# HUB-GA: A Heuristic for Universal Lists Broadcasting Using Genetic Algorithm

Saber Gholami and Hovhannes A. Harutyunyan

***Abstract*—Broadcasting is a fundamental problem in the information dissemination area. In classical broadcasting, a message must be sent from one network member to all other members as rapidly as feasible. Although this problem is NP-hard for arbitrary graphs, it has several applications in various fields. As a result, the universal lists model, which replicates some real-world restrictions like the memory limits of nodes in large networks, is introduced as a branch of this problem in the literature. In the universal lists model, each node is equipped with a fixed list and has to follow the list regardless of the originator.**

**As opposed to various applications for the problem of broadcasting with universal lists, the literature lacks any heuristic or approximation algorithm. In this regard, we suggest `HUB-GA`: A heuristic for universal lists broadcasting with genetic algorithm, as the first heuristic for this problem. `HUB-GA` works toward minimizing the universal lists broadcast time of a given graph with the aid of genetic algorithm. We undertake various numerical experiments on frequently used interconnection networks in the literature, graphs with clique-like structures, and synthetic instances with small-world model in order to cover many possibilities of industrial topologies. We also compare our results with state-of-the-art methods for classical broadcasting, which is proved to be the fastest model among all. Nevertheless of the substantial memory reduction in the universal list model compared to the classical model, our algorithm finds the same broadcast time as the classical model in diverse situations.**

***Index Terms*—Broadcasting, genetic algorithm, graph theory, heuristic, interconnection networks, universal lists.**

## I. Introduction

With the growth of computer networks in recent years, various problems have received significant attention. On the one hand, the design of interconnection networks is pivotal in several domains, such as high-performance computing (HPC) systems [1], [2]. The overall network performance is determined by the topology used in the interconnection network as well as the routing scheme [3]. On the other hand, studying the algorithms that move data around a network plays a significant role in the scalability of HPC applications [1], [2], and the network's quality of service (QoS). In particular, the time it takes to transmit a message between two communication sites, or the message delay, is one of several factors that affect a network's performance [4]. In this study, we present a method to reduce the message delay in the process of transmission of an identical message over the network from one computer to all other computers.

A crucial problem in this area is *information dissemination*, which is the process of distributing a message throughout a network. This problem has been shown to have several applications in diverse fields, such as parallelism [5], multiprocessor systems [6], and malware diffusion [7], to name a few. One of the fundamental problems in the field of information dissemination is *broadcasting*, where a message initially held by one network member must be distributed to all network members as soon as feasible through communication channels. In each unit of time, each informed member may send the message to one of its uninformed neighbors via a $call$. However, a node may receive messages from multiple senders simultaneously. The process ends after informing all network members.

Achieving optimal broadcasting in a network requires deep knowledge of the network topology. To illustrate, not only do network members need to be aware of the state of their neighbors, but they should also know the origin of the message, which is memory-inefficient. Hence, several settings of this problem have been defined in the literature to simulate real-world networks, i.e., messy broadcasting and the universal list model. In the first model, when a node is informed, it randomly selects one of its neighbors and sends the message to that node [8], [9]. In the latter model, network vertices are equipped with a universal list of their neighbors. When a vertex receives the message, it follows its list and passes the message to the vertices of its list [10]. Nevertheless, in both models, a member needs local knowledge of its neighbors.

The problem of broadcasting with universal list has several applications and could be applied to different networks, such as software defined networks (SDNs) [11]. The SDN environment and its related algorithms have attracted both academia and industrial communities in recent years [12]–[14]. In particular, the problem of broadcasting with universal lists has immediate applications in the update procedure of SDNs [14], [15], which has shown to be highly effective on the performance of the networks and their QoS [14], [16].

The emergence of machine learning has revolutionized research in almost all areas. In this study, we focus on the problem of broadcasting with universal list and propose a reinforcement learning-based algorithm for this problem. Although there are several studies for this problem, our work differs from all of them by that we propose the first heuristic for this problem. Our heuristic is based on genetic algorithm which is proved to be genuinely useful for finding near-optimal solutions for problems with huge search spaces.

Manuscript received August 19, 2022 revised October 17, 2022; approved for publication by Chien-Ming Chen, Division 2 Editor, November 3, 2022.1

S. Gholami and H. A. Harutyunyan are with Computer Science and Software Engineering, Concordia University Ringgold Standard Institution, Guy st, Montreal, Quebec H3G 1M8, Canada, email: sabergholami72@gmail.com and haruty@cs.concordia.ca.

S. Gholami is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000051



1229-2370/22/$10.00 © 2022 KICS

In genetic algorithm, a chromosome is a candidate solution for the problem. In this regard, we first suggest a novel way to encode a broadcast scheme as a chromosome. Then, in our heuristic, we start with generating several random chromosomes. Each chromosome is evaluated according to a *fitness function*. In this study, we have designed two different fitness functions to address two scenarios. Afterwards, we use two operations of genetic algorithm, namely *crossover* and *mutation*, to generate unseen solutions. In each round, the chromosomes with higher fitness scores are more likely to be survived to the next generation. Eventually, near optimal solutions are expected to be generated over multiple generations. The experimental results conducted in this paper demonstrate that our approach finds optimal or near-optimal broadcast time while being tested under divergent circumstances.

The rest of this paper is organized as follows: In Section II, the preliminaries of this work are studied, while the related works are discussed in Section III. We propose our methodology in Section IV, and the numerical results are reported in Section V. Lastly, Section VI concludes this paper.

## II. Problem Definition

### A. Classical Broadcasting

Broadcasting is a problem in which a sender, usually called the $originator$, has a piece of information in a network and wishes to inform all network members of this message. This is achieved by placing a series of calls over the network's communications links while respecting the following conditions [17]:

1) Each call involves exactly two members,
2) Each call needs precisely one unit of time,
3) A vertex can participate in only one call in each unit of time,
4) And a vertex can only make a call toward its adjacent vertices.

The process ends when all members are informed. Robledo et al. proved that the one-to-one communication is faster than a setup in which a sender may update all its neighbours simultaneously [18].

Given an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of vertices, and $E$ is the set of bidirectional communication links, the broadcast time of a vertex $u \in V$ is the minimum time required to complete broadcasting originating from $u$, and it is denoted by $B_{cl}(u, G)$. The broadcast scheme (or broadcast algorithm) of vertex $u$ is the series of calls that are placed in the network starting from vertex $u$. The broadcast time of graph $G$, $B_{cl}(G)$, is the maximum value that the latter variable could take:

$$B_{cl}(G) = \max_{u \in V} \{B_{cl}(u, G)\} \tag{1}$$

Finding $B_{cl}(u, G)$ in an arbitrary graph is an NP-hard problem [19], [20]. Unfortunately, this problem remains NP-hard in more restricted families of networks such as bounded degree [21] and planar and decomposable graphs [22]. The formal definition of the decision version of this problem is as follows:

**Minimum broadcast time (MBT) from [19], Problem [ND49]:**
*Given a graph $G = (V, E)$ with a subset $V_0 \subseteq V$, and a positive integer $K$. Can a message be "broadcast" from the base set $V_0$ to all other vertices in time $K$, i.e., is there a sequence $V_0, E_1, V_1, E_2, V_2, \cdots, E_K, V_K$ such that each $V_i \subseteq V$, each $E_i \subseteq E$, $V_K = V$, and, for $1 \leq i \leq K$, (1) each edge in $E_i$ has exactly one endpoint in $V_{i-1}$, (2) no two edges in $E_i$ share a common endpoint, and (3) $V_i = V_{i-1} \cup \{v : \{u, v\} \in E_i\}$?*

If $|V_0| = 1$, it is the case when broadcasting starts from a single originator, which remains NP-complete [19]. The NP-completeness proof of the decision version is presented in [20], and a reduction from the three-dimension matching (3DM) problem has been utilized for the proof. Also, this problem cannot be approximated in polynomial time for an arbitrary graph within a ratio of $3 - \epsilon$ for any $\epsilon > 0$ unless $P = NP$ [23].

In classical broadcasting, a $call$ is only initiated from an informed vertex to an uninformed vertex which requires comprehensive knowledge over the whole network for every single vertex. In other words, once informed, a vertex sends the message to some of its uninformed neighbors in a particular order. This ordering is different for each originator; that is, a vertex has to maintain $n$ different lists according to $n$ possible originators. Thus, not only each vertex has to know the originator to perform broadcasting, but it also should maintain significantly large lists. This is inefficient in real-world networks due to the increased message bits and the need for larger local memory [10].

### B. Broadcasting with Universal Lists

To handle the above-mentioned drawbacks, another variant of broadcasting is introduced in the literature. Every vertex of the network is given a $universal\ list$, and it has to follow the list, regardless of the originator. So, once a vertex receives the message, it transmits it to its neighbors with respect to the fixed ordering given in the list. There are three sub-models defined using universal lists:

- **Non-adaptive model:** Once a vertex $u$ receives the message, it will re-transmit it to all the vertices on its list, even if $u$ has received it from some of its neighbors. The broadcast time of a graph $G$ following this model is denoted by $B_{na}(G)$.
- **Adaptive model:** Once informed, a vertex $u$ will send the message to its neighbors according to its list, but it will skip the neighbors from which it has received the message. The broadcast time of a graph $G$ following this model is denoted by $B_a(G)$.
- **Fully adaptive model:** Once a vertex $u$ gets informed, it will send the message to its neighbors according to its list, but it will skip all informed neighbors. The broadcast time of a graph $G$ following this model is denoted by $B_{fa}(G)$.

There could be many unnecessary calls under the non-adaptive model, in which the message is returned to the sender.

It makes this model the slowest among all while having the best space complexity as the list is the only thing that should be maintained. Although the number of unnecessary calls drops in the adaptive model compared to the non-adaptive model, many calls might still be unnecessary since only the senders are avoided under this model. Hence, a vertex $u$ may transmit the message to one of its neighbors $v$, which is already informed but has not transmitted the message to $u$ yet. Furthermore, to keep track of the nodes that the message has been received from, each vertex requires additional storage. Lastly, in the fully-adaptive model, a vertex skips all of its informed neighbors. This accelerates the broadcast process compared to the adaptive model. Similar to the classical model, no unnecessary call is made following this model. It is proved that for any graph $G$ [10], [15]:

$$B_{cl}(G) \leq B_{fa}(G) \leq B_a(G) \leq B_{na}(G) \tag{2}$$

Consider graph $G = (V, E)$. A broadcast scheme for the non-adaptive, adaptive, or fully adaptive model can be viewed as a matrix $\sigma_{n\times\Delta}$, where row $i$ of $\sigma$ corresponds to an ordering of the neighbors of vertex $v_i$. Assuming this vertex has degree $d_i$, the cells $\sigma_{[i][d_i+1]}, \sigma_{[i][d_i+2]}, \cdots, \sigma_{[i][\Delta]}$ will be `Null`, where $\Delta = \max\{d_i : 1 \leq i \leq n\}$. Also, denote all possible schemes for a graph $G$ by $\Sigma_{(G)}$. When it is clear from the context, we may omit the subscript $(G)$.

Let $M$ be one of the three models using universal lists ($M \in \{na, a, fa\}$) and fix a graph $G$. For any broadcast scheme $\sigma \in \Sigma$, $B^{\sigma}_{M}(u, G)$ is the time steps needed to inform all the vertices in $G$ from the source $u$ while following the scheme $\sigma$ under model $M$. Moreover, the broadcast time of a graph $G$ under model $M$ with scheme $\sigma$ is defined as the maximum $B^{\sigma}_{M}(u, G)$ over all possible originators. Finally, $B_M(G)$ is the minimum $B^{\sigma}_{M}(G)$ over all possible schemes:

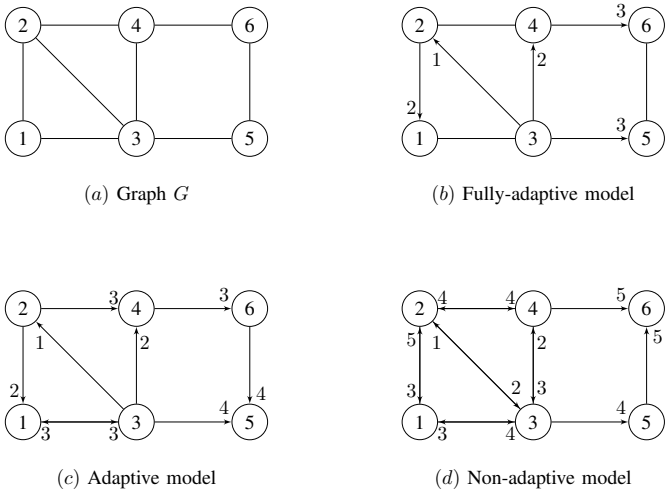
$$\begin{aligned} B^{\sigma}_{M}(G) &= \max_{u\in V}\{B^{\sigma}_{M}(u, G)\} \\ B_M(G) &= \min_{\sigma\in\Sigma}\{B^{\sigma}_{M}(G)\}. \end{aligned} \tag{3}$$

In terms of the space complexity, the relations in (2) are reversed [15]. To illustrate, in the non-adaptive model, the list is the only data structure that should be maintained that needs at most $\sum_{1\leq i\leq n} d_i$ space. In the adaptive model, in addition to the above lists, each vertex $v_i$ should maintain a set of size $d_i$ in order to keep track of the vertices that have sent the message to vertex $v_i$. Therefore, the required space is at most $2 \times \sum_{1\leq i\leq n} d_i$. This is the same for the fully-adaptive model, but those sets store the informed neighbors, which should be updated after each time unit. On the other hand, in the classical model, firstly, those lists differ according to every possible originator. Thus, the required space is $n\times\sum_{1\leq i\leq n} d_i$. Secondly, the message bits increase since the identity of the originator should be carried on with the message.

In summary, in universal lists models, the space complexity is $O(|E|)$, while in the classical model, it requires $O(|V|\cdot|E|)$. Hence, the choice of a suitable model heavily depends on the available resources and the requirements of the network.

TABLE I
AN ORDERING OF VERTICES FOR THE GRAPH OF FIG. 1.

| Sender | Ordering of receivers | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | | |
| 2 | 3 | 1 | 4 | |
| 3 | 2 | 4 | 1 | 5 |
| 4 | 3 | 2 | 6 | |
| 5 | 6 | 3 | | |
| 6 | 5 | 4 | | |



(a) Graph $G$ (b) Fully-adaptive model

(c) Adaptive model (d) Non-adaptive model

Fig. 1. (a) Graph $G$, (b) Broadcast scheme of $G$ under fully adaptive model: $B^{\sigma}_{fa}(G, 3) = 3$, (c) Broadcast scheme of $G$ under adaptive model: $B^{\sigma}_{a}(G, 3) = 4$, and (d) Broadcast scheme of $G$ under non-adaptive model: $B^{\sigma}_{na}(G, 3) = 5$.

### C. An Example

Table I shows an arbitrary broadcast scheme $\sigma$ for the graph given in Fig. 1(a). Suppose vertex 3 is the originator. Now, the broadcast processes starting from this vertex under all three models are described:

- **Fully-adaptive:** At time $t = 1$, vertex 3 sends the message to vertex 2 according to its list. At time $t = 2$, vertex 3 sends the message to vertex 4, and vertex 2 (by skipping the informed vertex 3) will send the message to vertex 1. At time $t = 3$, the rest of the vertices on the list of vertices 1 and 2 are already informed; thus, vertices 1 and 2 will remain idle. Also, vertex 3 will skip vertex 1 (since it is informed) and send the message to vertex 5, whereas vertex 4 will skip vertices 3 and 2 from its list and send the message to vertex 6. Since all 6 vertices are informed, $B^{\sigma}_{fa}(G, 3) = 3$. See Fig. 1(b).
- **Adaptive:** The broadcast process is the same for time units $t = 1, 2$. However, at time $t = 3$, vertex 1 sends the message to vertex 3, while vertex 2 sends the message to vertex 4 because vertex 2 has not received the message from vertex 4. Also, vertex 3 sends the message to vertex 1 for the same reason, whereas vertex 4 is the only vertex that informs an already uninformed vertex 6. Since vertex 5 is still uninformed, the process must continue for another time unit $t = 4$, in which vertex 3 and 6 hit the same target (vertex 5) which ends the process: $B^{\sigma}_{a}(G, 3) = 4$. See Fig. 1(c).
- **Non-adaptive:** This model is the slowest model yet the

TABLE II
AN OPTIMAL BROADCAST SCHEME FOR THE GRAPH GIVEN IN FIG. 1 UNDER FULLY-ADAPTIVE MODEL.

| Sender | Ordering of receivers | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | | |
| 2 | 1 | 4 | 3 | |
| 3 | 4 | 1 | 2 | 5 |
| 4 | 6 | 3 | 2 | |
| 5 | 6 | 3 | | |
| 6 | 4 | 5 | | |

easiest to follow because no vertex is being skipped. Therefore, once a vertex gets informed at time $t$, it will send the message to its neighbours at time $t+1, t+2, \cdots$, following its list given in $\sigma$. For instance, since vertex 3 is the originator who has the message at time $t = 0$, it will pass the message to vertices 2, 4, 1, and 5 at time units $t = 1, 2, 3, 4$, respectively. Also, since vertex 2 is informed at time $t = 1$, it will send the message to vertex 3,1, and 4 during time units $t = 2, 3, 4$, respectively. Following this process for other vertices is the same, therefore is omitted. Eventually, this process ends at time unit 5: $B^{\sigma}_{na}(G, 3) = 5$. See Fig. 1(d).

Two interesting points could be highlighted from this example: Firstly, while a broadcast scheme $\sigma$ might be optimal for a particular model, it could be inefficient under another model. For instance, Table I is optimal under model *fa* and vertex 3 as the originator, whereas it is not optimal under model *na* and the same originator. Secondly, a broadcast scheme could be tailored for a particular originator, whereas it is not efficient once other vertices are selected as the message originator. For example, by following Table I from vertex 6 as the originator, the fully-adaptive broadcasting will be finished in four time units. Even though it is possible to come up with another scheme that reduces this time to 3. The tough task is to design a scheme $\sigma'$ that yields a broadcast time of 3 from not only vertex 6 but all vertices of this graph under the fully-adaptive model. This optimal broadcast scheme is given in Table II. Following this scheme $B^{\sigma'}_{fa}(G) = 3$. Also, Table III gives an optimal broadcast scheme under the adaptive model with $B^{\sigma'}_{a}(G) = 3$, whereas Table IV is an optimal scheme under the non-adaptive model with $B^{\sigma'}_{na}(G) = 4$ (achieving the non-adaptive broadcast time of 3 for this graph is impossible).

Guessing an optimal scheme out of many possible solutions is an impossible task, even for a small graph. Besides, our results demonstrate that simple heuristics, such as sorting the lists based on the degree, are not effective as well. In this study, we propose a heuristic based on the Genetic algorithm that tackles this problem. In particular, our heuristic tries to find the minimum $B^{\sigma}_{M}(G)$ over several possibilities of $\sigma$ for a given graph $G$ under model $M \in \{na, a, fa\}$.

## III. RELATED WORKS

We divide this section into three categories: studies related to the classical broadcast problem, related works of the universal lists model, and problems and challenges of the genetic algorithm.

TABLE III
AN OPTIMAL BROADCAST SCHEME FOR THE GRAPH GIVEN IN FIG. 1 UNDER ADAPTIVE MODEL.

| Sender | Ordering of receivers | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | | |
| 2 | 4 | 3 | 1 | |
| 3 | 5 | 1 | 2 | 4 |
| 4 | 2 | 3 | 6 | |
| 5 | 3 | 6 | | |
| 6 | 5 | 4 | | |

TABLE IV
AN OPTIMAL BROADCAST SCHEME FOR THE GRAPH GIVEN IN FIG. 1 UNDER NON-ADAPTIVE MODEL.

| Sender | Ordering of receivers | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | | |
| 2 | 4 | 3 | 1 | |
| 3 | 1 | 5 | 2 | 4 |
| 4 | 3 | 6 | 2 | |
| 5 | 6 | 3 | | |
| 6 | 4 | 5 | | |

### A. Classical Broadcasting

Since the classical broadcast problem is a fundamental NP-hard problem, there are three general directions to follow.

Firstly, some researchers designed exact approaches. This includes a dynamic programming algorithm suggested in [24] and the ILP models suggested in [25]. Currently, the ILP method suggested in [25] is believed to be the best exact method. These exponential methods can only effectively solve this problem for networks with up to around 50 nodes in a reasonable amount of time, making them unsuitable for use in real-world networks.

Secondly, some researchers tried to solve the problem optimally for a particular family of networks. Slater et al. [20] made the first step in this category and suggested a linear algorithm for trees. Other researches include grid and tori [26], cube connected cycle [27], and shuffle exchange [28]. Later on, more algorithms for non-trivial topologies such as unicyclic graphs [29], fully connected trees [30], and tree of cycles [31] were developed, to name a few.

Finally, there is a long list of heuristics and approximation algorithms for this problem. Several heuristic [25], [32]–[35] and approximation algorithms for arbitrary graphs [36]–[38], or a specific family of graphs [39], [40] have been proposed in the literature. The algorithm with the best approximation ratio is proposed in [41] which guarantees a $(\frac{\log |V|}{\log \log |V|})$-approximation for this problem. For survey papers, we refer to [17], [42]–[44].

### B. Broadcasting with Universal Lists

Slater et al. [20], who informally posted the problem of broadcasting with universal lists, established $B_{cl}(T) = B_a(T)$ for any tree $T$. However, in [45], where the authors proposed an algorithm for optimal broadcasting in trees under the adaptive model, the first explicit statement of the problem of broadcasting with universal lists was provided. In order to construct the optimal broadcast scheme for cycles and grids under both models, Diks and Pelc [10] separated between non-adaptive and adaptive models. Additionally, they provided

TABLE V
NOTATION TABLE.

| Variable | Definition |
|---|---|
| $G$ | An undirected connected graph |
| $V$ | Set of vertices of graph $G$ |
| $E$ | Set of edges of graph $G$ |
| $n$ | Number of vertices in graph $G$ |
| $u \in V$ | A vertex in graph $G$ |
| $B_{cl}(u, G)$ | Classical broadcast time of vertex $u$ in graph $G$ |
| $B_{cl}(G)$ | Classical broadcast time of graph $G$ |
| $na$ | Non-adaptive broadcasting with universal lists |
| $a$ | Adaptive broadcasting with universal lists |
| $fa$ | Fully-adaptive broadcasting with universal lists |
| $M \in \{na, a, fa\}$ | Model of broadcasting with universal lists |
| $d_i$ | Degree of vertex $i$ |
| $\delta$ | Minimum degree of graph |
| $\Delta$ | Maximum degree of graph |
| $\Sigma_{(G)}$ | All possible schemes for graph $G$ |
| $\sigma_{n\times\Delta}$ | A broadcast scheme with $n$ rows and $\Delta$ columns |
| $B^{\sigma}_{M}(u, G)$ | Broadcast time of vertex $u$ following $\sigma$ under $M$ |
| $B^{\sigma}_{M}(G)$ | Broadcast time of graph $G$ following $\sigma$ under $M$ |
| $B_M(G)$ | Broadcast time of graph $G$ under $M$ |
| $g^{(i)}$ | Gene $i$, an ordering of neighbors of vertex $i$ |
| $\|p\|$ | Population size |
| $f_1(\sigma)$ | Fitness function, maximum broadcast time |
| $f_2(\sigma)$ | Fitness function, average broadcast time |
| $K$ | Number of participants of a $K$-way tournament |
| $c_p$ | Crossover point |
| $p_1$ | A chromosome: Parent number 1 |
| $c_1$ | A chromosome: Offspring number 1 |
| $g^{(i)}(p_j)$ | The $i^{\text{th}}$ gene of parent $j$ |
| $S_t$ | Stability variable |

upper bounds for complete graphs and tori under both adaptive and non-adaptive models. Also, the fully-adaptive model and its applications is discussed in [15], where the optimal broadcast scheme for trees, grids, and cube-connected cycles, in addition to a tight upper bound for tori, are provided.

In [46], the non-adaptive upper bounds of tori were enhanced. Later, Kim and Chwa [47] developed non-adaptive broadcast algorithms for paths and grids. They also developed upper bounds for complete graphs and hypercubes under non-adaptive models. The optimal broadcast time of hypercubes under the fully-adaptive model is established in [11]. Also, a general upper bound on arbitrary graphs as well as a tight lower bound for trees under the non-adaptive model were presented in [48]. They also suggested a polynomial-time algorithm for determining $B_{na}(T)$ for any tree $T$.

### C. Genetic Algorithm

Genetic algorithm (GA) has attracted an enormous amount of attention from researchers after being proposed by J. H. Holland [49]. The studies in this domain could be divided into several categories.

**Enhancing the algorithm:** Several researchers have tried to enhance the GA itself by proposing different approaches for various operations of GA, such as crossover [50]–[52], mutation [53], selection [54]–[59], or the stopping criteria [60]. For survey papers, we refer to [61]–[66].

**Graph-related problems:** GA helped researchers to develop heuristics for different problems. As far as graph-related problems are concerned, the graph coloring problem and some solutions working with GA are discussed in [67], [68], while some algorithms for the graph partitioning with GA are proposed in [69], [70]. Several algorithms for max-cut [71], max clique [72], [73], and max-cut clique [74] problems are proposed in the literature. Palmer and Kershenbaum [75] came up with a novel idea for representing trees in GA and showed that the proposed encoding could be effective for several problems.

**Routing:** As mentioned in [76], GA has been used in various network routing protocols. In particular, in [77], a GA approach for the shortest path problem under two different settings has been proposed. Also in [78], the network design problem with the goal of optimizing vehicle travel distance has been studied with GA, while unmanned aerial vehicle (UAV) networks [79], and energy-efficient resource allocation [80] have been studied using GA. digital data service (DDS), which is a famous communication service, is studied in [81] where the authors proposed a GA for the Steiner-tree problem that is tightly connected to the design of DDS networks.

In [82] the performance of several routing algorithms in wireless sensor networks (WSNs) are compared, such as GA, Dijkstra algorithm, ad hoc on-demand distance Vector (AODV), GA-based AODV routing (GA-AODV), grade diffusion (GD) algorithm, directed diffusion algorithm and GA combined with the GD algorithm. They have also studied the performance of those algorithms in the presence of faulty nodes and found out that combining GA with other algorithms is usually useful. In [83], efficient data transmission through WSNs with multiple objective genetic algorithm is studied by proposing a compressive sensing based algorithm. Multicast routing problem and an algorithm based on GA is suggested in [84]. A dynamic source routing protocol in mobile ad hoc network (MANET) is suggested in [85] which utilizes genetic algorithm-bacterial foraging optimization (GA-BFO). The routing problem in MANETs is also discussed in [86] using GA, considering energy consumption as one of the foremost vital limitations in MANETs. Furthermore, the effectiveness of GA-based routing solutions in vehicular ad-hoc networks (VANETs) is discussed in [87]. Also, QoS routing using GA in VANETs [88] and WSNs with applications in smart grids [89] is a well-studied problem in the literature.

In [90], the goal is to enhance energy efficiency with the lifespan of sensor nodes. They proposed an energy effective routing protocol, low energy adaptive clustering hierarchy (LEACH) in addition to an optimization GA. The authors of [91] proposed a GA for routing in WSNs with the aim of minimizing the energy consumption. The problem of routing in energy harvesting-wireless sensor networks (EH-WSN) has been studied in [92] using GA. Prolonging the lifetime of nodes by reducing energy consumption in WSNs is studied in [93] where GA has been used for the routing algorithm.

**Broadcasting:** In the area of broadcasting, Hoelting et al. proposed a GA for the problem of minimum broadcast time (MBT) [94]. They tested their algorithm for random graphs on 10 to 500 nodes, as well as three contrived sets of networks on 40, 80, and 120 nodes. They compared their results with the approximation matching (AM) algorithm presented in [24], and claimed that their algorithm outperforms AM, particularly considering the contrived networks. Moreover, Hasson and Sipper [4] proposed ACS: an ant colony system for the MBT

problem. They compared the performance of their algorithm with that of [94] and AM [24] for random graphs on 15 to 250 nodes and edge probability in the range of $(0.05 - 0.1)$. In most cases, the achieved broadcast time was better or the same, while the running time was enhanced compared to both algorithms.

More recently, Lima et al. [95] proposed a metaheuristic algorithm for the MBT problem with GA, namely BRKGA. They suggested various versions of their algorithm while combining it with different decoders, such as first receive first send (FRFS) or an integer linear programming (ILP) method. They compared their results with various methods such as Tree Block [25], NTBA [34], NEWH [35], and ACS [4] for several graph families of up to 1024 nodes. Their results show that BRKGA is able to outperform all counterpart heuristics for the MBT problem, while it can also be an alternative for larger networks where the exact methods cannot be applied. Finally, we should point out that the goal of [4], [94], [95] is to minimize the classical broadcast time of a particular vertex under the classical model, not to optimize the broadcast time of all network members at the same time.

## IV. Methodology

In this section, we propose a novel solution to the problem of broadcasting with universal lists. We look at this problem as follows: Assuming $\Sigma$ as the search space, the goal is to find a $\sigma \in \Sigma$ that minimizes an objective function, i.e. the broadcast time. In the following proposition, it is argued that the size of the search space skyrockets as the graph size grows. Subsequently, a thorough exploration of state space is impossible.

**Proposition 1.** *For a graph $G$ on $n$ vertices, where the degree of vertex $i$ is $d_i$, the size of search space for the problem of broadcasting using universal list is as follows:*

$$|\Sigma_{(G)}| = \prod_{i=1}^{n} \sum_{j=0}^{d_i} \left( \binom{d_i}{j} \times j! \right). \tag{4}$$

*Proof.* Note that, as mentioned in [10], a solution to the problem of broadcasting with universal lists may include some neighbors of a particular vertex, not all of them, necessarily.

For a vertex $i$ with $d_i$ neighbors, a valid universal list may contain any number of its neighbors ranging from 0 to $d_i$. If it includes $j$ neighbors, there are $\binom{d_i}{j}$ different ways to select those neighbors, while in each case, the selected neighbors may be arranged in $j!$ different ways. Also, this is true for each vertex $1, 2, \cdots, n$. □

Observe that the function given in (4) grows at least as fast as $\Omega\Big((\delta!)^n\Big)$, when $\delta$ is the minimum degree of the graph, that is, $|\Sigma_{(G)}|$ is exponential. The complexity of the same function and its importance are studied in [96] under the classical model. All in all, since the size of the search space is exponential, evolutionary computing could be useful in finding an optimal solution out of many possible solutions with a high probability [61]. Genetic algorithm is a particular class of evolutionary algorithms that has been shown to be effective in finding optimal solutions for complex problems in various domains such as biology, engineering, computer science, and social science.

**Algorithm 1:** HUB−GA

```
Generate random population;
Calculate fitness score;
while not converged do
    Crossover;
    Mutation;
    Calculate fitness score;
    Acceptance;
end
return The best chromosome
```

Genetic algorithm (GA) is a population-based search algorithm [65] that uses the idea of survival of the fittest [61], originally inspired by the Darwinian theory of evolution [66]. In this algorithm, every solution to the problem at hand corresponds to a chromosome, while every parameter is a gene [64]. The fitness of each individual is evaluated with a fitness function. In order to improve the quality of solutions, the best solutions are selected for reproduction using two main operations of GA: Crossover and mutation. GA tries to find a good solution by repeating this process over multiple generations.

In this study, we propose HUB−GA: A heuristic for universal list model of broadcasting with genetic algorithm. The general routine of HUB−GA is given in Algorithm 1, whereas Fig. 2 portrays how this framework is tailored according to our problem. Note that the most crucial task for developing a GA heuristic for a problem is to encode the problem properties into chromosomes and genes [77], [84]. In the rest of this section, the detail of the proposed algorithm is explained for the problem at hand.

### A. Genes, Chromosomes, and Population

Consider a graph $G$ with $n$ vertices and denote the degree of vertex $i$ by $d_i$. An arbitrary ordering of the neighbors of vertex $i$ represents a *gene* $i$, $g^{(i)}$, with size at most $d_i$. A *chromosome* is a collection of $n$ genes: $g^{(1)}, g^{(2)}, \cdots, g^{(n)}$, each corresponding to an ordering for a particular vertex. Table I portrays an arbitrary chromosome for the graph given in Fig 1(a), where row $i$ of the matrix corresponds to gene $g^{(i)}$. Using our notation, a chromosome is a matrix $\sigma$ with $n$ rows (or $n$ genes) and $\Delta$ columns. We need to stress the fact that a chromosome is different from an adjacency list of the graph. This is due to the fact that a gene does not include all neighbors of a particular vertex, necessarily, as opposed to the adjacency list.

In GA, a chromosome is a possible solution for the problem, meaning that any $\sigma \in \Sigma$ may be an optimal broadcast scheme. However, guessing the correct solution out of too many possible solutions is nearly impossible. Subsequently, the first step of HUB−GA is to generate several solutions randomly, which is called the first *population*. The main goal of the initialization step is to distribute the solutions in the

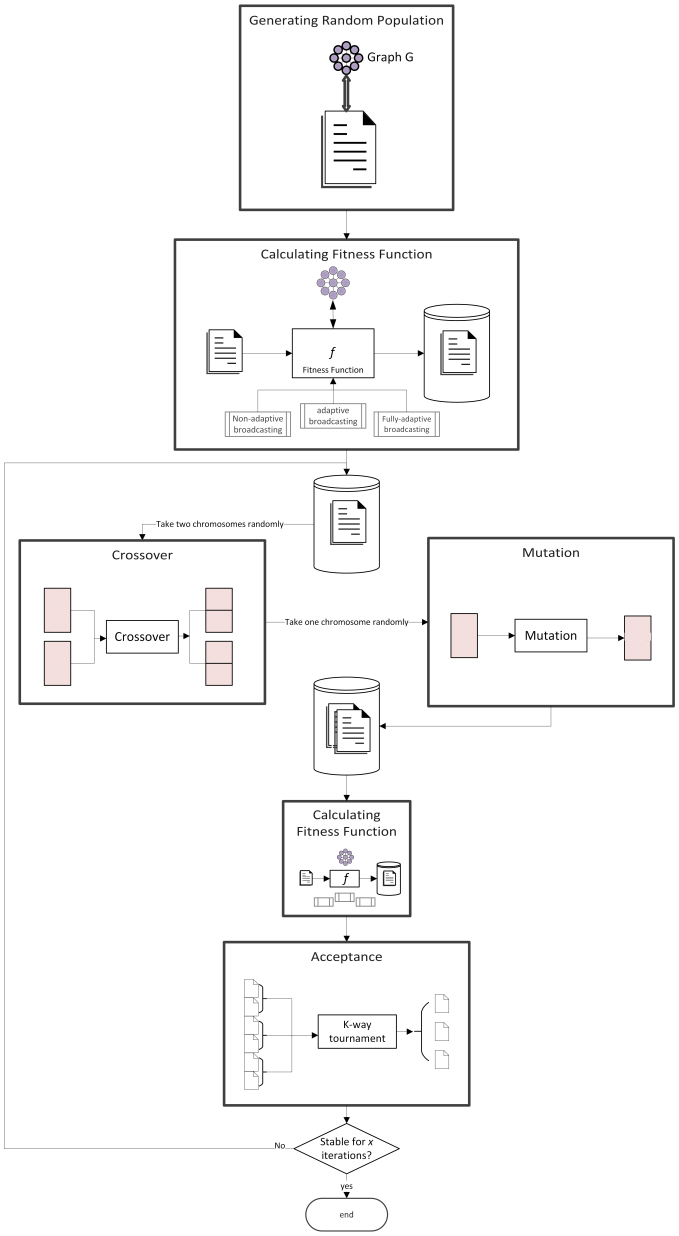



Fig. 2. A schema of our methodology.

search space as evenly as possible to increase the diversity of the population and have a better chance of finding promising regions [64]. Consequently, the bigger the size of the first generation, the higher the chance of finding a near-optimal solution in early iterations. The computational cost, however, surges as the size of the population grows. Therefore, choosing a reasonable size for the first population is a trade-off between cost and accuracy. In our experiments, we study the effect of population size, or $|p|$, on the performance of our algorithm.

Choosing a suitable encoding for genes, chromosomes, and population is crucial since they have to abide by some fundamental rules. For instance, a gene has to be mutable. Therefore, having modified the content of a gene, another valid solution must be generated. Moreover, it must be possible to create new and unseen solutions using two different chromosomes. Also, as mentioned in [75], it should be easy to go back and forth between the graph's encoding and the graph itself in a more conventional form suitable for evaluating the fitness function (which is called the encoding/decoding process). Lastly, the encoding should possess locality; that is, small changes in the encoding should result in small changes in the fitness score [75]. This will help GA to function more effectively. This study's novel encoding allows us to respect these ground rules without violating the problem's definitions.

### B. Fitness Function

The key element of any GA-based heuristic for a problem is the definition of its fitness function. The complexity of calculating the fitness function usually determines the complexity of the algorithm. The fitness function, $f(\sigma)$, evaluates the fitness of a chromosome $\sigma$, and it must be designed in a way that it evaluates the optimal solution as the best one. The ultimate objective of GA is to find a chromosome $\sigma$ which minimizes/maximizes a predefined fitness function $f(\sigma)$. We consider two different fitness functions in this study, as discussed in what follows. In both cases, a better chromosome has a smaller fitness score. Hence, the ultimate objective is to minimize the fitness function.

*1) Broadcast time:* For the first case, the fitness function is considered to be the broadcast time of the graph using $\sigma$ as the universal list:

$$f_1(\sigma) = \max_{u \in V(G)} \{B_M^\sigma(u, G)\} = B_M^\sigma(G). \tag{5}$$

To evaluate $f_1(\sigma)$, the broadcast process is simulated starting from every originator $u$, using $\sigma$ as the universal list under a model $M$. No need to mention that this is done in parallel for each originator. Then, the maximum broadcast time over every originator is considered as $f_1(\sigma)$. Therefore, in this case, the aim is to minimize the maximum broadcast time. Eventually, we expect to find a scheme in which, starting from any originator, the broadcast time is quite short. Ideally, $f_1(\sigma)$ approaches $B_M(G)$, since $B_M(G) = \min_{\sigma \in \Sigma}\{B_M^\sigma(G)\}$.

The reason for defining this function is quite obvious. In recent years, finding the broadcast time of a particular family of graphs has been an ongoing research question [20], [30], [39], [40], particularly under the universal list model [10], [45]–[48]. We claim that using the proposed algorithm, we can tackle this problem in two different ways: Firstly, our algorithm generates the actual broadcast scheme that minimizes its fitness function. The scheme could be used as a foundation for proposing general upper bounds on the broadcast time of a particular family of graphs. Secondly, the reachability of some lower bounds could be questioned with the results provided by this algorithm. For instance, based on our experimental results, we conjecture that the broadcast time of a complete graph $K_n$ on $n$ vertices under the fully-adaptive model should be strictly greater than that of the classical model. This idea could even be proved by generating all possible schemes for a particular $K_n$ and calculating their $f_1$ fitness score, which is out of scope for this study. On the other hand and by using $f_1$ as the fitness function, the network manager will be able to minimize the maximum broadcast time of any entity of the network, which is shown to be useful in different applications

such as wireless sensor networks [97], industry 4.0 [98], video packet distribution in vehicular ad-hoc networks [99], robotics [100], satellite link topology [101], and age of information performance [102].

*2) Average broadcast time:* In the second case, the average broadcast time of the network is used for the fitness function:

$$f_2(\sigma) = \frac{\sum_{u \in V(G)} B_M^{\sigma}(u, G)}{n}, \tag{6}$$

in which $n$ is the number of vertices of graph $G$. Evaluating $f_2(\sigma)$ is quite similar to that of $f_1(\sigma)$, except that instead of taking the maximum broadcast time of each originator, we use the average broadcast time. Using this fitness score, we aim to find a scheme $\sigma$ for which the behavior of all originators is fairly optimal in terms of broadcasting under model $M$.

The definition of the second function fits the environment of this study even better. To illustrate, by making a small change in a chromosome $\sigma$, the value associated with $f_2(\sigma)$ is more likely to change compared to $f_1(\sigma)$. This important feature, previously referred to as locality, could help in creating a more discoverable search space. Subsequently, the GA is anticipated to perform better in minimizing $f_2$ in general.

As an application concerning $f_2$, consider a communication network containing a clique and several paths attached to its nodes. In this network, the furthest nodes from the clique cannot inform all members quickly, nevertheless of the scheme. Using $f_1$ for this network means paying attention only to those at the furthest distance from the clique members. Consequently, the broadcast time of several vertices will not be minimized, even though it could be important for the network provider. Using $f_2$, on the other hand, will result in a scheme in which the behavior of every originator is objectively optimal.

### C. Crossover

There are two basic GA operators; the first one is Crossover, where two chromosomes are selected as the parents (*selection* phase), and then two children (called *offsprings*) are generated by *crossover*. Each of these steps is described hereafter.

In nature, the fittest individuals are more likely to get food and mate [64]. Inspired by this fact, the GA gives a higher chance to fitter chromosomes to be selected as the parents. There are several solutions proposed for this phase, a.k.a. *selection*. For instance, one solution is to use the Roulette Wheel method [62], where the chance of selecting a chromosome is proportional to its fitness score. Other well-known selection operations include, but are not limited to, Boltzmann selection [55], rank selection [56], fuzzy selection [57], and fitness uniform selection [58].

In this study, we used the famous $K$-way Tournament selection method [59]. In this method, $K$ individuals are selected from the population randomly. Then, the fittest chromosome, according to its fitness function, will be selected for the role of the first parent. The same procedure is repeated for selecting the second parent. This method is proved to be genuinely useful, particularly when the fitness score is a minimization function. Moreover, this method is very efficient for parallel computing [54]. In this paper, $K$ is fixed to 4 as a commonly used value [54], [59].

Once parent 1 ($p_1$) and parent 2 ($p_2$) are selected, a random integer is chosen from the range $[1, n)$ as the crossover point, denoted by $c_p$. Afterward, two new off-springs ($c_1$ and $c_2$) are generated as follows:

$$\begin{aligned} c_1 &= g^{(1)}(p_1), \cdots, g^{(c_p)}(p_1), g^{(c_p+1)}(p_2), \cdots, g^{(n)}(p_2) \\ c_2 &= g^{(1)}(p_2), \cdots, g^{(c_p)}(p_2), g^{(c_p+1)}(p_1), \cdots, g^{(n)}(p_1), \end{aligned} \tag{7}$$

in which $g^{(i)}(p_j)$ is the $i$th gene of parent $j$. Indeed, the first child contains the first $c_p$ genes of $p_1$, while other genes come from $p_2$. This will be the other way around for the second child. This method is called single-point crossover, which is one of the most common methods in addition to the double-point method [63]. To name a few other methods, three parents crossover [50], uniform crossover [51], and Masked crossover [52] could be mentioned.

We keep making new off-springs until the initial size of the population is doubled. Note that off-springs still are valid solutions since no new vertex is added to the neighborhood of a sender. However, their fitness function must be calculated in the next iteration.

### D. Mutation

The second operator of GA is mutation, in which a gene of an offspring is changed randomly with a small probability. The mutation operator preserves population diversity by introducing another level of randomness [64]. Also, this operator prevents the solutions from becoming similar and increases the probability of avoiding local solutions in the GA [64].

In our algorithm, the ordering of a gene is shuffled when it is supposed to go over mutation. Although mutation prevents GA from falling into a local optimum, there is a big chance of losing the best chromosome. Therefore, we performed two actions for mutation:

1) **Decreasing mutation probability:** The chance of performing mutation reduces over time. This is because in early iterations, the goal is to explore the state space as much as possible, which is achieved by mutation. However, during last iterations, the algorithm is likely to reach convergence with a relatively good solution. Therefore, mutation cannot be useful anymore.
2) **Elitism:** When a gene of an offspring is changed by mutation, the best solution is likely to be lost. Using elitism [53], a copy of the original individual is maintained before changing it. This copy will be carried out to the next generation, being treated as a normal chromosome. The goal is to prevent such solutions (elites) from being degraded. This method may accelerate the convergence of GA dramatically [53]. Also, as mentioned in [64], the reliability of GA heavily depends on the process of maintaining the best solutions in each generation. Using elitism, the chance of maintaining a good solution gets higher.

In our implementation, at early iterations, each gene could go over mutation with a 1% chance, and this number decreases smoothly over time. When a mutation is supposed to happen, a copy of the original chromosome is kept in the population.

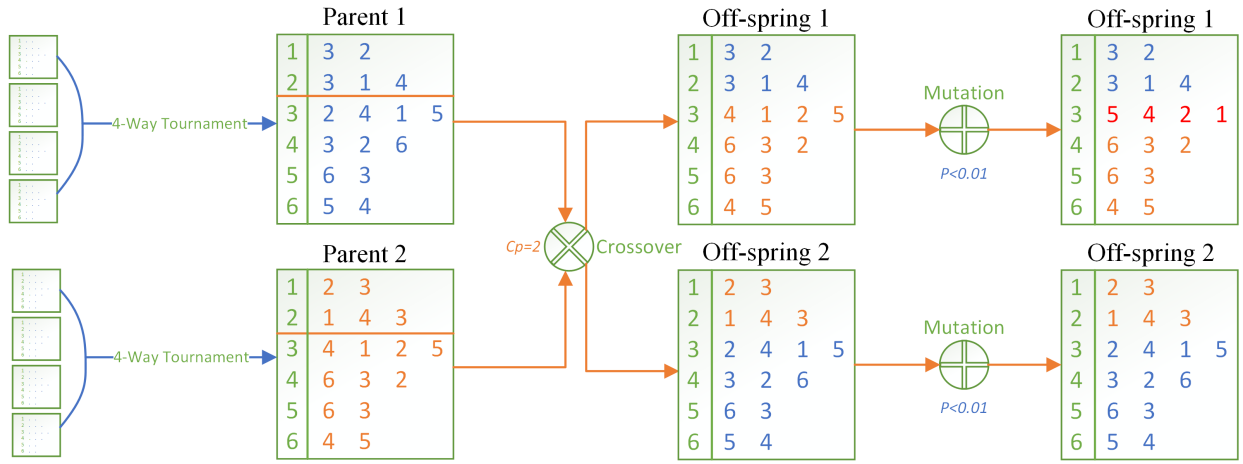



Fig. 3. The process of selection, crossover, and mutation in HUB−GA.

Fig. 3 shows the three main operators of HUB−GA, i.e. selection, crossover, and mutation. In this Figure, Parent 1 ($p_1$) and Parent 2 ($p_2$) are the same chromosomes as the ones given in Table I and Table II for the graph displayed in Fig. 1(a). Assume $p_1$ and $p_2$ are selected through a 4-Way tournament among other candidates. By performing crossover with $c_p = 2$ on $p_1$ and $p_2$, two new offsprings, $c_1$ and $c_2$, are generated. Each gene of $c_1$ and $c_2$ may go through mutation with a small probability, resulting in a random shuffle in $g^{(3)}$ of $c_1$.

### E. Acceptance

Having generated several off-springs using crossover and mutation, the population size escalates. To keep the current generation manageable with limited resources, one possible solution is to retain the original population size by allowing a fixed number of chromosomes to survive into the next generation.

In order to do so, we used a 4-way tournament as described earlier in Section IV-C. Thus, as long as the size of the next generation has not reached the original size, four random chromosomes are selected from the current generation, and the one with a better fitness score will survive into the next generation. This method gives a higher chance to the fittest individuals to survive, while any chromosome could be carried out to the next generation.

### F. Stopping Criterion

Several conditions are discussed in the literature regarding the best time to terminate the execution of GA [60], [61]. Each criterion has its own advantages and disadvantages. For instance, the most trivial way is to stop exploring when a fixed number of generations have been generated or a predefined time limit is exceeded. Although the idea is very simple and easy to implement, choosing a number for this purpose is not straightforward. Another well-known solution is to stop the execution when the lower bound of the problem is met. Utilizing this condition could be quite effective in many problems. In the case of broadcasting, however, there are two obstacles. Firstly, calculating the lower bound on $B_M(G)$ could be as difficult as guessing $B_{cl}(G)$ (See Eq. (2)), which is an NP-hard problem [19]. Secondly, to the best of our knowledge, there is no solution to evaluate the lower bound's reachability for a particular graph. Therefore, if the lower bound is unachievable, the GA never terminates.

In this study, we employed an adaptive stopping criteria. The execution of HUB−GA terminates if, after $S_t$ iterations, the fitness score of the fittest individual does not change. Thus, the algorithm can avoid unnecessary reproductions once a local (and possibly global) optimum has been reached. Nonetheless, choosing the parameter $S_t$ remains a confusing puzzle. Therefore, we studied the impact of changing parameter $S_t$, a.k.a the stability variable, on the performance of our algorithm. Once the stopping criterion is met, the best chromosome (solution) from the current generation as well as its fitness score is returned as the final answer.

### G. Remarks

The advantages of our method are as follows:

1) **The first heuristic for this problem:** As mentioned in Section III, there are several heuristics for classical broadcasting, such as [32], [33], even some researchers consider GA [94], [95], or other evolutionary algorithms [4]. However, to the best of our knowledge, there is no similar study on the problem of broadcasting with universal lists.
2) **Working for arbitrary graphs:** the proposed framework takes an arbitrary graph as the input and finds a nearly optimal broadcast scheme for that particular graph, that is, the structure of the graph and its features are not utilized in any step of HUB−GA. This is a huge advantage since all upper bounds provided in the literature (such as [10], [11], [15], [46]–[48]) are tailored for a specific family of networks.
3) **Working for any model under universal lists:** our heuristic could minimize its objective function (either $f_1$ or $f_2$) under all three models: non-adaptive, adaptive, or fully adaptive. To the best of our knowledge, this novelty separates HUB−GA from several related works

considering a specific model; either classical [4], [32], [33], [36]–[40], [94], [95], non-adaptive [10], [46]–[48], adaptive [10], [45], or fully-adaptive [11], [15].

4) **Possibility of defining various fitness scores:** the proposed framework can accept any fitness score defined by the user. For instance, to minimize the broadcast time of a particular vertex $u$ under model $M$, the user can define $f_3(\sigma) = B^{\sigma}_{M}(u, G)$. Moreover, more complex fitness functions such as $f_4(\sigma) = f_1(\sigma) \times f_2(\sigma)$ could be defined according to the requirements of the network manager. This useful feature is not considered in other related works, such as [4], [84], [94], [95].
5) **Efficiency:** the proposed framework is quite efficient in terms of run time. Our numerical results show that it could find optimal (or near-optimal) solutions in a few seconds for graphs with hundreds of nodes. Considering the limited resources used in this study, we believe this work is scalable to a good extent for real-world applications.
6) **Providing the broadcast scheme:** As opposed to several studies which propose closed formula upper bounds for some graphs alongside a descriptive broadcast scheme (such as [10], [15], [45]–[48]), our method generates the actual broadcast scheme. This is quite useful for real-world scenarios. Also note that the generated schemes could be used to prove many theoretical aspects related to this problem. Moreover, the process of encoding/decoding of chromosomes does not require any knowledge on the topic, in contrast with other related works such as the ones suggested in [84], [94], [95].

## V. Experimental Results

To show the efficiency of the proposed method, we have conducted several computer simulations. In this section, each of which will be discussed in detail. Table VI specifies the aims and scopes of each experiment.

### A. Experiment 1

This experiment studies the impact of our algorithm's parameters on its performance. There are two major parameters in our algorithm:

- Population size $|p|$: As the size of the initial population increases, the chance of finding a better solution in early iterations gets higher. However, the computational cost associated with the algorithm increases as well. Therefore, choosing the initial size of the population plays an important role in the algorithm's performance.
- Stability $S_t$: The stopping criterion of the proposed algorithm is its stability over the last $S_t$ iterations; that is, if the fitness score of the fittest individual does not change over $S_t$ generations, the algorithm stops and returns the best solution. Choosing an appropriate number for $S_t$ is a trade-off; the higher the value of $S_t$, the higher the chance of finding optimal solutions, and the longer it takes for the algorithm to terminate.

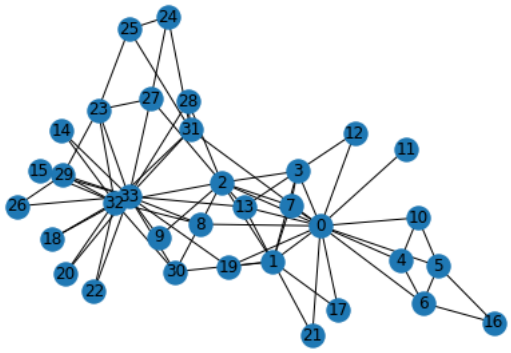



Fig. 4. Karate club network [103], used in Experiment 1.

For this experiment, the Karate club network [103] has been selected with 34 nodes and 78 edges (See Fig. 4). We run our algorithm under all three models $(na, a, fa)$ while changing those parameters. The parameter $|p|$ increases from 10 to 500, while we let $S_t$ change in the range of 1 to 15. The performance of the algorithm will be measured based on 4 criteria: The broadcast time ($f_1$), the run time required for calculating $f_1$, the average broadcast time ($f_2$), and the run time required for calculating $f_2$. The results of this experiment are reported in Figs. 5 and 6.

By analyzing the results given in Fig. 5, several interesting points could be drawn. Firstly, by increasing the population size, the chance of minimizing fitness functions (either $f_1$ or $f_2$) improves. But the slope of this improvement lessens for higher values of $|p|$. In other words, increasing the population size up to a certain value, such as 200, could be beneficial for the algorithm, but higher values of $|p|$ seem to be almost ineffective, while they need more time to be processed. For instance, we compare the performance of $fa$ model when $|p|$ increases from 200 to 500 considering $f_2$: Although the run time increases by 140%, the fitness score improves only by 4%. These numbers are 434% and 3.5% under adaptive, and 146% and 3.6% under the non-adaptive model, respectively. Therefore, based on this experiment, we set the value of $|p|$ to 200 for the rest of this study.

Secondly, by comparing Fig. 5(a) and (c), it is clear that the average broadcast time $(f_2)$ could be improved more, provided the fact that the algorithm has enough resources (time and computational power). Whereas decreasing the broadcast time $(f_1)$ seems to be a very difficult task, particularly once $f_1$ has reached a local minimum. This is due to the nature of the search space associated with $f_1$ and $f_2$. For $f_1$, the search space has drastic steps, usually not very close to each other (since a very small change in $\sigma$ might not affect $f_1(\sigma)$ sometimes). However, fitness function $f_2$ has a search space with a lot of small steps since by a slight modification in $\sigma$, $f_2(\sigma)$ is expected to change, even for a small amount.

Thirdly, with a quick look at Figs. 5(b) and (d), it is concluded that the Fully-adaptive model is the quickest model among all. This idea also has been proved before [11]. Broadcasting under this model is also affected less than two other models when changing $|p|$.

TABLE VI
AIMS AND SCOPES OF THE EXPERIMENTS.

| Experiment | What? | Why? | How? | Graph(s) |
|---|---|---|---|---|
| Experiment 1 | Parameter Tuning | To see the impact of changing HUB−GA parameters on its performance. | For a graph $G$, we change parameters $\|p\|$ and $S_t$, while reporting $f_1(\sigma)$ and $f_2(\sigma)$ and the run time. | Karate club network [103] |
| Experiment 2 | Performance comparison vs. Classical model | To see whether the found broadcast time under universal lists model approaches its optimal value or not. | By calculating the ratio of $B_M(G)/B_{cl}(G)$ for different interconnection networks. | Well-known interconnection networks for which the value of $B_{cl}(G)$ is known. |
| Experiment 3 | Performance comparison vs. degree-based heuristics | To see whether HUB−GA outperforms degree-based heuristics or not. | By comparing the performance of our heuristic with three heuristics for clique-like graphs. | Clique-like graphs: Ring of cliques [104], and Windmill graph [105] |
| Experiment 4 | Performance comparison vs. state-of-the-art heuristics | To see whether HUB−GA gets close to other heuristics for classical broadcasting or not. | By comparing the performance of our heuristic with two lower bounds and six upper bounds. | Interconnection Networks and Complex networks with small-world model [106] |

Also, we need to explain the reason for high fluctuations in Fig. 5 (b) and (d): According to the randomness of HUB−GA, it is possible that during early generations, a fit chromosome is constructed and it will survive to future generations. Therefore, the final answer could be found quickly. On the other hand, if this chromosome is not generated in early iterations, it could take multiple generations to construct it from several good solutions. So, regardless of the population size, the run time of GA fluctuates. Though the range of this fluctuation widens as $|p|$ grows, according to Figs. 5(b) and (d).

The next parameter of our algorithm is $S_t$, whose effect on the performance of HUB−GA is reported in Fig. 6. Similar to the population size, as $S_t$ grows, the chance of finding a solution with a smaller broadcast time ($f_1$ or $f_2$) increases. However, as Fig. 6 suggests, increasing $S_t$ to large values, such as 10 or higher, does not seem to have a notable effect on the performance of the algorithm while it requires more processing time. Consequently, based on the results of this experiment, we select a value of $S_t = 5$ throughout the rest of this study. Also, note that by looking at the line charts provided in Fig. 6, still, the fully-adaptive model is the quickest among all.

### B. Experiment 2

The objective of this experiment is to compare the GA heuristic for the universal list model with known bounds on the classical model for commonly used interconnection networks. Therefore, we measure the ratio of $B^{\sigma}_M(G)/B_{cl}(G)$ for several graphs. Ultimately, the found broadcast time under universal list models should approach a desired value. For instance, consider line graph $P_n$ on $n$ vertices under the non-adaptive model. Recall that $B_{cl}(P_n) = n - 1$, and $B_{na}(P_n) = \lceil \frac{3n}{2} \rceil - 2$ [10]. If our algorithm finds the optimal scheme, as $n$ approaches infinity, the ratio is supposed to approach 1.5: $\lim_{n\to\infty} \frac{B_{na}(P_n)}{B_{cl}(P_n)} = \lim_{n\to\infty} \frac{\lceil \frac{3n}{2} \rceil - 2}{n-1} = 1.5$.

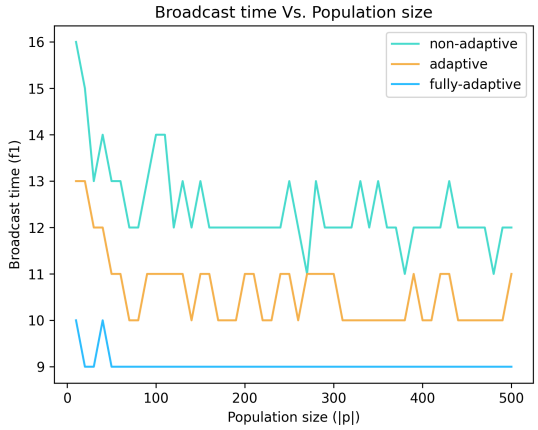
To this aim, for a graph $G$ on $n$ vertices (or dimension $d$, where $n$ is a function of dimension $d$), we run our heuristic under all three models $(na, a, fa)$, then compare the achieved value with the known value of $B_{cl}(G)$. We repeat this process for several instances of that graph and report the average of the ratio as the result. Note that for this experiment, we need to use some graphs for which the exact value of classical broadcast time is known; thus, the ratio could be calculated. The only exception is the De Bruijn graph $DB_d$ for which the exact value of $B_{cl}(G)$ is not known. Hence, we compare our results with the best lower bound suggested in [108]. The result of this experiment is reported in Table VII. An asterisk means that our algorithm has been able to find the optimal scheme or a scheme better than the current upper bound for a given graph. Also, the known bounds under all models of broadcasting for these families are presented in Table VIII.

By comparing the results of Table VII and VIII, it immediately follows that HUB−GA is able to achieve the optimal result for Path, Star, and Cycle under all three models since the ratio approaches the optimal value. For Grid and Tori, however, the values are close to optimal. The reason is that, as mentioned in [15] and [10], an optimal broadcast scheme for these networks is symmetric for all nodes. Finding a completely symmetric scheme for a large graph could be difficult for the GA due to its randomness. Also, since the universal list broadcast time of $HC_d$, $CCC_d$, $DB_d$, and $SE_d$ are not known, we can neither accept nor reject the optimality of HUB−GA for these families of networks. However, the achieved broadcast time for all these networks is better than the current upper bound and mostly between 1 to 1.5 times its classical time, which is very promising. A trivial upper bound for the non-adaptive broadcast time of an arbitrary graph is suggested in Theorem 2.2. of [10] as follows: $B_{na}(G) \leq 3 \cdot B_{cl}(G)$, for any graph $G$.

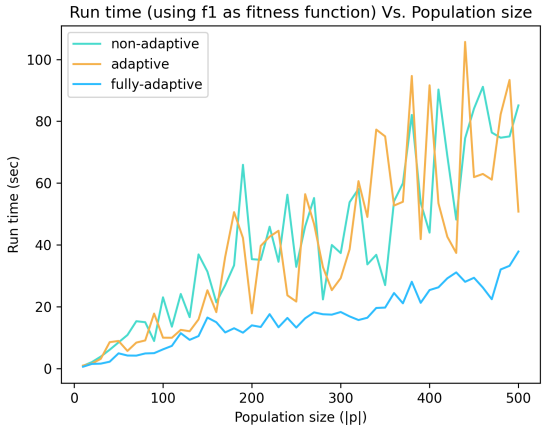
Lastly, the results of complete graphs are discussed. The known bounds for a complete graph are as follows:

$$\begin{aligned} \lceil \log n \rceil = B_{cl}(K_n) \leq B_{fa}(K_n) \leq B_a(K_n) \leq B_{na}(K_n) \\ \leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil, \end{aligned} \tag{8}$$

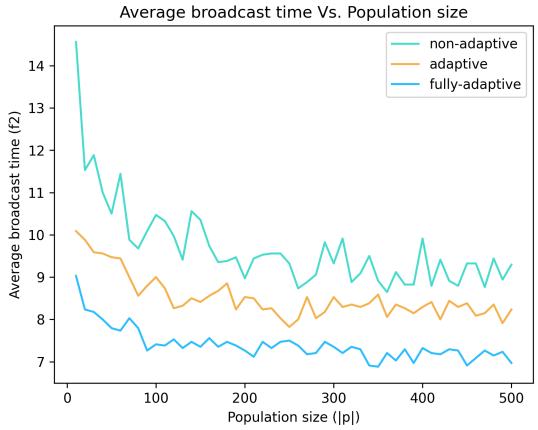
where the lower bound comes from the trivial lower bound on broadcast time, and the upper bound is presented in [10]. According to Table VII, the result of each model differs from other models. Hence, in (8), the inequalities are more likely to become strictly less than, particularly for the fully-



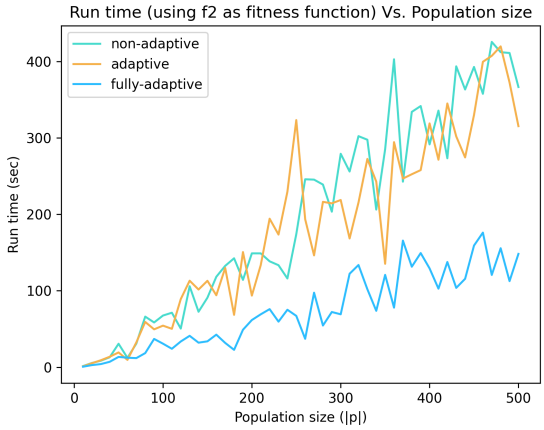

(a) Broadcast time ($f_1$)




(b) Run time for calculating $f_1$




(c) Average broadcast time ($f_2$)




(d) Run time for calculating $f_2$

Fig. 5. Results of experiment 1: Impact of population size ($|p|$).

TABLE VII
RESULTS OF EXPERIMENT 2.

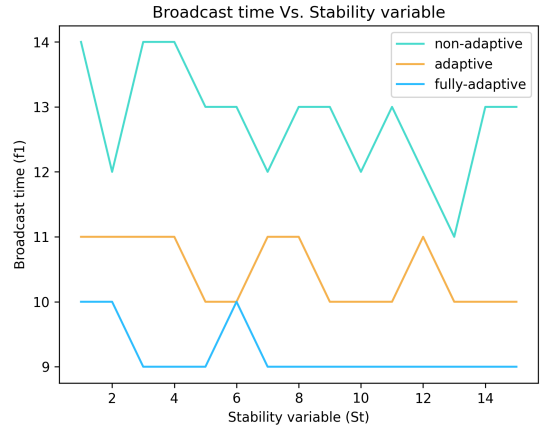
| Graph $G$ | $n$ | $\frac{B^{\sigma}_{fa}(G)}{B_{cl}(G)}$ | $\frac{B^{\sigma}_{a}(G)}{B_{cl}(G)}$ | $\frac{B^{\sigma}_{na}(G)}{B_{cl}(G)}$ |
|---|---|---|---|---|
| Path $P_n$ | $2 \leq n \leq 1000$ | **1.00*** | **1.00*** | **1.49*** |
| Cycle $C_n$ | $3 \leq n \leq 1000$ | **1.00*** | **1.00*** | **1.32*** |
| Star $S_n$ | $2 \leq n \leq 1000$ | **1.00*** | **1.00*** | **1.01*** |
| Complete Graph $K_n$ | $3 \leq n \leq 50$ | 1.14 | 1.39 | 1.42 |
| Grid $G_{n\times m}$ | $2 \leq n, m \leq 10$ | 1.07 | 1.08 | 1.35 |
| Tori $T_{n\times m}$ | $2 \leq n, m \leq 10$ | 1.09 | 1.24 | 1.55 |
| Hypercube $HC_d$ | $2 \leq d \leq 9$ | 1.06 | **1.41*** | **1.68*** |
| Cube Connected Cycle $CCC_d$ | $2 \leq d \leq 7$ | 1.14 | **1.18*** | **1.52*** |
| Shuffle Exchange $SE_d$ | $3 \leq d \leq 9$ | **1.06*** | **1.09*** | **1.44*** |
| De Bruijn $DB_d$ | $3 \leq d \leq 9$ | **1.09*** | **1.18*** | **1.51*** |

adaptive model. Based on our results, we make the following conjecture:

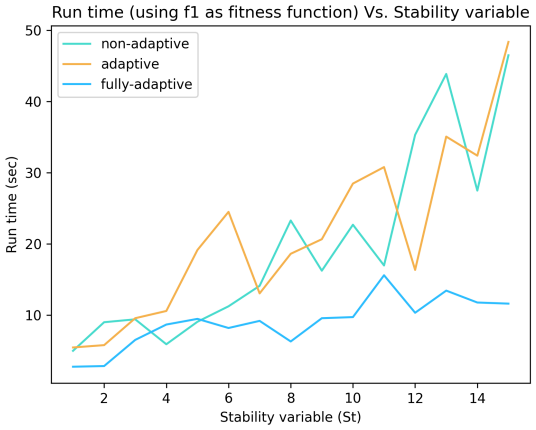
**Conjecture 1.** *For sufficiently large $n$, the broadcast time of a complete graph $K_n$ is bounded as follows:*

$$\lceil \log n \rceil = B_{cl}(K_n) < B_{fa}(K_n) < B_a(K_n) \leq B_{na}(K_n) \leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil. \quad (9)$$

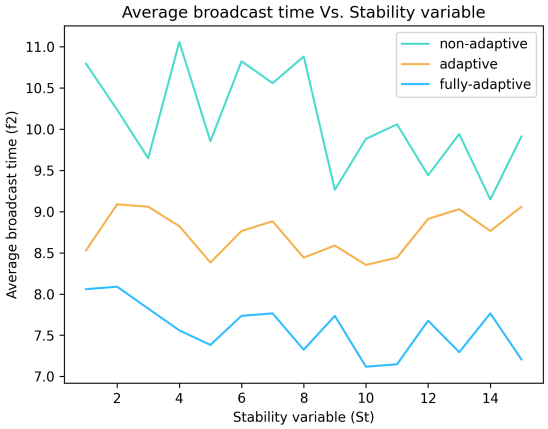
In order to experimentally validate this conjecture, we designed another experiment in which the performance of our GA is compared with the lower and upper bounds of (8) for a complete graph $K_n$ on size $3 \leq n \leq 150$. The results of this experiment are illustrated in Fig. 7. First, observe that these results correspond to the bounds of (8). Secondly, based on this result, the smallest value of $n$ for which $\lceil \log n \rceil = B_{cl}(K_n) < B_{fa}(K_n)$ is $n = 8$ where $B^{\sigma}_{fa}(K_8) = 4$, while $B_{cl}(K_8) = 3$. In order to prove that $B_{fa}(K_8) \neq 4$ (which also proves the first inequality of (9)), one solution is to examine all possible broadcast schemes for a complete graph $K_8$ under the fully-adaptive model. If no scheme $\sigma$ could be found that yields $B^{\sigma}_{fa}(K_8) = 3$,



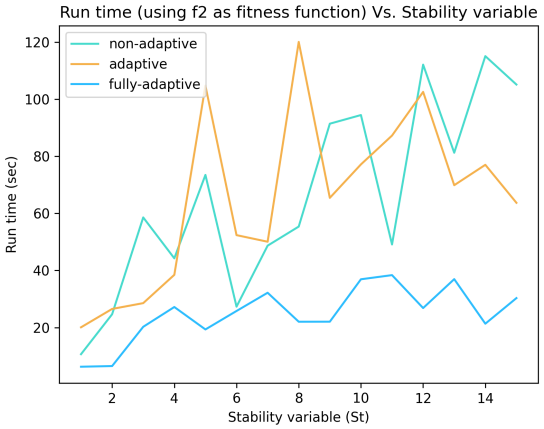

(a) Broadcast time ($f_1$)




(b) Run time for calculating $f_1$




(c) Average broadcast time ($f_2$)




(d) Run time for calculating $f_2$

Fig. 6. Results of experiment 1: Impact of stability variable ($S_t$).

the proof is completed. However, by considering (4), there are almost $2.08e40$ unique broadcast schemes for $K_8$, which is impossible to be generated without a careful constructive method. Thirdly, the truth of the second inequality could be realized by observing the growth rate of $B_{fa}(K_n)$ and comparing it with that of $B_a(K_n)$ in Fig. 7.

We need to stress the importance of Conjecture 1: If it is true, it implies that the definition of broadcast graphs ($bg$'s) are different under the universal list models. A $bg$ on $n$ vertices is a graph for which broadcasting could be finished in $\lceil \log n \rceil$ time units starting from any originator. In the classical model, the most trivial $bg$ on $n$ vertices is $K_n$ in which $\forall u \in V(K_n) : B_{cl}(u, K_n) = \lceil \log n \rceil$. If Conjecture 1 is true, it means that: 1) complete graphs are not immediate $bg$'s under the universal lists model, 2) for an arbitrary value of $n$, the existence of a graph on $n$ vertices for which the $\lceil \log n \rceil$ time could be obtained under the universal list model is questionable. It should be noted that this problem is also studied in [11] where the Hypercube $H_d$ was introduced as the first infinite family of minimum $bg$'s under the fully-adaptive model for any $n = 2^d$. They also proposed different $bg$'s for some values of $n$ such as $n = 2^{k-1}+2^{k-2}$ or $n = 2^{k-1}+2^{k-3}$ for any integer $k = \lceil \log n \rceil \geq 4$. However, the puzzling question discussed in this study concerning arbitrary values of $n$ still remains open.

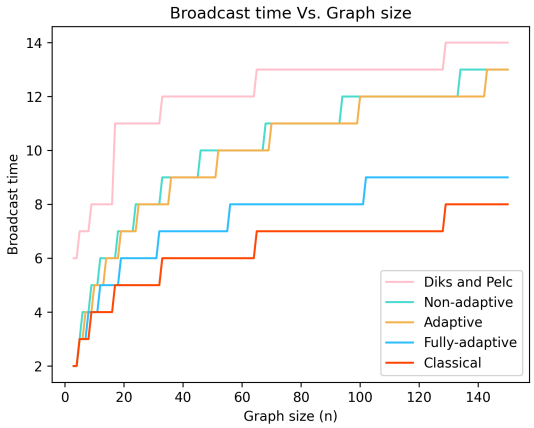



Fig. 7. Comparing the performance of HUB−GA with the known bounds on the broadcast time of a complete graph $K_n$ of size $3 \leq n \leq 150$.

TABLE VIII
KNOWN VALUES OF $B_M(G)$ FOR THE GRAPHS OF EXPERIMENT 2.

| Graph $G$ | $B_{cl}(G)$ | $B_{fa}(G)$ | $B_a(G)$ | $B_{na}(G)$ |
|---|---|---|---|---|
| Path $P_n$ | $n-1$, folklore | $n-1$, proved in [15] | $n-1$, proved in [10] | $\lceil \frac{3n}{2} \rceil - 2$, proved in [10] |
| Star $S_n$ | $n-1$, folklore | $n-1$, proved in [15] | $n-1$, proved in [10] | $n$, proved in [10] |
| Cycle $C_n$ | $\lceil \frac{n}{2} \rceil$, folklore | $\lceil \frac{n}{2} \rceil$, proved in [15] | $\lceil \frac{n}{2} \rceil$, proved in [10] | $\lfloor \frac{2n}{3} \rfloor$, proved in [10] |
| Grid $G_{n\times m}$ | $n+m-2$, proved in [26] | $n+m-2$, proved in [15] | $n+m-2$, proved in [10] | $n+m-1$, proved in [10] |
| Tori $T_{m\times n}$ | $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor$, if $m$ and $n$ are even $\lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 1$, otherwise proved in [26] | $\leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 2$, proved in [15] | $\leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 3$, proved in [10] | $\leq \lfloor \frac{n}{2} \rfloor + \lfloor \frac{m}{2} \rfloor + 5$, proved in [46] |
| Complete graph $K_n$ | $\lceil \log n \rceil$, folklore | $\leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$, proved in [15] | $\leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$, proved in [10] | $\leq \lceil \log n \rceil + 2\lceil \sqrt{\log n} \rceil$, proved in [10] |
| Hypercube $HC_d$ | $d$, folklore | $d$ | $\leq \frac{d(d-1)}{2} + 1$ | $\leq \frac{d(d+1)}{2} + 1$ |
| Cube Connected Cycle $CCC_d$ | $\lceil \frac{5d}{2} \rceil - 1$, proved in [27] | $\lceil \frac{5d}{2} \rceil - 1$ | $\leq 2\lceil \frac{5d}{2} \rceil - 1$ | $\leq 3\lceil \frac{5d}{2} \rceil - 3$ |
| Shuffle Exchange $SE_d$ | $2d-1$, proved in [28] | $\leq 4d-1$ | $\leq 4d-1$ | $\leq 6d-3$ |
| De Bruijn $DB_d$ | $\leq \frac{3}{2}(d+1)$, proved in [107] $\geq 1.3171d$, proved in [108] | $\leq 3d+1$ | $\leq 3d+1$ | $\leq 4d$ |

### C. Experiment 3

For a fixed value of $n$, the function discussed in (4), i.e. the size of the search space of this problem, reaches its maximum when the degree of each node maximizes, or a complete graph on $n$ vertices. In fact, as opposed to the classical model in which $B_{cl}(K_n)$ is known, there is no optimal bound for this network under universal list models. Besides, as mentioned in [29], the broadcast problem becomes more difficult when there are several intersecting cycles. In the previous experiment, we studied this problem for complete graphs $K_n$. Subsequently, the objective of this experiment is to study the performance of HUB-GA for graphs with clique-like subgraphs.

To this aim, two different families of graphs are considered: the ring of cliques [104] and the Windmill graph [105]. A ring of clique $RC_{n,m}$ consists of $n$ cliques of size $m$ that are connected to each other on a cycle. It has $n \cdot m$ vertices and $n \cdot \frac{m\cdot(m-1)}{2} + n$ edges. Also, a Windmill graph $W_{k,n}$ is a graph of $n$ cliques each of size $k$ that are all joined at one vertex. Alternatively, one may generate $n$ cliques of size $k-1$ and add one node to this graph which is connected to all other vertices. A $W_{k,n}$ has $n \cdot (k-1) + 1$ vertices and $\frac{nk(k-1)}{2}$ edges. Fig. 8 portrays $RC_{3,5}$ and $W_{5,3}$.

The classical broadcast time of these networks is not known. Therefore, we cannot make a similar experiment as the previous experiment. Instead, we compare the performance of our GA solution with three heuristics:

- **Ran.Seq.:** The ordering of a vertex is uniformly random.
- **Inc.Deg.:** Neighbors of a vertex are sorted in ascending order based on their degree.
- **Dec.Deg.:** Neighbors of a vertex are sorted in descending order based on their degree.

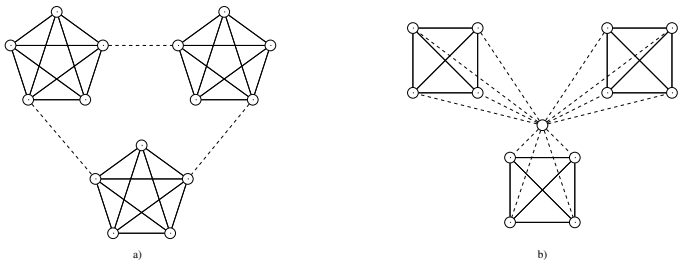



Fig. 8. (a) Ring of clique $RC_{3,5}$ and (b) Windmill graph $W_{5,3}$.

Note that as the performance of the Ran.Seq. is random in each run, we need to average its result over 5 runs. Moreover, the idea of degree-based heuristics is claimed to be efficient under the classical model, such as the AM algorithm provided in [24]. Besides, not only the worst behavior of random broadcasting in a network, a.k.a messy broadcasting, has received a lot of attention from researchers [8], [9], [109], but also the average-case random broadcasting time of various networks are studied in [110].

For each heuristic, we report three numbers: **Min** which is the minimum broadcast time of any vertex in the network using the scheme provided by a heuristic: $\min_{u\in V(G)}\{B^{\sigma}_M(u,G)\}$, **Avg** or the average broadcast time of all vertices of the graph following that scheme: $\frac{\sum_{u\in V(G)}\{B^{\sigma}_M(u,G)\}}{n}$, and **Max** which is the broadcast time of the worst originator in the graph following a particular scheme: $\max_{u\in V(G)}\{B^{\sigma}_M(u,G)\}$. Note that **Max** corresponds to fitness function $f_1$, whereas **Avg** corresponds to $f_2$. The results of this experiment are reported in Table X for $RC_{n,m}$ and Table IX for $W_{k,n}$, where $3 \leq n, m, k \leq 6$.

By looking at the results expressed in Table X, it is obvious

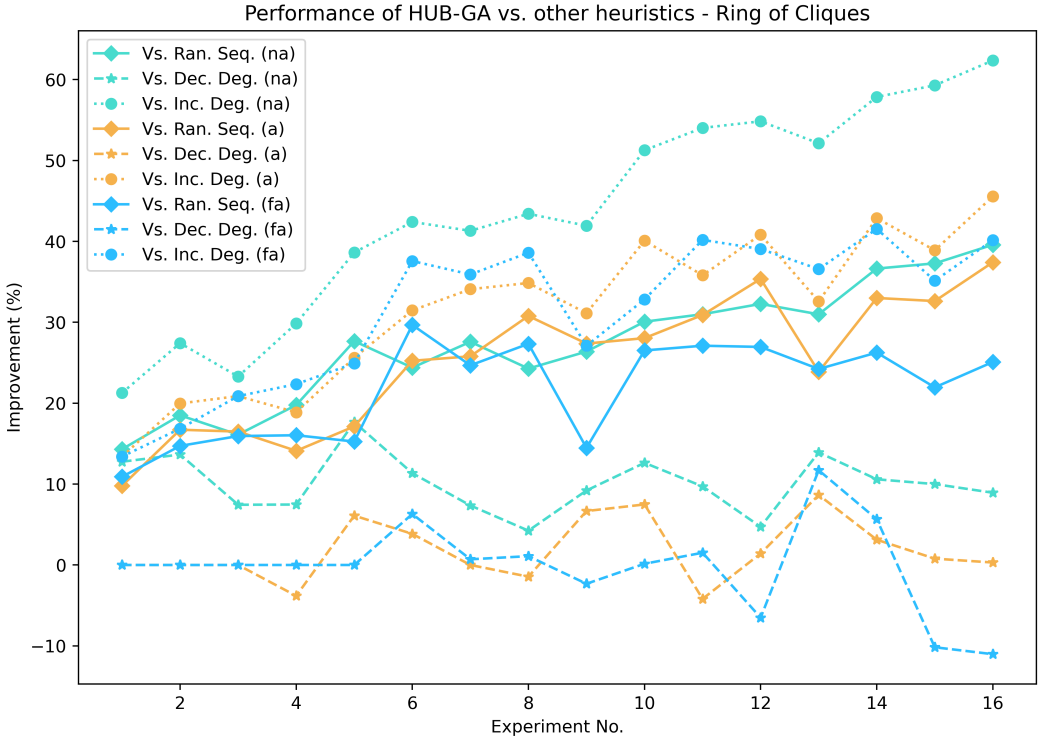



Fig. 9. Comparison of the performance of the HUB-GA and other heuristics in terms of average broadcast time - Ring of cliques.

that HUB-GA outperforms all other heuristics in terms of **Min**, **Avg**, and **Max** broadcast time in almost all cases. Moreover, among three competitor heuristics, Dec.Deg. is the most successful one for this particular graph family, while Inc.Deg. is even worse than a pure random ordering. The reason is that by using Dec.Deg. a vertex will first send the message to join vertices since they have the highest degree in the graph (Fig. 8(a), there are 6 join vertices). Then, it will send the message to its other neighbors in a random way. This is an efficient ordering, which could be optimal with a careful ordering of other neighbors. On the other hand, the Inc.Deg. heuristic is the least successful one since a vertex prioritizes its non-joint neighbors, which is not optimal behavior.

Lastly, the margin between the performance of the GA and Dec.Deg. heuristic is wider under the non-adaptive model compared to the fully-adaptive model. The underlying reason is interesting: consider a random scheme for non-adaptive broadcasting in a complete graph, and denote the non-adaptive broadcast time achieved by this scheme by $t_{na}^{ran}$. Also, denote the optimal broadcast time for the same graph under the same model by $t_{na}^{opt}$. Moreover, denote by $t_{fa}^{ran}$ the fully-adaptive broadcast time of a random scheme for the same graph, whereas $t_{fa}^{opt}$ is the optimal fully-adaptive broadcast time. We argue that the value of $t_{fa}^{ran} - t_{fa}^{opt}$ is more likely to be less than $t_{na}^{ran} - t_{na}^{opt}$ with infinite number of same experiments. This is because of the fact that in the fully-adaptive model, an inefficient behavior is to hit the same receiver by several senders. However, in the non-adaptive model, the message is likely to be sent back and forth between senders and receivers, resulting in a wider gap between the optimal scheme and a randomly selected ordering. Based on this observation, we expect the results of HUB-GA to have a wider margin with all three heuristics under the non-adaptive model compared to the fully-adaptive model in which the performance of all 4 heuristics could get much closer. The numerical results of this experiment reported in Table X also correspond to this observation.

Table IX gives the results of the same experiment for Windmill graph $W_{k,n}$, where $3 \leq k, n \leq 6$. In a $W_{k,n}$ there are two types of vertices: Join vertex (with a degree of $k(n-1)$), and other vertices (with a degree of $n-1$). Therefore, in competitor heuristics, the join vertex will choose a random ordering for its neighbors since all neighbors have the same degree. However, other vertices behave differently in two degree-based heuristics: in the Inc.Deg. heuristic, they will first distribute the message within their clique and then will send it to the join vertex, whereas this ordering is reversed in the Dec.Deg. heuristic. Therefore, we expect Dec.Deg. to perform slightly better than Inc.Deg.

The results given in Table IX correspond to our expectations, in which Dec.Deg. outperforms both Inc.Deg. and Ran.Seq. heuristics under all three models. Interestingly enough, a random ordering of the vertices seems to be more

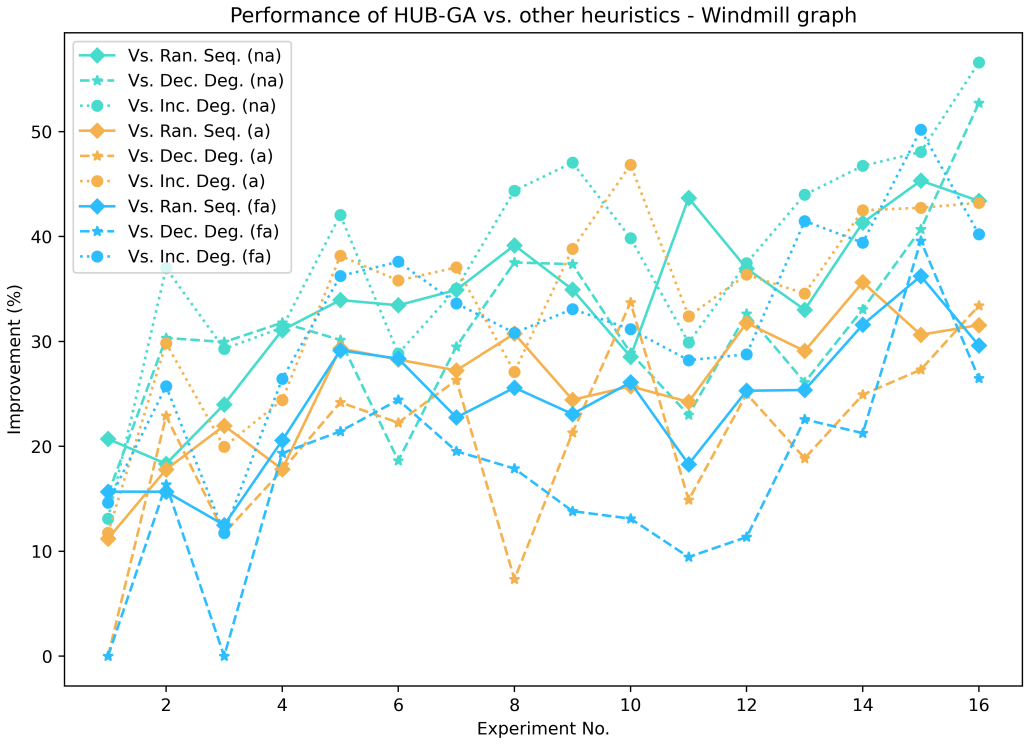



Fig. 10. Comparison of the performance of the HUB−GA and other heuristics in terms of average broadcast time - Windmill graph.

effective than the Inc.Deg. heuristic. This is due to the missing call toward the join vertex during early stages under the increasing degree heuristic. All in all, the performance of HUB−GA is better than all three heuristics, particularly under the non-adaptive model, for the same reason discussed earlier.

Also, Figs. 9 and 10 compare the results of HUB−GA with three heuristics for these two families of graphs. The comparison metric is the performance improvement which is calculated as follows for model $M$ when comparing the performance of HUB−GA against heuristic H:

$$\text{improvement}_M^{\text{H}} = \frac{\textbf{Avg}(B_M^{\text{H}}(G)) - \textbf{Avg}(B_M^{\text{HUB-GA}}(G))}{\textbf{Avg}(B_M^{\text{HUB-GA}}(G)),} \quad (10)$$

in which $M \in \{na, a, fa\}$ and H$\in$ {Ran.Seq., Dec.Deg., Inc.Deg.}.

As can be seen, HUB−GA is able to speed up the broadcast process up to almost 60% compared to degree-based heuristics for both networks. The main reason for high fluctuations in Fig. 9 and 10 are as follows:

1) **Randomness of HUB−GA:** As mentioned earlier, GA is a random search algorithm that starts with generating several random solutions for the problem at its first step. Although GA is expected to approach a nearly-optimal solution in the long run, it is possible that it does not find it in some situations. Therefore, the performance of HUB−GA is random. However, our extensive experiments show that our heuristic is quite effective dealing with real-world data sets.
2) **Randomness of other heuristics:** When the performance of HUB−GA is compared with Ran.Seq. the fluctuations are wider. This is due to the pure randomness of Ran.Seq. heuristic. Even though we took the average results of Ran.Seq. over 5 runs, it still exhibit its random nature which leads to high fluctuations.
3) **Random ordering of deterministic heuristics:** Consider other deterministic heuristics, Dec.Deg. or Inc.Deg.. In both heuristics, the ties are broken randomly. In graphs with several vertices with the same degree (such as the Windmill graph), these heuristics are almost random, with the only difference being the position of the joint vertex. This fact justifies more drastic fluctuations in Fig. 10 (Windmill graph) compared to Fig. 9 (Ring of cliques).
4) **x-axis:** In Figs. 9 and 10, the x-axis is the experiment number. According to Table X and IX, the experiment number does not have a monotonic relation with neither $|V|$ nor $|E|$. However, we decided to choose the experiment number as the x-axis (and for instance, not sort the experiments based on $|E|$) to avoid confusion. We believe this is the best way to represent our findings.

TABLE IX
RESULTS OF EXPERIMENT 3 FOR $W_{k,n}$, WHERE $3 \leq k, n \leq 6$.

| $W_{k,n}$ | | $\|V\|$ | $\|E\|$ | Non-adaptive model | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | | | | Avg | | | | Max | | | |
| $k$ | $n$ | | | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA |
| 3 | 3 | 7 | 9 | 4.4 | 5 | **4** | **4** | 5.94 | 5.57 | 5.42 | **4.71** | 7 | **6** | **6** | **6** |
| 4 | 3 | 9 | 12 | 5.8 | 7 | 8 | **5** | 7.2 | 8.44 | 9.33 | **5.88** | 7.8 | 9 | 10 | **7** |
| 5 | 3 | 11 | 15 | 7.6 | 9 | 8 | **6** | 8.96 | 9.72 | 9.63 | **6.81** | 10 | 10 | 10 | **8** |
| 6 | 3 | 13 | 18 | 8.8 | 10 | 10 | **7** | 11.49 | 11.61 | 11.53 | **7.92** | 12.4 | 12 | 12 | **9** |
| 3 | 4 | 10 | 18 | 5.6 | 6 | 7 | **4** | 7.72 | 7.3 | 8.8 | **5.1** | 9.2 | 8 | 10 | **6** |
| 4 | 4 | 13 | 24 | 6.2 | 6 | 6 | **5** | 9.12 | 7.46 | 8.53 | **6.07** | 11 | 8 | 9 | **7** |
| 5 | 4 | 16 | 30 | 8 | 8 | 9 | **6** | 11.23 | 10.37 | 11.25 | **7.31** | 13 | 11 | 12 | **8** |
| 6 | 4 | 19 | 36 | 11.6 | 10 | 11 | **8** | 14.53 | 14.15 | 15.89 | **8.84** | 16.2 | 15 | 17 | **10** |
| 3 | 5 | 13 | 30 | 6.4 | 7 | 9 | **5** | 9.33 | 9.69 | 11.46 | **6.07** | 11.4 | 11 | 13 | **8** |
| 4 | 5 | 17 | 40 | 6.8 | 7 | 9 | **6** | 9.95 | 10 | 11.82 | **7.11** | 12.4 | 11 | 13 | **9** |
| 5 | 5 | 21 | 50 | 10.4 | 9 | 8 | **7** | 14.45 | 10.57 | 11.61 | **8.14** | 16.8 | 11 | 12 | **9** |
| 6 | 5 | 25 | 60 | 11 | 11 | 11 | **8** | 14.4 | 13.48 | 14.52 | **9.08** | 16.6 | 14 | 15 | **11** |
| 3 | 6 | 16 | 45 | 6.6 | 8 | 7 | **5** | 9.51 | 8.62 | 11.37 | **6.37** | 11.8 | 9 | 12 | **8** |
| 4 | 6 | 21 | 60 | 9.4 | 10 | 9 | **6** | 12.49 | 10.95 | 13.76 | **7.33** | 14.6 | 11 | 14 | **9** |
| 5 | 6 | 26 | 75 | 11.6 | 12 | 12 | **7** | 15.6 | 14.38 | 16.42 | **8.53** | 18.2 | 15 | 17 | **10** |
| 6 | 6 | 31 | 90 | 13.2 | 16 | 18 | **9** | 17.54 | 21 | 22.87 | **9.93** | 19.6 | 22 | 24 | **11** |

| $W_{k,n}$ | | $\|V\|$ | $\|E\|$ | Adaptive model | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | | | | Avg | | | | Max | | | |
| $k$ | $n$ | | | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA |
| 3 | 3 | 7 | 9 | 4.2 | **4** | **4** | **4** | 4.82 | **4.28** | 4.85 | **4.28** | 5.4 | **5** | **5** | **5** |
| 4 | 3 | 9 | 12 | 5.8 | 6 | 7 | **5** | 6.35 | 6.77 | 7.44 | **5.22** | 7.2 | 7 | 8 | **6** |
| 5 | 3 | 11 | 15 | 6.6 | **6** | 7 | **6** | 7.92 | 7 | 7.72 | **6.18** | 9 | 8 | 8 | **7** |
| 6 | 3 | 13 | 18 | 8 | 8 | 9 | **7** | 8.7 | 8.69 | 9.46 | **7.15** | 9.4 | 10 | 10 | **8** |
| 3 | 4 | 10 | 18 | 5.2 | 5 | 7 | **4** | 6.65 | 6.2 | 7.6 | **4.7** | 7.8 | 7 | 8 | **5** |
| 4 | 4 | 13 | 24 | 6.6 | 6 | 7 | **5** | 7.5 | 6.92 | 8.38 | **5.38** | 8.6 | 8 | 9 | **6** |
| 5 | 4 | 16 | 30 | 8 | 7 | 9 | **6** | 9.62 | 9.5 | 11.12 | **7** | 11 | 11 | 12 | **8** |
| 6 | 4 | 19 | 36 | 9.2 | **7** | 8 | **7** | 10.41 | 7.78 | 9.89 | **7.21** | 11.6 | **8** | 10 | **8** |
| 3 | 5 | 13 | 30 | 6.2 | 6 | 8 | **5** | 7.53 | 7.23 | 9.3 | **5.69** | 8.6 | 8 | 10 | **7** |
| 4 | 5 | 17 | 40 | 6.8 | 8 | 10 | **6** | 8.71 | 9.76 | 12.17 | **6.47** | 10.2 | 11 | 14 | **7** |
| 5 | 5 | 21 | 50 | 8.6 | 8 | 9 | **7** | 10.11 | 9 | 11.33 | **7.66** | 11.6 | 10 | 13 | **9** |
| 6 | 5 | 25 | 60 | 10.6 | 10 | 11 | **8** | 12.31 | 11.2 | 13.2 | **8.4** | 14.2 | 12 | 15 | **9** |
| 3 | 6 | 16 | 45 | 6.2 | 5 | 6 | **5** | 8.01 | 7 | 8.68 | **5.68** | 9.6 | 8 | 10 | **6** |
| 4 | 6 | 21 | 60 | 8.6 | 8 | 10 | **6** | 10.72 | 9.19 | 12 | **6.9** | 12.2 | 10 | 13 | **8** |
| 5 | 6 | 26 | 75 | 9.2 | 9 | 11 | **7** | 11.36 | 10.84 | 13.76 | **7.88** | 13 | 12 | 15 | **9** |
| 6 | 6 | 31 | 90 | 10.6 | 12 | 13 | **8** | 12.58 | 12.93 | 15.16 | **8.61** | 14.4 | 14 | 17 | **10** |

| $W_{k,n}$ | | $\|V\|$ | $\|E\|$ | Fully-adaptive model | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | | | | Avg | | | | Max | | | |
| $k$ | $n$ | | | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA |
| 3 | 3 | 7 | 9 | 4.2 | **4** | **4** | **4** | 4.91 | **4.14** | 4.85 | **4.14** | 5.6 | **5** | **5** | **5** |
| 4 | 3 | 9 | 12 | 5.4 | **5** | 6 | **5** | 6.06 | 6.11 | 6.88 | **5.11** | 7 | 7 | 8 | **6** |
| 5 | 3 | 11 | 15 | 6.4 | **6** | 6 | **6** | 6.96 | **6.09** | 6.9 | **6.09** | 7.6 | **7** | **7** | **7** |
| 6 | 3 | 13 | 18 | 8 | 8 | 9 | **7** | 8.9 | 8.766 | 9.61 | **7.07** | 9.8 | 10 | 10 | **8** |
| 3 | 4 | 10 | 18 | 5.2 | 5 | 6 | **4** | 6.21 | 5.6 | 6.9 | **4.4** | 7.2 | 7 | 8 | **5** |
| 4 | 4 | 13 | 24 | 6 | 6 | 7 | **5** | 7.3 | 6.92 | 8.38 | **5.23** | 8.6 | 8 | 9 | **6** |
| 5 | 4 | 16 | 30 | 6.4 | **6** | 8 | **6** | 8 | 7.68 | 9.31 | **6.18** | 9.4 | 9 | 10 | **7** |
| 6 | 4 | 19 | 36 | 8.2 | 8 | 9 | **7** | 9.69 | 8.78 | 10.42 | **7.21** | 11 | 10 | 11 | **8** |
| 3 | 5 | 13 | 30 | 5.6 | **5** | 6 | **5** | 6.89 | 6.15 | 7.92 | **5.3** | 8.2 | 7 | 9 | **6** |
| 4 | 5 | 17 | 40 | 7.2 | **6** | 7 | **6** | 8.43 | 7.17 | 9.05 | **6.23** | 9.8 | 8 | 10 | **7** |
| 5 | 5 | 21 | 50 | 7.4 | **7** | 8 | **7** | 8.91 | 8.04 | 10.14 | **7.28** | 10.2 | 9 | 11 | **8** |
| 6 | 5 | 25 | 60 | 9 | **8** | 9 | **8** | 10.87 | 9.16 | 11.4 | **8.12** | 12.6 | 10 | 12 | **9** |
| 3 | 6 | 16 | 45 | 6.2 | 7 | 7 | **5** | 7.45 | 7.18 | 9.5 | **5.56** | 8.6 | 8 | 10 | **7** |
| 4 | 6 | 21 | 60 | 7.6 | 7 | 8 | **6** | 9.53 | 8.28 | 10.76 | **6.52** | 11 | 9 | 12 | **7** |
| 5 | 6 | 26 | 75 | 9.8 | 11 | 13 | **7** | 11.51 | 12.15 | 14.73 | **7.34** | 13.4 | 14 | 16 | **8** |
| 6 | 6 | 31 | 90 | 10 | 11 | 12 | **8** | 12.19 | 11.67 | 14.35 | **8.58** | 13.6 | 13 | 16 | **10** |

### *D. Experiment 4*

The objective of this experiment is to compare our heuristic with some state-of-the-art heuristics in the literature. Since some instances used in previous works, such as [4], [24], [94] are no longer available, we used two types of networks:

1) **Interconnection Networks (44 instances):** We replicated several instances from [4], [25], [34], [35], [95]. These networks include 8 instances of hypercube $HC_d$, 5 instances of cube connected cycle $CCC_d$, 7 instances of De Bruijn $DB_d$, 8 instances of Shuffle Exchange $SE_d$ [1], and 16 instances of Harary $H_{k,n}$ [2].
2) **Connected Complex Networks (30 instances):** well-established instances of connected complex networks from network repository[3] [106]. In order to address various industry scenarios, we take into account instances based on small-world networks with 100 nodes [95], [111]. Note that the small-world model could be used to represent communication networks in real-world applications, as suggested by [95], [112], [113].

For each instance we report $|V|$ (the number of vertices), $|E|$ (the number of edges), and density (edge density of the graph: $\frac{2|E|}{|V|(|V|-1)}$.) As mentioned before, our heuristic is the first of its kind in the literature for broadcasting with universal lists. Thus, we can only compare our heuristic with similar results under the classical model. We compared our results with two lower bounds:

- **TLB:** The trivial lower bound on the classical broadcast time of graph $G$ on $n$ vertices: $B_{cl}(G) \geq \lceil \log n \rceil$.
- **LBB [95]:** The lower bound on the classical model suggested in [95], namely LBB−BFS. The goal here is to find the maximum shortest path between any receiver and the originator (vertex $v$) by performing a BFS algorithm. Then, a lower bound on the value of $B_{cl}(v, G)$ is found,

[1]Data available here: https://github.com/sabergholami/
[2]Data available here: https://github.com/alfredolimams/
[3]Data available here: https://networkrepository.com/rand.php/

TABLE X
RESULTS OF EXPERIMENT 3 FOR $RC_{n,m}$, WHERE $3 \leq n, m \leq 6$.

| $RC_{n,m}$ | | $\|V\|$ | $\|E\|$ | Non-adaptive model | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | | | | Avg | | | | Max | | | |
| n | m | | | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA |
| 3 | 3 | 9 | 12 | **5** | **5** | 6 | **5** | 6.22 | 6.11 | 6.77 | **5.33** | 7.4 | 7 | 7 | **6** |
| 4 | 3 | 12 | 16 | 6.6 | 7 | 9 | **6** | 8.38 | 7.91 | 9.41 | **6.83** | 10.4 | 9 | 10 | **8** |
| 5 | 3 | 15 | 20 | 8.4 | 8 | 10 | **7** | 9.93 | 9 | 10.86 | **8.33** | 12 | **10** | 12 | **10** |
| 6 | 3 | 18 | 24 | 10.2 | **9** | 12 | **9** | 12.04 | 10.44 | 13.77 | **9.66** | 13.8 | 12 | 15 | **10** |
| 3 | 4 | 12 | 21 | 6.2 | 6 | 9 | **5** | 8.06 | 7.08 | 9.5 | **5.83** | 9.6 | 8 | 10 | **7** |
| 4 | 4 | 16 | 28 | 8.4 | **7** | 12 | **7** | 10.33 | 8.81 | 13.56 | **7.81** | 12.6 | 10 | 15 | **9** |
| 5 | 4 | 20 | 35 | 10.8 | **8** | 15 | **8** | 13.05 | 10.2 | 16.1 | **9.45** | 15.8 | **12** | 17 | **12** |
| 6 | 4 | 24 | 42 | 12.2 | **10** | 18 | **10** | 14.91 | 11.79 | 19.95 | **11.29** | 17.8 | 14 | 22 | **13** |
| 3 | 5 | 15 | 33 | 7.2 | 7 | 12 | **6** | 9.78 | 7.93 | 12.4 | **7.2** | 11.8 | **9** | 14 | **9** |
| 4 | 5 | 20 | 44 | 8.8 | 8 | 16 | **7** | 12.37 | 9.9 | 17.75 | **8.65** | 15.4 | **11** | 19 | **11** |
| 5 | 5 | 25 | 55 | 11.6 | 9 | 21 | **8** | 14.55 | 11.12 | 21.84 | **10.04** | 17.6 | 13 | 23 | **12** |
| 6 | 5 | 30 | 66 | 14 | **11** | 24 | **11** | 17.91 | 12.73 | 26.86 | **12.13** | 21.8 | **14** | 29 | **14** |
| 3 | 6 | 18 | 48 | 8.8 | 8 | 15 | **6** | 10.94 | 8.77 | 15.77 | **7.55** | 13 | 10 | 17 | **9** |
| 4 | 6 | 24 | 64 | 11 | 9 | 20 | **7** | 14.52 | 10.29 | 21.83 | **9.2** | 17.8 | 12 | 24 | **11** |
| 5 | 6 | 30 | 80 | 14 | 11 | 25 | **9** | 17.22 | 12 | 26.53 | **10.8** | 20 | **13** | 27 | **13** |
| 6 | 6 | 36 | 96 | 16.4 | **11** | 30 | **11** | 20.59 | 13.66 | 33.05 | **12.44** | 23.8 | 16 | 36 | **14** |

| $RC_{n,m}$ | | $\|V\|$ | $\|E\|$ | Adaptive model | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | | | | Avg | | | | Max | | | |
| n | m | | | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA |
| 3 | 3 | 9 | 12 | 4.4 | **4** | 5 | **4** | 4.8 | **4.33** | 5 | **4.33** | 5 | **5** | 5 | **5** |
| 4 | 3 | 12 | 16 | 5.6 | **5** | 6 | **5** | 6.4 | **5.33** | 6.66 | **5.33** | 7 | **6** | 7 | **6** |
| 5 | 3 | 15 | 20 | 6.8 | **6** | 8 | **6** | 7.58 | **6.33** | 8 | **6.33** | 8 | **7** | 8 | **7** |
| 6 | 3 | 18 | 24 | 8 | **7** | 9 | **7** | 8.86 | **7.33** | 9.38 | 7.61 | 10 | **8** | 10 | **8** |
| 3 | 4 | 12 | 21 | 5.6 | 5 | 6 | **4** | 6.13 | 5.41 | 6.83 | **5.08** | 6.8 | **6** | 7 | **6** |
| 4 | 4 | 16 | 28 | 7 | 6 | 9 | **5** | 8.36 | 6.5 | 9.12 | **6.25** | 9.4 | **7** | 10 | **7** |
| 5 | 4 | 20 | 35 | 8.4 | 7 | 11 | **6** | 9.77 | **7.25** | 11 | **7.25** | 10.8 | **8** | 11 | **8** |
| 6 | 4 | 24 | 42 | 10.4 | **8** | 12 | **8** | 12.15 | **8.29** | 12.91 | 8.41 | 13.6 | **9** | 13 | **9** |
| 3 | 5 | 15 | 33 | 6.6 | **5** | 8 | **5** | 7.71 | 6 | 8.13 | **5.6** | 8.6 | 7 | 9 | **6** |
| 4 | 5 | 20 | 44 | 7.8 | **6** | 11 | **6** | 9.45 | 7.35 | 11.35 | **6.8** | 11 | **8** | 12 | **8** |
| 5 | 5 | 25 | 55 | 9.8 | **7** | 12 | **7** | 11.52 | **7.64** | 12.4 | 7.96 | 13.2 | **8** | 13 | 9 |
| 6 | 5 | 30 | 66 | 12 | **8** | 14 | **8** | 13.97 | 9.16 | 15.26 | **9.03** | 15.8 | **10** | 17 | **10** |
| 3 | 6 | 18 | 48 | 6.8 | **6** | 9 | **6** | 8.46 | 7.05 | 9.55 | **6.44** | 9.6 | 8 | 10 | **7** |
| 4 | 6 | 24 | 64 | 9 | 7 | 13 | **6** | 11.12 | 7.69 | 13.04 | **7.45** | 13 | **8** | 14 | **8** |
| 5 | 6 | 30 | 80 | 11 | **8** | 14 | **8** | 13.15 | 8.93 | 14.5 | **8.86** | 14.8 | **10** | 15 | **10** |
| 6 | 6 | 36 | 96 | 13.8 | 9 | 17 | **8** | 15.66 | 9.83 | 18 | **9.8** | 17.6 | **11** | 19 | **11** |

| $RC_{n,m}$ | | $\|V\|$ | $\|E\|$ | Fully-adaptive model | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Min | | | | Avg | | | | Max | | | |
| n | m | | | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA | Ran.Seq. | Dec.Deg. | Inc.Deg. | HUB-GA |
| 3 | 3 | 9 | 12 | 4.4 | **4** | 5 | **4** | 4.86 | **4.33** | 5 | **4.33** | 5 | **5** | 5 | **5** |
| 4 | 3 | 12 | 16 | 5.8 | **5** | 6 | **5** | 6.25 | **5.33** | 6.41 | **5.33** | 7 | **6** | 7 | **6** |
| 5 | 3 | 15 | 20 | 6.6 | **6** | 8 | **6** | 7.53 | **6.33** | 8 | **6.33** | 8 | **7** | 8 | **7** |
| 6 | 3 | 18 | 24 | 8 | **7** | 9 | **7** | 8.73 | **7.33** | 9.44 | **7.33** | 9.2 | **8** | 10 | **8** |
| 3 | 4 | 12 | 21 | 5.4 | **4** | 6 | 5 | 5.9 | **5** | 6.66 | **5** | 6.6 | 6 | 7 | **5** |
| 4 | 4 | 16 | 28 | 6.6 | **5** | 8 | **5** | 7.82 | 5.87 | 8.81 | **5.5** | 9 | 7 | 10 | **6** |
| 5 | 4 | 20 | 35 | 8.2 | **6** | 11 | **6** | 9.36 | 7.1 | 11 | **7.05** | 10.4 | **8** | 11 | **8** |
| 6 | 4 | 24 | 42 | 10.2 | **7** | 13 | **7** | 11.23 | 8.25 | 13.29 | **8.16** | 13 | **9** | 14 | **9** |
| 3 | 5 | 15 | 33 | 6 | **5** | 7 | **5** | 6.7 | **5.6** | 7.86 | 5.73 | 7.2 | **6** | 9 | **6** |
| 4 | 5 | 20 | 44 | 7.6 | **6** | 9 | **6** | 9.05 | 6.66 | 9.9 | **6.65** | 10.2 | **7** | 11 | **7** |
| 5 | 5 | 25 | 55 | 9.4 | **7** | 12 | **7** | 10.7 | 7.92 | 13.04 | **7.8** | 12.2 | **9** | 14 | **9** |
| 6 | 5 | 30 | 66 | 11 | **8** | 14 | **8** | 12.54 | **8.6** | 15.03 | 9.16 | 14 | **9** | 16 | 10 |
| 3 | 6 | 18 | 48 | 6.8 | 6 | 8 | **5** | 7.76 | 6.66 | 9.27 | **5.88** | 8.6 | **7** | 10 | **7** |
| 4 | 6 | 24 | 64 | 8.6 | **7** | 12 | **7** | 10.17 | 7.95 | 12.83 | **7.5** | 11.6 | 9 | 14 | **8** |
| 5 | 6 | 30 | 80 | 10.4 | **7** | 13 | 8 | 11.66 | **8.26** | 14.03 | 9.1 | 13 | **9** | 15 | 10 |
| 6 | 6 | 36 | 96 | 11.8 | **8** | 16 | 9 | 13.71 | **9.25** | 17.16 | 10.27 | 15.4 | **10** | 19 | 11 |

which is also a valid lower bound on the $B_{cl}(G)$.

Also, we compare our results with six upper bounds reported in the literature:

- **`TreeBlock` [25], [114]:** the constructive heuristic proposed in [25], with a more detailed version presented in [114] considering various network families. `TreeBlock` works based on detecting cut-points of graph $G$ and breaking the problem down into several smaller problems in trees that are formed from the cut-points.
- **`NTBA` [34]:** This algorithm works based on forming a layer graph from the original graph by performing a BFS. Then, it traverses the layer graph and produces a broadcast scheme for the originator.
- **`NEWH` [35]:** This algorithm builds upon the idea of `NTBA` but applies a new non-random strategy in order to generate a spanning tree for broadcasting. The advantage of this algorithm is that almost half of vertices are informed by the shortest path from the originator, while the rest are informed by a path at most 3 hops longer.
- **`ILP` [25], [95]:** Consider exact algorithms on the classical broadcast problem. In addition to the dynamic programming algorithm presented in [24], the most successful approach is the linear programming model of de Sousa et al. [25]. However, only cases with up to about 50 vertices can be solved using this model in a fair amount of time, which is insufficient for actual industrial applications. The `ILP` algorithm [95] works based on the algorithm suggested in [25]. It is an integer linear programming algorithm for the classical broadcast problem, solved by IBM Cplex 12.9.
- **`ACS` [4]:** The ant colony system algorithm proposed by Hasson and Sipper [4]. Note that the results reported by the authors of [4] illustrate that `ACS` outperforms the algorithm by Hoelting et al. [94]. Thus, we did not consider their algorithm.
- **`BRKGA` [95]:** The biased random key genetic algorithm proposed in [95] with the first receive first send decoder. Considering the results reported in the corresponding

TABLE XI
RESULTS OF EXPERIMENT 4 FOR INTERCONNECTION NETWORKS.

| Instance | $\|V\|$ | $\|E\|$ | Density | *LB* on $B_{cl}(v,G)$ | | *UB* on $B_{cl}(v,G)$ | | | | | | HUB-GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | TLB | LBB | TreeBlock | NTBA | NEWH | ILP | ACS | BRKGA | $B^{\sigma}_{fa}(G)$ | $B^{\sigma}_{a}(G)$ | $B^{\sigma}_{na}(G)$ |
| $HC_2$ | 4 | 4 | 0.6667 | 2 | - | - | - | - | - | - | - | 2 | 2 | 2 |
| $HC_3$ | 8 | 12 | 0.4285 | 3 | - | - | - | - | - | - | - | 3 | 4 | 4 |
| $HC_4$ | 16 | 32 | 0.2667 | 4 | - | - | - | - | 4 | - | - | 4 | 5 | 6 |
| $HC_5$ | 32 | 80 | 0.1613 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 9 |
| $HC_6$ | 64 | 192 | 0.0952 | 6 | 6 | 6 | 7 | 6 | 6 | 6 | 6 | 7 | 9 | 11 |
| $HC_7$ | 128 | 448 | 0.0551 | 7 | 7 | 7 | 9 | 7 | 7 | 7 | 7 | 8 | 11 | 14 |
| $HC_8$ | 256 | 1024 | 0.0314 | 8 | 8 | 8 | 11 | 8 | 8 | 8 | 9 | 9 | 13 | 16 |
| $HC_9$ | 512 | 2304 | 0.0176 | 9 | 9 | 9 | 14 | 9 | 9 | 9 | 10 | 10 | 15 | 18 |
| $CCC_2$ | 8 | 12 | 0.4285 | 3 | - | - | - | - | - | - | - | 4 | 4 | 5 |
| $CCC_3$ | 24 | 36 | 0.1304 | 5 | 6 | - | 6 | 7 | 6 | 6 | 6 | 8 | 8 | 10 |
| $CCC_4$ | 64 | 94 | 0.0476 | 6 | 8 | - | 7 | 9 | 9 | 9 | 9 | 11 | 11 | 15 |
| $CCC_5$ | 160 | 240 | 0.0189 | 8 | 10 | - | 11 | 12 | 11 | 12 | 11 | 14 | 15 | 19 |
| $CCC_6$ | 384 | 576 | 0.0078 | 9 | 13 | - | 14 | 14 | 13 | 14 | 13 | 17 | 18 | 24 |
| $DB_3$ | 8 | 16 | 0.5714 | 3 | - | - | 4 | 4 | - | - | - | 4 | 4 | 5 |
| $DB_4$ | 16 | 32 | 0.2583 | 4 | 4 | 4 | 5 | 5 | - | 5 | 5 | 6 | 6 | 7 |
| $DB_5$ | 32 | 64 | 0.1270 | 5 | 5 | 7 | 7 | 7 | - | 6 | 6 | 7 | 8 | 10 |
| $DB_6$ | 64 | 128 | 0.0630 | 6 | 6 | 8 | 8 | 8 | - | 8 | 8 | 9 | 10 | 13 |
| $DB_7$ | 128 | 256 | 0.0314 | 7 | 7 | 12 | 10 | 10 | - | 10 | 9 | 11 | 12 | 16 |
| $DB_8$ | 256 | 512 | 0.0157 | 8 | 8 | 12 | 12 | 12 | - | 12 | 11 | 13 | 14 | 19 |
| $DB_9$ | 512 | 1024 | 0.0078 | 9 | 9 | 14 | 13 | 13 | - | 14 | 13 | 15 | 17 | 22 |
| $SE_2$ | 4 | 5 | 0.8334 | 2 | - | - | - | - | - | - | - | 3 | 3 | 4 |
| $SE_3$ | 8 | 12 | 0.4285 | 3 | - | - | 5 | 5 | - | - | - | 5 | 5 | 6 |
| $SE_4$ | 16 | 21 | 0.1750 | 4 | 7 | - | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 9 |
| $SE_5$ | 32 | 46 | 0.0927 | 5 | 9 | - | 9 | 9 | 9 | 9 | 9 | 9 | 10 | 13 |
| $SE_6$ | 64 | 93 | 0.0461 | 6 | 11 | - | 11 | 11 | 11 | 11 | 11 | 12 | 12 | 16 |
| $SE_7$ | 128 | 190 | 0.0234 | 7 | 13 | - | 13 | 13 | 13 | 13 | 13 | 15 | 15 | 20 |
| $SE_8$ | 256 | 381 | 0.0117 | 8 | 15 | - | 15 | 15 | 15 | 15 | 15 | 17 | 17 | 25 |
| $SE_9$ | 512 | 766 | 0.0059 | 9 | 17 | - | 18 | 18 | 17 | 17 | 18 | 20 | 21 | 28 |
| $H_{10,30}$ | 30 | 150 | 0.3448 | 5 | 3 | 6 | - | - | 5 | 5 | 5 | 7 | 9 | 9 |
| $H_{11,50}$ | 50 | 275 | 0.2245 | 6 | 3 | 7 | - | - | 6 | 6 | 6 | 8 | 10 | 11 |
| $H_{20,50}$ | 50 | 500 | 0.4082 | 6 | 3 | 8 | - | - | 6 | 6 | 6 | 8 | 10 | 11 |
| $H_{21,50}$ | 50 | 525 | 0.4286 | 6 | 2 | 7 | - | - | 6 | 6 | 6 | 7 | 10 | 10 |
| $H_{2,100}$ | 100 | 100 | 0.0202 | 7 | 50 | 50 | - | - | 50 | 50 | 50 | 50 | 50 | 67 |
| $H_{2,17}$ | 17 | 17 | 0.1250 | 4 | 8 | 9 | - | - | 9 | 9 | 9 | 9 | 9 | 11 |
| $H_{2,30}$ | 30 | 30 | 0.0690 | 5 | 15 | 15 | - | - | 15 | 15 | 15 | 15 | 15 | 20 |
| $H_{2,50}$ | 50 | 50 | 0.0408 | 6 | 25 | 25 | - | - | 25 | 25 | 25 | 25 | 25 | 29 |
| $H_{3,17}$ | 17 | 26 | 0.1912 | 4 | 4 | 5 | - | - | 5 | 5 | 5 | 6 | 6 | 8 |
| $H_{3,30}$ | 30 | 45 | 0.1034 | 5 | 8 | 9 | - | - | 9 | 9 | 9 | 9 | 9 | 12 |
| $H_{3,50}$ | 50 | 75 | 0.0612 | 6 | 13 | 14 | - | - | 14 | 14 | 14 | 14 | 15 | 17 |
| $H_{5,17}$ | 17 | 43 | 0.3162 | 4 | 3 | 5 | - | - | 5 | 5 | 5 | 5 | 6 | 7 |
| $H_{6,17}$ | 17 | 51 | 0.3750 | 4 | 3 | 5 | - | - | 5 | 5 | 5 | 6 | 6 | 7 |
| $H_{7,17}$ | 17 | 60 | 0.4412 | 4 | 2 | 5 | - | - | 5 | 5 | 5 | 5 | 6 | 6 |
| $H_{8,30}$ | 30 | 120 | 0.2759 | 5 | 4 | 6 | - | - | 5 | 6 | 5 | 8 | 9 | 10 |
| $H_{9,30}$ | 30 | 135 | 0.3103 | 5 | 3 | 6 | - | - | 5 | 5 | 5 | 7 | 8 | 9 |

study, this could be considered the most successful and recent heuristic for the classical broadcast problem. BRKGA_FRFS is believed to outperform several well-known heuristics for the classical broadcast problem such as Tree Block [25], NTBA [34], and NEWH [35].

The numerical results of this experiment are reported in Tables XI and XII. The results of all lower and upper bounds are reproduced from their original paper, respectively. A hyphen in Tables XI and XII indicates that we do not have the result for an instance or the method could not produce a feasible solution. Also, in the case that the result of a particular algorithm is not reported for an instance, we used the result reported in [95], as they re-implemented all algorithms.

We need to point out that the value achieved by all upper bounds concerns $B_{cl}(v,G)$ for a particular $v$ as the originator (for most instances, $v = \{1\}$). Recall that $\forall v : B_{cl}(v,G) \leq B_{cl}(G)$. Also, from (2), it is clear that the value of $B^{\sigma}_{M}(G)$ is lower bounded by $B_{cl}(G)$ for $M = \{fa, a, na\}$. Thus, the values achieved by HUB-GA are not supposed to outperform the six upper bounds. In fact, it could be impossible to achieve those values if the chosen originator in a competitive algorithm is not the worst originator. Finally, in all competitor heuristics, the goal is to minimize the broadcast time of a vertex in a given graph. This is somewhat in contrast with the nature of the universal list model, in which the goal is to optimize the behavior of all nodes simultaneously with one universal list. However, we performed this experiment to see whether $B^{\sigma}_{M}(G)$ could get close to $B_{cl}(v,G)$ or not.

Our numerical results demonstrate that as opposed to the significant memory reduction in the universal list model compared to the classical model, as well as the need for only local knowledge for nodes, the universal list broadcast time of all instances considered in this study is very close to that of classical. In fact, most cases have between 1 to 3 time units difference.

For most instances reported in Table XI, HUB-GA finds a sufficiently close broadcast time under all three models

TABLE XII
RESULTS OF EXPERIMENT 4 FOR CONNECTED COMPLEX NETWORKS.

| Instance | $\|V\|$ | $\|E\|$ | Density | *LB* on $B_{cl}(v,G)$ | *UB* on $B_{cl}(v,G)$ | | | | | | HUB-GA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | TLB | LBB | TreeBlock | NTBA | NEWH | ILP | ACS | BRKGA | $B^{\sigma}_{fa}(G)$ | $B^{\sigma}_{a}(G)$ | $B^{\sigma}_{na}(G)$ |
| SW-100-3-0d1-trial1 | 100 | 100 | 0.0202 | 7 | 61 | - | - | - | 61 | 61 | 61 | 68 | 68 | 104 |
| SW-100-3-0d2-trial1 | 100 | 100 | 0.0202 | 7 | 31 | - | - | - | 31 | 31 | 31 | 40 | 40 | 60 |
| SW-100-3-0d2-trial3 | 100 | 100 | 0.0202 | 7 | 31 | - | - | - | 31 | 31 | 31 | 49 | 49 | 74 |
| SW-100-4-0d1-trial1 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 9 | 10 | 9 | 14 | 14 | 19 |
| SW-100-4-0d1-trial2 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 8 | 9 | 8 | 13 | 14 | 18 |
| SW-100-4-0d1-trial3 | 100 | 200 | 0.0404 | 7 | 9 | - | - | - | 10 | 11 | 10 | 15 | 16 | 20 |
| SW-100-4-0d2-trial1 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 8 | 9 | 8 | 12 | 13 | 17 |
| SW-100-4-0d2-trial2 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 8 | 9 | 9 | 12 | 13 | 16 |
| SW-100-4-0d2-trial3 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 9 | 9 | 9 | 12 | 13 | 17 |
| SW-100-4-0d3-trial1 | 100 | 200 | 0.0404 | 7 | 6 | - | - | - | 8 | 9 | 8 | 12 | 13 | 16 |
| SW-100-4-0d3-trial2 | 100 | 200 | 0.0404 | 7 | 6 | - | - | - | 8 | 8 | 8 | 11 | 12 | 15 |
| SW-100-4-0d3-trial3 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 8 | 9 | 8 | 11 | 12 | 15 |
| SW-100-5-0d1-trial1 | 100 | 200 | 0.0404 | 7 | 8 | - | - | - | 9 | 10 | 9 | 14 | 15 | 19 |
| SW-100-5-0d1-trial2 | 100 | 200 | 0.0404 | 7 | 9 | - | - | - | 10 | 11 | 10 | 15 | 15 | 22 |
| SW-100-5-0d1-trial3 | 100 | 200 | 0.0404 | 7 | 11 | - | - | - | 12 | 13 | 12 | 15 | 16 | 21 |
| SW-100-5-0d2-trial1 | 100 | 200 | 0.0404 | 7 | 8 | - | - | - | 9 | 10 | 10 | 13 | 14 | 17 |
| SW-100-5-0d2-trial2 | 100 | 200 | 0.0404 | 7 | 9 | - | - | - | 9 | 10 | 10 | 12 | 13 | 17 |
| SW-100-5-0d2-trial3 | 100 | 200 | 0.0404 | 7 | 7 | - | - | - | 8 | 9 | 9 | 12 | 13 | 18 |
| SW-100-5-0d3-trial1 | 100 | 200 | 0.0404 | 7 | 6 | - | - | - | 8 | 8 | 8 | 11 | 12 | 15 |
| SW-100-5-0d3-trial2 | 100 | 200 | 0.0404 | 7 | 6 | - | - | - | 8 | 8 | 8 | 11 | 12 | 16 |
| SW-100-5-0d3-trial3 | 100 | 200 | 0.0404 | 7 | 6 | - | - | - | 8 | 8 | 8 | 11 | 12 | 15 |
| SW-100-6-0d1-trial1 | 100 | 300 | 0.0606 | 7 | 5 | - | - | - | 7 | 8 | 8 | 12 | 13 | 16 |
| SW-100-6-0d1-trial2 | 100 | 300 | 0.0606 | 7 | 6 | - | - | - | 8 | 9 | 8 | 12 | 13 | 16 |
| SW-100-6-0d1-trial3 | 100 | 300 | 0.0606 | 7 | 6 | - | - | - | 7 | 8 | 8 | 12 | 14 | 17 |
| SW-100-6-0d2-trial1 | 100 | 300 | 0.0606 | 7 | 6 | - | - | - | 7 | 8 | 7 | 11 | 13 | 15 |
| SW-100-6-0d2-trial2 | 100 | 300 | 0.0606 | 7 | 4 | - | - | - | 7 | 8 | 7 | 10 | 12 | 14 |
| SW-100-6-0d2-trial3 | 100 | 300 | 0.0606 | 7 | 4 | - | - | - | 7 | 8 | 7 | 10 | 12 | 15 |
| SW-100-6-0d3-trial1 | 100 | 300 | 0.0606 | 7 | 4 | - | - | - | 7 | 8 | 7 | 10 | 11 | 14 |
| SW-100-6-0d3-trial2 | 100 | 300 | 0.0606 | 7 | 5 | - | - | - | 7 | 8 | 7 | 10 | 11 | 13 |
| SW-100-6-0d3-trial3 | 100 | 300 | 0.0606 | 7 | 5 | - | - | - | 7 | 8 | 7 | 10 | 11 | 14 |

($M = \{fa, a, na\}$) compared to the best upper bound reported in the literature. This gap is even smaller for Harary networks $H_{k,n}$ and Shuffle Exchange graphs $SE_d$, where the fully-adaptive and adaptive broadcast time is even equal to the classical upper bounds in several instances. For complex networks in Table XII, HUB−GA is able to decrease its objective function to 10, 11, and 14 for various instances under fully-adaptive, adaptive, and non-adaptive models, respectively. The value of classical broadcasting is mostly close to 8 in those instances. This result is promising in the sense that with a careful design of the broadcast scheme under the universal list model, the best-known upper bounds reported in the literature for the classical model are achievable.

## VI. CONCLUSION

In this paper we proposed HUB−GA: A heuristic for the problem of universal lists broadcasting that uses genetic algorithm. Our numerical results have demonstrated that our algorithm is able to find optimal or near-optimal broadcast time for several well-known interconnection networks. It also outperforms degree-based heuristics for various networks with clique-like subgraphs, in which the search space of the problem is almost maximized. Finally, as opposed to the significant memory reduction in the universal list model in comparison with the classical model, our result is very close to the state-of-the-art heuristics for the classical broadcast problem for interconnection networks and several instances of synthetic networks.

## REFERENCES


[1] J. Rocher-Gonzalez, J. Escudero-Sahuquillo, P. J. García, and F. J. Quiles, "On the impact of routing algorithms in the effectiveness of queuing schemes in high-performance interconnection networks," in *Proc. IEEE HOTI*, 2017.

[2] J.-K. Lee, T. Hong, and G. Li, "Traffic and overhead analysis of applied pre-filtering ACL firewall on HPC service network," *J. Commun. Netw.*, vol. 23, no. 3, pp. 192–200, 2021.

[3] F. Al Faisal, M. H. Rahman, and Y. Inoguchi, "A new power efficient high performance interconnection network for many-core processors," *J. Parallel Distrib. Comput.*, vol. 101, pp. 92–102, 2017.

[4] Y. Hasson and M. Sipper, "A novel ant algorithm for solving the minimum broadcast time problem," in *Proc. PPSN*, 2004.

[5] E. A. Varvarigos and D. P. Bertsekas, "Dynamic broadcasting in parallel computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 6, no. 2, pp. 120–131, 1995.

[6] M.-S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. Comput.*, vol. 39, no. 1, pp. 10–18, 1990.

[7] V. Karyotis and M. Khouzani, *Malware diffusion models for modern complex networks: Theory and applications*. Morgan Kaufmann, 2016.

[8] R. Ahlswede, H. Haroutunian, and L. H. Khachatrian, "Messy broadcasting in networks," *Commun. Cryptography*, vol. 276, pp. 13–24, 1994.

[9] H. A. Harutyunyan and A. L. Liestman, "Messy broadcasting," *Parallel Process. Lett.*, vol. 8, no. 02, pp. 149–159, 1998.

[10] K. Diks and A. Pelc, "Broadcasting with universal lists," *Netw.*, vol. 27, no. 3, pp. 183–196, 1996.

[11] S. Gholami and H. A. Harutyunyan, "Broadcast graphs with nodes of limited memory," in *Proc. CompleNet*, 2022.

[12] A. Al-Jawad, I.-S. Comşa, P. Shah, O. Gemikonakli, and R. Trestian, "An innovative reinforcement learning-based framework for quality of service provisioning over multimedia-based SDN environments," *IEEE Trans. Broadcast.*, vol. 67, no. 4, pp. 851–867, 2021.

[13] L. H. Binh and T.-V. T. Duong, "Load balancing routing under constraints of quality of transmission in mesh wireless network based on software defined networking," *J. Commun. Netw.*, vol. 23, no. 1, pp. 12–22, 2021.

[14] D. Li, S. Wang, K. Zhu, and S. Xia, "A survey of network update in SDN," *Frontiers Comput. Sci.*, vol. 11, no. 1, pp. 4–12, 2017.

[15] S. Gholami and H. A. Harutyunyan, "Fully-adaptive model for broadcasting with universal lists," in *Proc. SYNASC*, 2022.

[16] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, 2013.

[17] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Netw.*, vol. 18, no. 4, pp. 319–349, 1988.

[18] F. Robledo, P. Rodríguez-Bocca, and P. Romero, "Optimal broadcast strategy in homogeneous point-to-point networks," in *Proc. LOD*, 2020.

[19] M. R. Garey and D. S. Johnson, "Computers and intractability. a guide to the theory of NP-completeness," *J. Symbolic Logic*, vol. 48, no. 2, pp. 498–500, 1983.

[20] P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi, "Information dissemination in trees," *SIAM J. Comput.*, vol. 10, no. 4, pp. 692–701, 1981.

[21] M. J. Dinneen, "The complexity of broadcasting in bounded-degree networks," *arXiv preprint math/9411222*, 1994.

[22] A. Jakoby, R. Reischuk, and C. Schindelhauer, "The complexity of broadcasting in planar and decomposable graphs," *Discrete Appl. Math.*, vol. 83, no. 1–3, pp. 179–206, 1998.

[23] M. Elkin and G. Kortsarz, "A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 672–689, 2005.

[24] P. Scheuermann and G. Wu, "Heuristic algorithms for broadcasting in point-to-point computer networks," *IEEE Comput. Archit. Lett.*, vol. 33, no. 9, pp. 804–811, 1984.

[25] A. de Sousa *et al.*, "Heuristics for the minimum broadcast time," *Electron. Notes Discrete Math.*, vol. 69, pp. 165–172, 2018.

[26] A. M. Farley and S. T. Hedetniemi, "Broadcasting in grid graphs," in *Proc. 9th S.E. Conf. Combinatorics, Graph Theory, and Computing, Utilitas Mathematica*, 1978, pp. 275–288.

[27] A. L. Liestman and J. G. Peters, "Broadcast networks of bounded degree," *SIAM J. Discrete Math.*, vol. 1, no. 4, pp. 531–540, 1988.

[28] J. Hromkovič, C.-D. Jeschke, and B. Monien, "Optimal algorithms for dissemination of information in some interconnection networks," *Algorithmica*, vol. 10, no. 1, pp. 24–40, 1993.

[29] H. A. Harutyunyan and E. Maraachlian, "On broadcasting in unicyclic graphs," *J. Combinatorial Optimization*, vol. 16, no. 3, pp. 307–322, 2008.

[30] S. Gholami, H. A. Harutyunyan, and E. Maraachlian, "Optimal broadcasting in fully connected trees," *J. Interconnection Netw.*, pp. 2150037, 2022.

[31] H. A. Harutyunyan and E. Maraachlian, "Linear algorithm for broadcasting in networks with no intersecting cycles," in *Proc. PDPTA*, 2009.

[32] R. Beier and J. F. Sibeyn, "A powerful heuristic for telephone gossiping," in *Proc. SIROCCO*, 2000.

[33] H. A. Harutyunyan and B. Shao, "An efficient heuristic for broadcasting in networks," *J. Parallel Distrib. Comput.*, vol. 66, no. 1, pp. 68–76, 2006.

[34] H. A. Harutyunyan and W. Wang, "Broadcasting algorithm via shortest paths," in *Proc. IEEE ICPADS*, 2010.

[35] H. A. Harutyunyan and C. Jimborean, "New heuristic for message broadcasting in networks," in *Proc. IEEE AINA*, 2014.

[36] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber, "Multicasting in heterogeneous networks," in *Proc. ACM STOC*, 1998.

[37] G. Kortsarz and D. Peleg, "Approximation algorithms for minimum-time broadcast," *SIAM J. Discrete Math.*, vol. 8, no. 3, pp. 401–427, 1995.

[38] R. Ravi, "Rapid rumor ramification: Approximating the minimum broadcast time," in *Proc. FOCS*, 1994.

[39] P. Bhabak and H. A. Harutyunyan, "Approximation algorithms in graphs with known broadcast time of the base graph," in *Proc. CALDAM*, 2022.

[40] S. Gholami and H. A. Harutyunyan, "A broadcasting heuristic for hypercube of trees," in *Proc. IEEE CCWC*, 2021.

[41] M. Elkin and G. Kortsarz, "Sublogarithmic approximation for telephone multicast," *J. Comput. Syst. Sci.*, vol. 72, no. 4, pp. 648–659, 2006.

[42] P. Fraigniaud and E. Lazard, "Methods and problems of communication in usual networks," *Discrete Appl. Math.*, vol. 53, no. 1, pp. 79–133, 1994.

[43] H. A. Harutyunyan, A. L. Liestman, J. G. Peters, and D. Richards, "Broadcasting and gossiping," in *Handbook of Graph Theory*, 2013, pp. 1477–1494.

[44] J. Hromkovič, R. Klasing, B. Monien, and R. Peine, "Dissemination of information in interconnection networks (broadcasting & gossiping)," in *Combinatorial network theory*, 1996, pp. 125–212.

[45] A. Rosenthal and P. Scheuermann, "Universal rankings for broadcasting in tree networks," in *Proc. Allerton*, 1987.

[46] H. A. Harutyunyan and P. Taslakian, "Orderly broadcasting in a 2D torus," in *Proc. IV*, 2004.

[47] J.-H. Kim and K.-Y. Chwa, "Optimal broadcasting with universal lists based on competitive analysis," *Netw.*, vol. 45, no. 4, pp. 224–231, 2005.

[48] H. A. Harutyunyan, A. L. Liestman, K. Makino, and T. C. Shermer, "Nonadaptive broadcasting in trees," *Netw.*, vol. 57, no. 2, pp. 157–168, 2011.

[49] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.

[50] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proc. GECCO*, 1999.

[51] E. Semenkin and M. Semenkina, "Self-configuring genetic algorithm with modified uniform crossover operator," in *Proc. ICSI*, 2012.

[52] S. J. Louis and G. J. Rawlins, "Designer genetic algorithms: Genetic algorithms in structure design." in *Proc. ICGA*, 1991.

[53] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 367–385, 2003.

[54] Y. Fang and J. Li, "A review of tournament selection in genetic programming," in *Proc. ISICA*, 2010.

[55] D. E. Goldberg, "A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing," *Complex Syst.*, vol. 4, pp. 445–460, 1990.

[56] R. Kumar, "Blending roulette wheel selection & rank selection in genetic algorithms," *Int. J. Mach. Learning Comput.*, vol. 2, no. 4, pp. 365–370, 2012.

[57] H. Ishibuchi and T. Yamamoto, "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," *Fuzzy Sets Syst.*, vol. 141, no. 1, pp. 59–88, 2004.

[58] M. Hutter, "Fitness uniform selection to preserve genetic diversity," in *Proc. CEC*, 2002.

[59] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex syst.*, vol. 9, no. 3, pp. 193–212, 1995.

[60] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, "On stopping criteria for genetic algorithms," in *SBIA*, 2004.

[61] M. Zbigniew, "Genetic algorithms+ data structures= evolution programs," in *Computational Statistics*, 1996, pp. 372–373.

[62] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*, 1991, vol. 1, pp. 69–93.

[63] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Comput.*, vol. 27, no. 6, pp. 17–26, 1994.

[64] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*, 2019, pp. 43–55.

[65] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Applicat.*, vol. 80, no. 5, pp. 8091–8126, 2021.

[66] J. J. Grefenstette, "Genetic algorithms and machine learning," in *Proc. COLT*, 1993.

[67] M. Assi, B. Halawi, and R. A. Haraty, "Genetic algorithm analysis using the graph coloring method for solving the university timetable problem," *Procedia Comput. Sci.*, vol. 126, pp. 899–906, 2018.

[68] R. Marappan and G. Sethumadhavan, "Solution to graph coloring using genetic and tabu search procedures," *Arabian J. Sci. Eng.*, vol. 43, no. 2, pp. 525–542, 2018.

[69] E.-G. Talbi and P. Bessiere, "A parallel genetic algorithm for the graph partitioning problem," in *Proc. ICS*, 1991.

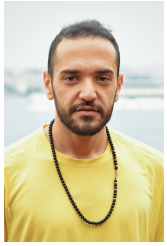
[70] H. Maini, K. Mehrotra, C. Mohan, and S. Ranka, "Genetic algorithms for graph partitioning and incremental graph partitioning," in *Proc. ACM/IEEE Supercomputing*, 1994.

[71] Y.-H. Kim, Y. Yoon, and Z. W. Geem, "A comparison study of harmony search and genetic algorithm for the max-cut problem," *Swarm Evol. Comput.*, vol. 44, pp. 130–135, 2019.

[72] B. Q. Pinto, C. C. Ribeiro, I. Rosseti, and A. Plastino, "A biased random-key genetic algorithm for the maximum quasi-clique problem," *European J. Operational Research*, vol. 271, no. 3, pp. 849–865, 2018.

[73] E. Marchiori, "A simple heuristic based genetic algorithm for the maximum clique problem," in *Proc. ACM SAC*, 1998.

[74] G. Fortez, F. Robledo, P. Romero, and O. Viera, "A fast genetic algorithm for the max cut-clique problem," in *Proc. LOD*, 2020.
[75] C. C. Palmer and A. Kershenbaum, "Representing trees in genetic algorithms," in *Proc. IEEE CEC. IEEE WCCI*, 1994.
[76] A. Lambora, K. Gupta, and K. Chopra, "Genetic algorithm a literature review," in *Proc. IEEE COMITCon*, 2019.
[77] N. Selvanathan and W. J. Tee, "A genetic algorithm solution to solve the shortest path problem in OSPF and MPLS," *Malaysian J. Comput. Sci.*, vol. 16, no. 1, pp. 58–67, 2003.
[78] N. Shanmugasundaram, K. Sushita, S. P. Kumar, and E. Ganesh, "Genetic algorithm-based road network design for optimising the vehicle travel distance," *Int. J. Veh. Inform. Commun. Syst.*, vol. 4, no. 4, pp. 344–354, 2019.
[79] X. Wen *et al.*, "Improved genetic algorithm based 3-D deployment of UAVs," *J. Commun. Netw.*, vol. 24, no. 2, pp. 223–231, 2022.
[80] Y. Jiao and I. Joe, "Energy-efficient resource allocation for heterogeneous cognitive radio network based on two-tier crossover genetic algorithm," *J. Commun. Netw.*, vol. 18, no. 1, pp. 112–122, 2016.
[81] C.-H. Chu, G. Premkumar, and H. Chou, "Digital data networks design using genetic algorithms," *European J. Operational Research*, vol. 127, no. 1, pp. 140–158, 2000.
[82] N. Muruganantham and H. El-Ocla, "Routing using genetic algorithm in a wireless sensor network," *Wireless Pers. Commun.*, vol. 111, no. 4, pp. 2703–2732, 2020.
[83] M. A. Mazaideh and J. Levendovszky, "A multi-hop routing algorithm for WSNs based on compressive sensing and multiple objective genetic algorithm," *J. Commun. Netw.*, vol. 23, no. 2, pp. 138–147, 2021.
[84] B. Sun and L. Li, "A QoS multicast routing optimization algorithm based on genetic algorithm," *J. Commun. Netw.*, vol. 8, no. 1, pp. 116–122, 2006.
[85] D.-g. Zhang, S. Liu, X.-h. Liu, T. Zhang, and Y.-y. Cui, "Novel dynamic source routing protocol (DSR) based on genetic algorithm-bacterial foraging optimization (GA-BFO)," *Int. J. Commun. Syst.*, vol. 31, no. 18, pp. e3824, 2018.
[86] A. Bhardwaj and H. El-Ocla, "Multipath routing protocol using genetic algorithm in mobile ad hoc networks," *IEEE Access*, vol. 8, pp. 177534–177548, 2020.
[87] A. Kumar, P. Dadheech, R. Kumari, and V. Singh, "An enhanced energy efficient routing protocol for VANET using special cross over in genetic algorithm," *J. Stat. Manag. Syst.*, vol. 22, no. 7, pp. 1349–1364, 2019.
[88] G. Zhang, M. Wu, W. Duan, and X. Huang, "Genetic algorithm based QoS perception routing protocol for VANETs," *Wireless Commun. Mobile Comput.*, vol. 2018, 2018.
[89] U. Baroudi, M. Bin-Yahya, M. Alshammari, and U. Yaqoub, "Ticket-based QoS routing optimization using genetic algorithm for WSN applications in smart grid," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 4, pp. 1325–1338, 2019.
[90] J. Bhola, S. Soni, and G. K. Cheema, "Genetic algorithm based optimized LEACH protocol for energy efficient wireless sensor networks," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 3, pp. 1281–1288, 2020.
[91] T. Wang, G. Zhang, X. Yang, and A. Vajdi, "Genetic algorithm for energy-efficient clustering and routing in wireless sensor networks," *J. Syst. Software*, vol. 146, pp. 196–214, 2018.
[92] Y. Wu and W. Liu, "Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks," *IET Wireless Sensor Syst.*, vol. 3, no. 2, pp. 112–118, 2013.
[93] A. Rezaeipanah, H. Nazari, and G. Ahmadi, "A hybrid approach for prolonging lifetime of wireless sensor networks using genetic algorithm and online clustering." *J. Comput. Sci. Eng.*, vol. 13, no. 4, pp. 163–174, 2019.
[94] C. J. Hoelting, D. A. Schoenefeld, and R. L. Wainwright, "A genetic algorithm for the minimum broadcast time problem using a global precedence vector," in *Proc. ACM SAC*, 1996.
[95] A. Lima, A. L. Aquino, B. Nogueira, and R. G. Pinheiro, "A matheuristic approach for the minimum broadcast time problem using a biased random-key genetic algorithm," *Int. Trans. Operational Research*, 2022.
[96] C. D. Morosan, "On the number of broadcast schemes in networks," *Inform. Process. Lett.*, vol. 100, no. 5, pp. 188–193, 2006.
[97] W. Shang, P. Wan, and X. Hu, "Approximation algorithms for minimum broadcast schedule problem in wireless sensor networks," *Frontiers Math. China*, vol. 5, no. 1, pp. 75–87, 2010.
[98] M. F. Hocaoğlu and I. Genç, "Smart combat simulations in terms of industry 4.0," in *Simulation for Industry 4.0*, 2019, pp. 247–273.
[99] M. A. Salkuyeh and B. Abolhassani, "Optimal video packet distribution in multipath routing for urban VANETs," *J. Commun. Netw.*, vol. 20, no. 2, pp. 198–206, 2018.
[100] D. Bucantanschi, B. Hoffmann, K. R. Hutson, and R. M. Kretchmar, "A neighborhood search technique for the freeze tag problem," in *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, 2007, pp. 97–113.
[101] X. Chu and Y. Chen, "Time division inter-satellite link topology generation problem: Modeling and solution," *Int. J. Satellite Commun. Netw.*, vol. 36, no. 2, pp. 194–206, 2018.
[102] Z. Liu, L. Huang, B. Li, and B. Ji, "Anti-aging scheduling in single-server queues: A systematic and comparative study," *J. Commun. Netw.*, vol. 23, no. 2, pp. 91–105, 2021.
[103] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.
[104] B. Kamiński, P. Prałat, and F. Théberge, *Mining Complex Networks*, Chapman and Hall/CRC, 2021.
[105] J.-C. Bermond, "Graceful graphs, radio antennae and french windmills," in *Proc. One day Combinatorics Conference, Research Notes in Mathematics*, 1979.
[106] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. AAAI*, 2015.
[107] J.-C. Bermond and C. Peyrat, "Broascasting in de bruijn networks," *Congressus Numerantium*, vol. 66, pp. 283–292, 1988.
[108] R. Klasing, B. Monien, R. Peine, and E. A. Stöhr, "Broadcasting in butterfly and debruijn networks," *Discrete Appl. Math.*, vol. 53, no. 1–3, pp. 183–197, 1994.
[109] F. Comellas, H. A. Harutyunyan, and A. L. Liestman, "Messy broadcasting in multidimensional directed tori," *J. Interconnection Netw.*, vol. 4, no. 01, pp. 37–51, 2003.
[110] C. Li, T. E. Hart, K. J. Henry, and I. A. Neufeld, "Average-case" messy" broadcasting," *J. Interconnection Netw.*, vol. 9, no. 4, pp. 487–505, 2008.
[111] C. G. Freitas, A. L. Aquino, H. S. Ramos, A. C. Frery, and O. A. Rosso, "A detailed characterization of complex networks using information theory," *Scientific Reports*, vol. 9, no. 1, pp. 1–12, 2019.
[112] D. L. Guidoni, R. A. Mini, and A. A. Loureiro, "On the design of resilient heterogeneous wireless sensor networks based on small world concepts," *Comput. Netw.*, vol. 54, no. 8, pp. 1266–1281, 2010.
[113] R. S. Cabral, A. Aquino, A. Frery, O. Rosso, and J. Ramírez, "Structural changes in data communication in wireless sensor networks," *Open Physics*, vol. 11, no. 12, pp. 1645–1652, 2013.
[114] A. de Sousa *et al.*, "A tree-block decomposition-based heuristic for the minimum broadcast time," *Int. J. Metaheuristics*, vol. 7, no. 4, pp. 379–401, 2020.



**Saber Gholami** received his B.Sc. and M.Sc from K.N. Toosi University of Technology and Tehran Polytechnic University, Iran, respectively. Recently, he completed his Ph.D. at Concordia University under the supervision of Professor Hovhannes Harutyunyan. During his Ph.D., Saber worked on social networks analysis, algorithm design, broadcasting, routing algorithms for networks, and graph theory. His research interests include, but are not limited to, reinforcement learning in graphs, modeling and analysis of large complex and social networks, algorithmic graph theory, message dissemination, broadcasting, and deep learning for graphs.

**Hovhannes Harutyunyan** is a Professor in the Department of Computer Science and Software Engineering at Concordia University, Montreal. He received his Bachelor and Master in Applied Mathematics from Yerevan State University, Armenia, and his Ph.D. in Discrete Mathematics from the Armenian Academy of Sciences. Before Concordia, Dr. Harutyunyan worked at Armenian Academy of Sciences, Simon Fraser University, and Brandon University. He was a researcher at Swiss Federal Institute of Technology (ETH), Zurich, and Bielefeld University, Germany. Dr. Harutyunyan's main research area is in discrete and combinatorial algorithms, with the main focus on message dissemination problems in networks.