A Weighted Artificial Bee Colony Algorithm for Influence Maximization

Riccardo Cantini[a], Fabrizio Marozzo[a], Silvio Mazza[a], Domenico Talia[a], Paolo Trunfio[a]

[a] *University of Calabria, via P. Bucci, Rende (87036), Italy*

Abstract

Social media platforms are increasingly used to convey advertising campaigns for products or services. A key issue is to identify an appropriate set of influencers within a social network, investing resources to get them to adopt a product. Influence maximization is an optimization problem that aims at finding a small set of users that maximize the spread of influence in a social network. In this paper we propose an influence maximization algorithm, named *Weighted Artificial Bee Colony* (WABC), that is based on a bio-inspired technique for identifying a subset of users which maximizes the spread. The proposed algorithm has been applied to a case study that analyzes the propagation of information among Twitter users during the Constitutional Referendum held in Italy in 2016. Our analysis is aimed at identifying the main influencers of the *yes* and *no* factions, and deriving the main information diffusion strategies of each faction during the political campaign. WABC outperformed ranking-proxy techniques based on classical centrality measures, i.e., PageRank, Rank and Degree. Even compared to DIRIE, which exploits a more complex algorithm, WABC was able to find a more accurate set of users which allows to maximize the spread in almost all the considered configurations.

Keywords: Influence maximization, information diffusion, information spread, social network analysis, bio-inspired computing, heuristic algorithms.

1. Introduction

Millions of people every day interact on social media platforms by generating large amounts of data [12], which can be exploited for extracting valuable information in different application contexts, such as information diffusion [1], sentiment [16] and opinion mining [4, 27], news gathering [34] and misinformation blocking [26].

Email addresses: rcantini@dimes.unical.it (Riccardo Cantini), fmarozzo@dimes.unical.it (Fabrizio Marozzo), smazza@dimes.unical.it (Silvio Mazza), talia@dimes.unical.it (Domenico Talia), trunfio@dimes.unical.it (Paolo Trunfio)

A very active research area that seeks to exploit the data available on social media is viral marketing. Viral marketing or viral advertising is a business strategy that uses social media to promote a product or service. An efficient way for performing a good marketing campaign is to identify an appropriate set of influencers among users and invest resources to make them adopt a product/service. This can lead to a cascade process, influencing consumer preferences in a large part of the network [33, 15].

Influence maximization is an optimization problem that aims at finding a small set of users that maximize the spread of influence in a social network [2]. Initially proposed as a stochastic optimization problem in [11], it consists in identifying a set of k users with the greatest overall influence, by analyzing the structure of the network and user interconnections, as well as user-specific features such as demographic properties [32].

Influence maximization is an NP-Hard problem, with two sources of hardness: *i*) the complexity of computing the spread, i.e. the number of influenced users; *ii*) the combinatorial nature of identifying the best solution, that maximizes the influence, among all possible combinations. For this reason, implementing efficient influence maximization algorithms requires the use of heuristic methods and also of parallel computing models. An effective parallel computing paradigm to be used here is the Bulk Synchronous Parallel (BSP) model, that simplifies the implementation of parallel applications by exploiting distributed-memory parallelism. An efficient implementation of BSP is provided by the Apache Hama framework.

This paper describes the functioning and the implementation of an influence maximization algorithm, namely *Weighted Artificial Bee Colony (WABC)*, aimed at identifying a subset of users which maximizes the spread. It is based on a bio-inspired approach based on the Artificial Bee Colony algorithm [18] that has been modified for implementing the influence maximization task [31], by introducing several changes and improvements with respect to previous related work. In particular, the proposed algorithm exploits an effective approach to evaluate the fitness value, which can be considered as the resolution of a reachability problem centered on the paths of maximum probability. We also addressed the influence overlap problem of classical influence ranking-proxy algorithms, avoiding the negative effects caused by influence redundancy during the maximization process. Moreover, the proposed algorithm is less sensitive to parameter tuning in comparison to related work, as it dynamically sets the depth at which to explore the graph, focusing more on the most promising paths. All of these factors contribute in making the model able to produce an accurate estimate of the total spread for the final seed set, which is useful for estimating the number of users who will actually be influenced.

The WABC algorithm has been applied to a case study that analyzes the propagation of information in Twitter during the Constitutional Referendum held in Italy in 2016, for identifying the main influencers of the two factions, i.e. *yes* and *no*, and deriving the main information diffusion strategies of each faction during the political campaign. We experimentally evaluated the accuracy of the WABC algorithm through its implementation in Apache Hama. For analyz-

ing qualitative aspects, we classified the identified influencers according to their profile (journalistic page, political activist, popular or normal user) to better determine the type of political campaign. We carried out several simulations in order to measure their influence strength. For what concerns quantitative analysis, we compared the obtained results with both standard ABC algorithm and other related state-of-art techniques in terms of *computing time*, *evaluated spread* and relative error on the *expected spread*. Specifically, WABC turned out to be more time consuming than its classical version (ABC), but much more accurate in determining the expected spread, with an up to 24% decrease of the relative estimation error. Furthermore, it outperformed ranking-proxy techniques based on classical centrality measures, i.e., PageRank, Rank and Degree, with an up to 40% improvement. Even compared to DIRIE, which is based on the Independent Cascade model and exploits a more complex algorithm, WABC was able to find a more accurate set of users which allows to maximize the spread in almost all the considered configurations. Overall, the obtained results confirm the effectiveness of the proposed approach in identifying the leading influencers of a social network and understanding the main information diffusion strategies.

The remainder of the paper is organized as follows. Section 2 describes the main information diffusion models used in literature. Section 3 discusses influence maximization related work. Section 4 describes the proposed algorithm. Section 5 presents the experimental evaluation on a case study, and Section 6 concludes the paper.

2. Information diffusion models

Interactions among users of a social network can be represented as a directed graph $G = (V, E)$, where V is the set of users in the network and E represents the relationship among them as edges directed from one vertex to another. The influence exercises by a user on the other members of the network is modeled as a function $p : E \to [0, 1]$ that associates a weight to each relationship $(u, v) \in E$. Given a user node $u \in V$, we define with $N^{in}(u)$ and $N^{out}(u)$ the sets of users $v \in V$ for which there exists a relationship $(v, u) \in E$ and $(u, v) \in E$ respectively.

In a diffusion model, nodes can be partitioned according to their current state: *influenced* (i.e., nodes that have been activated during the diffusion process), *active* (i.e., nodes that can propagate influence and activate others), and *idle* (i.e., nodes that have not yet been activated). The diffusion process starts from a small set of active nodes, called seed set $S \subseteq V$. Therefore, each node of the seed set can iteratively influence its out-neighbors and the process generally stops when there are no new active users. Diffusion models can be divided in two classes: i) *progressive* models, which does not allow a user to become idle once activated; ii) *non progressive*, in which deactivation is allowed at any time.

The most used diffusion models in literature are progressive, since the growth of the active set is monotonic, which ensures the termination of the propagation process in a finite number of steps when the number of users is finite. In the following we outline the two most used models, *Independent Cascade* (IC) and *Linear Threshold* (LT).

Independent Cascade. The independent Cascade model (IC) originally described by Kempe et al. [19], is characterized by the independence of activation among nodes. Given the input network graph $G = (V, E)$ an initial set of active nodes, i.e. the seed set S, is chosen. Therefore, the IC model generates the active sets A_t for each step $t \geq 1$ following a randomized diffusion dynamics, with $A_0 = S$. Specifically at the step t, for each inactive node $v \notin A_{t-1}$, each node $u \in N^{in}(v)$ activated at the previous step attempts to activate v through a Bernoulli trial with probability of success equal to $p(u, v)$. If the test is successful, the node v is added to the active set of the current iteration.

Linear Threshold Model. Similar to IC, the Linear Threshold model (LT) [19], takes the network graph $G = (V, E)$ and the initial seed set S_0 as input. The probability on the in-edges is normalized so that $\sum_{u \in N^{in}(v)} p(u, v) \leq 1, \forall v \in V$. Therefore, LT generates the active sets A_t for each step $t \geq 1$, according to the following mechanism. Initially each node $v \in V$ independently selects a threshold θ_v by sampling a uniform distribution in the interval [0.1]. At the step t, for each inactive node $v \in V$, if the sum of the weights of the active in-neighbors reaches the threshold θ_v, i.e. $\sum_{u \in N^{in}(v) \cap A_{t-1}} p(u, v) \geq \theta_v$, then v is activated and is included in the active set A_t. Intuitively, threshold θ_v models the likelihood with which v is influenced by its active neighborhood: a high threshold value represents a greater resistance to the influence in the propagation process. The random choice of the threshold reflects the lack of information on users' tendency to be influenced and is the only source of non-determinism in the model.

The aforementioned propagation models need techniques for establishing the influence probability and therefore the weights to be assigned to the edges of the network. A widely used practice is weighted cascade (WC), where the probability of influence $p(u, v)$ is defined as $\frac{1}{|N^{in}(v)|}$, where $|N^{in}(v)|$ is the in-degree of v. The main idea behind this weighting scheme is that an important node (i.e., a public figure) is more likely to influence a user that tends to express interest only for its contents, assuming that edges are oriented according to a relationship of interest. Recent studies proposed the estimation of the influence probabilities starting from logs. The first proposal was formalized by [30] who represented the acquisition of influence weights from existing logs as a likelihood maximization problem, exploiting the Expectation Maximization algorithm.

2.1. Spread function properties

Independent Cascade and Linear Threshold are progressive models that share two important properties in terms of influence spread, that is the function σ, which estimates the expected number of users who will be active at the end of the information diffusion process. For both models the spread function is:

1. *monotonic*, the inclusion of a new node v in the active set S can not lead to a decrement of the spread function: $\sigma(S \cup v) \geq \sigma(S), \forall v \in V, S \subseteq V$.
2. *submodular*, the marginal gain obtained by adding a new node v to a set S is at least equal to the marginal gain obtained by adding the same element to a superset T of S: $\sigma(S \cup v) - \sigma(S) \geq \sigma(T \cup v) - \sigma(T),\ \forall v \in V, S \subseteq T$.

3. Related work

The problem of identifying a set of k elements that maximizes the spread σ is an NP-Hard optimization problem. However, thanks to the properties of monotonicity and submodularity of σ, a greedy hill-climbing procedure, which selects at each iteration the most promising node in terms of influence spread, provides a pseudo-optimal solution S^*, achieving a $(1 - \frac{1}{e})$ approximation ratio.

Despite the theoretical bound provided by the greedy algorithm, the influence maximization task remains hard to solve. In fact, besides the complexity related to the maximization of the spread σ, which derives from the combinatorial nature of the problem, another crucial point is the calculation of σ with respect to the addition of a node v in the active set, which is a #P-hard counting problem. For this reason different resolution techniques have been developed. They can be grouped into three main categories [25], according to the approach used in the evaluation of the spread function: simulation-, proxy- and sketch-based. Another interesting class related to *context-aware* approaches is described in the following.

Simulation-based. The key idea of this approach is to perform a series of Monte Carlo simulations for evaluating the spread function for a given seed set. Considering the IC model, given a graph G, this approach consists in considering an initial seed set S and removing the edges with probability $1 - p(u, v)$. This way a set of instances can be generated and the spread can be estimated on these sampled instances. The advantage of such models is their generality, as this process can be applied to any information propagation model, and also the bound provided by the greedy algorithm is preserved. However, the main problem here is the computational efficiency related to the large number of simulations needed to obtain a good estimate. Kempe et al. [19] extended the greedy algorithm using simulations for evaluating the marginal gain for a given node added to the active set. In particular, a seed set S is built by considering the most promising nodes with respect to their marginal gain on the spread function, estimated after r simulations, as the average cardinality of the active set. The number of simulations is a crucial parameter in such a mechanism, which affects computational complexity. For this reason several methods have been proposed aimed at reducing r. The CELF technique [24] aims to estimate an upper bound of the marginal gain determined by adding a node to the current seed set. This avoids the evaluation of some nodes whose influence is considered insignificant, thus exploiting the submodularity of the spread function and a power law assumption on the degree distribution of the network graph. Another technique used to reduce the complexity of this kind of approach is the Community-based greedy algorithm (CGA) [6]. This technique is based on the divide-and-conquer paradigm for reducing the complexity of the Monte Carlo simulations by partitioning the graph according to a community structure and evaluating the spread only within communities.

Proxy-based. The main idea behind this class of algorithms is to define proxy models such as PageRank or shortest path, for approximating the spread function σ. Therefore, its main advantage is the reduced complexity of the proxy

model, but there are no guarantees of optimality. Proxy based algorithms can be divided into *influence ranking proxy* and *diffusion model reduction proxy*.

i) Influence Ranking Proxy are models that provide a rank to each user in the graph G in order to estimate a metric for their influence rate and subsequently generate the seed set directly from that ranking. There are different approaches based on ranking that can be directly derived from the graph, such as degree, PageRank, and other centrality measures. However, these techniques are usually not very suitable to solve the problem of influence maximization, as the ranking defined on users often does not take into account any overlap of influence; two users with a high ranking could influence an almost identical set of users, providing an incorrect solution to the problem. To deal with this issue, the DegreeDiscount technique [9] has been proposed, which introduces a penalty on σ for a given node v proportional to the overlap of influence with the other nodes in the active set. The IRIE [17] algorithm is based on the Independent Cascade model and it includes two steps: *i*) the *influence ranking* of each node is computed and the node with maximum ranking is determined; *ii*) for each node the $p^S_{act}(u)$ contribution is evaluated, that is the probability that u is active following the diffusion process started from the seed set S.

ii) Diffusion model reduction proxy tries to reduce the complexity in computing the spread σ following two main approaches: i) reducing the stochastic propagation model in a deterministic one; ii) estimating the influence form a local subgraph. The Shortest-Path Model (SPM) [22] considers the shortest path between two nodes in the activation process. In the MIA/PMIA [10] algorithm, for each pair of nodes (u, v), u can influence v only along the maximum influence path (MIP), defined on a tree where unpromising paths are pruned using a threshold. The IPA algorithm [20] is similar to MIA/PMIA, but can evaluate multiple paths besides the MIP. The LDAG algorithm [10] restricts the influence to acyclic graphs, by building a DAG for a node v, exploiting the Dijkstra algorithm for the shortest-path length. Goyal et al. [14] proposed the Sim-Path algorithm, where the influence of a set of nodes, propagated through the LT model, is calculated by enumerating all the simple paths starting from each node within the set. However, as this is a #P-Hard problem, the SimPath limits this enumeration to a restricted neighborhood, cutting those paths with a probability lower than a threshold. Lee et al. [23] proposed a fast greedy approximation algorithm for influence maximization exploiting the concept of *2-hop influence spread*. It is based on the interesting observation that an item is generally diffused from a seed within a very small number of hops in online social network services. Specifically, even if we consider only users who are within 2-hops away from seeds, the estimated influence spread is experimentally expected to be at least 81% of the exact influence spread.

Sketch-based. The goal of sketch based models is to preserve the theoretical bounds provided by simulation-based methods, while providing computational efficiency. In order to avoid the repetition of several Monte Carlo simulations, these techniques calculate different sketches based on a specific diffusion model and evaluate the spread function σ by exploring them. Depending on how the sketches are generated, this class of algorithms can be divided into

two main categories: *Forward Influence Sketch (FI-Sketch)* and *Reverse Reachable Sketch (RR-Sketch)*. The main idea behind forward influence sketches is to build sketches extracting the subgraph induced by an instance of the diffusion process with respect to the considered propagation model, then estimate the spread of a seed set using that subgraph. One of the most famous algorithms is NewGreIC [9], which extends the Independent Cascade model building a given number of sketches by sampling the graph G for evaluating the marginal gain of each node. Borges et al. [5] discovered that it is not necessary to estimate the influence using sketches generated starting from the entire graph. They developed the reverse reachable sketch approach, a technique in which the influence of each seed set S is estimated by selecting a random subset of nodes and analyzing which of these can be reached. By creating multiple random RRs on different nodes, if a node u has a great impact on the other nodes, then it will have a high probability of appearing within these RR sets. Similarly, if a seed set S covers a maximum number of RR sets, it is likely to be the optimal seed set. Borges et al. [5] proposed the RIS algorithm, which generates random RRs until the total number of edges examined during the generation process does not reach a threshold.

Context-aware. The common factor which characterizes all the aforementioned algorithms, categorized in taxonomy [25], is that the influence propagation process is often modeled in an unrealistic way, never referring to a specific context. For this reason, other influence maximization algorithms have been proposed in the literature, for dealing with several tasks in specific contexts. In topic aware influence maximization (TAIM) the IM problem is extended considering what are the topics to be propagated. TAIM introduces the topics to exploit the interests of users interacting in the social networks while computing the spread. TAIM models are TIC (topic aware Independent Cascade) and TLT (topic aware Linear Threshold) [3]. The standard IM algorithm does not take the time dimension into account. Such an assumption could be unreasonable in some cases, time-aware diffusion models introduce the concept of step as a temporal measure, and restrict the process of diffusion within these steps. Chen et al. [8] proposed IC-M the model, where for each edge between the nodes u and v, a meeting probability $p(u, v)$ is defined, and the IM problem consists in identifying the optimal seed set capable of activating the greatest number of nodes in at the most τ steps. Kim et al. [21] proposed the CT-IC model, based on the concept of continuous time, by defining an activation delay of the nodes and a delay distribution.

Data-driven. The main goal of data-driven approaches is to exploit propagation traces available in historical data for learning how influence flows in the network, and thus estimating the expected spread of influence. As an example, Goyal et al. [13] proposed a data-driven approach based on the credit distribution model, which can learn different levels of influenceability of users, also taking into account the temporal aspects. Data-driven approaches are more flexible and able to adapt to different networks and application scenarios, compared to those models that randomly assign influence probabilities by generating large errors in spread prediction. However, they require more computational resources

and a large number of propagation traces representative of user interactions.

Comparison. The *Weighted Artificial Bee Colony (WABC)* proposed in this paper effectively exploits a bio-inspired approach to deal with the influence maximization task. It can be classified as an influence ranking-proxy algorithm and is characterized by several changes and improvements with respect to previous related work. Primarily, the proposed algorithm exploits a more effective way for evaluating fitness, which can be considered as the resolution of a reachability problem, where the maximum probability path is considered among all possible ones connecting two distinct nodes. This feature leads to two main benefits:

- The total spread can be accurately estimated. It is a crucial result for an influence maximization task, as it measures the expected number of influenced users, without providing incorrect assessments.
- The influence overlap problem is addressed. This is a common issue of classical influence ranking-proxy algorithms, which can lead to negative effects caused by influence redundancy during the maximization process.

Furthermore, the proposed algorithm is less sensitive to parameter tuning in comparison to related approaches, as it does not use a fixed a-priori depth at which to explore the graph like in [18]. In particular, WABC exploits a threshold on the influence probability, dynamically focusing more on the most promising paths. Moreover, for what concerns the application domain, it was combined with a polarization analysis and a user classification process, in order to identify also the main information diffusion strategies in a scenario characterized by multiple opposing factions. Also the evaluation tests carried out confirmed the effectiveness of the proposed algorithm with respect to the main techniques present in literature (see Section 5). WABC was able to find a more suitable set of users for maximizing the spread in almost all the considered configurations.

4. Proposed algorithm

In recent years, nature has been a great source of inspiration for the development of different algorithms aimed at solving many real world optimization problems [28]. These bio-inspired techniques are related to Swarm Intelligence (SI), a particular field of Artificial Intelligence (AI) based on observing the behavior of social animals such as ants and bees. Swarm Intelligence can be defined as the collective behavior of decentralized and self-organized systems, in which the interaction among components causes the emergence of a complex behavior.

4.1. Artificial Bee Colony (ABC)

Among the various swarm intelligence algorithms, Artificial Bee Colony (ABC) results to be one of the most studied and one of the most applied to real world problems. It is a meta-heuristic algorithm, introduced in 2005 by Derviş Karaboğa [18] and applied to the influence maximization task by Sankar

et al. [31], inspired by the food supply model of bee colonies. It consists of three main components: food sources, employed bees and unemployed bees. In the colony system, the quality of a *food resource* depends on several factors, like the distance from the hive, the amount of food and the ease of extraction. Each resource is assigned to a bee, whose task is to store the information related to that resource. The main behavior of such a model is the search of a source rich in nectar and the abandonment of a poor source. *Employed bees* collect nectar from a flower and bring it to the hive, carrying details about the source of food and sharing this information with other bees. *Unemployed bees* are those bees that are not currently picking up nectar from any flower and can be divided into two types: *Scout bees* whose job is to search new nearby sources of food; *Onlooker bees* which wait for choosing a food source based on information brought to the hive by employed bees. The most interesting aspect of this behavior is the exchange of information in the swarm, which takes place within the hive through a particular technique called *waggle dance* [29]. It consists in a physical movement of bees, whose duration is proportional to the goodness of the food source. Therefore the details relating to all the identified sources are communicated through this dance to onlooker bees which in this way can select the most promising source of food.

The ABC algorithm can be adapted to explore a social network for identifying a subset of nodes with maximum influence, based on the waggle dance mechanism. Each node of the social network is considered as a source of food. The employed bees, used to identify the opinion leaders of the network, are initially assigned on the basis of a ranking vector. Scout bees are used for exploring the neighborhood of employed bees for obtaining better solutions, while onlooker bees are used to indicate influenced nodes. For reader's convenience, Table 1 reports the meaning of the main symbols used throughout the sections.

Symbol	*Meaning*
V	Set of graph nodes
E	Set of graph edges
S	Seed set
A_t	Set of active nodes at time t
EB	Set of employed bees eb
SB	Set of scout bees sb
fit_x	Fitness value of the node x
$F(t)$	Global fitness value at time t
$p(x,u)$	Probability of the path between x and u
$\sigma(S)$	*Expected spread*, i.e. an estimate defined by the algorithm starting from the seed set S
$\tilde{\sigma}(S)$	*Evaluated spread*, i.e. an estimate defined via simulation starting from the seed set S
ω	Convergence distance
θ	Cutting threshold
$\mathcal{N}_x$	Set of all neighbors of the node x
$\mathcal{N}_x^{in}$	Set of in-neighbors of the node x
$\mathcal{N}_x^{out}$	Set of out-neighbors of the node x
$\mathcal{N}_x^d$	Set of distinct nodes reachable from x in d steps

Table 1: Table of symbols

Algorithm 1 shows the pseudo-code of the ABC algorithm. The input is composed of: a graph $G = (V, E)$, a ranking vector R, with $|R| = |V|$, and an integer k representing the seed set size. The output consists of: i) a set of nodes S with $S \subset V$ and $|S| = k$, which maximizes the spread (i.e., the number of influenced users); ii) the expected spread $\sigma(S)$.

ALGORITHM 1: Artificial Bee Colony (ABC)

Input : Graph $G = (V, E)$, a ranking vector R, an integer k
Output: Seed set S, Expected spread $\sigma(S)$

1 $EB \leftarrow$ *top-k nodes ordered by ranking*
2 $SB \leftarrow \emptyset$
3 **for** $eb \in EB$ **do**
4 $\quad SB \leftarrow SB \cup \mathcal{N}_{eb}^{out}$
5 $Fit_{EB} \leftarrow \emptyset$
6 **for** $eb \in EB$ **do**
7 $\quad fit_{eb} \leftarrow evalFitness(\{eb\}, G)$
8 $\quad Fit_{EB} \leftarrow Fit_{EB} \cup fit_{eb}$
9 `/* Local optimum search */`
10 **while** *not convergence reached* **do**
11 $\quad SB \leftarrow orderByRanking(SB)$
12 $\quad$ **for** $sb \in SB$ **do**
13 $\quad\quad fit_{sb} \leftarrow evalFitness(\{sb\} \cup EB, G)$
14 $\quad\quad$ **if** $\exists\ eb \mid fit_{sb} \geq fit_{eb}$ **then**
15 $\quad\quad\quad EB \leftarrow EB \setminus \{argmin_{eb} Fit_{EB}\} \cup \{sb\}$
16 $\quad SB \leftarrow \emptyset$
17 $\quad$ **for** $eb \in EB$ **do**
18 $\quad\quad SB \leftarrow SB \cup \mathcal{N}_{eb}^{out}$
19 `/* Estimate global optimum */`
20 $S \leftarrow EB$
21 **for** $s \in S$ **do**
22 $\quad \sigma(S) \leftarrow \sigma(S) + fit_s$
23 **return** $S, \sigma(S)$

The algorithm starts by defining two sets:

- The set of employed bees $EB \subset V$ is initialized with the best k nodes of the input ranking vector R, identifying the initial seed set (line 1).
- The set of scout bees $SB \subset V$ is initialized with an empty set and filled by joining the out-neighborhood $\mathcal{N}_{eb}^{out}$ of each employed bee (lines 2-4).

Then the vector Fit_{EB} is obtained evaluating the fitness function, for each employed bee in EB (lines 5-8), whose goal is to iteratively determine local optima during the diffusion process. To that end, the algorithm starts an iterative phase (lines 10-18), performing at each iteration the following operations:

- The set of scout bees SB is ordered by ranking value (line 11).

- For each scout bee in descending order of ranking the fitness fit_{sb} is evaluated (lines 12-13).
- If fit_{sb} exceeds the fitness value of one of the scout bees then the roles are exchanged (lines 14-15).

This phase is repeated until the evaluation of the whole set SB. The set of scout bees is therefore repopulated with the out-neighborhood of the new employed bees (line 16-18) and the process iterates until a termination criterion is reached. Once this criterion is reached the final seed set S is filled with the employed bees eb in EB (line 20). The expected spread $\sigma(S)$ is evaluated by summing up the fitness fit_s of each seed $s \in S$ (rows 21-22). Finally the algorithm returns the final seed set S and the expected spread $\sigma(S)$ (line 23).

4.2. Weighted Artificial Bee Colony (WABC)

Weighted Artificial Bee Colony (WABC) is the extension of the classical ABC algorithm [31] we designed. The main advantages are related to how the fitness function is calculated.

ALGORITHM 2: ABC fitness evaluation

Input : Graph $G = (V, E)$, a distance d, a set of nodes X
Output: Fitness value fit_X

1 $covered \leftarrow \emptyset$
2 /* For each $x \in X$ store each node $u \in V$ reachable by x in d steps, i.e. the d_{out}-neighborhood of x ($\mathcal{N}_x^d$) */
3 **for** $x \in X$ **do**
4 $covered_x \leftarrow \mathcal{N}_x^d \subset X$
5 /* Each node evaluate its fitness as the number of unique nodes that compose its d_{out}-neighborhood $\mathcal{N}_x^d$ */
6 **for** $x \in X$ **do**
7 $fit_x \leftarrow 0$
8 **for** $u \in covered_x$ **do**
9 **if** $!\exists\ z \in X | u \in covered_z$ **then**
10 $fit_x \leftarrow fit_x + 1$
11 $fit_X \leftarrow \sum_{x \in X} fit_x$
12 **return** fit_X

In ABC the fitness evaluation (see Algorithm 2) is based on the difference between sets. The input is composed of the graph $G = (V, E)$, a distance d and the set of nodes X with respect to which fitness is evaluated. For each node $x \in X$, the $covered_x$ set is filled with each node $u \in V$ reachable by x in d steps, i.e. the d_{out}-neighborhood of x ($\mathcal{N}_x^d$), where d is generally equals to one (lines 3-4). Therefore, each node $x \in X$ evaluates its fitness (fit_x) as the difference between the $covered_x$ and $covered_z$ for each node $z \in X$ with $z \neq x$, i.e. the number of nodes that only x can reach in d steps (lines 6-10). Finally, the global fitness value fit_X is obtained by summing up $fit_x \forall x$ and returned (lines 11-12).

ALGORITHM 3: WABC fitness evaluation

Input : Graph $G = (V, E)$, a threshold θ, a set of nodes X
Output: Fitness value fit_X

1 $covered \leftarrow \emptyset$
2 /* Find the best activation path for each pair employed bee $x \in X$, node $u \in V$ */
3 **for** $x \in X$ **do**
4 **for** $u \in V$ **do**
5 $p(x,u) \leftarrow max_{\mathcal{P} \in Paths(x,u)} \prod_{(i,j) \in \mathcal{P}} p(i,j)$
6 $covered_x \leftarrow covered_x \cup \langle u, p(x,u) \mid p(x,u) \geq \theta \rangle$
7 /* The employed bee with the highest influence probability increases its fitness */
8 **for** $x \in X$ **do**
9 $fit_x \leftarrow 0$
10 **for** $\langle u, p(x,u) \rangle \in covered_x$ **do**
11 **if** $!(\exists \langle u, p(z,u)\ z, u \in X \rangle \in covered_z \mid p(z,u) > p(x,u))$ **then**
12 $fit_x \leftarrow fit_x + p(x,u)$
13 $fit_X \leftarrow \sum_{x \in X} fit_x$
14 **return** fit_X

Differently, in WABC (see Algorithm 3), each node evaluates the weighted sum with respect to the unique nodes it can activate. Similarly the input is composed of: the graph $G = (V, E)$, a threshold θ and the set of nodes X. For each employed node $x \in X$, the $covered_x$ set is filled with the pair $\langle u, p(x,u) \rangle$ for each node u covered by x, where $p(x,u)$ represents the maximum influence probability of x on u. The evaluation of the fitness can be considered as the resolution of a reachability problem where the maximum probability path is considered among all the paths $\mathcal{P} \in Paths(u,v)$, connecting x and u. However, we must note that considering all the possible influence paths between an employed bee and the other nodes is often computationally infeasible. For this reason, the algorithm takes as input positive threshold $\in \Re, \theta \geq 0$, which defines the minimum probability of influence, i.e., a cutting value for those paths having a negligible activation probability. Therefore, given this threshold, each employed x determines the set of nodes reachable along a path of total probability at least equal to θ, where the probability of a path $\mathcal{P}$ from x to u, i.e. $p(x,u)$, is given by the product of the weights associated to each edge $(i,j) \in \mathcal{P}$ (line 3-6).

At this point, the fitness value fit_x for each $x \in X$ is computed. Specifically, for each pair $\langle u, p(x,u) \rangle \in covered_x$, fit_x is incremented of $p(x,u)$ if x has the highest influence on the node u, i.e. there not exists another node z such that $p(z,u) > p(x,u)$. So, when two nodes reach the same target, only the most influential will increase its fitness by a value of $p(x,u)$ (lines 8-12). Finally, the global fitness value fit_X is obtained by summing up all the obtained fit_x and returned (lines 13-14).

The described fitness function introduces two main improvements with respect to the classical ABC approach. Firstly, the weighted activation, with a strength equal to the influence probability, leads to a more accurate estimate of the final spread, compared to the classical binary activation used in ABC. Moreover, the algorithm dynamically focuses more on the most promising paths, leading to a more effective and efficient exploration of the social graph compared to ABC, which considers the neighborhood of each employer node within a fixed small number of hops, often equals to one.

4.3. Implementation details

In the following we show some implementation details of the proposed algorithm, implemented by using the Vertex-Centric model provided by Apache Hama. The system requires as input a file representing the graph, in which the main characteristics of each node are provided, i.e., *id*, *ranking*, and a list of *outgoing edges*. For each edge is also defined the ranking of the target vertex. Further parameters to be specified are: the threshold θ, a parameter ω which controls convergence and the size of the seed set k used for identifying the initial seed vector.

During the setup phase, nodes initialize their data structures and one of them is elected as master, taking on the role of coordinator. The vertices have a simple behavior: they receive a message and check the corresponding flag for determining the behavior to be adopted.

During the first phase each seed vertex provides the master with information about its neighborhood, specifying the ranking for each neighbor. Once notified by the master, a seed vertex evaluates its reachability set, sending along its outgoing edges a new message characterized by the flag *influence* and by a value, which represents the probability of activation. Each vertex who receives this message proceeds by storing the information, forwarding the message if it has not already received a greater influence value for that node. Once the phase of propagation of the values is ended, i.e., when there are no more messages within the network, each non-seed node $v \in V \backslash S$ selects a seed node $s \in S$ to belong to, that is the seed with maximum probability of influence on v. Therefore, these values are conveyed to an aggregator which evaluates the fitness of the different seeds.

Once the first iterative phase is over, the master node elects the scout bees with the highest ranking and notifies the beginning of the influence evaluation procedure. At this point, the scout bees send an influence message along the outgoing edges in order to evaluate their fitness. Once the aggregator has completed the evaluation, the master node determines whether it is necessary to proceed with the role switch or not, communicating it to the other nodes.

The process iterates until the entire set of scout bees is evaluated, then the stop condition is evaluated. The convergence of the algorithm is controlled by ω, which specifies the minimum increment in percentage of the spread between two subsequent iterations. Let $F(t)$ be the global fitness value, evaluated as the sum of the fitness of each employed bee, at the current iteration t, and $F(t-1)$ the global fitness value at the previous step: the algorithm stops if $\frac{F(t)-F(t-1)}{F(t-1)} < \omega$.

5. Experimental evaluation

In this section, we evaluated the performances of the proposed algorithm implemented by the Apache Hama framework and applied to the influence maximization task. Experiments have been designed for answering the following research questions: *i*) what are the main advantages of the WABC algorithm with respect to its original version (ABC)? *ii*) how does WABC perform compared to the other state-of-art ranking-proxy approaches?

5.1. Case study

We applied the methodology described in the previous section to the constitutional referendum that was held in Italy on 4th December 2016. This is a real-world case study that involves significant data and complex influencing behaviors. The Italian voters were asked whether they approve a constitutional law that amends the Italian Constitution to reform the composition and powers of the Parliament of Italy, as well as the division of powers between the State, regions, and administrative entities[1]. The main supporter of the referendum (i.e., in favor of *yes*) was the Democratic Party (in Italian Partito Democratico or PD) and its leader, also Italian prime minister Matteo Renzi. On the other side, in favor of *no* were the main opposition parties (e.g., Movimento 5 Stelle, Forza Italia) and several citizen committees. The referendum saw a high voter turnout (approximately 65% of voters) and a majority of the votes opposed to the reform (i.e., voting *no*), which exceeded 59% of the expressed preferences. A political effect of the referendum's result was the resignation of the Italian prime minister.

The political event under analysis $\mathcal{P}$ is a two-faction event $F = \{yes, no\}$. In order to investigate the information diffusion processes involved in the political campaign of both factions, also identifying the main influencers, we build two polarity-based subgraphs, G_{yes} and G_{no}. The subgraphs generation process is based on: *i*) the identification of retweet relationships among users and *ii*) the binary classification of tweets based on a set of faction keywords.

As a first step, we collected the main keywords $\mathcal{K}$ used as hashtags in tweets related to $\mathcal{P}$. Such keywords have been grouped as follows:

- $\mathcal{K}_{neutral}$ = {#referendumcostituzionale, #siono, #riformacostituzionale, #referendum, #4dicembre, #referendum4dicembre}
- $\mathcal{K}_{yes}$ = {#bastaunsi, #iovotosi, #leragionidelsi, #italiachedicesi, #iodicosi, }
- $\mathcal{K}_{no}$ = {#iovotono, #iodicono, #bastaunno, #famiglieperilno, #leragionidelno}

Given the keywords $\mathcal{K} \coloneqq \mathcal{K}_{neutral} \cup \mathcal{K}_{yes} \cup \mathcal{K}_{no}$, we collected 338,592 tweets containing at least one of these keywords posted from 23rd October (5 weeks before the voting day) to 3rd December 2016 (one day before).

[1] http://www.interno.gov.it/it/italiani-voto-referendum-costituzionale

Collected tweets were pre-processed as described in [27]. Afterwards tweets are classified as follows: if a tweet t contains only keywords that are in favor of a specific faction $f \in F$, then t is classified as in favor of f; otherwise, t is classified as neutral. For instance, Table 2 shows some examples of tweets we collected with their classification (translated in English for the Reader's convenience).

Text	*Keywords*	*Class*
Why is important to be well informed on #ReferendumCostituzionale	#referendum costituzionale	neutral
#IoVotoNO: all the reasons to vote against this reform	#iovotono	no
For a stronger Italy in Europe! #iovotosi #referendum #democrazia	#iovotosi #referendum #democrazia	yes

Table 2: Examples of tweets on the Italian constitutional referendum.

The first tweet expresses the importance of going to vote by using a *neutral* hashtag (#ReferendumCostituzionale) and so it is classified as *neutral*. The second tweet shows the dissatisfaction of a user with the reform by using a hashtag supporting *no* (#iovotono) and is classified as in favor of *no*. The third tweet contains a hashtag supporting *yes* (#iovotosi), a *neutral* hashtag (#referendum), and a co-hashtag (#democrazia) and for this reason is classified as in favor of *yes*.

Starting from the set of classified tweets, we generated two subgraphs G_{yes} and G_{no}, relating to the users who supported the *yes* and *no* faction respectively. For example, the graph $G_{yes} = (V, E)$ is obtained as follows:

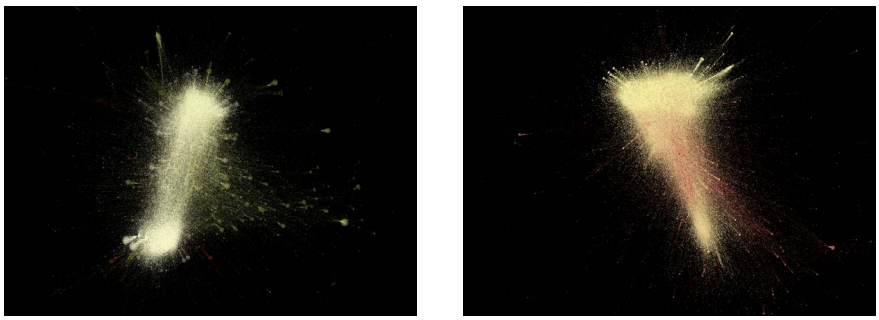
- The set of nodes V is represented by users who have at least published a tweet or a retweet classified in favor of *yes*.
- The set of direct edges E is determined using the *retweet* relationship. In particular, there is an edge (u_1, u_2) from user u_1 to user u_2 if there exists at least one tweet classified in favor of *yes* published by u_1 and retweeted by u_2.
- The influence weight is established by using a weighted cascade criterion. In particular, the probability of influence $p(u, v) \in \mathcal{R}$ associated to the edge (u, v) is defined as $\frac{1}{|\mathcal{N}_v^{in}|}$, where $|\mathcal{N}_v^{in}|$ is the in-degree of v, i.e., the number of retweets published by user v.

The generation process for the G_{no} graph is analogous. Once derived the graphs G_{yes} and G_{no}, the leading influencers and the main information diffusion strategies of the *yes* and *no* factions can be identified.

5.2. Graph properties

The *yes* graph G_{yes} has $117,000$ nodes and $270,000$ edges, with a density of $1.95 \cdot 10^{-5}$; the *no* graph G_{no} has $130,000$ nodes and $440,000$ edges, with

a density of $2.72 \cdot 10^{-5}$. The low density values are related to an accentuated sparsity of the graphs, induced by the presence of many isolated nodes (i.e., users publishing tweets that have not caught the attention of any user). For this reason, the analysis we carried out is focused on the *Giant Component* (GC), i.e. the largest connected component of the two graphs, represented in Figure 1.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

Figure 1: Representation of the two graphs and their giant components.

Table 3 shows the main properties of the *yes* and *no* GC subgraphs, while Table 4 shows the top-5 most influential nodes computed through the PageRank scoring strategy. The underlying assumption of this algorithm is that more important users are likely to receive more links from the others.

Feature	G_{yes}	G_{no}
Num. of nodes	72,225	78,899
Num. of edges	269,218	437,608
Diameter	18	18
Average in-degree	3.98	5.67
Clustering coefficient	0.05	0.06
Average path length	5.92	5.28

Table 3: Giant components properties of the two graphs

Pos.	G_{yes}	G_{no}
1	bastaunsi	Mov5Stelle
2	matteorenzi	matteosalvimini
3	davidefaraone	marionecomix
4	fanpage	beppe_grillo
5	repubblicait	bastaunsi

Table 4: Top-5 most influential nodes calculated using PageRank

5.3. Parameter sensitive analysis

We investigated the effect of different parameters on performance. The WABC algorithm requires as input the network graph $G = (V, E)$, an integer k which represents the seed set cardinality, a threshold θ, a real number ω which control convergence and a diffusion model. In our case study, we used the following parameters: $\omega = 2 \cdot 10^{-2}$, $k = 10$ and Weighted Cascade as a diffusion model. As with the hyperparameter optimization of many algorithms, we set the threshold θ by performing various experiments, varying its value within a given range. This process, based on grid search, requires the optimization of a score function, $f(\theta)$, derived as follows. We analyzed the behavior of the algorithm varying θ with respect to the expected spread given by the algorithm ($\sigma_\theta(S)$), an estimate of the real spread achieved through 20 000 simulations ($\tilde{\sigma}_\theta(S)$), and the overall execution time (T_θ). These different metrics, are jointly modeled in the following score function:

$$f(\theta) = Err_{rel}(\theta) + Time_{rel}(\theta) + Cov_{rel}(\theta)$$

where $\Theta = \{\theta_1, ..., \theta_n\}$ is the set of all considered thresholds. The above formula takes three factors into account:

- The relative error of the expected spread with respect to its estimate achieved by simulating the diffusion process starting from the k seeds identified by the algorithm:

$$Err_{rel}(\theta) = \frac{|\sigma_\theta(S) - \tilde{\sigma}_\theta(S)|}{\tilde{\sigma}_\theta(S)}$$

- The relative execution time, i.e. the ratio between the overall execution time with the current threshold θ and the maximum time taken by the other instances executed with different values of θ:

$$Time_{rel}(\theta) = \frac{T^\theta}{max_{\theta \in \Theta} T^\theta}$$

- The percentage decrement between the number of covered nodes with the current threshold and the maximum number of nodes covered with different values of θ:

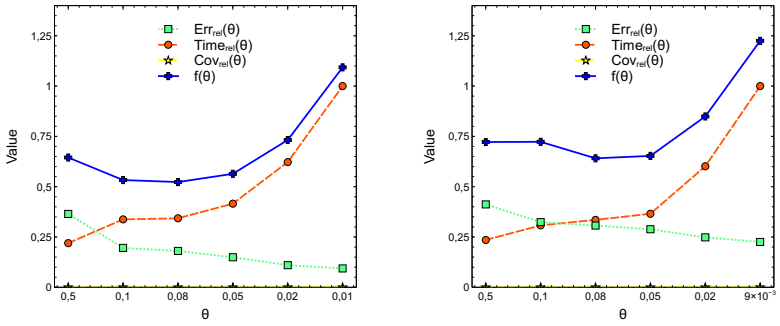
$$Cov_{rel}(\theta) = (1 - \frac{\tilde{\sigma}_\theta(S)}{max_{\theta \in \Theta} \tilde{\sigma}_\theta(S)})$$

The threshold values are sorted in descending order. We estimated the optimal value of the threshold as:

$$\hat{\theta} = argmin_{\theta \in \Theta} f(\theta)$$

Figure 2 shows the trend of $f(\theta)$ (in blue) varying θ for *yes* and *no* graphs. For decreasing values of θ, the relative error of expected spread (in green) decreases, while the overall execution time (in red) increases. Thus, through the minimization of $f(\theta)$, we are able to find a suitable value for θ, reached at

$\hat{\theta} = 8 \cdot 10^{-2}$ for both graphs, which provides a good trade-off between accuracy and complexity. Subsequently, this configuration can be used to evaluate quantitative and qualitative aspects of the obtained results, such as comparing the members of the final seed set with the major activists and journalistic pages affiliated with the corresponding faction.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

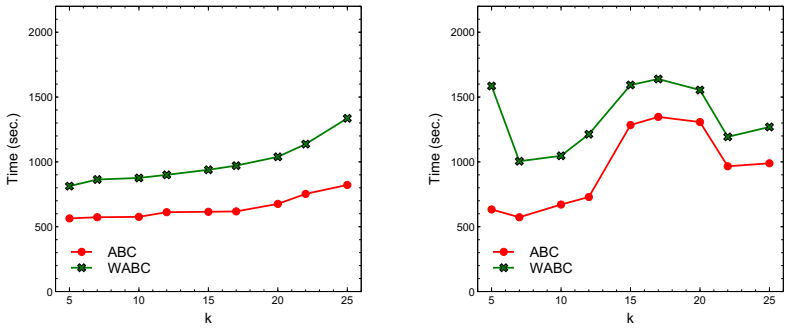
Figure 2: Trend of the $f(\theta)$ score function and its components (Err_{rel}, $Time_{rel}$, Cov_{rel}).

5.4. WABC vs. ABC

In this section we investigate the main advantages of the WABC algorithm with respect to its original version (ABC), by varying the cardinality of the seed set k, where $k \in K$ and $K = \{5, 7, ..., 25\}$. Our analysis focuses on evaluating: *i*) the *execution time*; *ii*) the *evaluated spread* ($\tilde{\sigma}(S)$), an estimate of the real spread achieved via simulation; and *iii*) the *expected spread* ($\sigma(S)$), an estimate given by the algorithm. The computing system used for the experimental evaluation is a machine equipped with 4 CPUs (AMD 6376), each one with 16 cores of 2.3GHz, 256 GB of memory and 1TB of disk space.

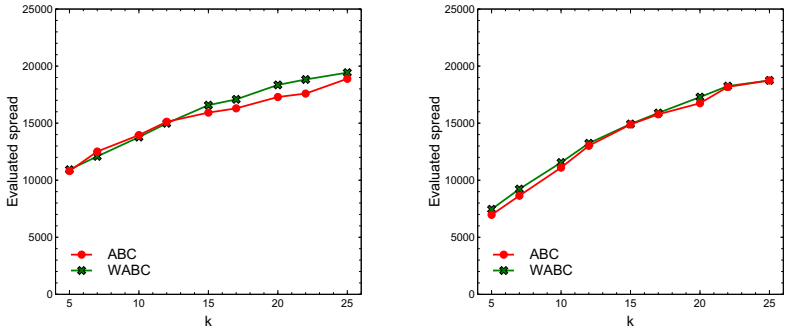
Figure 3 shows the execution time for the two algorithms by varying the value of k. The ABC algorithm provides better performance thanks to the greater simplicity in computing the fitness function that analyzes only the neighbors of the seeds at a fixed distance. Besides this factor, greater simplicity emerges in terms of convergence, as ABC generally converges after a single iteration.

Figure 4 shows the number of influenced nodes, estimated by simulating the information diffusion process, starting from the seed set identified by the two algorithms. In this case, WABC achieved similar results with respect to ABC, finding sets of seeds which allow to reach almost the same number of nodes in all the considered configurations. The average in-degree of the subgraph induced by the set of influenced users for G_{yes} and G_{no} graphs is equal to 3.86 and 6.15 respectively. These values highlight the tendency of users, especially in the G_{no} graph, to retweet content from different sources. This results in a small number of exclusive retweet relationships with individual users, characterized by a high probability of influence.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

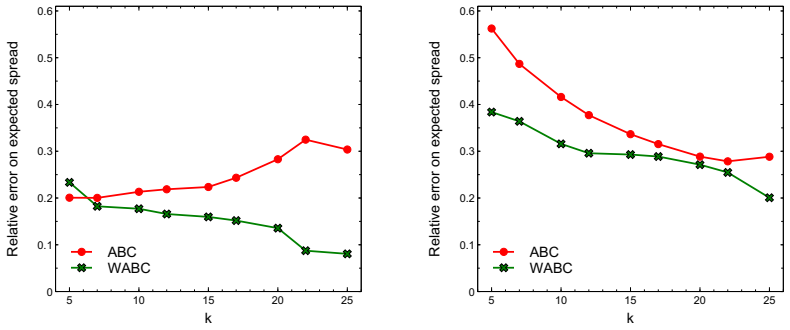
Figure 3: Comparison between WABC and its classical version ABC in terms of execution time.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

Figure 4: Comparison between WABC and ABC in terms of evaluated spread $\tilde{\sigma}_k(S)$.

A third aspect to analyze concerns the quality of the expected spread. The majority of ranking-proxy models, such as Degree, PageRank, Rank and IRIE are unable to provide an estimate of the expected spread, highlighting the need for identifying appropriate solutions for this issue. The ABC algorithm provides an estimate of the spread, returning the number of unique reachable nodes, at a distance one, starting from the identified seed set. Figure 5 shows a comparison between WABC and ABC in terms of relative error on the expected spread. In formulas: $\frac{|\sigma_k(S)-\tilde{\sigma}_k(S)|}{\tilde{\sigma}_k(S)}$, varying $k \in K$, with respect to the seed set S, where $\sigma_k(S)$ is the expected spread given by the algorithm and $\tilde{\sigma}_k(S)$ is the evaluated spread, i.e. an estimate of its real value determined through 20 000 simulations. It can be clearly observed that the WABC algorithm provides more accurate spread estimates compared to ABC, with an up to 24% decrease (on average 10%) of the relative estimation error.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

Figure 5: Comparison between WABC and ABC in terms of estimation error on the expected spread.

Summing up, the following emerged from the comparison between WABC and ABC. The two algorithms achieved very similar results in terms of number of influenced users in almost all configurations tested during our simulations. WABC showed to be more time-consuming as it exploits a more sophisticated approach for fitness evaluation. Consequently, WABC is able to obtain much more precise estimates of the expected spread, a crucial aspect that makes the algorithm more suitable for use in real contexts.

5.4.1. Qualitative analysis

We also carried out a qualitative analysis of our results discriminating the category of the influencers identified by WABC, simulating also the diffusion process in order to visualize their hypothetical influence within the network.

We firstly compared the two seed sets generated by WABC and ABC for G_{yes} and G_{no} graphs.

G_{yes}		G_{no}	
ABC	WABC	ABC	WABC
lucatelese	serracchiani	dukana2	dukana2
bastaunsi	fnicodemo	beppe_grillo	beppe_grillo
ArsenaleKappa	TwitterItalia	matteosalvinimi	matteosalvinimi
antonio_bordin	matteorenzi	marionecomix	comitatono
GiorgiaMeloni	pdnetwork	figprov	ale_dibattista
nonleggerlo	ArsenaleKappa	antonio_bordin	antonio_bordin
matteorenzi	repubblicait	tuseivitaearia	luigidimaio
TwitterItalia	Tgcom24	ArsenaleKappa	ArsenaleKappa
tuseivitaearia	tuseivitaearia	arsenaletv	Mov5Stelle
molumbe	bastaunsi	ComitatoDelNo	GiorgiaMeloni
overlap: 50%		overlap: 50%	

Table 5: Comparison between the seed sets identified by WABC and ABC.

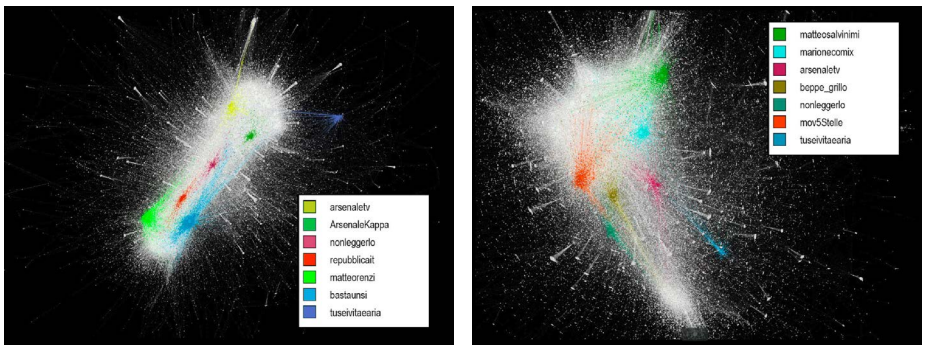
As shown in Table 5, the two algorithms produced quite different results, with a seed set overlap equal to 50% for both graphs. It is also worth noting that the results generated by WABC are more accurate from the point of view of political polarization. In fact, a correct correspondence is present between each member of the seed set of both factions and his/her actual political alignment. Afterwards, we divided influencers identified by WABC in four categories, *news pages* (information or satire), *political activist*, *popular user* and *normal user*, finding out the composition shown in Table 6.

	news pages	*political activist*	*popular user*	*normal user*
G_{yes}	50%	20%	20%	10%
G_{no}	20%	60%	10%	10%

Table 6: Classification of the influencers for G_{yes} and G_{no} graphs.

By observing this categorization, we can determine the type of the political campaign for the two factions, characterized in the case of *no* by a greater effort of leading politicians, such as Matteo Salvini, Alessandro Di Battista, Luigi Di Maio, Beppe Grillo and Giorgia Meloni. The *yes* faction instead saw Matteo Renzi as the main leader, followed by news pages such as La Repubblica and Tgcom24, confirming the communication strategy of *yes* faction, centralized on the head of government.

Lastly, Figure 6 shows the results of a simulation executed starting from the seed set identified for the two graphs. By coloring each node according to the seed node (influencer) that determined its activation, we can see that the aforementioned political activists and news page can activate a remarkable portion of the network.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

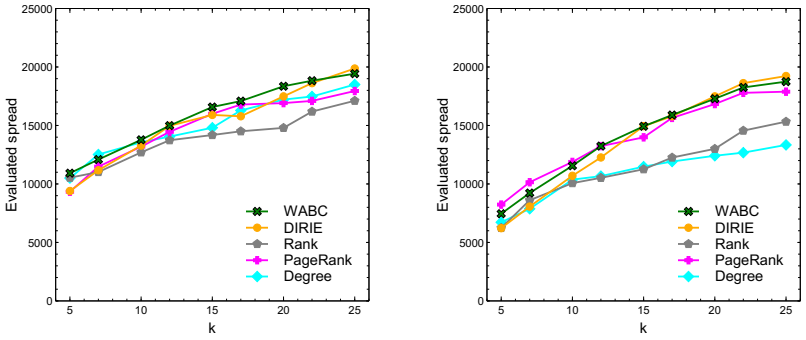
Figure 6: Simulation of the influence diffusion process starting from the seed set identified for the two graphs.

5.5. *WABC vs ranking-proxy models*

For better assessing the effectiveness of WABC we also carried out a comparison, in terms of evaluated spread, with the most relevant ranking-proxy models used in literature:

- *Degree*: uses the out-degree of each node as the seed set selection criterion.
- *PageRank*: uses the pagerank [7] of each node, evaluated on the reversed graph, as the seed set selection criterion.
- *Rank*: uses the rank, proposed in [31] of each node as the seed set selection criterion.
- *DIRIE*: is a distributed version of the IRIE algorithm [17].

Figure 7 shows the results obtained by WABC in comparison with these ranking-proxy techniques by varying k. Compared to the aforementioned techniques, WABC turned out to be the most effective, providing the best solution in almost any configuration. In particular, WABC outperformed ranking-proxy techniques based on simple classical centrality measures, i.e. PageRank, Rank and Degree, with an up to 40% improvement over the latter. Even compared to DIRIE, which is based on the Independent Cascade model and exploits a more complex algorithm, WABC was able to find a more accurate seed set which allows to maximize the spread in almost all the considered configurations.



(a) Graph of *yes* (G_{yes}). (b) Graph of *no* (G_{no}).

Figure 7: Comparison between WABC and the most relevant state-of-art ranking-proxy techniques in terms of evaluated spread.

5.6. *Worst, average and best case analysis*

The extensive experimental evaluation described in the previous sections empirically demonstrated the efficiency and effectiveness of the proposed algorithm. In this section we provide an analysis of the worst, average and best cases from the viewpoint of result quality and performance.

The main factor that can influence the quality of the results produced by the algorithm is the choice of the initial seed set with respect to the underlying community structure $\mathcal{C} = c_1, c_2, \ldots, c_k$, generally composed by a number of communities $k > 1$. In the worst case, the initial seed set S is composed of nodes belonging to the same community. This can lead to the algorithm being trapped within that community, losing the contribution of important nodes located in different communities and converging towards a local optimum. On the contrary, the best case occurs when the nodes belonging to S cover all the communities in $\mathcal{C}$, which allows the social graph to be fully explored maximizing the effectiveness of the algorithm. On average, the number of initial seeds is less than the number of communities in the graph, $k > |S|$, so the seed set is unlikely to cover all communities in $\mathcal{C}$, but at the same time it should not consider a single community $c_i \in \mathcal{C}$.

Taking into account performance, the algorithm can suffer from convergence problems. Convergence time can be heavily influenced by the choice of the parameter $\omega > 0$, which specifies the minimum increment in percentage of the spread between two subsequent iterations, defined as $\frac{F(t)-F(t-1)}{F(t-1)}$. As a consequence, this parameter controls the trade-off between accuracy and execution time: smaller values of ω may lead to more accurate solutions and a higher number of iterations. In the worst case, the greedy algorithm maximizes the total spread performing a high number of small steps, characterized by an increment $\epsilon \geq \omega$, which prevents stopping on a local maximum. In such a situation the algorithm slowly converges towards an optimal solution whose fitness value is only slightly better than the local maximum, albeit paying a very high cost in terms of execution time. On the contrary, in the best situation we would find the global optimum in the first iterations, which will lead to an increment $\epsilon = 0 < \omega$. On average, if the selected value of ω identifies a good trade-off, the algorithm will converge on an acceptable local optimum in a reasonable number of iterations.

6. Conclusion

Influence maximization is an optimization problem aimed at finding a k-seed set which maximizes the spread of influence in a social network. This problem is a central one in understanding how information flows within a network of users, and is related to a wide range of applications in viral marketing, advertisement and news spread. In this paper we proposed a bio-inspired influence maximization algorithm, namely *Weighted Artificial Bee Colony (WABC)*, improving fitness evaluation with respect to classical Artificial Bee Colony (ABC). WABC has been applied to a real case study related to the Constitutional Referendum held in Italy in 2016. By analyzing the propagation of information within the Twitter network, we identified the main influencers for the *yes* and *no* factions, deriving also the main information diffusion strategies of each faction during the political campaign. The tests carried out confirmed the effectiveness of the proposed algorithm, which outperformed ranking-proxy techniques

based on standard centrality measures, i.e., PageRank, Rank and Degree. Even compared to DIRIE, which is based on the Independent Cascade model and exploits a more complex algorithm, WABC was able to find a more accurate set of users which allows to maximize the spread in almost all the considered configurations. Furthermore, WABC proved to be much more accurate than ABC, with an improvement of up to 24% in estimating the expected spread.

As a future work, the relationship between the diffusion of influence and political polarization can be further investigated, analyzing how the tendency of users to polarize in favor of a faction affects the dynamics of information diffusion and vice versa. Furthermore, we can analyze how this phenomenon varies over time for identifying patterns on the evolution of users' polarization. Consequently, once identified the most influential users supporting the different factions, the outcome of a political event can be predicted. Finally, our algorithm can be adapted and applied to scenarios other than the political one, in order to analyze for example the spread of news, the adoption of new technologies, or the promotion of new products.

Acknowledgement

This work has been supported by the ASPIDE Project funded by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 801091.

Data and code availability statement

The data that support the findings of this study are publicly available. In particular, this data was gathered using Twitter publicly available APIs. For the purpose of using the code of our method, an open-source version of WABC is available at `https://github.com/SCAlabUnical/WABC`.

References

[1] Arnaboldi, V., Conti, M., Passarella, A., Dunbar, R.I., 2017. Online social networks and information diffusion: The role of ego networks. Online Social Networks and Media 1, 44 – 55. doi:`https://doi.org/10.1016/j.osnem.2017.04.001`.

[2] Banerjee, S., Jenamani, M., Pratihar, D.K., 2020. A survey on influence maximization in a social network. Knowledge and Information Systems , 1–39.

[3] Barbieri, N., Bonchi, F., Manco, G., 2013. Topic-aware social influence propagation models. Knowledge and information systems 37, 555–584.

[4] Belcastro, L., Cantini, R., Marozzo, F., Talia, D., Trunfio, P., 2020. Learning political polarization on social media using neural networks. IEEE Access 8, 47177–47187.

[5] Borgs, C., Brautbar, M., Chayes, J., Lucier, B., 2014. Maximizing social influence in nearly optimal time, in: Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms, SIAM. pp. 946–957.

[6] Bozorgi, A., Samet, S., Kwisthout, J., Wareham, T., 2017. Community-based influence maximization in social networks under a competitive linear threshold model. Knowledge-Based Systems 134, 149–158.

[7] Brin, S., Page, L., 1998. The anatomy of a large-scale hypertextual web search engine .

[8] Chen, W., Lu, W., Zhang, N., 2012. Time-critical influence maximization in social networks with time-delayed diffusion process, in: Twenty-Sixth AAAI Conference on Artificial Intelligence.

[9] Chen, W., Wang, Y., Yang, S., 2009. Efficient influence maximization in social networks, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 199–208.

[10] Chen, W., Yuan, Y., Zhang, L., 2010. Scalable influence maximization in social networks under the linear threshold model, in: 2010 IEEE international conference on data mining, IEEE. pp. 88–97.

[11] Domingos, P., Richardson, M., 2001. Mining the network value of customers, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 57–66.

[12] Ghani, N.A., Hamid, S., Hashem, I.A.T., Ahmed, E., 2019. Social media big data analytics: A survey. Computers in Human Behavior 101, 417–428.

[13] Goyal, A., Bonchi, F., Lakshmanan, L.V., 2011a. A data-based approach to social influence maximization. arXiv preprint arXiv:1109.6886 .

[14] Goyal, A., Lu, W., Lakshmanan, L.V., 2011b. Simpath: An efficient algorithm for influence maximization under the linear threshold model, in: 2011 IEEE 11th international conference on data mining, IEEE. pp. 211–220.

[15] Guille, A., Hacid, H., Favre, C., Zighed, D.A., 2013. Information diffusion in online social networks: A survey. ACM Sigmod Record 42, 17–28.

[16] Hutto, C.J., Gilbert, E., 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text, in: Eighth international AAAI conference on weblogs and social media.

[17] Jung, K., Heo, W., Chen, W., 2012. Irie: Scalable and robust influence maximization in social networks, in: 2012 IEEE 12th International Conference on Data Mining, IEEE. pp. 918–923.

[18] Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization. Technical Report. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

[19] Kempe, D., Kleinberg, J., Tardos, É., 2003. Maximizing the spread of influence through a social network, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 137–146.

[20] Kim, J., Kim, S.K., Yu, H., 2013. Scalable and parallelizable processing of influence maximization for large-scale social networks?, in: 2013 IEEE 29th international conference on data engineering (ICDE), IEEE. pp. 266–277.

[21] Kim, J., Lee, W., Yu, H., 2014. Ct-ic: Continuously activated and time-restricted independent cascade model for viral marketing. Knowledge-Based Systems 62, 57–68.

[22] Kimura, M., Saito, K., 2006. Tractable models for information diffusion in social networks, in: European conference on principles of data mining and knowledge discovery, Springer. pp. 259–271.

[23] Lee, J.R., Chung, C.W., 2014. A fast approximation for influence maximization in large social networks, in: Proceedings of the 23rd international conference on World Wide Web, pp. 1157–1162.

[24] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N., 2007. Cost-effective outbreak detection in networks, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 420–429.

[25] Li, Y., Fan, J., Wang, Y., Tan, K.L., 2018. Influence maximization on social graphs: A survey. IEEE Transactions on Knowledge and Data Engineering 30, 1852–1872.

[26] Litou, I., Kalogeraki, V., Katakis, I., Gunopulos, D., 2017. Efficient and timely misinformation blocking under varying cost constraints. Online Social Networks and Media 2, 19 – 31. doi:https://doi.org/10.1016/j.osnem.2017.07.001.

[27] Marozzo, F., Bessi, A., 2018. Analyzing polarization of social media users and news sites during political campaigns. Social Network Analysis and Mining 8, 1–13. ISSN:1869-5469.

[28] Mavrovouniotis, M., Li, C., Yang, S., 2017. A survey of swarm intelligence for dynamic optimization: Algorithms and applications. Swarm and Evolutionary Computation 33, 1–17.

[29] Riley, J.R., Greggers, U., Smith, A.D., Reynolds, D.R., Menzel, R., 2005. The flight paths of honeybees recruited by the waggle dance. Nature 435, 205–207.

[30] Saito, K., Nakano, R., Kimura, M., 2008. Prediction of information diffusion probabilities for independent cascade model, in: International conference on knowledge-based and intelligent information and engineering systems, Springer. pp. 67–75.

[31] Sankar, C.P., Asharaf, S., Kumar, K.S., 2016. Learning from bees: An approach for influence maximization on viral campaigns. PloS one 11.

[32] Stoica, A.A., Chaintreau, A., 2019. Fairness in social influence maximization, pp. 569–574. doi:10.1145/3308560.3317588.

[33] Sun, J., Tang, J., 2011. A survey of models and algorithms for social influence analysis, in: Social network data analytics. Springer, pp. 177–214.

[34] Zubiaga, A., 2019. Mining social media for newsgathering: A review. Online Social Networks and Media 13, 100049. doi:https://doi.org/10.1016/j.osnem.2019.100049.