that ILR is the structural information loss criterion introduced in [7]. Similar to the NSIL criterion, to maintain the usefulness of the anonymized social network in our experiments, we maximize the 1-ILR value, which is equivalent to the ILR minimization. Besides, we execute the SCAN algorithm 30 times and then use only the best result (1-ILR) of this algorithm, which can be seen in Fig. 7.

The comparison between the EDPSOVNS, EDPSOSA, EDPSO, Anatomy, Ek-means, SCAN, and SANGreea on Karate Club, Tailor Shop, Prison, Joint Senate Press Releases, Collaboration in Jazz, Innovation Among Physicians, Facebook Pages Food, Wikipedia Vote, and Political Blogs networks is demonstrated in Fig. 7. The obtained results reveal that the proposed algorithms, especially the EDPSOVNS, achieve improved 1-NSIL and 1-ILR values when compared to the previous algorithms. In other words, these algorithms anonymize a social network while preserving a higher utility than the Anatomy, Ek-means, SCAN, and SANGreea methods. Again, for the sake of brevity, in the following paragraphs, we analyze the results obtained from Karate Club, Collaboration in Jazz, and Political Blogs networks in more detail.

Considering the Karate Club network dataset with $k = 5$ (Fig. 7 (a)) and focusing on the 1-NSIL criterion, EDPSOVNS performed the best with a value of 0.7129, indicating it retained information most effectively during the clustering process among all methods. EDPSOSA closely followed at 0.7113, and EPDSO was third at 0.7095. Among the well-known greedy and heuristic methods, Anatomy had the highest score (0.66032), followed by Ek-means (0.65112), SCAN (0.6328), and SANGreea (0.6145). This implies that, for this dataset, the proposed algorithms, especially EDPSOVNS, generally retained more information than the previous methods when measured by the 1-NSIL criterion. Regarding the 1-ILR criterion, EDPSOVNS once again performed the best with a score of 0.9433. This was closely followed by EDPSOSA (0.9427) and EPDSO (0.9406). Among the previously introduced algorithms, SCAN had the highest score (0.934), followed by Ek-means (0.92552), Anatomy (0.91545), and SANGreea (0.9104). Therefore, according to the 1-ILR criterion, the proposed algorithms, especially the EDPSOVNS, generally maintained a better balance between data utility and privacy compared to the previous methods on this dataset.

Upon analysis of the Collaboration in Jazz network with $k = 16$ (Fig. 7(e)), it is worth noting that for the 1-NSIL criterion, EDPSOVNS stands out with the highest score of 0.7054. This is followed by EDPSOSA with 0.6976, EPDSO with 0.682, SCAN with 0.663, Anatomy with 0.5938, Ek-means with 0.5933, and finally SANGreea with 0.5743. The results for this criterion clearly show that EDPSOVNS outperforms the other methods, including the Anatomy, Ek-means, SCAN, and SANGreea algorithms. Besides, from the obtained results of the 1-ILR criterion, it is evident that, again, EDPSOVNS performs the best with a score of 0.9923. It is followed closely by EDPSOSA with 0.992, then EPDSO with 0.9917, SCAN with 0.9912, Ek-means with 0.9875, Anatomy with 0.9871, and SANGreea with 0.9869. These results align with those from the 1-NSIL criterion, again showing EDPSOVNS as the best algorithm to anonymize the network.

From the results presented for the Political Blogs network, shown in Fig. 7(i), a comparison of the algorithms based on their 1-NSIL and 1-ILR values reveals some clear trends. Regarding the 1-NSIL metric, EDPSOVNS exhibits the highest performance with a score of 0.98982, closely followed by EDPSOSA with a value of 0.98976. The EPDSO comes next, yielding a score of 0.98796. This indicates that these proposed methods have lower information loss and perform better at preserving the overall structure of the data. Besides, looking at the previous heuristic and greedy algorithms, SCAN performs relatively well with a score of 0.9625; however, it falls behind the proposed methods, followed by Anatomy and Ek-means with scores of 0.95825 and 0.95811, respectively. The lowest score belongs to SANGreea, with a 1-NSIL value of 0.95764, showing the most information loss amongst the compared algorithms. The story is much the same with the 1-ILR criterion. EDPSOVNS leads the way with a score of 0.99717, closely trailed by EDPSOSA and EPDSO with values of 0.99712 and 0.99709, respectively. Among the traditional algorithms, SCAN scores highest with a value of 0.99668, followed by Ek-means with 0.99609 and Anatomy with 0.99602. SANGreea shows the highest information loss with a value of 0.996.

Eventually, from the results, it is evident that the proposed algorithms have privilege over considered non *meta*-heuristic algorithms in terms of both 1-NSIL and 1-ILR criteria, and they are able to find anonymized networks with the best trade-off between privacy and utility among all the compared algorithms.

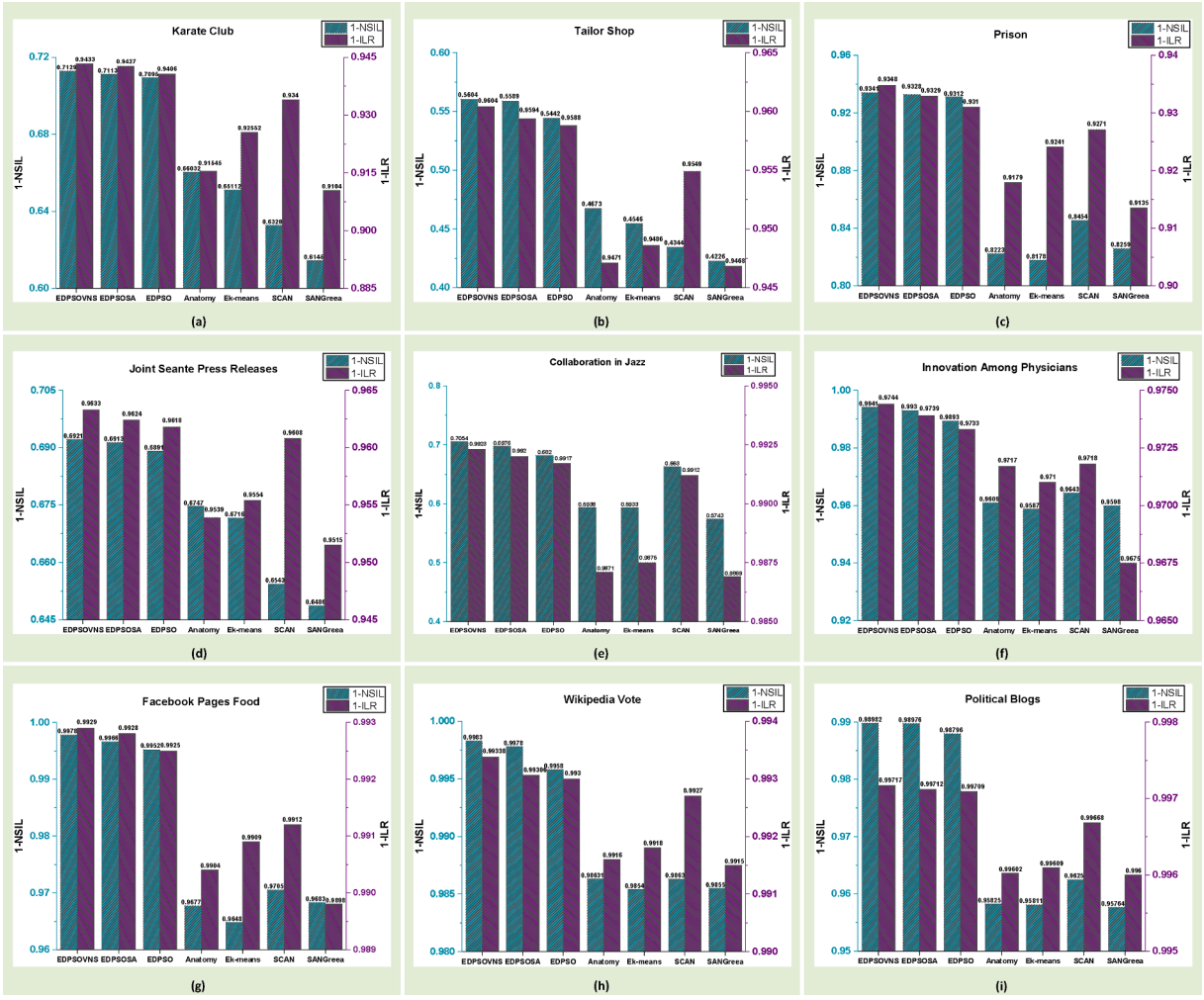

Fig. 7. Comparison of the Methods Regarding the 1-NSIL and 1-ILR Criteria.

5.5. *Statistical analysis of results*

The last part of this section involves using the Friedman test to determine if there are any noteworthy distinctions among the outcomes of the proposed algorithms (EDPSOVNS, EDPSOSA, EDPSO) with other analyzed approaches (DPSO, PSOGA, GA, Anatomy, Ek-means, SANGreea, and SCAN), for all datasets used in this study. The Friedman test is a commonly used non-parametric two-way hypothesis test that aims to determine whether the null hypothesis (H0: all algorithms perform similarly) can be rejected. Therefore, in this study, the Friedman test is applied to the average 1- NSIL values gathered from 30 distinct runs on each dataset, reported in Tables 8–16 and Fig. 7. Table 18 illustrates the average rankings attained by ten studied algorithms across all the datasets.

A lower average rank in the table indicates better performance of the algorithm in achieving higher fitness values (1-NSIL). The statistical Friedman test approach explained in [49] utilizes chi-squared distribution (χ_F^2), as shown in Equations (29). However, this distribution is highly conservative and suggested an alternative statistic based on the F-distribution with $K_{al}-1$ and $(K_{al}-1)(N_D-1)$ degrees of freedom, as shown in Equations (30).

$$\chi_F^2 = \frac{12N_D}{K_{al}(K_{al}+1)}\left[\sum_j R_j^2 - \frac{K_{al}(K_{al}+1)^2}{4}\right] \tag{29}$$

Table 18
Friedman Average Rankings for Ten Algorithms Over All Datasets.

Algorithm	EDPSOVNS	EDPSOSA	EDPSO	DPSO	PSOGA	GA	Anatomy	Ek-means	SANGreea	SCAN
Avg Ranks	1.06	1.94	3.00	4.11	5.78	7.17	7.06	8.39	9.22	7.06

Table 19
chi-squared (χ_F^2), F-distribution (F_F), and P-value of the Friedman Test.

Criteria	χ_F^2	F_F	P-value
Average 1-NSIL	66.94	38.1	6.45E-24

$$F_F = \frac{(N_D - 1)\chi_F^2}{N_D(K_{al} - 1) - \chi_F^2} \tag{30}$$

Here, N_D, K_{al}, and R_j represent the number of datasets, the number of algorithms, and the average ranking of algorithm j, respectively. Table 19 displays the χ_F^2 and F_F values derived from the average rankings of the algorithms and the corresponding P-value.

Hypothesis testing algorithms typically use significance levels of 0.1, 0.05, and 0.01 to determine statistically significant results. Table 19 demonstrates that the null hypothesis is rejected with a 99 % significance level. Such a small p-value confirms that the null hypothesis can be definitively rejected, indicating that the results produced by the analyzed algorithms are significantly different. Additionally, based on the average ranking values presented in Table 18, it can be concluded that the proposed approaches (EDPSOVNS, EDPSOSA, EDPSO) outperform the other algorithms analyzed in almost all of the datasets considered in this study.

6. Conclusion

Our study was driven by the growing need for effective privacy preservation within the realm of social networks, where the volume of users' data continues to expand. We primarily focused on the structural k-anonymity method, which shows promising potential in preserving network structural information.

In this regard, we proposed the EDPSO algorithm to solve the structural k-anonymity problem more efficiently and effectively. This is achieved through the intelligent application of several problem-specific operators by introducing a novel adaptive heuristic algorithm to update the position vector of each particle more effectively, creating a balance between exploration and exploitation capabilities and preventing premature convergence. This algorithm adaptively adjusts the selection probabilities of operators based on the fitness values obtained in the current iteration as well as the average and the count of fitness improvements in previous iterations. Also, two network-specific local search strategies, based on the SA and VNS algorithms, were developed to support the proposed EDPSO algorithm, enhancing both the exploration and exploitation capabilities, preventing the algorithm from getting stuck in local optima, and increasing the convergence rate. In addition, a vector-based solution representation was utilized in this study, which addresses the limitations of the matrix-based solution representation used in prior studies through intrinsic satisfaction of the structural k-anonymity constraint and, thus, preventing demanding validity checks or modification steps. It also leads to the use of fewer numeric values, reducing the complexity of the algorithm and decreasing the hardware resources needed for the execution process.

Finally, to evaluate the performance of these new algorithms, extensive empirical tests were conducted on various real-world networks. Our proposed algorithms, EDPSOVNS and EDPSOSA, were thoroughly evaluated against previous *meta*-heuristic methods (DPSO, PSOGA, and GA) and non metaheuristic algorithms (Anatomy, Ek-means, SCAN, and SANGreea). In this regard, various performance metrics, including 1-NSIL values, convergence rates, and process time, were examined over nine real-world networks from 30 different runs. Our algorithms demonstrated significantly superior results, achieving higher fitness values, faster convergence rates, and generating high-quality k-anonymous networks. Eventually, we provided the Freedman statistical test evidence for the superiority of the proposed algorithms, thus confirming the enhanced capacity of our proposed algorithms to effectively balance data utility and privacy.

While our work yields promising results, it is not devoid of limitations. Generally, *meta*-heuristic-based anonymization algorithms require a significant amount of time to process very large social networks due to their iterative nature. Our proposed method is no exception to this limitation. The computational complexity of the proposed EDPSO algorithm, along with the local search strategies (VNS and SA), is the main bottleneck that necessitates further improvements. Therefore, in our future endeavors, we aim to address these limitations by exploring more efficient and agile methods capable of handling real-world large-scale and dynamic social networks. This will involve incorporating a mix of heuristics into our proposed method. Additionally, we plan to investigate the utilization of hyperheuristic methods to generate effective heuristics. We also suggest exploring the integration of other anonymization techniques with the structural k-anonymity model and introducing a novel information loss criterion to evaluate the effectiveness of the anonymization approach. Additionally, the performance of state-of-the-art *meta*-heuristic methods should be analyzed for anonymizing online social networks.

CRediT authorship contribution statement

Navid Yazdanjue: Conceptualization, Data curation, Methodology, Writing – original draft, Software, Visualization. **Hossein Yazdanjouei:** Writing – review & editing, Conceptualization, Formal analysis, Methodology, Software. **Ramin Karimianghadim:** Writing – review & editing, Data curation. **Amir H. Gandomi:** Project administration, Supervision, Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

We partially acknowledge funding from the Digital Finance CRC, which is supported by the Cooperative Research Centers program, an Australian Government initiative.

Appendix A

This section provides a comprehensive example to clarify the steps of the proposed EDPSO's updating rule mechanisms. In this regard, we will show all the mechanisms on the $\overrightarrow{X_1^t} = [3,1,3,2,1,3,2]$ solution. In the first section, we will provide an example of the velocity vector updating mechanism; then, using the updated velocity and the proposed heuristic algorithm, an example of the position vector updating mechanism is presented in the following.

1) The Velocity Vector Updating Mechanism.

The example shown in Fig. 1 indicates the function of this operator, where it is considered the position vector of the particle P_1 as $\overrightarrow{X_1^{t-1}} = [3,1,3,2,1,3,2]$. Given the personal best position of this particle as $\overrightarrow{Pbest_1} = [1,1,2,2,3,2,3]$, and the global best position of the swarm as $\overrightarrow{Gbest} = [2,3,3,2,3,1,2]$, the $\overrightarrow{Pbest_1} \oplus \overrightarrow{X_1^{t-1}}$ and $\overrightarrow{Gbest} \oplus \overrightarrow{X_1^{t-1}}$ are calculated as follows.

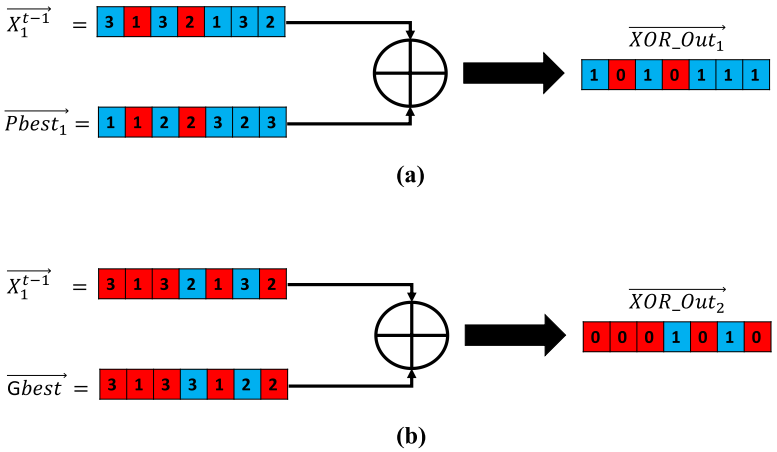

Fig. 1. An illustration of the XOR Operator "⊕".

First, both $\overrightarrow{X_1^{t-1}}$ and $\overrightarrow{Pbest_1}$ vectors are compared with each other element-wise. If the two elements are equal, the corresponding element in the XOR_Out_1 will be set to '0'; otherwise, it will be set to '1'. Therefore, as shown in Fig. 1(a), the third and seventh elements of the output vector are set to '0', while the others are set to '1' (i.e.,$\overrightarrow{XOR\_Out_1} = [1,0,1,0,1,1,1]$). Likewise, as illustrated in Fig. 1(b), the $\overrightarrow{XOR\_Out_2}$ stores the result of $\overrightarrow{Gbest} \oplus \overrightarrow{X_1^{t-1}}$, which is equal to $[0,0,0,1,0,1,0]$.

For instance, assuming that the previous velocity vector of the P_1 equals $\overrightarrow{Vel_1^{t-1}} = [1,0,0,1,1,0,1]$ and the parameters $IW, c_1, c_2, r_1,$ andr_2 have the values defined in Table 1; the updated velocity vector can be calculated as shown in Fig. 2.

Table 1
Constant Parameters of the Example.

Parameter	Value
IW	0.75
c_1	1.45

(*continued on next page*)

Table 1 (*continued*)

Parameter	Value
r_1	0.3
c_2	1.7
r_2	0.8

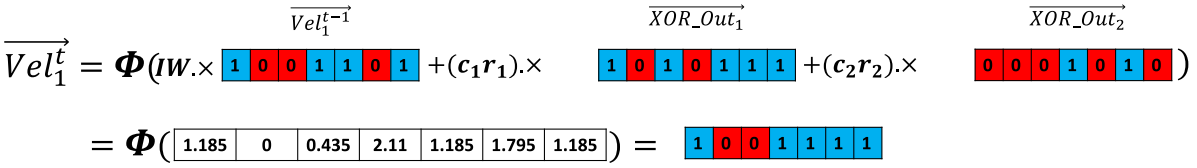

Fig. 2. Velocity Updating.

2) The Proposed Position Vector Updating Mechanism.

Using as example the position vector of P_1(i.e., $\overrightarrow{X_1^{t-1}} = [3,1,3,2,1,3,2]$), and according to its updated velocity vector $\overrightarrow{Vel_1^t} = [1,0,0,1,1,1,1]$, the candidate node whose $n\_id_{c\_node} = 1$ must be updated since its corresponding element in the velocity vector (v_1) is equal to 1. Thereupon, a random node among nodes with $n\_id = 2$ to $n\_id = 7$ is selected randomly. In the following steps, we consider the node with $n\_id_{r\_node} = 5$ as the selected *rand_node*.

Following the example in step one, Fig. 3 illustrates the results of updating $\overrightarrow{X_1^{t-1}}$ by applying the mentioned operators, considering $n\_id_{c\_node}$ and $n\_id_{r\_node}$ as 1 and 5, respectively.

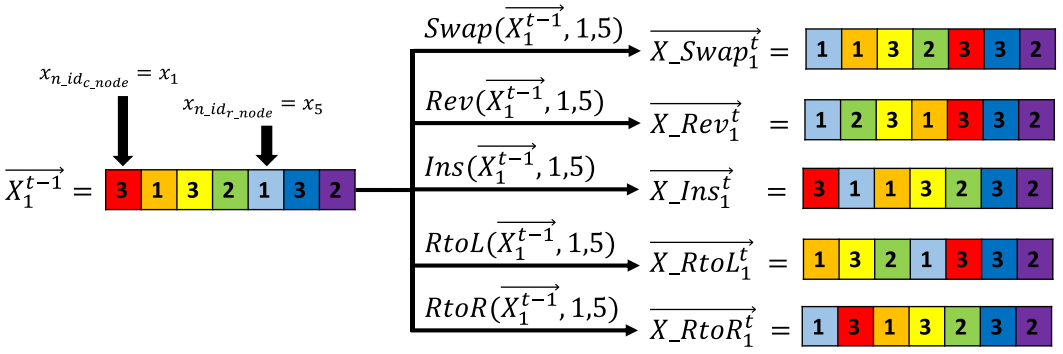

Fig. 3. Applying the Operators to $\overrightarrow{X_1^{t-1}}$.

To elucidate the procedure explained above, we use the results of the previous example (Fig. 3) and calculate the fitness value (1-NSIL) for each of the updated position vectors. As depicted in Fig. 4, the obtained fitness values are stored in $\overrightarrow{FV^t}$.

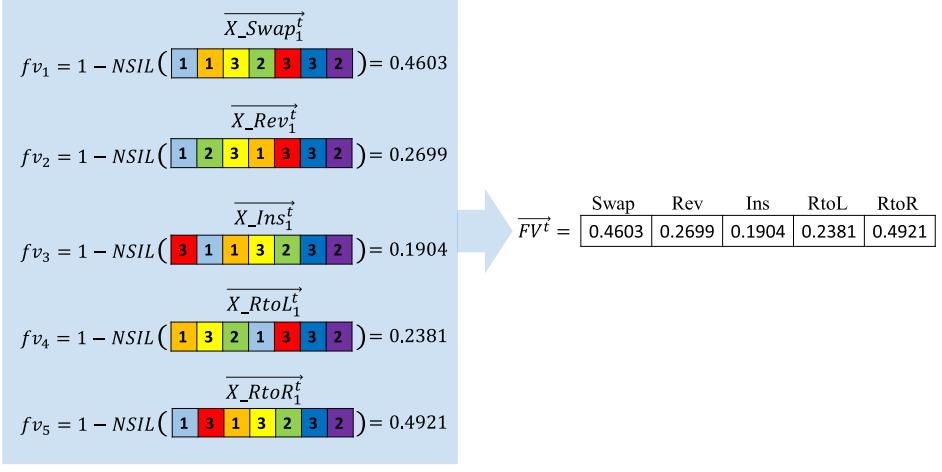

Fig. 4. Calculating Fitness Value for Each Operator Result.

Afterwards, we assume that the maximum number of iterations is set to 20 ($it_{Max} = 20$), and the exponential transformation

function is applied to the obtained fitness value vector ($\overrightarrow{FV^t}$) in iterations 1, 10, and 20. These iteration numbers are equivalent to $\alpha = 1$, $\alpha = 10$, and $\alpha = 20$, respectively. As shown in Fig. 5, the transformed fitness values for each of these α values are stored in $\overrightarrow{TFV^t}$ vector, and subsequently, each of the values in each $\overrightarrow{TFV^t}$ vector is normalized, with the output values being stored in the $\overrightarrow{NFV^t}$ vector. It is quite clear that the higher the α value, the greater the difference among $\overrightarrow{TFV^t}$, and consequently $\overrightarrow{NFV^t}$ values.

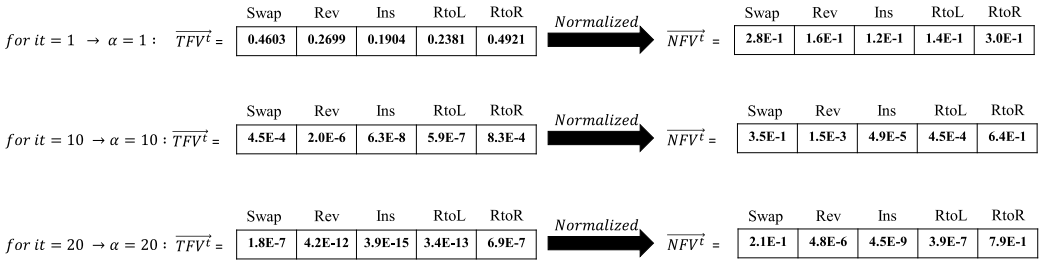

Fig. 5. Calculating the Transformed and Normalized Fitness Value Vectors.

As an example for this step, we assume $\overrightarrow{NoI^{t-1}} = [7,2,6,4,3]$. Considering the obtained $\overrightarrow{FV^t}$ ($\overrightarrow{FV^t} = [0.4603,0.2699,0.1904,0.23811,0.4921]$) and the fitness value of $\overrightarrow{X_1^{t-1}}$ ($1 - NSIL\left(\overrightarrow{X_1^{t-1}}\right) = 0.4127$), both $\overrightarrow{NoI^t}$ and $\overrightarrow{NNoI^t}$ are calculated as shown in Fig. 6.

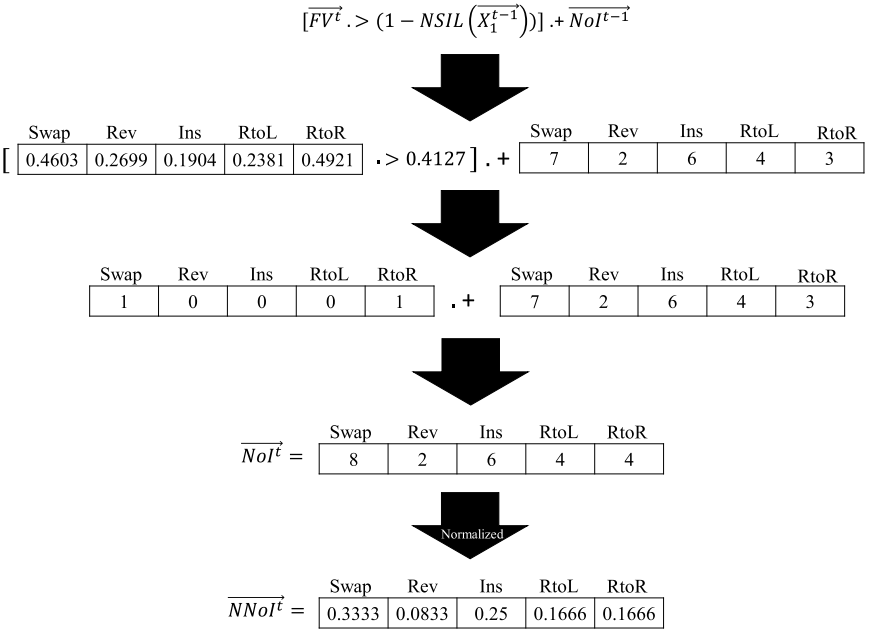

Fig. 6. Calculating the Number of Improvement Vector and Its Normalized Vector.

To further clarify this step, it is useful to consider the example depicted in Fig. 7, assuming $\overrightarrow{AoI^{t-1}} = [0.4396, 0.2286, 0.2922, 0.3664, 0.3546]$. First, the fitness value of $\overrightarrow{X_1^{t-1}}$ ($1 - NSIL\left(\overrightarrow{X_1^{t-1}}\right) = 0.4127$) is subtracted from the values stored in $\overrightarrow{FV^t} = [0.4603, 0.2699, 0.1904, 0.23811, 0.4921]$. Second, the $\boldsymbol{L}$ function is applied to the subtraction result. Eventually, the output values of $\boldsymbol{L}$ function are added to the $\overrightarrow{AoI^{t-1}}$ and the results are stored in $\overrightarrow{AoI^t}$.

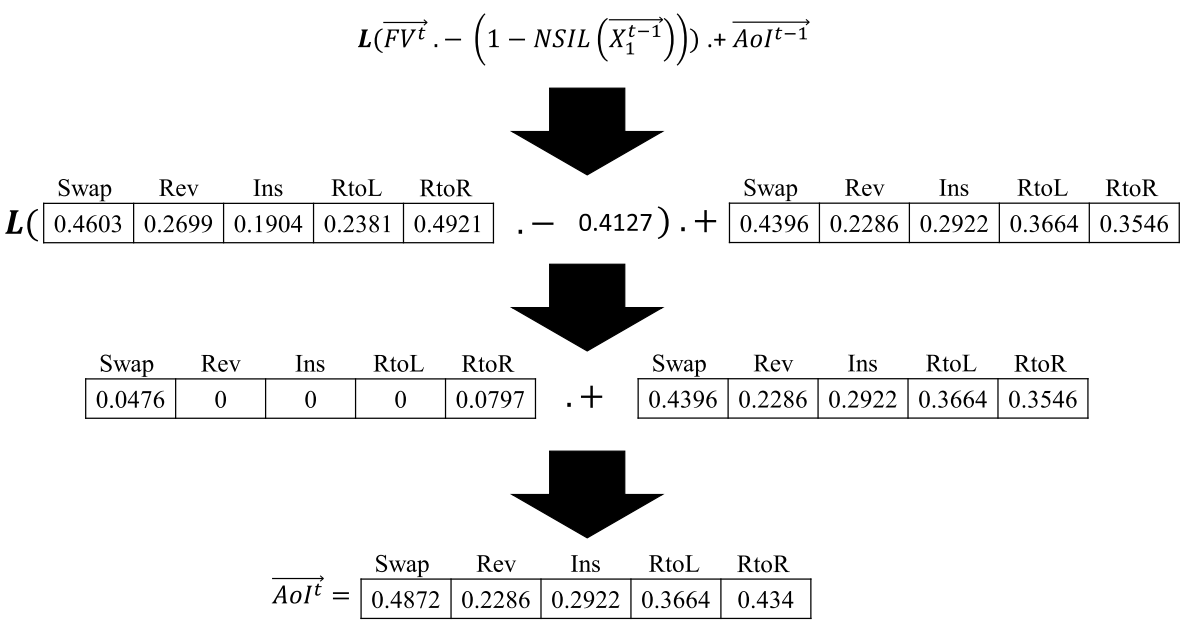

Fig. 7. Calculating the Amount of Improvement Vector.

Following the example presented in Fig. 7 and considering calculated $\overrightarrow{NoI^t} = [8, 2, 6, 4, 4]$, the details of calculating $\overrightarrow{AvgAoI^t}$ and $\overrightarrow{NAvgAoI^t}$ are illustrated in Fig. 8.

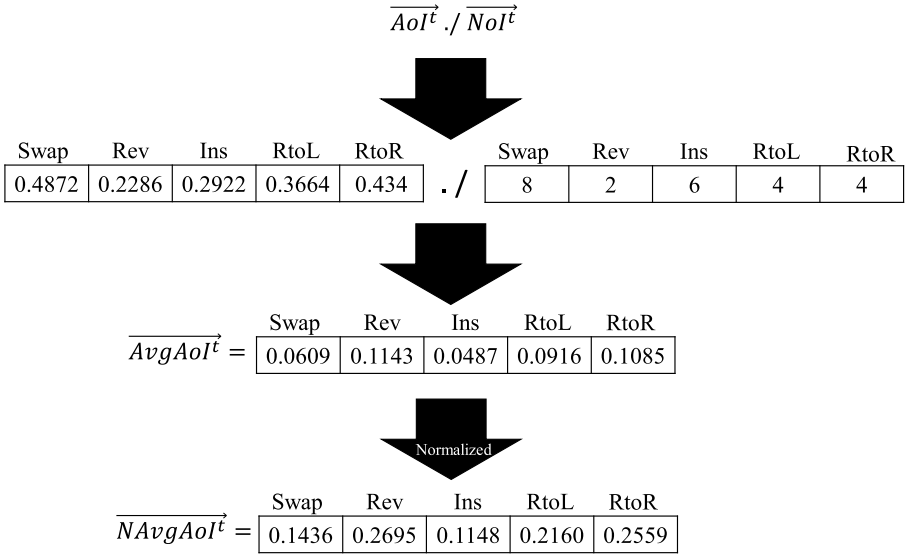

Fig. 8. Calculating the Average Amount of Improvement Vector and Its Normalized Vector.

Continuing the examples given in previous steps, we assume that the current iteration number is equal to 10 and the maximum iteration number is 20 ($it_{Max} = 20$). Therefore, in the 10 th iteration, the values of w_1, w_2, and w_3 are 0.7549, 0.12265, and 0.12265, respectively. Also, the amount of α is equal to 10, and the $\overrightarrow{NFV^t}$ is $[3.5E-1, 1.5E-3, 4.9E-5, 4.5E-4, 6.4E-1]$. Moreover, the $\overrightarrow{NNoI^t}$ and $\overrightarrow{NAvgAoI^t}$ are $[0.3333, 0.0833, 0.25, 0.1666, 0.1666]$ and $[0.1436, 0.2695, 0.1148, 0.2160, 0.2559]$, repectively. Eventually, the $\overrightarrow{OSP^t}$ can be calculated as shown in Fig. 9.

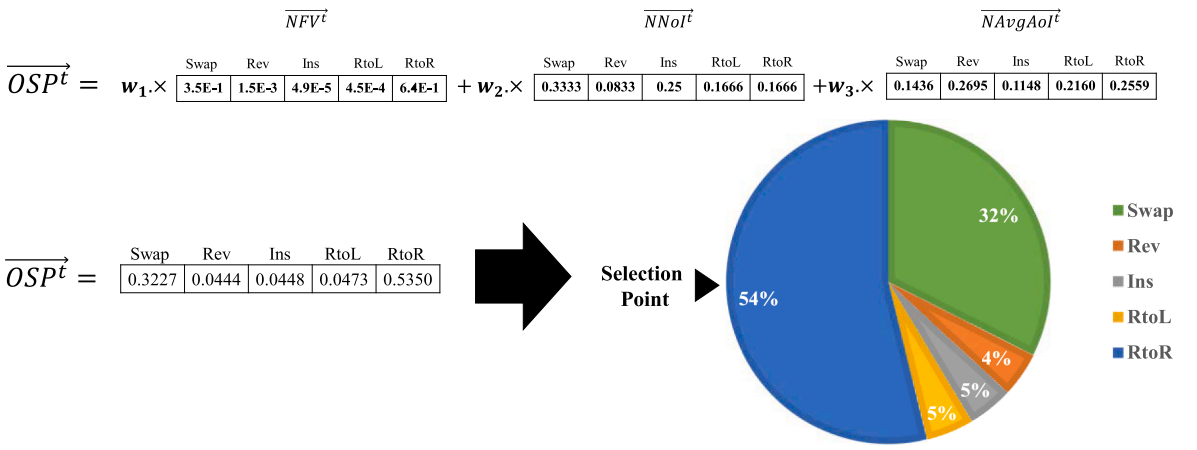

Fig. 9. Calculating the Operator Selection Probability Vector.

References

[1] A. Pfitzmann, M. Hansen, Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology, Version v0 31 (2008) 15.
[2] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, Anonymizing social networks, *Computer science department faculty publication series,* (2007), p. 180.
[3] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra, A survey of graph-modification techniques for privacy-preserving on networks, *Artificial Intelligence Review,* vol. 47, no. 3, (2017), pp. 341-366.
[4] N. Yazdanjue, H. Yazdanjouei, H. Gharoun, M.S. Khorshidi, M. Rakhshaninejad, A.H. Gandomi, A Comprehensive Bibliometric Analysis on Social Network Anonymization, Current Approaches and Future Directions (2023) arXiv preprint arXiv:2307.13179.
[5] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa, Injecting uncertainty in graphs for identity obfuscation, *arXiv preprint arXiv:1208.4145,* 2012.
[6] A. Campan and T. M. Truta, Data and structural k-anonymity in social networks, in *International Workshop on Privacy, Security, and Trust in KDD*, 2008, pp. 33-54.
[7] M. Ros-Martin, J. Salas, J. Casas-Roma, Scalable non-deterministic clustering-based k-anonymization for rich networks, Int. J. Inf. Secur. 18 (2019) 219–238.
[8] E. Zheleva and L. Getoor, Preserving the privacy of sensitive relationships in graph data, in *International workshop on privacy, security, and trust in KDD*, 2007, pp. 153-171.
[9] V. K. Sihag, A clustering approach for structural k-anonymity in social networks using genetic algorithm, in *Proceedings of the CUBE international information technology conference*, 2012, pp. 701-706.
[10] N. Yazdanjue, M. Fathian, and B. Amiri, Evolutionary algorithms for k-anonymity in social networks based on clustering approach, *The Computer Journal,* vol. 63, no. 7, (2020), pp. 1039-1062.
[11] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, L. Jiao, Greedy discrete particle swarm optimization for large-scale social network clustering, Inf. Sci. 316 (2015) 503–516.
[12] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, Optimization by simulated annealing, *science,* vol. 220, no. 4598, (1983), pp. 671-680.
[13] N. Mladenović and P. Hansen, Variable neighborhood search, *Computers & operations research,* vol. 24, no. 11, (1997), pp. 1097-1100.
[14] J. Casas-Roma, An evaluation of vertex and edge modification techniques for privacy-preserving on graphs, J. Ambient Intell. Hum. Comput. (2019) 1–17.
[15] S. Kumar, P. Kumar, Privacy preserving in online social networks using fuzzy rewiring, IEEE Trans. Eng. Manag. (2021).
[16] P. Samarati and L. Sweeney, Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, (1998).
[17] P. Samarati, Protecting respondents identities in microdata release, *IEEE transactions on Knowledge and Data Engineering,* vol. 13, no. 6, (2001), pp. 1010-1027.
[18] L. Sweeney, k-anonymity: A model for protecting privacy, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 10, no. 05, (2002), pp. 557-570.
[19] S.A. Moqurrab, T. Naeem, M.S. Malik, A.A. Fayyaz, A. Jamal, G. Srivastava, UtilityAware: A Framework for Data Privacy Protection in e-Health, Inf. Sci. (2023) 119247.
[20] S. Rajabzadeh, P. Shahsafi, M. Khoramnejadi, A graph modification approach for k-anonymity in social networks using the genetic algorithm, Soc. Netw. Anal. Min. 10 (1) (2020), https://doi.org/10.1007/s13278-020-00655-6.
[21] M. Kiabod, M. Naderi Dehkordi, B. Barekatain, A fast graph modification method for social network anonymization, Expert Syst. Appl. 180 (2021) 115148, https://doi.org/10.1016/j.eswa.2021.115148.
[22] H. Kaur, N. Hooda, H. Singh, k-anonymization of social network data using Neural Network and SVM, Journal of Information Security and Applications 72 (2023) 103382, https://doi.org/10.1016/j.jisa.2022.103382.
[23] M. Kiabod, M.N. Dehkordi, B. Barekatain, K. Raahemifar, FSopt_k: Finding the Optimal Anonymization Level for a Social Network Graph, Appl. Sci. 13 (6) (2023) 3770, https://doi.org/10.3390/app13063770.
[24] J. Medková, J. Hynek, HAkAu: hybrid algorithm for effective k-automorphism anonymization of social networks, Soc. Netw. Anal. Min. 13 (1) (2023) 63.
[25] J. Yan, L. Zhang, Y. Tian, G. Wen, J. Hu, An Uncertain Graph Approach for Preserving Privacy in Social Networks Based on Important Nodes, in, International Conference on Networking and Network Applications (NaNA) 2018 (2018) 107–111, https://doi.org/10.1109/NANA.2018.8648723.
[26] P. Parchas, N. Papailiou, D. Papadias, and F. Bonchi, Uncertain Graph Sparsification, *IEEE transactions on knowledge and data engineering,* vol. 30, no. 12, (2018), pp. 2435-2449, doi: 10.1109/TKDE.2018.2819651.
[27] L. Qu, J. Yang, Y. Wang, Homogeneous network publishing privacy protection based on differential privacy uncertainty, Inf. Sci. 636 (2023) 118925, https://doi.org/10.1016/j.ins.2023.04.004.
[28] A. Campan, T. M. Truta, and N. Cooper, P-sensitive K-anonymity with generalization constraints, *Transactions on data privacy,* vol. 3, no. 2, (2010), pp. 65-89.
[29] T. Tassa and D. J. Cohen, Anonymization of Centralized and Distributed Social Networks by Sequential Clustering, *IEEE transactions on knowledge and data engineering,* vol. 25, no. 2, (2013), pp. 311-324, doi: 10.1109/TKDE.2011.232.
[30] A. Campan, Y. Alufaisan, and T. M. Truta, Preserving communities in anonymized social networks, *Transactions on data privacy,* vol. 8, no. 1, (2015), pp. 55-87.
[31] J. Casas-Roma, F. Rousseau, Community-preserving generalization of social networks, in: In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining,* 2015, pp. 1465–1472.

[32] D. Mohapatra and M. R. Patra, Anonymization of attributed social graph using anatomy based clustering, *Multimedia tools and applications,* vol. 78, no. 18, (2019), pp. 25455-25486, doi: 10.1007/s11042-019-07745-4.
[33] R.K. Langari, S. Sardar, S.A.A. Mousavi, R. Radfar, Combined fuzzy clustering and firefly algorithm for privacy preserving in social networks, Expert Syst. Appl. 141 (2020) 112968.
[34] R. Gangarde, A. Sharma, A. Pawar, R. Joshi, S. Gonge, Privacy preservation in online social networks using multiple-graph-properties-based clustering to ensure k-anonymity, l-diversity, and t-closeness, Electronics (basel) 10 (22) (2021) 2877, https://doi.org/10.3390/electronics10222877.
[35] B. Kadhiwala, S.J. Patel, Privacy-preserving collaborative social network data publishing against colluding data providers, Int. J. Inf. Comput. Secur. 19 (3–4) (2022) 346–378, https://doi.org/10.1504/IJICS.2022.127174.
[36] Z. Wang, et al., Graph-Clustering Anonymity Privacy Protection Algorithm With Fused Distance-Attributes, J. Phys. 2504 (1) (2023) 012058.
[37] R. Gangarde, A. Sharma, and A. Pawar, Enhanced Clustering Based OSN Privacy Preservation to Ensure k-Anonymity, t-Closeness, l-Diversity, and Balanced Privacy Utility, *Computers, materials & continua,* vol. 75, no. 1, (2023), pp. 2171-2190, doi: 10.32604/cmc.2023.035559.
[38] M. Clerc and J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE transactions on evolutionary computation,* vol. 6, no. 1, (2002), pp. 58-73, doi: 10.1109/4235.985692.
[39] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence (The Morgan Kaufmann series in evolutionary computation), Morgan Kaufmann, San Francisco, 2001.
[40] W. W. Zachary, An Information Flow Model for Conflict and Fission in Small Groups, *Journal of anthropological research,* vol. 33, no. 4, (1977), pp. 452-473, doi: 10.1086/jar.33.4.3629752.
[41] B. Kapferer, Strategy and transaction in an African factory: African workers and Indian management in a Zambian town, Manchester University Press, Manchester, 1972.
[42] D. MacRae, Direct Factor Analysis of Sociometric Data, *SOCIOMETRY,* vol. 23, no. 4, (1960), pp. 360-371, doi: 10.2307/2785690.
[43] J. Grimmer, A Bayesian Hierarchical Topic Model for Political Texts: Measuring Expressed Agendas in Senate Press Releases, *Political analysis,* vol. 18, no. 1, (2010), pp. 1-35, doi: 10.1093/pan/mpp034.
[44] P. M. Gleiser and L. Danon, COMMUNITY STRUCTURE IN JAZZ, *Advances in complex systems,* vol. 6, no. 4, (2003), pp. 565-573, doi: 10.1142/S0219525903001067.
[45] J. Coleman, E. Katz, and H. Menzel, The Diffusion of an Innovation Among Physicians, *SOCIOMETRY,* vol. 20, no. 4, (1957), pp. 253-270, doi: 10.2307/2785979.
[46] R. Rossi, N. Ahmed, The Network Data Repository with Interactive Graph Analytics and Visualization 29 (2015), https://doi.org/10.1609/aaai.v29i1.9277.
[47] L. Adamic, N. Glance, The political blogosphere and the 2004 U.S. election: divided they blog, in: In *International Conference on Knowledge Discovery and Data Mining*, 2005, pp. 36–43, https://doi.org/10.1145/1134271.1134277.
[48] R. N. Kackar, Off-Line Quality Control, Parameter Design, and the Taguchi Method, *Journal of quality technology,* vol. 17, no. 4, (1985), pp. 176-188, doi: 10.1080/00224065.1985.11978964.
[49] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *Journal of the american statistical association,* vol. 32, no. 200, (1937), pp. 675-701.