Efficient and Flexible Compression of Very Sparse Networks of Big Data

Carson K. Leung, Fan Jiang, and Yibin Zhang

Abstract In the current era of big data, huge amounts of valuable data and information have been generated and collected at a very rapid rate from a wide variety of rich data sources. Social networks are examples of these rich data sources. Embedded in these big data are implicit, previously unknown and useful knowledge that can be mined and discovered by data science techniques such as data mining and social network analysis. Hence, these techniques have drawn attention of researchers. In general, a social network consists of many users (or social entities), who are often connected by "following" relationships. Finding those famous users who are frequently followed by a large number of common followers can be useful. These frequently followed groups of famous users can be of interest to many researchers (or businesses) due to their influential roles in the social networks. However, it can be challenging to find these frequently followed groups because most users are likely to follow only a small number of famous users. In this chapter, we present an efficient and flexible compression model for supporting the analysis and mining of very sparse networks of big data, from which the frequently followed groups of users can be discovered.

Keywords Big data · Social media analytics · Followship · Data compression · Data mining · Frequent patterns

C. K. Leung (✉)
Department of Computer Science, University of Manitoba, Winnipeg, MB, Canada
e-mail: kleung@cs.umanitoba.ca

F. Jiang
Department of Computer Science, University of Northern British Columbia (UNBC), Prince George, BC, Canada

Y. Zhang
Department of Computer Science, University of Manitoba, Winnipeg, MB, Canada

Department of Computer Science, University of Toronto, Toronto, ON, Canada

M. Çakırtaş, M. K. Ozdemir (eds.), *Big Data and Social Media Analytics*, Lecture Notes in Social Networks, https://doi.org/10.1007/978-3-030-67044-3_9

1 Introduction

Nowadays, big data are everywhere [1]. These big data can be characterized by the well-known 3 V's, 4 V's, 5 V's, etc. (e.g., value, velocity, veracity [2–5], visualization [6, 7], volume). It is because huge volumes of valuable data have been generated and collected at a very high velocity from a wide variety of rich data sources for various real-life applications and services. Examples of these rich data sources include financial services [8], healthcare sectors [9–14], public services [15, 16], and social networks [17–26]. Due to their popularity, social networks can be considered as indispensable products in people's life. In these social networks, people express their personal opinions, browse interesting contents, and follow other favorite individuals (or organizations) on social networks. Consequently, plenty of valuable information and useful knowledge is embedded in social networks. Many researchers, who care about public behavior and psychology (e.g., business, economic, social science), may be interested in mining and analyzing social networks [27, 28] through data science [8, 29, 30], data mining [31–33] and/or machine learning [34, 35] techniques.

In general, *social networks* consist of users or social entities (e.g., individuals, corporations, organizations). Due to the popularity of social networking, the number of users in social networking sites keeps growing. As of July 2020[1] , there were:

- 2.60 billion monthly active users (MAU) in Facebook,
- 2.00 billion MAU in WhatsApp and YouTube,
- 1.30 billion MAU in Facebook Messenger,
- 1.20 billion MAU in WeChat,
- 1.08 billion MAU in Instagram (aka Insta or IG),
- 800 million MAU in TikTok,
- 694 million MAU in Tencent QQ (aka QQ),
- 675 million LinkedIn members[2] ,
- 517–550 million MAU in Sina Weibo and Tencent Qzone,
- 430 million MAU in Reddit,
- 400 million MAU in Telegram[3] ,
- 397 million MAU in Snapchat,
- 367 million MAU in Pinterest, and
- 326 million MAU in Twitter.

The users use these, as well as other similar, social networking sites for a wide variety of purposes. For instance:

[1]https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[2]https://www.linkedin.com/company/linkedin/about/

[3]https://www.statista.com/statistics/234038/telegram-messenger-mau-users/

- Discord, Facebook Messenger, iMessage, KakaoTalk, Line, Skype, Snapchat, Telegram, Tencent QQ, Viber, WeChat, and WhatsApp are mostly used for instant messaging;
- Facebook, LinkedIn, Sina Weibo, Tencent Qzone, Tencent Weibo, Twitter, and Tumblr mostly used for micro-blogging;
- Flickr, Google Photos, Instagram, and Pinterest are mostly used for image sharing; whereas
- TikTok and YouTube are mostly used for video sharing.

Regardless of the purposes why the users use these social networking sites, a common observation on these social network users is that they are linked by some relationships (e.g., friendship, common interest) on the social networks. Examples of these linkages include:

- mutual friendships, and
- "following" patterns or relationships.

For example, in Facebook (as one of the aforementioned social networking sites), users can create a personal profile and add other Facebook users as friends. To elaborate, a user X can add another user Y as a friend by sending Y a friend request. Upon Y's acceptance of X's friend request, X and Y can become mutual friends. In addition to exchanging messages among *mutual friends*, Facebook users can also join common-interest user groups and categorize their friends into different customized lists (e.g., classmates, co-workers). The number of (mutual) friends may vary from one Facebook user to another. It is not uncommon for a user A to have hundreds or thousands of friends.

Besides mutual friendships, another common linkage between users in social networks is a *"following" pattern* or *"following" relationship*, which captures the linkage that a social network user X follows another user Y. Let us elaborate by continuing with the aforementioned example on Facebook users. Although many of the Facebook users are linked to some other Facebook users via the mutual friendship (i.e., if a user X is a friend of another user Y, then user Y is also a friend of user X), there are also situations in which such a relationship is no longer mutual. To handle these situations, Facebook added the functionality of 'subscribe' in 2011, which was relabelled as 'follow' in 2012. Specifically, a user can subscribe or follow public postings of some other Facebook users—usually, famous celebrities, public institutions, product and services, news media, and well-known bloggers—without the need of adding them as friends. A user X may follow other users who do not know user X. In this situation, the link between these social entities is no longer mutual but a directional "following" pattern.

With these "following" patterns, many decisions and recommendations can be made. Consider the following scenario. When many friends of a Facebook user X follow some individual users or groups of famous users, it is likely that user X may also be interested in following these individual users or groups of famous users. This leads to a collection of most-followed pages[4] , which include Facebook accounts of some sports players (e.g., soccer player Ronaldo), popular performers (e.g., musicians, actors and actresses), public figures, and politicians. One can discover groups of famous users (or social entities), which are followed by a significant number of common users. We call these groups of famous users the *frequently followed groups*. Upon the discovery of these frequently followed groups, if any social network user X follows some members of these groups, we could recommend other members of these groups to user X.

To find these frequently followed groups, we need an efficient and flexible way to represent the big social network. However, members in these frequently followed groups are only some tiny percentages of users in the entire social network. Hence, a major challenge is how to capture and represent the "following" relationships and mine frequently followed groups from the captured and represented relationships. Consequently, some logical questions include:

- How to capture and represent "following" relationships among users in a big social network?
- How to efficiently and flexibly compress such representation of "following" relationships among users who are sparsely distributed in a big social network?
- How to effectively mine and analyze the compressed representation to discover those frequently followed groups of users?

To answer these questions, we presented an efficient and flexible compression model for supporting the analysis and mining of very sparse networks of big data, from which the frequently followed groups of users can be discovered. As the current chapter is an extension of our paper titled "flexible compression of big data" [36] published in the *Proceedings of 2019 IEEE/ACM Conference on Advances in Social Networks Analysis and Mining* (*ASONAM'19*), our *key contributions* of the current chapter include:

- multi-line position list word-aligned hybrid compressed bitwise representations—called MPLWAH(k)—of a very sparse but big social network embedded with "following" patterns; and
- an efficient and flexible data science solution for supporting big data mining and analysis on the discovery of frequent "following" groups from the compressed data structure.

The remainder of the current chapter is organized as follows. We first provide some background and discuss some related works. Then, we describe our efficient and flexible compression model for supporting the analysis and mining of very sparse networks of big data, from which the frequently followed groups of users can be discovered. Afterwards, we show our evaluation results on three real-life datasets. Finally, we draw our conclusions and share some ideas for future work.

[4]https://socialblade.com/facebook/top/50/likes

2 Background and Related Work

Compression techniques have been applied to various areas, include compression of data [37, 38], image and video compression [39–42], as well as sequence compressions (e.g., DNA sequences) [43, 44]. Compressed data in these application areas help speed up the information retrieval of data in the areas. However, as they were not designed for social network analysis and mining, most of them cannot be easily adapted to compressing social networking sites.

For related works that focus on compressing networks or graphs [45, 46], most of them aim to compressed the networks for community detection. As such, the compressed networks may not be easily analyzed or mined for frequent patterns or "following" patterns (e.g., frequent "following" groups of famous users).

Regarding related works that focus on compressing social networks for frequent pattern mining and analysis, we [47] presented a social network mining strategy in the IEEE/ACM ASONAM 2016. The strategy applies the word-aligned hybrid (WAH) compression model to reduce the sparsity of "following" data. The idea behind this compression model is to divide the long bitmap into groups of 31 bits, then encode long-run consecutive zero groups (a group without 1 bit) into a compressed word. If a "1"-bit appears in a group, then the group is stored without compression.

In the IEEE/ACM ASONAM 2017, we [48] presented an improved social network "following" pattern solution, which applies an improved position list word-aligned hybrid (IPLWAH) compression model. This compression model encodes both

- long run consecutive zero bits, as well as
- their succeeding group if there are at most k single-bits in the succeeding group.

The solution further reduces the sparsity.

As a logical but non-trivial extension of these works, our MPLWAH(k) compression solution for supporting social network analysis and mining—presented in the current chapter—is based on the above two strategies. We encode long run consecutive zero groups and their succeeding four groups if there are at most four "1"-bits in them.

3 Our Efficient and Flexible Compression Model

To illustrate the key idea behind our efficient and flexible compression model, let us consider a sample social network with many users. Among them, the following six followees (i.e., famous users who are frequently followed other users) are of interest:

- Alice (user #84650),
- Bob (user #169237),

- Carol (user #169248),
- Don (user #253661),
- Eva (user #253667), and
- Fred (user #253673).

Let us consider the following seven followers who follow these six famous users in the social network:

- Gigi (user #7) follows Bob and Eva;
- Henry (user #12308) follows Alice, Carol and Eva;
- Iris (user #90009) follows Alice and Eva;
- John (user #101010) follows Bob, Carol and Eva;
- Kat (user #7600011) follows Alice, Bob, Carol, Don and Fred;
- Leo (user #112012012) follows Alice only; and
- Monica (user #123456789) follows Alice, Bob, Carol, Don, Eva and Fred.

Here, the relationship between Gigi and Bob is unidirectional such that Gigi follows Bob but Bob does not follow Gigi. If Alice happens to follow Leo, then the relationship between Alice and Leo would be (bidirectional) mutual friends such that they would follow each other.

3.1 Graph Representation of a Social Network

A logical representation of a social network is a *graph* $G_1 = (V, E)$ with a set V of vertices and a set E of edges. Each vertex represents an individual user (i.e., a social entity) in the social network, and each edge represents a "following" relationship between a pair of vertices. In theory, there can be at most $(|V|^2 - |V|)$ directional edges with |V| vertices in a social network graph. So, with millions of MAU in many real-life social networking sites, there can be trillions of directional edges. For instance, if $|V| \sim 10^6$, then $|E| \sim 10^{12}$. In reality, most social network users would not follow millions of other users. Hence, the number of edges is usually much smaller.

Reconsider the aforementioned sample social networks. The corresponding graph G_1 contains of millions of vertices representing user #1 to user #123456789 (Monica), including at least $6 + 7 = 13$ vertices representing the six followees and seven followers (i.e., Alice, Bob, ..., Leo, and Monica). The graph G_1 also contains $|E| = 22$ directional edges representing the 22 "following" relationships (e.g., Gigi is following Bob).

Another alternative representation for capturing the "following" relationships would be a *bipartite graph* $G_2 = (U, V, E)$. With it, U represents a set of followers, V represents a set of followees, and E represents a set of "following" relationships. Each edge $e \in E$ captures the information that a follower $u \in U$ follows a followee $v \in V$. The maximum number of edges is bounded above by $|U| \times |V|$.

3.2 Matrix Representation of a Social Network

A third logical representation of a social network is an *adjacency matrix*. When putting followers in the row and followees in the column, each entry in (row r, column c) represents a "following" relationship that r follows c. More precisely, the social network is represented by a Boolean matrix such as a "1" entry in in (row r, column c) represents the presence of a "following" relationship that r follows c. A "0" entry in (row r, column c) represents the absence of any "following" relationship between r and c, meaning that r does not follow c. This Boolean matrix is of size $|V| \times |V|$, where V is the set of users (i.e., social entities) in the social network.

Reconsider the aforementioned sample social networks. With millions of users (e.g., user #1 to at least user #123456789 named Monica), the adjacency matrix can be large. When $|V| \sim 10^6$, $|E| \sim 10^{12}$. It is relieved that, in terms of memory consumption, this matrix can be stored as a Boolean matrix, i.e., a bit matrix or a collection of bit vectors. Usually, the adjacency matrix representing the social networks would be sparse. For instance, reconsider the aforementioned sample social networks. Among all $|V| \times |V|$ bits in the matrix, only 22 of them are "1"-bits representing the 22 "following" relationships in the network.

3.3 Bit Vector Representation of a Follower in a Social Network

A fourth logical representation of a social network is a *collection of bit vectors*. As observed from the previous section, a social network can be represented as a Boolean matrix (or bit matrix), which in turn can be considered as a collection of rows. Each row can then be represented as a bit vector capturing the followees followed by the user for that row. With this representation, each vector is of length $|V|$. Given that the Boolean matrix representing the social network is usually sparse, the corresponding bit vector for each row of the matrix is also expected to be sparse (i.e., sparse bit vector). In terms of memory consumption, the "following" relationships can be captured by $|U|$ bit vectors, each of which of length $|V|$. Here, U represents a set of followers, and V represents a set of users to be followed by followers.

Reconsider the aforementioned sample social networks. The 22 "following" relationships can be captured by 6 bit vectors representing 6 followers (Alice, Bob, ..., Fred). With millions of users (e.g., user #1 to at least user #123456789 named Monica), each bit victor can be long (e.g., $|V| \sim 10^6$ bits). It is relieved that, in terms of memory consumption, the total number "1"-bits among the 6 bit vectors is only 22, which represents the 22 "following" relationships in the social network. For instance, the bit vector for Monica is of length at least 123,456,789 bits. Among these bits, there are only six "1"-bits, which are located at the 84,650th, 169,237th, 169,248th, 253,661st, 253,667th and 253,673rd positions representing the 84,650th, 169,237th, 169,248th, 253,661st, 253,667th and 253,673rd users—namely, Alice, Bob, Carol, Don, Eva and Fred. The remaining bits (at least 123,456,783 bits) on this bit vector for Monica are "0"-bits. This bit vector can be stored in 123,456,789 $\div$ 32 $\approx$ 3,858,025 words (32-bit words). Similar memory requirements for the other five followers, for a total of 6 $\times$ 3,858,025 words = 23,148,150 words (32-bit words).

3.4 Word-Aligned Hybrid (WAH) Compressed Bitmap Representation of a Follower in a Social Network

The (uncompressed) bit vector representing followers in a social network is observed to be usually sparse. In order to alleviate the data sparsity, the bit vector can be compressed by the word-aligned hybrid (WAH) compression so as to save memory usage, and thus enhance mining performance. With this compression model, the bit vector can be divided into groups of 31 bits. Depending on the contents (i.e., composition of "0"- and "1"-bits), these groups can be classified into literal and zero-fill ones. Groups of consecutive zero-fill 31 bits can form some zero-fill words, whereas remaining groups of 31 bits with a mixture of "0"- and "1"-bits form some literal words. More precisely,

- A literal word is represented as a 32-bit word, in which (i) the 1st bit is "0" indicating that it is a literal word and (ii) the remaining 31 bits contain a mixture of "0"- and "1"-bits.
- A zero-fill word is also represented as a 32-bit word, in which (i) the 1st bit is "1" indicating that it is a fill word, (ii) the 2nd bit is "0" indicating that the fill word is a zero-fill word, and (iii) the remaining 30 bits indicate the number of consecutive groups of 31 zeros.

The resulting WAH compressed bitmap consists of a sequence of these literal words and zero-fill words.

3.4.1 An Example of WAH Compressed Bitmap

Reconsider the (uncompressed) bit vector for Monica, in there are six "1"-bits located at the 84,650th, 169,237th, 169,248th, 253,661st, 253,667th and 253,673rd positions representing the 84,650th, 169,237th, 169,248th, 253,661st, 253,667th and 253,673rd users—namely, Alice, Bob, Carol, Don, Eva and Fred. The remaining bits are all "0"-bits. By dividing this bit vector into groups of 31 bits, we observed that there are:

- 2730 groups of consecutive zero-fill 31 bits; succeeded by
- a mixture of "0"- and "1"-bits in the next group of 31 bits (or precisely, a "1"-bit in the 20th position and "0" for the remaining 30 bits);
- 2728 groups of consecutive zero-fill 31 bits;

- another mixture of “0”- and “1”-bits in the next group of 31 bits (or precisely, a “1”-bits in the 8th and 19th positions and “0” for the remaining 29 bits);
- 2722 groups of consecutive zero-fill 31 bits;
- a third mixture of “0”- and “1”-bits in the next group of 31 bits (or precisely, a “1”-bit in the 19th position and “0” for the remaining 30 bits); succeeded by
- a fourth mixture of “0”- and “1”-bits in the next group of 31 bits (or precisely, a “1”-bit in the 25th position and “0” for the remaining 30 bits); and finally succeeded by
- a fifth mixture of “0”- and “1”-bits in the next group of 31 bits (or precisely, a “1”-bit in the 31st position and “0” for the remaining 30 bits).

With the observations, this uncompressed bit vector for Monica can be compressed via the WAH compression into a compressed bitmap consisting of a sequence of 8 literal and zero-fill 32-bit words:

- The first 32-bit word in the sequence is a zero-fill word 10 00 0000 0000 0000 0000 1010 1010 1010, where 1010 1010 1010 $_{(2)} = 2730_{(10)}$ indicates 2730 groups of consecutive zero-fill 31 bits.
- The second 32-bit word in the sequence is a literal word 0 00000 00000 00000 00001 00000 00000 0, where (i) the prefix “0” indicates that it is a literal word and (ii) the “1”-bit is in the 20th position within the suffix 31-bits.
- The third 32-bit word in the sequence is a zero-fill word 10 00 0000 0000 0000 0000 1010 1010 1000, for 1010 1010 1000 $_{(2)} = 2728_{(10)}$ groups of consecutive zero-fill 31 bits.
- The fourth 32-bit word in the sequence is a literal word 0 00000 00100 00000 00010 00000 00000 0, where the two “1”-bits are in the 8th and 19th positions within the suffix 31-bits.
- The fifth 32-bit word in the sequence is a zero-fill word 10 00 0000 0000 0000 0000 1010 1010 0010, for 1010 1010 0010 $_{(2)} = 2722_{(10)}$ groups of consecutive zero-fill 31 bits.
- The sixth 32-bit word in the sequence is a literal word 0 00000 00000 00000 00010 00000 00000 0, where the “1”-bit is in the 19th position within the suffix 31-bits.
- The seventh 32-bit word in the sequence is a literal word 0 00000 00000 00000 00000 00001 00000 0, where the “1”-bit is in the 25th position within the suffix 31-bits.
- The last 32-bit word in the sequence is a literal word 0 00000 00000 00000 00000 00000 00000 1, where the “1”-bit is in the 31st (i.e., last) position within the suffix 31-bits.

This example illustrates that the uncompressed bit vector for Monica (which requires 3,858,025 words) can be compressed into a WAH compressed bitmap consisting of only 8 words.

3.5 Improved Position List Word-Aligned Hybrid (IPLWAH) Compressed Bitmap Representation of a Follower in a Social Network

Compressing a bit vector representing followers in a social network into a WAH bitmap saves memory usage, and thus reduces runtime for social network analysis and social network data mining. Space required can further be reduced by the improved position list word-aligned hybrid (IPLWAH) compression. The key idea is to utilize some portions of the suffix 31-bits within those zero-fill words by encoding a few (i.e., $k \leq 5$) "1"-bits in a literal word succeeding a zero-fill word. There are variants of this compression model, depending on the number k of "1"-bits allowed to be utilized into the zero-fill word. So, with the IPLWAH(k) model, the uncompressed bit vector can be divided into groups of 31 bits, which can be classified into literal and zero-fill ones. Groups of consecutive zero-fill 31 bits can form some zero-fill words, whereas remaining groups of 31 bits with a mixture of "0"- and "1"-bits form some literal words. More precisely,

- A literal word is represented as a 32-bit word, in which (i) the 1st bit is "0" indicating that it is a literal word and (ii) the remaining 31 bits contain a mixture of "0"- and "1"-bits.
- A zero-fill word is also represented as a 32-bit word, in which (i) the 1st bit is "1" indicating that it is a fill word, (ii) the 2nd bit is "0" indicating that the fill word is a zero-fill word, (iii) the next 5 k bits capture the positions of the k "1"-bits in the literal word succeeding the consecutive groups of 31 zeros, and (iv) the last (30 – 5 k) bits indicate the number of these consecutive groups of 31 zeros.

The resulting IPWAH(k) compressed bitmap consists of a sequence of these literal words and zero-fill words.

3.5.1 An Example of IPLWAH(1) Compressed Bitmap

Reconsider the WAH compressed bitmap for Monica. We observed that it is represented as a sequence of 8 words encoding that:

- 2730 groups of consecutive zero-fill 31 bits are succeeded by a literal word with a "1"-bit in the 20th position within the suffix 31-bits; then
- 2728 groups of consecutive zero-fill 31 bits;
- a literal word with two "1"-bits in the 8th and 19th positions;
- 2722 groups of consecutive zero-fill 31 bits are succeeded by a literal word with a "1"-bit in the 19th position within the suffix 31-bits;
- a literal word with a "1"-bit in the 25th position within the suffix 31-bits; and finally succeeded by
- another literal word with a "1"-bit in the 31st (i.e., last) position within the suffix 31-bits.

With the observations, followees of Monica can be compressed via the IPLWAH(1) compression into a compressed bitmap consisting of a sequence of 6 literal and zero-fill 32-bit words:

- The first 32-bit word in the sequence is a zero-fill word 10 10100 0 0000 0000 0000 1010 1010 1010, where (i) prefix "10" indicates that it is a zero-fill word with (ii) 1010 1010 1010 $_{(2)} = 2730_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a literal word with a "1" in the bit position 10,100 $_{(2)} = 20_{(10)}$.
- The second and third 32-bit word in the sequence are the same as the third and fourth words in the WAH compressed bitmap for Monica:
  10 00 0000 0000 0000 0000 1010 1010 1000
  0 00000 00100 00000 00010 00000 00000 0
- The fourth 32-bit word in the sequence is a zero-fill word 10 10011 0 0000 0000 0000 1010 1010 0010, where (i) 1010 1010 0010 $_{(2)} = 2722_{(10)}$ groups of consecutive zero-fill 31 bits are succeeded by (ii) a literal word with a "1" in the bit position 10011 $_{(2)} = 19_{(10)}$.
- The fifth and sixth 32-bit words in the sequence are the same as the seventh and eighth (i.e., last) words in the WAH compressed bitmap for Monica:
  0 00000 00000 00000 00000 00001 00000 0
  0 00000 00000 00000 00000 00000 00000 1

In this example, IPLWAH(1) compressed bitmap combines the first and second words of the corresponding WAH bitmap into a single word as the first word of the IPLWAH(1) bitmap. Similarly, the fifth and sixth words of the corresponding WAH bitmap into another single word as the fourth word of the IPLWAH(1) bitmap. The example illustrates that the uncompressed bit vector for Monica (which requires 3,858,025 words) can be compressed into a WAH compressed bitmap consisting of 8 words, which can further reduced into an IPLWAH(1) compressed bitmap consisting of only 6 words.

3.5.2 An Example of IPLWAH(2) Compressed Bitmap

Note that followees of Monica can also be compressed via the IPLWAH(2) compression into a compressed bitmap consisting of a sequence of 5 literal and zero-fill 32-bit words:

- The first 32-bit word in the sequence is the same as the first word in the IPLWAH(1) compressed bitmap for Monica:
  10 10100 00000 0000 0000 1010 1010 1010
- The second 32-bit word in the sequence becomes a zero-fill word 10 01000 10011 0000 0000 1010 1010 1010, where (i) prefix "10" indicates that it is a zero-fill word with (ii) 1010 1010 1000 $_{(2)} = 2728_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a literal word with two "1"s in bit positions 01000 $_{(2)} = 8_{(10)}$ and 10,011 $_{(2)} = 19_{(10)}$.

- The third, fourth and fifth 32-bit words in the sequence are the same as the fourth, fifth and sixth (i.e., last) words in the IPLWAH(1) compressed bitmap for Monica:
  10 10011 00000 0000 0000 1010 1010 0010
  0 00000 00000 00000 00000 00001 00000 0
  0 00000 00000 00000 00000 00000 00000 1

In this example, IPLWAH(2) compressed bitmap combines the second and third words of the corresponding IPLWAH(1) bitmap into a single word as the second word of the IPLWAH(2) bitmap. The example illustrates that the uncompressed bit vector for Monica (which requires 3,858,025 words) can be compressed into a WAH compressed bitmap consisting of 8 words, which can further reduced into IPLWAH(1) and IPLWAH(2) compressed bitmaps consisting of only 6 words and 5 words, respectively.

3.6 Multi-group Position List Word-Aligned Hybrid (MPLWAH) Compressed Bitmap Representation of a Follower in a Social Network

Compressing a bit vector representing followers in a social network into a WAH or IPLWAH(k) bitmap—for $1 \leq k \leq 5$—saves memory usage, and thus reduces runtime for social network analysis and social network data mining. We observed that IPLWAH(k) utilizes the space of some portions of the suffix 31-bits within those zero-fill words by encoding a few (i.e., $k \leq 5$) "1"-bits in a single literal word succeeding a zero-fill word. A logical question is that what if those few "1"-bits are not in the same single literal word but distributed among consecutive literal words succeeding a zero-fill word?

In response to this question, we present the multi-group position list word-aligned hybrid (MPLWAH) compression, which further reduces the space required by IPLWAH(k). It does so by utilizing some portions of the suffix 31-bits within those zero-fill words to encode a few (i.e., $k \leq 5$) "1"-bits in consecutive literal words succeeding a zero-fill word. Again, there are variants of this compression model, depending on the number k of "1"-bits allowed to be utilized into the zero-fill word. So, with the MPLWAH(k) model, the uncompressed bit vector can be divided into groups of 31 bits, which can be classified into literal and zero-fill ones. Groups of consecutive zero-fill 31 bits can form some zero-fill words, whereas remaining groups of 31 bits with a mixture of "0"- and "1"-bits form some literal words. More precisely,

- A literal word is represented as a 32-bit word, in which (i) the 1st bit is "0" indicating that it is a literal word and (ii) the remaining 31 bits contain a mixture of "0"- and "1"-bits.

- A zero-fill word is also represented as a 32-bit word, in which (i) the 1st bit is "1" indicating that it is a fill word, (ii) the 2nd bit is "0" indicating that the fill word is a zero-fill word, (iii) the next (k–1) bits indicate whether the corresponding "1"-bits are in the succeeding group or not, (iv) the next 5 k bits capture the position of the k "1"-bits in consecutive literal words succeeding the consecutive groups of 31 zeros, and (v) the last (31 – 6 k) bits indicate the number of these consecutive groups of 31 zeros.

The resulting MPWAH(k) compressed bitmap consists of a sequence of these literal words and zero-fill words.

Note that, for a specific variant when $k = 1$, then a zero-fill word for MPWAH(1) is represented as a 32-bit word, in which (i) the first two bits are "10" indicating that it is a zero-fill word, (ii) the next 5 bits capture the position of a "1"-bit in consecutive literal words succeeding the consecutive groups of 31 zeros, and (iii) the last 25 bits indicate the number of these consecutive groups of 31 zeros. Observant readers may notice that this representation of zero-fill word for MPWAH(1) is identical to that for IPWAH(1). However, representations of the zero-fill words for MPWAH(k) and IPWAH(k) are different for other variants when $k > 1$ (but bounded above by 5), i.e., $1 < k \leq 5$. For MPWAH(k), while k is bounded by 5 in theory, but k is bounded above by 4 in practice.

3.6.1 An Example of MPLWAH(2) Compressed Bitmap

Reconsider the IPLWAH(2) compressed bitmap for Monica. We observed that it is represented as a sequence of 5 words encoding that:

- 2730 groups of consecutive zero-fill 31 bits are succeeded by a literal word with a "1"-bit in the 20th position within the suffix 31-bits; then
- 2728 groups of consecutive zero-fill 31 bits are succeeded by a literal word with two "1"-bits in the 8th and 19th positions within the suffix 31-bits;
- 2722 groups of consecutive zero-fill 31 bits are succeeded by a literal word with a "1"-bit in the 19th position within the suffix 31-bits, and another succeeding literal word with a "1"-bit in the 25th position within the suffix 31-bits; and finally succeeded by
- a third literal word with a "1"-bit in the 31st (i.e., last) position within the suffix 31-bits.

With the observations, followees of Monica can be compressed via the MPLWAH(2) compression into a compressed bitmap consisting of a sequence of 4 literal and zero-fill 32-bit words:

- The first 32-bit word in the sequence is a zero-fill word 10 0 10100 0000 0000 0000 1010 1010 1010, where (i) prefix "10" indicates that it is a zero-fill word with (ii) 1010 1010 1010 $_{(2)} = 2730_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a literal word with a "1" in the bit position 10,100 $_{(2)} = 20_{(10)}$.

- The second 32-bit word in the sequence is a zero-fill word 10 0 01000 10011 000 0000 1010 1010 1000, where (i) prefix “10” indicates that it is a zero-fill word with (ii) 1010 1010 1000 $_{(2)} = 2728$ $_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a literal word with a “1” in the bit position 01000 $_{(2)} = 8$ $_{(10)}$ and (iv) another “1” in the bit position 10,011 $_{(2)} = 19$ $_{(10)}$ on (v) the same group/literal word as the one with the “1”-bit in the 8th position (as indicated the “0” in the third bit of the 32-bit word).
- The third 32-bit word in the sequence is a zero-fill word 10 1 10011 11001 000 0000 1010 1010 0010, where (i) prefix “10” indicates that it is a zero-fill word with (ii) 1010 1010 0010 $_{(2)} = 2722$ $_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a (first) literal word with a “1” in the bit position 10,011 $_{(2)} = 19$ $_{(10)}$, which in turn is succeeded by (iv) a second literal word (as indicated the “1” in the third bit of the 32-bit word) with (v) a “1” in the bit position 11,001 $_{(2)} = 25$ $_{(10)}$.
- The fourth 32-bit word in the sequence is the same as the fifth (i.e., last) word in the IPLWAH(2) compressed bitmap for Monica:
  0 00000 00000 00000 00000 00000 00000 1

In this example, MPLWAH(2) compressed bitmap combines the third and fourth words of the corresponding IPLWAH(2) bitmap into a single word as the third word of the MPLWAH(1) bitmap. Here, the two single “1”-bits are on two consecutive literal words succeeded by groups of consecutive zero-fill 31 bits. MPLWAH(2) manages to combine and encode them into a single word. The example illustrates that the uncompressed bit vector for Monica (which requires 3,858,025 words) can be compressed into a WAH, IPLWAH(1) and IPLWAH(2) compressed bitmap consisting of 8, 6 and 5 words, which can further reduced into a MPLWAH(2) compressed bitmap consisting of only 4 words.

3.6.2 An Example of MPLWAH(3) Compressed Bitmap

Note that followees of Monica can also be compressed via the MPLWAH(3) compression into a compressed bitmap consisting of a sequence of 3 literal and zero-fill 32-bit words:

- The first 32-bit word in the sequence is similar but not identical to the first word in the MPLWAH(2) compressed bitmap for Monica in the sense that MPLWAH(3) uses two bits (precisely, the third and fourth bits in the 32-bit word) to indicate (i) whether or not the second “1”-bit is on a literal word succeeding the one containing the first “1”-bit and (ii) whether or not the third “1”-bit is on a literal word succeeding the one containing the second “1”-bit. Hence, the resulting 32-bit word in the sequence is a zero-fill word 10 0 0 10100 00000 00000 0 1010 1010 1010, where (i) prefix “10” indicates that it is a zero-fill word with (ii) 1010 1010 1010 $_{(2)} = 2730$ $_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a literal word with a “1” in the bit position 10100 $_{(2)} = 20$ $_{(10)}$.

- The second 32-bit word in the sequence is similar but not identical to the first word in the MPLWAH(2) compressed bitmap for the same reason. Hence, the resulting second 32-bit word in the sequence is a zero-fill word 10 0 0 01000 10011 00000 0 1010 1010 1000.
- The third 32-bit word in the sequence is a zero-fill word 10 1 1 10011 11001 11111 0 1010 1010 0010 is formed by combining the third and fourth words of MPLWAH(2) bitmap. Here, (i) prefix "10" indicates that it is a zero-fill word with (ii) 1010 1010 0010 $_{(2)}$ = 2722 $_{(10)}$ groups of consecutive zero-fill 31 bits succeeded by (iii) a (first) literal word with a "1" in the bit position 10011 $_{(2)}$ = 19 $_{(10)}$, which in turn is succeeded by (iv) a second literal word (as indicated the "1" in the third bit of the 32-bit word) with (v) a "1" in the bit position 11001 $_{(2)}$ = 25 $_{(10)}$ such that (vi) this second literal word is then succeeded by a third literal word (as indicated the "1" in the fourth bit of the 32-bit word) with (vii) a "1" in the bit position 11111 $_{(2)}$ = 31 $_{(10)}$.

In this example, MPLWAH(3) compressed bitmap combines the third and fourth words of the corresponding MPLWAH(2) bitmap into a single word as the third word of the MPLWAH(3) bitmap. The example illustrates that the uncompressed bit vector for Monica (which requires 3,858,025 words) can be compressed into a WAH, IPLWAH(1), IPLWAH(2) and MPLWAH(2) compressed bitmap consisting of 8, 6, 5 and 4 words, which can further reduced into a MPLWAH(3) compressed bitmap consisting of only 3 words.

3.6.3 Other Examples of MPLWAH(3) Compressed Bitmaps

For completeness and for preparation of our social network mining, we compute the MPLWAH(3) compressed bitmaps for capturing followees of other followers in our aforementioned sample social networks:

- MPLWAH(3) compressed bitmap for Monica can be represented as:
  10 0 0 10100 00000 00000 0 1010 1010 1010
  10 0 0 01000 10011 00000 0 1010 1010 1000
  10 1 1 10011 11001 11111 0 1010 1010 0010
- MPLWAH(3) compressed bitmap for Leo can be represented as:
  10 0 0 10100 00000 00000 0 1010 1010 1010
- MPLWAH(3) compressed bitmap for Kat can be represented as:
  10 0 0 10100 00000 00000 0 1010 1010 1010
  10 0 0 01000 10011 00000 0 1010 1010 1000
  10 0 0 10011 00000 00000 0 1010 1010 0010
  0 0 11111 00000 00000 0 0000 0000 0001
- MPLWAH(3) compressed bitmap for John can be represented as:
  10 0 0 01000 10011 00000 1 0101 0101 0011
  10 0 0 11001 00000 00000 0 1010 1010 0011
- MPLWAH(3) compressed bitmap for Iris can be represented as:
  10 0 0 10100 00000 00000 0 1010 1010 1010

10 0 0 11001 00000 00000 1 0101 0100 1100

- MPLWAH(3) compressed bitmap for Henry can be represented as:
  10 0 0 10100 00000 00000 0 1010 1010 1010
  10 0 0 10011 00000 00000 0 1010 1010 1000
  10 0 0 11001 00000 00000 0 1010 1010 0011
- MPLWAH(3) compressed bitmap for Gigi can be represented as:
  10 0 0 01000 00000 00000 1 0101 0101 0011
  10 0 0 11001 00000 00000 0 1010 1010 0011

These MPLWAH(3) compressed bitmaps for the six followers can be denoted in their hexadecimal notation as follows:

- MPLWAH(3) compressed bitmap for Monica can be denoted as:
  0x8A000AAA 844C0AA8 B9E7EAA2
- MPLWAH(3) compressed bitmap for Leo can be denoted as:
  0x8A000AAA
- MPLWAH(3) compressed bitmap for Kat can be denoted as:
  0x8A000AAA 844C0AA8 89800AA2 8F800001
- MPLWAH(3) compressed bitmap for John can be denoted as:
  0x844C1553 8C800AA3
- MPLWAH(3) compressed bitmap for Iris can be denoted as:
  0x8A000AAA 8C80154C
- MPLWAH(3) compressed bitmap for Henry can be denoted as:
  0x8A000AAA 89800AA8 8C800AA3
- MPLWAH(3) compressed bitmap for Gigi can be denoted as:
  0x84001553 8C800AA3

4 Our Data Science Solution for Social Network Mining on MPLWAH Compressed Bitmaps

To find frequently followed groups, we present a data science solution for social network mining on MPLWAH compressed bitmaps. Specifically, our solution algorithm mines a collection of MPLWAH(k) compressed bitmaps, each bitmap represents information—capturing a list of followees—of a follower.

The algorithm recursively mines the frequently followed groups as follows. It first locates the first followee X (in terms of user ID) in each bitmap, and groups the followers following the same first followee X. If the number of followers in a group of the first followee X (e.g., who has the smallest user ID) meets or exceeds a user-specified frequency threshold, then the followee X is considered a *frequently followed followee*. Then, the algorithm focuses on those followers who are following this frequently followed followee X. The algorithm then locates the second followee Y in the bitmap of these focused followers, and groups the followers following the same second followee Y. If the number of followers in a group of the second followee Y meets or exceeds a user-specified frequency threshold, then the group of followees {X, Y} is considered a *frequently followed group of followees*. This step is repeated recursively to find all the frequently followed groups of followees containing/including the followee X.

After finding all the frequently followed groups of followees containing/including the followee X, the algorithm backtracks to focus on the suffix of each bitmap (i.e., bits representing followees with user IDs bigger than that of X). It locates the first followee Y in each suffix of the bitmap, and groups the followers following the same first followee Y. If the number of followers in a group of the first followee Y (e.g., who has the smallest user ID within the suffix) meets or exceeds a user-specified frequency threshold, then the followee Y is considered a frequently followed followee. Then, the algorithm focuses on those followers who are following this frequently followed followee Y. The algorithm then locates the second followee Z in the bitmap of these focused followers, and groups the followers following the same second followee Z. If the number of followers in a group of the second followee Z meets or exceeds a user-specified frequency threshold, then the group of followees {Y, Z} is considered a frequently followed group of followees. This step is repeated recursively to find all the frequently followed groups of followees containing/including the followee Y.

The aforementioned step (i.e., backtracking to focus on the suffix of each bitmap) is repeated recursively to find all the frequently followed groups of followees containing/including the remaining followees.

4.1 An Example of Discovering Frequently Followed Groups of Followees from a Social Network Represented by a Collection of MPLWAH(3) Compressed Bitmaps

Reconsider our aforementioned social network sample with seven followers (Gigi, Henry, Iris, John, Kat, Leo and Monica). Each follower is following a list of followees represented by MPLWAH(3) compressed bitmaps. To discover frequently followed groups of followees, our data science solution first locates first followee in each bitmap:

- As the first 32-bit word in the MPLWAH(3) compressed bitmap for Monica is 10 0 0 10100 00000 00000 0 1010 1010 1010, her first followee (in terms of user ID) is at position $31 \times 1010\,1010\,1010_{(2)} + 10100_{(2)} = 31 \times 2730 + 20 = 84{,}650$, i.e., Alice (user #84650).
- As the first 32-bit word in the MPLWAH(3) compressed bitmaps for Leo, Kat, Iris and Henry is identical to that for Monica, their first followee is also at position 84,650, i.e., Alice (user #84650).
- In contrast, the first 32-bit word in the MPLWAH(3) compressed bitmaps for John and Gigi is 10 0 0 01000 00000 00000 1 0101 0101 0011, their first followee is

at position $31 \times 1\,0101\,0101\,0011_{(2)} + 01000_{(2)} = 31 \times 5459 + 8 = 169{,}237$, i.e., Bob (user #169237).

Our data science solution then groups the followers who are following the same first followee:

- Henry, Iris, Kat, Leo and Monica follow Alice (user #84650 having the smallest user ID among the followees of the seven followers in this sample social network); whereas
- Gigi and John follow another user having a bigger user ID.

For this sample social network, if the user specifies the frequency threshold at 3, then {Alice} (user #84650) is a *frequently followed followee* because she is followed by 5 followers (Henry, Iris, Kat, Leo and Monica).

Then, our solution focuses on these 5 followers, and observes their second followees: Kat and Monica follow Bob (user #169237 having the second smallest user ID among the followees of these five followers who also follow Alice). However, with only 2 followers, {Alice, Bob} is *not* considered as a frequently followed group of followees.

Continue with the 5 followers afterwards, and observe their next followees (after Bob): Henry and Monica follow Carol (user #169248 having the next smallest user ID among the followees of these five followers who also follow Alice). Hence, {Alice, Carol} is a *frequently followed pair of followees* because Alice & Carol are followed together by 3 followers (Henry, Kat and Monica).

Focusing on these 3 followers, and observes their third followees: Kat and Monica follow Eva (user #253667 having the third smallest user ID among the followees of these three followers who also follow both Alice and Carol). However, with only 2 followers, {Alice, Carol, Eva} is *not* considered as a frequently followed group of followees.

Then, backtrack and continue with the 5 followers (Henry, Iris, Kat, Leo and Monica), and observe their next followees (after Carol): Kat and Monica follow Don (user #253661 having the next smallest user ID among the followees of these five followers who also follow Alice). However, with only 2 followers, {Alice, Don} is *not* considered as a frequently followed group of followees.

Continue with the 5 followers (Henry, Iris, Kat, Leo and Monica), and observe their next followees (after Don): Henry, Iris and Monica follow Eva (user #253667 having the next smallest user ID among the followees of these five followers who also follow Alice). Hence, {Alice, Eva} is a *frequently followed pair of followees* because Alice & Eva are followed together by 3 followers (Henry, Iris and Monica). Among these 3 followers, only Monica follows one more followee (Fred the user #253673). Hence, {Alice, Eva, Fred} is *not* is a frequently followed group.

After discovering all frequently followed groups containing Alice, our data science solution backtracks, locates the next followee (after Alice) in each bitmap, and groups the followers who are following the same next followee:

- Gigi, John, Kat and Monica follow Bob (user #169237 having the smallest user ID among the followees—after Alice—of the seven followers in this sample social network); whereas
- Henry follows a user having a bigger user ID;
- Iris follows another user having an even bigger user ID; but
- Leo does not follow any other user.

Hence, {Bob} is a *frequently followed followee* because he is followed by 4 followers (Gigi, John, Kat and Monica).

Then, our solution focuses on these 4 followers, and observes their second followees: John, Kat and Monica follow Carol (user #169248 having the second smallest user ID among the followees of these four followers who also follow Bob). Hence, {Bob, Carol} is a *frequently followed pair of followees* because Bob & Carol are followed together by 3 followers (John, Kat and Monica). Focusing on these 3 followers, and observes their next few followees: With only Kat and Monica follow Don, {Bob, Carol, Don} is *not* a frequently followed group. Similarly, with only John and Monica follow Eva, {Bob, Carol, Eva} is also *not* a frequently followed group.

Backtrack to focuses on these 4 followers (Gigi, John, Kat and Monica), observe their next few followees:

- With only Kat and Monica follow Don, {Bob, Don} is *not* a frequently followed group.
- However, with Gigi, John and Monica follow Eva, {Bob, Eva} is a *frequently followed pair of followees*. Focusing on these 3 followers (Gigi, John and Monica), with only Monica follows Fred, {Bob, Eva, Fred} is *not* a frequently followed group.

After discovering all frequently followed groups containing Alice and groups containing Bob, our data science solution backtracks, locates the next followee (after Bob) in each bitmap, and groups the followers who are following the same next followee:

- Henry, John, Kat and Monica follow Carol (user #169248 having the smallest user ID among the followees—after Bob—of the seven followers in this sample social network); whereas
- Gigi and Iris follow another user having a bigger user ID; but
- Leo does not follow any other user

Hence, {Carol} is a *frequently followed followee* because she is followed by 4 followers (Henry, John, Kat and Monica). Applying the same procedure recursively, our data science solution finds {Carol, Eva} as a *frequently followed pair of followees*.

After discovering all frequently followed groups containing Alice, groups containing Bob, and groups containing Carol, our data science solution applies the same procedure—i.e., backtracks, locates the next followee after Carol (then Don, etc.) in each bitmap. Consequently, it finds {Eva} as a *frequently followed followee*.

To summarize, by applying the social network mining procedure recursively, our data science solution discovers nine *frequently followed groups*:

- {Alice}, who is followed by Henry, Iris, Kat, Leo and Monica;
- {Alice, Carol}, who are followed by Henry, Kat and Monica;
- {Alice, Eva}, who are followed by Henry, Iris and Monica;
- {Bob}, who is followed by Gigi, John, Kat and Monica;
- {Bob, Carol}, who are followed by John, Kat and Monica;
- {Bob, Eva}, who are followed by Gigi, John and Monica;
- {Carol}, who is followed by Henry, John, Kat, Leo and Monica;
- {Carol, Eva}, who are followed by Henry, John and Monica; and
- {Eva}, who is followed by Gigi, Henry, Iris, John and Monica.

5 Evaluation

To evaluate the performance of our efficient and flexible compression model, we compared the memory usage and runtime of the uncompressed bit vectors with bit compressed bitmaps using WAH, IPLWAH(k) for $1 \leq k \leq 5$, and MPLWAH(k) for $1 \leq k \leq 4$ by using the same datasets used for evaluation in related works. Specifically, we used real-life datasets from the Stanford Large Network Dataset Collection [49] from the Stanford Network Analysis Platform (SNAP):

- ego-Facebook [50], which captures anonymized social circles (i.e., friend lists) containing 88,234 edges among 4039 nodes from Facebook.
- ego-Gplus [50], which captures shared social circles containing 13,673,453 directed edges among 107,614 nodes from Google+ (aka G+). Here, there are 9,168,660 edges among 69,501 nodes in the largest strongly connected component (i.e., strongly connected graph or digraph, in which every node having an in-degree of at least 1).
- ego-Twitter [50], which captures social circles containing 1,768,149 directed edges among 81,306 nodes from Twitter. Here, there are 1,685,163 edges among 68,413 nodes in the largest strongly connected component.

To avoid distraction, among the consistent evaluation results on these three datasets, we show those on the ego-Twitter dataset. In the evaluation, we varied the size of the social network by selecting some subsets of the ego-users.

5.1 *Evaluation on Memory Consumption*

First, we measured the total number of 32-bit words in the uncompressed bit vector vs. the compressed bitmaps using different compression models ranging from WAH, to IPLWAH(k) for $1 \leq k \leq 5$, and to MPLWAH(k) for $1 \leq k \leq 4$. Note that IPLWAH(1) is identical to MPLWAH(1).

Each ego-user in Twitter usually follows many (e.g., hundreds of) other Twitter users as followees, and multiple ego-users may follow the same frequently followed groups of followees. To evaluate the scalability of the compression models, we varied the size of the social network by selecting some subsets of the ego-users. Specifically,

- we randomly selected 20 ego-users as our first data point. Here, the size of the uncompressed bit vector (in the adjacency matrix) of this first data point is around 20 followers × their 3406 followees followed by at least one of those 20 selected followers. In the actual storage, by using a 32-bit word to represent the 'following' relationship, the uncompressed bit vector of this first data point consumes:
  20 followers × $\lceil$their 3406 followees ÷ 32 bits$\rceil$ = 2,140 words.
- Similarly, we randomly selected 50 ego-users as our second data point. Here, the size of the uncompressed bit vector of this second data point is around 50 followers × their 6171 followees followed by at least one of those 50 selected followers. By using a 32-bit word to represent the 'following' relationship, the uncompressed bit vector of this second data point consumes:
  50 followers × $\lceil$their 6171 followees ÷ 32 bits$\rceil$ = 9,650 words.
- We also randomly selected 200 ego-users as our third data point. Here, the size of the uncompressed bit vector of this third data point is around 200 followers × their 22,062 followees followed by at least one of those 200 selected followers. By using a 32-bit word to represent the 'following' relationship, the uncompressed bit vector of this second data point consumes:
  200 followers × $\lceil$their 22,062 followees ÷ 32 bits$\rceil$ = 138,000 words.
- Similarly, we randomly selected 500 ego-users as our fourth data point. Hence, the size of the uncompressed bit vector of this fourth data point is around 500 followers × their 48,418 followees followed by at least one of those 500 selected followers. By using a 32-bit word to represent the 'following' relationship, the uncompressed bit vector of this second data point consumes:
  500 followers × $\lceil$their 48,418 followees ÷ 32 bits$\rceil$ = 757,000 words.
- Finally, we used all 973 ego-users as our fifth data point. Hence, the size of the uncompressed bit vector of this fifth data point is around 973 followers × their 81,306 followees followed by at least one of those 973 followers, for a total of 1,768,149 'following' relationships. By using a 32-bit word to represent the 'following' relationship, the uncompressed bit vector of this second data point consumes:
  973 followers × $\lceil$their 81,306 followees ÷ 32 bits$\rceil$ = 2,472,393 words.

Sizes of uncompressed bit vectors of other data points, as well as of other compressed bitmaps, are shown in Table 1 and plotted in Fig. 1.

Figure 1 shows that, when the number of ego-users increases, the total numbers of words required to capture their 'following' relationships also increases. The numbers grow rapidly for the uncompressed bit vectors. The numbers are lower

Table 1 Evaluation results on the total number of 32-bit words

	#ego-users (i.e., #followers)				
#words	20	50	200	500	973
Uncompressed	2140	9650	138,000	757,000	2,472,393
WAH	1538	6327	40,575	119,198	240,835
IPLWAH(1)	1373	5148	25,822	68,704	132,464
IPLWAH(2)	1302	4738	23,718	64,816	127,115
IPLWAH(3)	1274	4586	23,446	64,472	126,717
IPLWAH(4)	1263	4550	23,409	64,426	126,668
MPLWAH(2)	1198	4121	20,502	58,724	118,565
MPLWAH(3)	1033	3319	17,379	52,860	110,270
MPLWAH(4)	888	2738	15,116	48,551	105,943

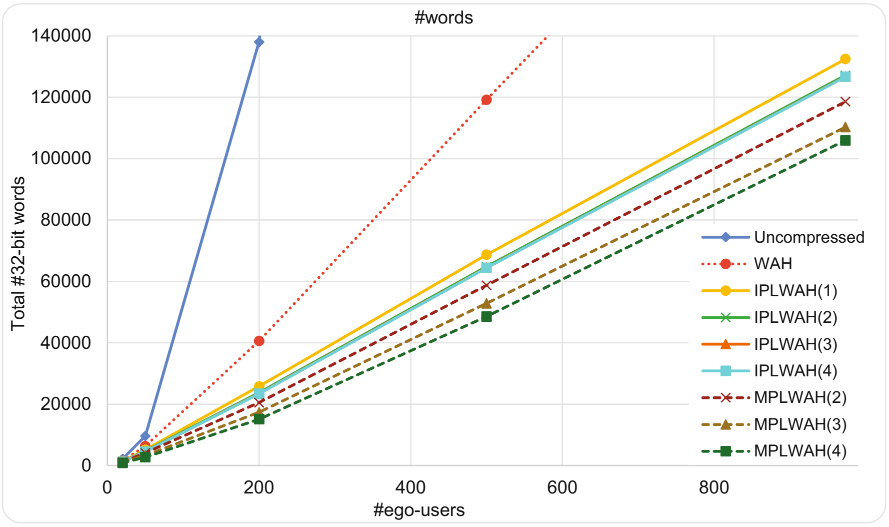

Fig. 1 Evaluation results on the total number of 32-bit words

for WAH. Those for IPLWAH(k) and MPLWAH(k) are much lower. Between the IPLWAH(k) and MPLWAH(k) compression models, the latter provide more effective compression and lead to less memory consumption required by the total numbers of 32-bit words.

Next, in addition to the total number of 32-bit words, we examined the compression ratios by measuring the relative memory consumption required by the uncompressed bit vector vs. the compressed bitmaps using different compression models ranging from WAH, to IPLWAH(k) for $1 \leq k \leq 5$, and to MPLWAH(k) for $1 \leq k \leq 4$. Both Table 2 and Fig. 2 show that, when the number of ego-users increases, the relative memory usage required to capture their 'following' relationships decreases. For high number of ego-users (i.e., followers), the relative memory usage drops from 100% (for the uncompressed bit vectors) to ~9.7% (for

Table 2 Evaluation results on the relative memory usage

	#ego-users				
%memory usage	20	50	200	500	973
Uncompressed	100%	100%	100%	100%	100%
WAH	71.87%	65.56%	29.40%	15.75%	9.74%
IPLWAH(1)	64.16%	53.35%	18.71%	9.08%	5.36%
IPLWAH(2)	60.84%	49.10%	17.19%	8.56%	5.14%
IPLWAH(3)	59.53%	47.52%	16.99%	8.52%	5.13%
IPLWAH(4)	59.02%	47.15%	16.96%	8.51%	5.12%
MPLWAH(2)	55.98%	42.70%	14.86%	7.76%	4.80%
MPLWAH(3)	48.27%	34.39%	12.59%	6.98%	4.46%
MPLWAH(4)	41.50%	28.37%	10.95%	6.41%	4.29%

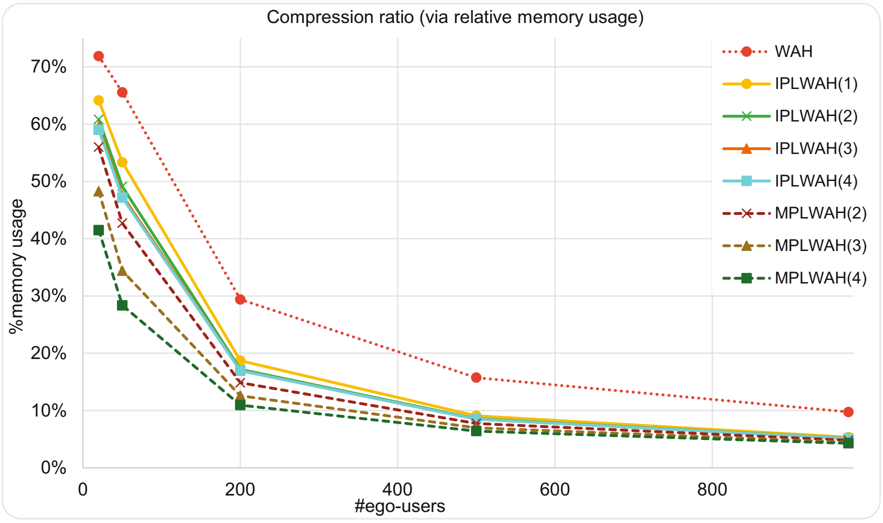

Fig. 2 Evaluation results on the relative memory usage

WAH compressed bitmap), and to ~5% (for IPLWAH compressed bitmaps) and further ~4% (for MPLWAH compressed bitmaps).

5.2 *Evaluation on Runtime*

Besides the memory consumption, we also evaluated the runtime of using our data science solution with different compression models. Experiments were run on a machine with Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz and a memory of 16GB double data rate fourth generation synchronous dynamic random-access memory (DDR4 SDRAM) @ 2400 GHz. Moreover, the runtime includes CPU and

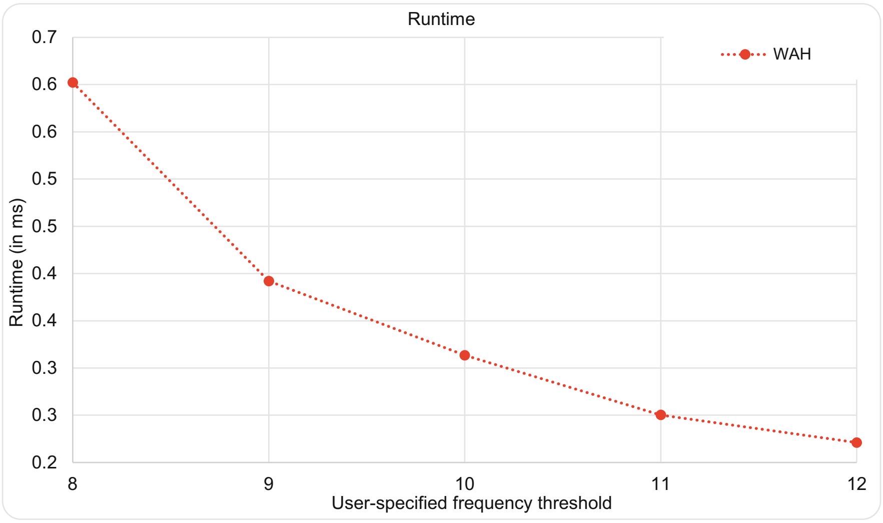

Fig. 3 Evaluation results on the runtime: WAH

I/Os, as well as encoding and decoding (compressing and uncompressing) of the bitmap. Results were the average of 10 runs. Figures 3, 4 and 5 all show that, when the user-specified frequency threshold increases, runtime decreases because the number of discovered frequently followed groups of followees increases. Among the IPLWAH(k) compression models, when k increases, the runtime slightly increases because of slightly more time spent on decoding the more compressed bitmap. This comment also applies to the runtimes of MPLWAH(k) compression models.

5.3 *Evaluation on Scalability*

In addition to evaluating the memory consumption and runtime, we also evaluated the scalability of our compression models and our data science solution for social network mining. Figures 1 and 2 show not only the memory consumption but also the scalability. Specifically, Fig. 1 shows that, when the number of ego-users (i.e., the number of followers) increases, the corresponding number of 32-bit words required to capture the followees of these followers also increases proportionally. This demonstrates the scalability of our compression models.

Figure 2 shows that, when the number of ego-users increases, the corresponding relative memory usage decreases. Moreover, IPLWAH bitmaps are more compressed than WAH bitmaps. Similarly, MPLWAH bitmaps are even more compressed than IPLWAH bitmaps. Hence, MPLWAH bitmaps use the least memory space, whereas IPLWAH bitmaps use more and WAH bitmaps use even more memory space. Among the variants of IPLWAH(k) bitmaps, the higher the k value,

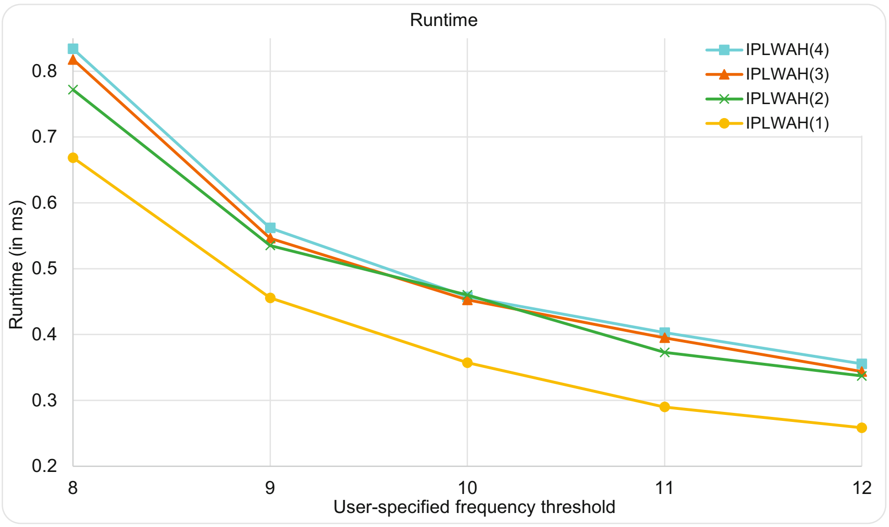

Fig. 4 Evaluation results on the runtime: IPLWAH(k)

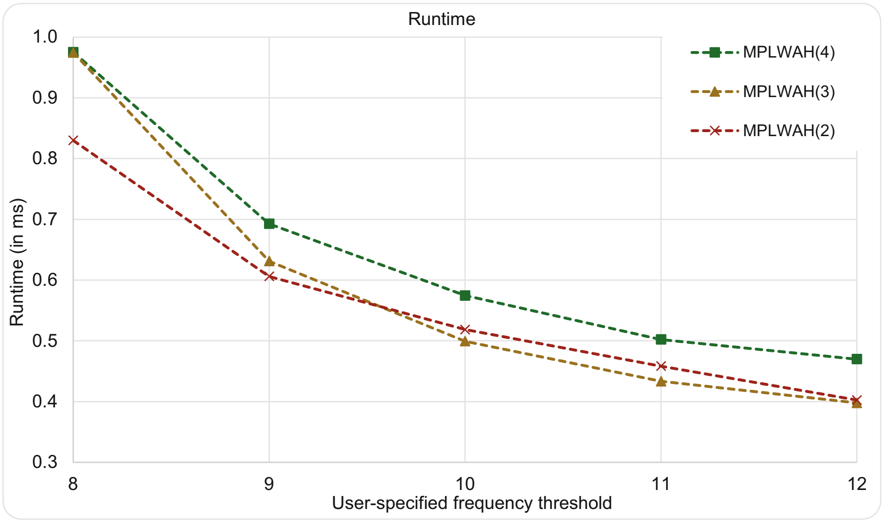

Fig. 5 Evaluation results on the runtime: MPLWAH(k)

the lower is the memory usage. Similar observations apply to the variants of MPLWAH(k) bitmaps: The higher the k value, the lower is the memory usage. This figure demonstrates, once again, the scalability of our compression models.

Figures 6, 7 and 8 shows that, when fixing the user-specified frequency threshold (to 9 followers), an increase in the number of ego-users (i.e., followers) leads to an increase in the number of frequently followed groups of followees followed by these

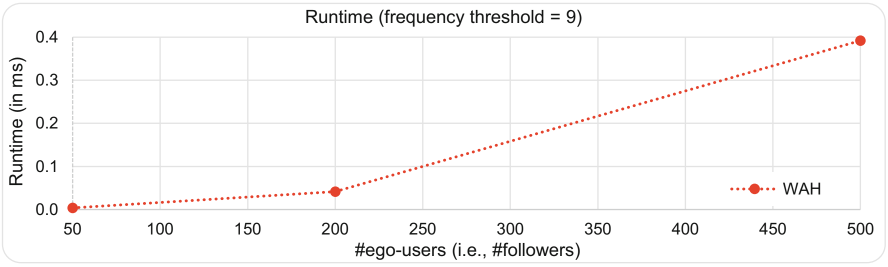

Fig. 6 Evaluation results on the scalability: WAH

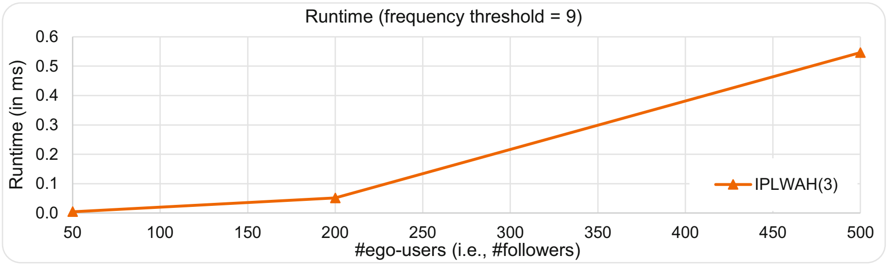

Fig. 7 Evaluation results on the scalability: IPLWAH(3)

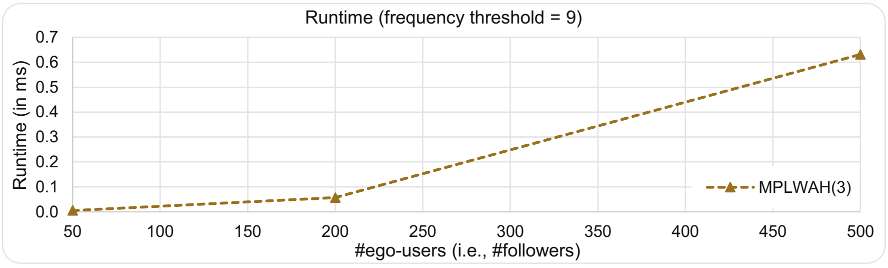

Fig. 8 Evaluation results on the scalability: MPLWAH(3)

followers. Hence, the corresponding runtime increases in running our data science solution to support social network mining in the discovery of frequently followed groups of followees. These figures demonstrate the scalability of our data science solution in supporting social network analysis and mining.

6 Conclusion

Nowadays, social networks are widely used by different users, who often express their personal opinions, browse interesting contents, and follow other favorite individuals on social networks. Consequently, plenty of valuable data can be generated and collected, and information and knowledge embedded in these big social network data can be discovered via social network analysis and mining. One such information is the frequently followed groups of followees because these followees are usually influential and are of interest to many other followers. However, the discovering of these frequently followed groups could be challenging because the corresponding "following" relationships (from which the frequently followed groups can be mined) are sparse in very big social networks.

Hence, in the current chapter, we focused on efficient and flexible compression of very sparse networks of big data. In particular, we first discussed different representations of social networks: graphs (including bipartite graphs), adjacency matrices, and collections of bit vectors. Among them, collections of bit vectors are considered as logical representations of very sparse networks. We then presented several compression models: word-aligned hybrid (WAH), improved position list word-aligned hybrid (IPLWAH(k) for $1 \leq k \leq 5$), and multi-group position list word-aligned hybrid (MPLWAH(k) for $1 \leq k \leq 4$) compression models. Among them, MPLWAH(k) is observed to be an efficient and flexible compression model to effectively capture very sparse networks of big data. In addition, we also described a data science solution in supporting social network analysis and mining to discover frequently followed groups of followees from the MPLWAH(k) compressed bitmaps capturing important information about the "following" relationships between followers who follow their followees. Evaluation results show the effectiveness of the MPLWAH(k) compression models and their corresponding data science solution—in terms of memory consumption, runtime, and scalability—in supporting social network analysis and mining.

As ongoing and future work, we explore models to further compress the social networks, especially very sparse networks of big data. Moreover, we also explore ways to further enhance social network analysis and mining.

Acknowledgement This work is partially supported by (1) Natural Sciences and Engineering Research Council (NSERC) of Canada, and (2) University of Manitoba.

References

1. Xylogiannopoulos, K. F., Karampelas, P., & Alhajj, R. (2019). Multivariate motif detection in local weather big data. In *IEEE/ACM ASONAM 2019* (pp. 749–756). ACM.
2. Han, K., et al. (2019). Efficient and effective algorithms for clustering uncertain graphs. *Proceedings of the VLDB Endowment, 12*(6), 667–680.
3. Ke, X., Khan, A., & Quan, L. L. H. (2019). An in-depth comparison of s-t reliability algorithms over uncertain graphs. *Proceedings of the VLDB Endowment, 12*(8), 864–876.
4. Leung, C. K. (2014). Uncertain frequent pattern mining. In *Frequent pattern mining* (pp. 417–453).
5. Leung, C. K., Mateo, M. A. F., & Brajczuk, D. A. (2008). A tree-based approach for frequent pattern mining from uncertain data. In *PAKDD 2008. LNCS (LNAI)* (Vol. 5012, pp. 653–661).
6. Leung, C. K., & Carmichael, C. L. (2009). FpVAT: A visual analytic tool for supporting frequent pattern mining. *ACM SIGKDD Explorations, 11*(2), 39–48.

7. Leung, C. K., et al. (2020). Big data visualization and visual analytics of COVID-19 data. In *IV 2020* (pp. 387–392). https://doi.org/10.1109/IV51561.2020.00073.
8. O'Halloran, S., et al. (2017). Computational data sciences and the regulation of banking and financial services. In *From social data mining and analysis to prediction and community detection* (pp. 179–209).
9. Gupta, P., et al. (2020). Vertical data mining from relational data and its application to COVID-19 data. In *Big data analyses, services, and smart data* (pp. 106–116). https://doi.org/10.1007/978-981-15-8731-3_8.
10. Leung, C. K., et al. (2020). Data science for healthcare predictive analytics. In *IDEAS 2020* (pp. 8:1–8:10). ACM.
11. Olawoyin, A. M., Leung, C. K., & Choudhury, R. (2020). Privacy-preserving spatio-temporal patient data publishing. In *DEXA 2020, Part II. LNCS* (Vol. 12392, pp. 407–416).
12. Pawliszak, T., et al. (2020). Operon-based approach for the inference of rRNA and tRNA evolutionary histories in bacteria. *BMC Genomics 21*, (Supplement 2), 252:1–252:14.
13. Souza, J., Leung, C. K., & Cuzzocrea, A. (2020). An innovative big data predictive analytics framework over hybrid big data sources with an application for disease analytics. In *AINA 2020. AISC* (Vol. 1151, pp. 669–680).
14. Vural, H., Kaya, M., & Alhajj, R. (2019). A model based on random walk with restart to predict circRNA-disease associations on heterogeneous network. In *IEEE/ACM ASONAM 2019* (pp. 929–932). ACM.
15. Hoang, K., et al. (2020). Cognitive and predictive analytics on big open data. In *ICCC 2020. LNCS* (Vol. 12408, pp. 88–104).
16. Leung, C. K., et al. (2020). Data mining on open public transit data for transportation analytics during pre-COVID-19 era and COVID-19 era. In *INCoS 2020. AISC* (Vol. 1263, pp. 133–144).
17. Fan, C., et al. (2018). Social network mining for recommendation of friends based on music interests. In *IEEE/ACM ASONAM 2018* (pp. 833–840). IEEE.
18. Fariha, A., et al. (2013). Mining frequent patterns from human interactions in meetings using directed acyclic graphs. In *PAKDD 2013, Part I. LNCS (LNAI)* (Vol. 7818, pp. 38–49).
19. Ghaffar, F., et al. (2018). A framework for enterprise social network assessment and weak ties recommendation. In *IEEE/ACM ASONAM 2018* (pp. 678–685). IEEE.
20. Jiang, F., Leung, C. K., & Tanbeer, S. K. (2012). Finding popular friends in social networks. In *CGC 2012* (pp. 501–508). IEEE.
21. Leung, et al. (2018). Mining 'following' patterns from big but sparsely distributed social network data. In *IEEE/ACM ASONAM 2018* (pp. 916–919). IEEE.
22. Leung, C. K., Tanbeer, S. K., & Cameron, J. J. (2014). Interactive discovery of influential friends from social networks. *Social Network Analysis and Mining, 4*(1), 154:1–154:13.
23. Patel, H., Paraskevopoulos, P., & Renz, M. (2018). GeoTeGra: A system for the creation of knowledge graph based on social network data with geographical and temporal information. In *IEEE/ACM ASONAM 2018* (pp. 617–620). IEEE.
24. Rafailidis, D., & Crestani, F. (2018). Friend recommendation in location-based social networks via deep pairwise learning. In *IEEE/ACM ASONAM 2018* (pp. 421–4428). IEEE.
25. Tanbeer, S. K., Leung, C. K., & Cameron, J. J. (2014). Interactive mining of strong friends from social networks and its applications in e-commerce. *Journal of Organizational Computing and Electronic Commerce, 24*(2–3), 157–173.
26. Vaanunu, M., & Avin, C. (2018). Homophily and nationality assortativity among the most cited researchers' social network. In *IEEE/ACM ASONAM 2018* (pp. 584–586). IEEE.
27. Leung, C. K., et al. (2018). Big data analytics of social network data: Who cares most about you on Facebook? In *Highlighting the importance of big data management and analysis for various applications* (pp. 1–15). https://doi.org/10.1007/978-3-319-60255-4_1.
28. Mai, M., et al. (2020). Big data analytics of Twitter data and its application for physician assistants: Who is talking about your profession in twitter? In *Data management and analysis* (pp. 17–32). https://doi.org/10.1007/978-3-030-32587-9_2.
29. O'Halloran, S., et al. (2019). A data science approach to predict the impact of collateralization on systemic risk. In *From security to community detection in social networking platforms* (pp. 171–192).

30. Leung, C. K. (2020). Data science for big data applications and services: Data lake management, data analytics and visualization. In *Big data analyses, services, and smart data* (pp. 28–44). https://doi.org/10.1007/978-981-15-8731-3_3.
31. Das, A., et al. (2019). Water governance network analysis using Graphlet mining. In *IEEE/ACM ASONAM 2019* (pp. 633–640). ACM.
32. Leung, C. K. (2020). Big data computing and mining in a smart world. In *Big data analyses, services, and smart data* (pp. 15–27). https://doi.org/10.1007/978-981-15-8731-3_2.
33. Leung, C. K. (2018). Frequent Itemset mining with constraints. In *Encyclopedia of database systems* (2nd ed., pp. 1531–1536).
34. Arora, U., Paka, W. S., & Chakraborty, T. (2019). Multitask learning for blackmarket tweet detection. In *IEEE/ACM ASONAM 2019* (pp. 127–130). ACM.
35. Leung, C. K., MacKinnon, R. K., & Wang, Y. (2014). A machine learning approach for stock price prediction. In *IDEAS 2014* (pp. 274–277). ACM.
36. Leung, C. K., Jiang, F., & Zhang, Y. (2019). Flexible compression of big data. In *IEEE/ACM ASONAM 2019* (pp. 741–748). ACM.
37. Cao, Y., et al. (2020). Hybrid deep learning model assisted data compression and classification for efficient data delivery in mobile health applications. *IEEE Access, 8*, 94757–94766.
38. Jiang, H., & Lin, S. (2020). A rolling hash algorithm and the implementation to LZ4 data compression. *IEEE Access, 8*, 35529–35534.
39. Birman, R., Segal, Y., & Hadar, O. (2020). Overview of research in the field of video compression using deep neural networks. *Multimedia Tools and Applications, 79*(17–18), 11699–11722.
40. Fu, H., Liang, F., & Lei, B. (2020). An extended hybrid image compression based on soft-to-hard quantification. *IEEE Access, 8*, 95832–95842.
41. Kumar, K. S., Kumar, S. S., & Kumar, N. M. (2020). Efficient video compression and improving quality of video in communication for computer encoding applications. *Computer Communications, 153*, 152–158.
42. Liu, T., & Wu, Y. (2020). Multimedia image compression method based on biorthogonal wavelet and edge intelligent analysis. *IEEE Access, 8*, 67354–67365.
43. Hossein, S. M., et al. (2020). DNA sequences compression by GP^2 R and selective encryption using modified RSA technique. *IEEE Access, 8*, 76880–76895.
44. Kounelis, F., & Makris, C. (2020). Comparison between text compression algorithms in biological sequences. *Information and Computation, 270*, 104466:1–104466:8.
45. Hernández, C., & Marín, M. (2013). Discovering dense subgraphs in parallel for compressing web and social networks. In *SPIRE 2013. LNCS* (Vol. 8214, pp. 165–173).
46. Liu, Z., Ma, Y., & Wang, X. (2020). A compression-based multi-objective evolutionary algorithm for community detection in social networks. *IEEE Access, 8*, 62137–62150.
47. Leung, C. K., et al. (2016). Mining "following" patterns from big sparse social networks. In *IEEE/ACM ASONAM 2016* (pp. 923–930). IEEE.
48. Leung, C. K., & Jiang, F. (2017). Efficient mining of "following" patterns from very big but sparse social networks. In *IEEE/ACM ASONAM 2017* (pp. 1025–1032). ACM.
49. Leskovec, J., & Krevl, A. (2014). *SNAP datasets: Stanford large network dataset collection*. http://snap.stanford.edu/data.
50. McAuley, J., & Leskovec, J. (2012). Learning to discover social circles in ego networks. In *NIPS 2012* (pp. 548–556).