UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Big Data Analytics and Business Intelligence

Online Social Networks Influence Maximization, Bio-Inspired Approaches

Anno Accademico 2016/2017

relatore
Ch.mo prof. Vincenzo Moscato
Ch.mo prof. Antonio Picariello
Ing. Giancarlo Sperlì

candidato
Guido del Puente
matr. M63000500

Contents

Chapter 1

Big Data

1.1 Information explosion

In 1944, a librarian of the Wesleyan University, Fremont Rider, estimated the growth rate of the American universities libraries. He speculated that they were doubling in size every sixteen years. Given this, the Yale Library in 2040 will have approximately 200,000,000 volumes, which will occupy over 6,000 miles of shelves [56]. This was one of the first attempts of observation about the phenomenon of data or information explosion. Today we know that every second around 60.000 search are done on Google, 8.000 tweets on Twitter and 44.000 Gb of Internet traffic is made [19]. All these data are stored not only for the purpose for which they were collected or achieved, in Google's case, once a search query has been processed. Data are no longer regarded as static or stale. Rather, data become a raw material used to create new information, new knowledge.

We live in the information society, we are inundated with services that give us data, the quantitative change has led to a qualitative one. We want even more accurate and new information. Very powerful computers are a blessing to many fields of inquiry. They are also a curse; fast computations spew out massive amounts of data. One of the first appearance of the term "Big Data" refers to this

explosion in the quantity of available and potentially relevant data, largely the result of recent and unprecedented advancements in data recording and storage technology. Today we can "save all the bits" [21].

There is no rigorous definition of big data. But there are some characteristics, properties that delineate Big Data as Gartner says about it:

"*Big data" is high-volume, -velocity and -variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.*" — *Gartner* [4]

The first three factors highlighted are Volume, Velocity and Variety. About the **Volume** we have already dealt with some aspects. **Velocity** is frequently equated to how fast the data are coming in. Yet, velocity is also about the real-time analytics, the rate of changes and linking data sets that are coming with different speeds [64]. Consider the census. The ancient Egyptians are said to have conducted censuses, as did the Chinese. The ability to record information is one of the lines of demarcation between primitive and advanced societies. Today, thanks to the digitalization, with a little imagination, a cornucopia of things can be rendered into data form. Therefore, we speak of **Variety**. Companies are digging out amazing insights from text, locations or log files. Most of this data already belong to organizations, but they are sitting there unused.

These factors have necessitated the change of the tools used for analysis and storage. This is the origin of new processing technologies like Google's MapReduce and its open-source equivalent, Hadoop, which came out of Yahoo. They let one manage far larger quantities of data than before, and the data need not be placed in tidy rows or classic database tables. New technologies imply expensive solution but those costs are balanced by the effective answers received to the problems posed.

After all, it must be remembered that, as Richard W. Hamming, mathematician and pioneer computer scientist, said "The purpose of computing is insight,

not numbers". The main goals for now are to learn how to identify and formulate big data problems. Centuries of established practices and our most basic understanding of how to make decisions and comprehend reality were overturned. The challenge of this Big Data Era is to dismiss the habit to search the law that describes the evolution of an event in exchange of probability and simple correlations: "not knowing why but only what" [45]. The technology can't prescind from the human factor, it is essential. It still needs humans to ask right questions.

1.2 Tools and Techniques

Different architectural models are suited for analytical environments. But single-node computers cannot easily accommodate massive amounts of data because they are limited in their capacity. In contrast, platforms composed of collections of computers are used for high-performance in which among a pool of resources the massive amounts of data and requirements for processing can be distributed. Parallel platforms use different processing units that share persistent storage and memory components. This approach is called shared-everything. But there are bottlenecks that exist because of the sharing and there are limits to the degree of scalability too. instead, using a shared-nothing approach, each processor has its own dedicated memory and disk storage. In this configuration, the scalability is horizontal and commodity hardware can be used. The nodes can be connected together via a variety of network topologies. Specialty software appliances may differ in the specifics of the configurations. However, the general architecture distinguishes the management of computing resources and the management of the data across the network. We will focus on the core aspects of Hadoop's utilities: HDFS, Hadoop distributed file systems, and MapReduce [42].

1.3 Hadoop

History Hadoop was created by Doug Cutting and has its origins in Apache Nutch, an open source web search engine. Nutch's developers set about writing an open source implementation of the GFS, Google's distributed filesystem, starting from the publication of a paper in 2003 that described the architecture. After in 2004 Google published the paper that introduced MapReduce to the world, by the middle of 2005 all the major Nutch algorithms had been ported to run using MapReduce and NDFS. But NDFS and the MapReduce implementation in Nutch were applicable beyond the realm of search, and in February 2006 they moved to form an independent subproject called Hadoop out of Nutch. In the following years, Hadoop was made its own top-level project at Apache, confirming its success and its diverse, active community. Hadoop was being used by many companies first from Yahoo! that was its springboard. Today, Hadoop is widely used in mainstream enterprises. Hadoop has been recognized by the industry as a general purpose storage and analysis platform for big data, ever more product use or incorporate Hadoop in some way [81].

1.3.1 HDFS

HDFS distributes the data among a pool of data nodes, in this way enables the storage of large files. The two principal types of nodes are the Name Node and the Data Node. In a cluster, there is just one Name Node and one or more Data Node. The Name Node is the coordinator of the interactions between the Data Nodes. He provides the management of a typical hierarchical file organization and namespace. A file in HDFS is stored as multiple blocks named "chunks" each one is located in a Data Node. The mapping of blocks to files as well as metadata about each file is maintained by the Name Node.

HDFS through data replication provides a level of fault tolerance. A replication

is also managed by the Name Node, witch in relation to the cluster's configuration optimize the marshaling and communication of replicated data. In larger environments consisting of multiple racks of data servers this is increasingly important, since on the same rack communication among nodes is generally faster than between server node sin different racks. There are a number of key tasks for failure management that increase the reliability and robustness of the system. Such as:

- Monitoring: Between the data nodes to the name node there is a continuous "heartbeat" communication.
- Rebalancing: When there is an increased demand for the data a process moves them to improve performance.
- Managing integrity: To verify that the data stored corresponds to the data shared or received, HDFS uses checksums.

1.3.2 MapReduce

Hadoop MapReduce is a software framework dependence on two basic operations that are applied to sets or lists of data value pairs:

1. Map, which describes the computation or analysis applied to a set of input *key/value* pairs to produce a set of intermediate *key/value* pairs.
2. Reduce, in which the set of values associated with the intermediate *key/value* pairs output by the Map operation are combined to provide the results.

The execution environment of MapReduce employs a *master/slave* execution model, in which one master node (called the Job Tracker) manages a pool of slave computing resources (called Task Trackers) that are called upon to do the actual work. The MapReduce framework operates exclusively on <*key, value*> pairs, that is, the framework views the input to the job as a set of<*key, value*> pairs

and produces a set of <*key, value*> pairs as the output of the job, conceivably of different types [28].

But there are limitations within this existing MapReduce model. It is suited to applications where there is locality between the processing and the data, not all applications are easily mapped to the MapReduce model.

Example WordCount This example is to get a flavor for how a MapReduce application works. WordCount [28] is a simple application that counts the number of occurrences of each word in a given input set. Suppose this is the input data file:

```
Hello World Bye World
Hello Hadoop Goodbye Hadoop
```

The first step is a map if the input file is chunked by line the result of the first Task Tracker is:

```
< Hello, 1>
< World, 1>
< Bye, 1>
< World, 1>
```

The second map emits:

```
< Hello, 1>
< Hadoop, 1>
< Goodbye, 1>
< Hadoop, 1>
```

The implementation of reduce method just sums up the values, which are the occurrence counts for each key, the words in this example. So, the output of the job is:

```
< Bye, 1>
< Goodbye, 1>
< Hadoop, 2>
< Hello, 2>
< World, 2>
```

1.3.3 Ecosystem

Many other tools were developed to enrich and exploit the potential of HDFS and MapReduce. Today a real ecosystem has been created around, we'll mention the main projects:

- YARN (Yet Another Resource Negotiator) [77] is a cluster management technology, is one of the key features in the second-generation Hadoop. The scheduling and resource management have been separated from the management of MapReduce pipelines. Hadoop version 2 with YARN still provides full MapReduce capability and backwards compatibility with version 1, it also opens the door to many other "application frameworks" that are not based on MapReduce processing [48].

- HBase is a Java-based, open source, NoSQL, nonrelational, column-oriented, distributed database built on top of the Hadoop Distributed Filesystem (HDFS), that supports structured data storage for large tables. HBase brings to the Hadoop ecosystem most of the BigTable capabilities [66].

- Hive is a data warehouse infrastructure that provides data summarization and an SQL dialect, called Hive Query Language, for querying data stored in a Hadoop cluster [10].

- Tez is a generalized data-flow programming framework, standing on Hadoop

YARN, which supply a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases.

- ZooKeeper is a centralized service for maintaining configuration details, naming, suppling distributed synchronization, and giving group services. A high-performance coordination service for distributed applications.

1.3.4 Scala

Scala is a Java-like programming language which unifies object-oriented and functional programming. It is a pure object-oriented language in the sense that every value is an object. Types and behavior of objects are described by classes. Classes can be composed using mixin composition. Scala is designed to work seamlessly with two less pure but mainstream object-oriented languages: Java and C#. Scala is a functional language in the sense that every function is a value. Nesting of function definitions and higher-order functions are naturally supported. Scala also supports a general notion of pattern matching which can model the algebraic types used in many functional languages. Scala has been designed to interoperate seamlessly with Java (an alternative implementation of Scala also works for .NET). Scala classes can call Java methods, create Java objects, inherit from Java classes and implement Java interfaces. None of this requires interface definitions or glue code. Scala has been developed from 2001 in the programming methods laboratory at EPFL. Version 1.0 was released in November 2003. The second version of the language was released in March 2006 [49].

1.3.5 Spark

Apache Spark is a general-purpose cluster computing framework, which supports applications with working sets while giving similar scalability and fault tolerance properties to MapReduce [95]. Apache Spark began as a research project at UC

Berkeley in the AMPLab, which centers on big data analytics. Their goal was to design a programming model that supports a much wider category of applications than MapReduce, while preserving its automatic fault tolerance. In details, MapReduce is unproductive for multi-pass applications that need low-latency data sharing through multiple parallel operations. These applications are rather ordinary in analytics, and include:

- Iterative algorithms, comprehending many machine learning algorithms and graph algorithms like PageRank.
- Interactive data mining, where a user wanna load data into RAM across a cluster and query it repeatedly.
- Streaming applications that keep aggregate state over time.

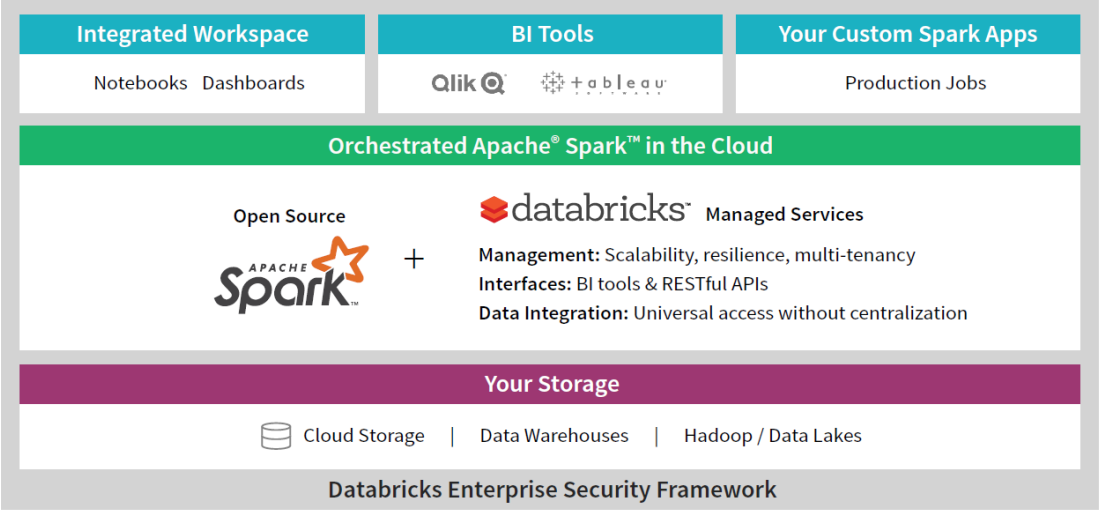
The main abstraction in Spark is that of a resilient distributed dataset (RDD), which constitutes a read-only collection of objects partitioned through a group of machines that can be rebuilt if a partition is lost. Users can arbitrarily cache an RDD in memory across machines and reuse it in multiple MapReduce-like parallel operations. RDDs allow Spark to outperform existing models by up to 100x in multi-pass analytics.

Spark is implemented in Scala 1.3.4. It supply high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a large ammount of higher-level tools among wich Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

The GraphX abstraction bring together the data-parallel and graph parallel computation across a data model that presents graphs and collections as first class objects and a set of primitive operators that allows their composition. GraphX represents graphs internally through two Spark distributed collections (RDDs) - an edge collection and a vertex collection.

1.3.6 Databricks

Databricks was established by the team who created Apache Spark1.3.5 and is the largest contributor to the project. Their mission at Databricks is to empower individuals and organizations to rapidly build and deploy advanced analytics solutions. They do this by their product, a virtual analytics platform called Databricks.



By virtualizing storage, Databricks allows access to data anywhere. Databricks provides a highly secure and reliable production environment in the cloud, managed and mantained by Spark experts. Thanks to a collaborative and integrated environment, Databricks streamlines the process of exploring data, prototyping, and operationalizing data-driven applications in Spark. Databricks offers a flexible job scheduler that permits a seamless transition from prototyping to production deployment without incremental work. Databricks empowers enterprises with security-enabled data democratization so that they can confidently build advanced analytics solutions when security considerations are fundamental.

Chapter 2

Social Network Analysis

A definition for the Social Network Analysis (SNA) can be "study of human relationships by means of graph theory" [76]. SNA is comparable to numerous statistical techniques, examine the social media is one way to apply SNA methods - not only the data are available without difficulty, but the possibilities for the study are many and profitable. The domain of social network analysis, ten years ago, was a scientific isolated area. It was nonconformist, rejected from both conventional sociology and conventional computer science, moreover data of social network was difficult to collect and tough to find. The coming of the Social Internet has renovated everything; an API is provided by every social media site for a simple retrieval of data; even many states are releasing data very useful to SNA techniques. The web as Tim Berners-Lee saw it as a place where people could interact. He called it "a collaborative medium, a place where we all meet and read and write" [27]. To answer many questions about our sociality, the science of SNA observes and studies the patterns of relationship that are related to our personality, education, background, race, ethnicity. SNA also look beyond social media, inside a company to find out how the social network around the water dispenser and lunchroom influences the organization's performance, analyze the universe of terrorists, revolutionaries, and radicals or the conformation of trends and fads as

off line phenomena

2.1 Basics of network structure

A graph is the model for the structure of a social network. to specify relationships among a collection of entities a graph is the mathematical way. It is formed of a set of entities, named nodes or vertices. Both will be used contingent on the background of the network analysis work you are reading, but are interchangeable Some pairs of these objects are connected by links (or edges). More formally, a graph consisting of n nodes and m edges is a pair (V,E) where V = v1, v2, . . . , vn is the set of nodes and $E \subseteq V \times V$ is the set of edges [57] However, edges can have a number of supplementary features, which can be used in analysis Edges can be tagged. The relationship between the vertices, can be described by the label. Edges can also be either directed or undirected. An undirected edge shows a mutual relationship, where as a directed edge shows a relationship that one node has with the other that is not necessarily reciprocated. The network is defined by the type of edge used as either a directed network or an undirected network. Some relationships are asymmetric by nature, it is easy to give examples. Professor/student or father/son roles presume a directionality of a relationship, and do not grant for a symmetric tie back. An example, in an online social media, is the following on Instagram. Other relationships are symmetric. LinkedIn connections and Facebook friendship ask reciprocal confirmation. Formally a graph is undirected if E is a set of unordered pairs; otherwise it is directed (i.e. edges are ordered pairs). Edges can also be valued or weighted. The weight is a number that reports numerical information concerning a relationship. Frequently, this is the strength of a relationship, but indicate many things and can come from a variety of sources. In the Twitter world, "Alice follows Bob" is a binary relationship while "Alice retweeted 4 tweets from Bob" is valued. But for relationships in the real

world it's very hard to define and quantify the quality of an interpersonal relationship. A useful substitute for strength of an interpersonal relationship is frequency of communication. Relationships within a graph are generally represented by a $n \times n$ adjacency matrix g $= [g_{ij}]i, j \in V$, where $g_{ij} \in 0, 1$ represents the existence of an edge from node i to node j. Each g_{ij} value can also be non-binary. In this case, it represents the weight or value of the edge.

Table 2.1: Adjacency Matrix

	Alice	Bob	Carl	David	Ernest
Alice	0	1	1	1	0
Bob	0	0	1	0	1
Carl	1	1	0	1	1
David	0	0	0	0	1
Ernest	0	0	1	0	0

2.2 Network properties

In literature, there are many network properties that describe how vertices are linked to one another and to the entire network. Degree is the most straightforward, the simplest. The number of edges linked to a vertex is the degree of that vertex. For undirected graphs the previous definition is formally correct. In directed graphs, need to draw a distinction, there are two measures of degree: in-degree and out-degree. The in-degree is quantified as the number of edges coming into the vertex. The out-degree is the number of edges originating from the vertex going outward to other vertices.

2.3 Centrality

Base of the network analysis is the principle of Centrality . The term centrality generally reminds one of importance or prominence of a vertex in a network, more precisely, the status of being situated in strategic position inside the network [55].

To calculate centrality of a node there are many ways affirmed in the literature. In the follow four types of centrality are taken in consideration:

- degree centrality
- closeness centrality
- betweenness centrality
- eigenvector centrality

One of the easiest to calculate is **Degree centrality**. The degree centrality of a node is merely its degree, the number of links it has. Being influenced only by adjacent neighbors of a node, this type of centrality highlights the most "evident" nodes in the network, as those acting as the main point of relational information; conversely, nodes with low degrees are marginal in the network. This can be a valid measure.

$$c_D(v) = deg(v)$$

However, the degree centrality is subject to the graph size. For this reason, since the highest degree for a network (without loops) is |V| - 1, we can define the relative centrality as:

$$\widehat{c_D}(v) = \frac{c_D(v)}{|V|-1} = \frac{deg(v)}{|V|-1}.$$

The previously measure is not dependent of the graph size, and in this way, it can be confronted across networks of dissimilar sizes.

Closeness centrality specify how close a node is to all other nodes in the network. In contrast to degree centrality, closeness centrality takes into account as well indirect links between vertices in the network, the nodes that can quickly interact with all others because of their shorter distance to the other vertices will score higher [59] It is measured as the average of the shortest path length from the node to every other node in the network

$$c_C(v) = \frac{1}{\sum_{u \in V} d(u,v)}$$

Where d(v; u) indicate the distance among vertices v;u. If a node has all the other vertices as neighbors, it has the highest closeness. For the same reasons of the previous centrality measure, the relative closeness centrality is defined as:

$$\widehat{c_C}(v) = (|V| - 1)c_C(v) = \frac{|V|-1}{\sum_{u \in V} d(u,v)}$$

An additional significant property, over shortest distance, refers to the capacity of a node to be master of the flow of data through the network. The concept behind betweenness centrality is to estimate the centrality of a vertex v as the fraction of the shortest paths between all pairs of vertices that pass through v [25]. Betweenness centrality measures how significant a node is to the shortest paths among the network. More detail we can define it like:

$$c_B(v) = \sum_{u,z \in V, u \neq v, z \neq v} \frac{m_{u,z}(v)}{m_{u,z}(V)}$$

where mu;z(v) is the number of shortest paths between u and z and passing through v, and mu;z(V) is the total number of shortest paths between u and z. Likewise, to the other, it's proposed to normalize the betweenness to extract a relative betweenness centrality:

$$\widehat{c_B}(v) = \frac{2C_B(v)}{(|V|-1)(|V|-2)}$$

which should be divided by 2 for directed networks.

Commonly, a node's importance should hinge on the importance of the nodes that point to it, and their importance should also hinge on the nodes that point to them, etcetera "ad infinitum" [61].In this case literature generally allude to rank. But also, Eigenvector centrality gives consideration to the importance of node's neighbors while measuring its importance. It is measured from the adjacency matrix finding out the principal eigenvector through a matrix calculation. Follow the characteristic equation used to determine the eigensystem of a matrix, in which r is an eigenvector

$$r = A^T r \Rightarrow (I - A^T)r = 0$$

where r is a vector of size |V| storing all rank scores, and A is the adjacency matrix, where I is the identity matrix of size |V|. Bonacich [7] proposed a generalization of equation, hypothesizing that the rank of each node is not necessarily equal to the weighted sum of the neighbor nodes but just proportional. This is the definition of eigenvector centrality, expressed in this equation:

$$\lambda r = A^T r$$

2.3.1 PageRank

The Google's ranking algorithm was patented by Page and Brin, in the 1998 when they publish PageRank [9]. This one is based on the idea of measuring the probability that a "random surfer", someone that navigates internet clicking randomly on links, will land on a specific page. More inbound links the page has from other popular pages, more probable it is that someone will arrive there just unintentionally. To better simulate the users' clicks one have to consider that all people stop browse after a while, if it were not so they will arrive on every page at last. to contemplate this PageRank introduces a damping factor of 0.85, people will continue clicking on links at each page by a chance of 85% [63]. So, there are three key concepts at the base of PageRank. The first two are also in common with the previously discussed eigenvector centrality methods, namely: a page is important if it is linked by other pages, and the importance of a page is determined by summing the rank values of all pages that point to that page. The third concept is that the importance of a page is propagated to its out-neighbors as distributed proportionally.

2.4 Influence

2.4.1 Small world

Further notable parameters in social networks discernment are the diameter, that is the maximum distance between any pair of vertices in the graph and the average path length, that is the average number of hops necessary to get from one vertex to another over all pairs of vertices in the graph by the shortest route.

The small-world experiment (1967) [75] consists of several experiments conducted by Stanley Milgram, a social psychologist at Harvard University, to find out the average path length between two Americans.

Milligram sent hundreds of letters containing the request to be forwarded to a person known to the recipient, starting from Nebraska in order to reach a target person in Boston as quickly as possible.After being passed through an average of six individuals the letters arrived at the final recipient, this phenomenon is known by the name of six degrees of separation.

2.4.2 Influence analysis examples

For decades both epidemiologists and marketing experts have intensively studied diffusion theories [57]. An example is the popular study published by Christakis and Fowler in the New England Journal of Medicine [15]. They draw out a real "offline" social network from medical records, derived by the relationships between the patients: sibling, spouse, friendship, neighbor, and so on. The purpose of the study was to comprehend the correlation between non-contagious health conditions, e.g. obesity and social neighbors. For example, they discovered that person who has an obese friend is 171% more likely to be obese compared to a randomly picked individual; 37% and 40% respectively in cases of obese spouse.

A renowned instance of success in the business area, using information diffu-

sion is the Hotmail phenomenon [29]. When Hotmail was founded used a simple idea to increase the brand celebrity, i.e. appending to the end of each mail message sent by their users the text "Join the world's largest e-mail service with MSN Hotmail. http://www.hotmail.com". The adoption of Hotmail became viral. Hotmail became the number one e-mail provider, with 8 million users, in only 18 months [13].

The information and influence diffusion have been used and subject of research in disparate area like: social media analytics; the spread of rumors, stories; the adoption of innovations in companies; viral marketing; behavioral targeting; social recommendation, search; the study of human and non-human animal epidemics; etc.

The principal concept is to start up, in a social network, a small group of nodes that head the viral diffusion influencing a large number of other nodes

More precisely, we represent a social network as a directed graph G=(V;E) the vertices in V represent the persons and the social relationship are the links $E \subseteq V \times V$. Moreover, is defined a function $\rho : E \rightarrow [0; 1]$ that connects a probability $\rho(u, v)$ with each edge (u;v) acting as the influence exercised by the individual u on v.

The problem of selecting a small subset of nodes, named seed-set, in a social network that could maximize the influence propagation is called Influence maximization

Other side of the coin The presence of a social link does not have as a consequence a certain behavior of diffusion: The observation can be described using correlation as opposed to causation. In essence, the existence of influence and its benefit for applications rely on the datasets.

2.4.3 Centrality and Influence

Degree centrality measures are simple to calculate because it is required just to compute the direct edges of the vertices in the network. Vertices with high degree centrality have higher chance of receiving and sending information running in the network. Therefore, high degree centrality vertices are efficient in interacting rapidly the others vertices so are classified as vertices with great influence. In contrast the degree centrality measures have a drawback they take in consideration just the direct links of a vertex, while transversal connections are not contemplated at all.

2.4.4 Diffusion

Now we'll focus on the diffusion of information or influence in the network this can be studied looking at discrete time steps, with time t = 0; 1; 2; 3; Two possible state are defined for a vertex $v \in V$: active and inactive. Of course, the active state stand for influenced individual a node that adopts the new information, idea, or buys the product, spreaded across the network. Instead, inactive state represents the ones that are not affected by the diffusion. We define $S_t \subseteq V$ the collection of active vertices at instant t, named as the active set at time t. S_0 is the group of vertices at the starting instant this is called seed set of influence diffusion [13]. Below are the two main formalized ways of diffusion: Independent Cascade Model and Linear Threshold Model.

The Independent Cascade Model (IC) [38] follows some simple rules, going on in discrete steps: A vertex becoming active in the step t, has one opportunity to influence its neighborhood; the success of activation is given only by the probability function ρ, the history does not affect activation. If the diffusion from u to v occurs the node v at next step (t+1) will be active. It goes without saying that if a vertex has multiple just activated neighbors, independently, each will make his attempt

to influence.

The independent cascade model is appropriate to trace mere diffusion where a unique source trigger the activations. Nevertheless, in many circumstances is needed to describe an activation provoked by multiple independent element. The model presented by Kempe et al. [38], namely the linear threshold model, aims to delineate this scenario.

In the linear threshold model, every relationship $(u; v) \in E$ is linked with a value $\in [0; 1]$ named influence weight indicated by w_{uv} that specify the magnitude of u on influencing v. These values are normalized so that for each vertex the sum of values on the incoming edges is at most 1. i.e. $\sum_{u \in N^{in}(v)} w(u, v) \leq 1$ for all $v \in V$

The linear threshold model (LT) adheres to the rules, that are described below, to produce the active sets St for all t. First of all, the inputs are the graph G= (V; E), the influence weights and the seed set S_0. Each vertex $v \in V$ sort a threshold randomly belonging to the interval [0; 1]. At every step for each inactive vertex, if the sum of the weight of the incoming edges from active vertices is greater than or equal to the threshold, then the vertex change its state to active. So, the threshold corresponds to the probability that the vertex is influenced by its active neighbors. Higher is the value, more active neighbors are necessary for the activation [13].

2.5 Influence maximization

The influence maximization is a stochastic optimization problem formally defined as follows: The inputs needed are a social graph G= (V; E), a diffusion model σ and a budget k; the goal is to find a seed set $S_0 \in V with |S_0| \leq k$, so that the influence diffusion of S_0, using the diffusion model $\sigma(S_0)$ is maximized. Formally:

$$S^* = argmax_{S0 \subseteq V, |S0|=k} \sigma(S_0)$$

Wang et al. [78] and Chen et al. [14] prove that the influence computation is #Phard for both IC and LT models. For this reason, various approaches to suboptimal solutions to the problem have been proposed. The greedy approach places reliance on the important properties of the influence diffusion function $\sigma()$. In fact, if the function is monotone and submodular then a simple greedy algorithm can pick a good seed set of vertices [38] . An approximation to the optimum of (1-1/e) is given by such an algorithm. So, at least (1-1/e) $>$ 63% of the number of vertices are activated by the resulting seed set S0 that any other k-size set could activate.

Algorithm 1 Greedy algorithm for Influence Maximization

```
1: procedure GreedyAlgorithm(Graph, k)
2:     S_0 = ∅
3:     while |S_0| < k do
4:         u ← argmax_{w∈V/S_0}(σ(S_0 ∪ w) − σ(S_0))
5:         S_0 = S_0 ∪ u
6:     return S_0
```

The Algorithm a greedy implementation adds a vertex to the set selecting the one which give largest marginal gain $\sigma(S_0 \cup u) - \sigma(S_0)$. The k nodes are chosen repeating this process in a loop. Anyway the function $\sigma()$ in NP-hard and the greedy algorithm call this function repeatedly. To solve this problem has been proposed [38] an implementation that calculate roughly the influence propagation using the Monte Carlo simulations of the diffusion process.

Algorithm 2 Monte Carlo Greedy algorithm for Influence Maximization

```
1: procedure MC-Greedy(G, k)
2:     for i ← 1 to k do
3:         u ← argmax_{w∈V/S} MC − Estimate(S ∪ {w}, G)
4:         S ← S ∪ {u}
5:     return S
1: procedure MC-Estimate(S, G)
2:     count ← 0
3:     for j ← 1 to R do
4:         simulate diffusion process on graph G with seed set S
5:         n_a ← the number of activate nodes after the diffusion ends
6:         count ← count + n_a
7:     return count/R
```

The vast majority of existing algorithms cannot compute in a short time to give an approximation of $(11/e\epsilon)$, for social network composed by thousands of vertices and links are necessary days of elaboration. Algorithms that provide better times does not ensure a worst-case performance. Methods that provide a good tradeoff are the TIM, TIM+ [70]and the IMM [69]

2.5.1 TIM and TIM+

Basically, TIM [70], consist in two phases to be specific *parameter estimation* and *node selection*, form this the name Two-phase Influence Maximization. The **Parameter Estimation** calculate a lower-bound of the maximum expected diffusion to each of all vertices sets of size k, and derive a parameter Θ through the lower-bound. The **Node Selection** phase derives the resulting size-k vertices set, with a large diffusion of influence, making use of a number Θ of vertices sets picked from the social network G.

TIM works in the same way but add a new step between these two that sharps the lower-bound found, improving the algorithm performance

2.5.2 IMM

Proposed by the same authors, Tang et al. [69], IMM stand for "Influence Maximization via Martingales", It surpass the deficiencies of TIM and TIM+. As the name implies, it springs from a series of approximation methods that use as a basis martingales [82], a classic statistical model. So IMM, preserving the two-stage division, encloses a completely dissimilar parameter estimation phase. The latter is able to obtain a lower bound that is asymptotically tight.

Following some definition to better comprehend the algorithm's implementation. Reverse reachable set: Let v be a node in V. A reverse reachable (RR) set for v is generated by first sampling a graph g from G, and then taking the set of nodes in g that can reach v. A random RR set is an RR set for a node selected uniformly at random from V.

Table 2.2: Used Notation

Notation	Description
G = (V, E)	a social network G with a node set V and an edge set E
n, m	the numbers of nodes and edges in G, respectively
k	the size of the seed set for influence maximization
R	the set of RR sets generated by IMM's sampling phase
Θ	the number of RR sets in R
$F_R(S)$	the fraction of RR sets in R that are covered by a node set S
OPT	the maximum expected influence of any size-k node set S

Algorithm 3 IMM Node Selection

1: **procedure** NODESELECTION(R, k)
2: Initialize a node set $S_k^* = 0$
3: **for** i $\leftarrow$ 1 to k **do**
4: Identify the vertex v that maximizes $F_R(S_k^* \cup v) - F_R(S_k^*)$
5: insert v into S_k^*
6: return S_k^*

Algorithm 4 IMM Sampling

procedure SAMPLING(G, k, ϵ, l)
  Initialize a set $R = \emptyset$ and an integer LB = 1
  Let $\epsilon' = \sqrt{2}\epsilon$
  for i $\leftarrow$ 1 to $\log_2 n - 1$ **do**
    Let x = $n/2^i$
    Let $\theta_i = \lambda'/x$ where λ' is as defined in equation 2.4
    while $|R|\theta_i$ **do**
      Select a node v from G uniformly at random
      Generate an RR set for v, and insert it into R
    Let S_i = NodeSelection(R)
    if $nF_R(S_i) \geq (1+\epsilon')x$ **then**
      LB = $nF_R(S_i)/(1+\epsilon')$
  Let $\theta = \lambda^*/LB$ where λ^* is defined in equation 2.3
  while $|R|\theta$ **do**
    Select a node v from G uniformly at random
    Generate an RR set for v, and insert it into R
  return R

$$\alpha = \sqrt{l \log n + \log 2} \tag{2.1}$$

$$\beta = \sqrt{(1-1/e)(\log \binom{n}{k}) + l \log n + \log 2} \tag{2.2}$$

$$\lambda^* = 2n((1-1/e)\alpha + \beta)^2 \epsilon^{-2} \tag{2.3}$$

$$\lambda' = \frac{(2+\frac{2}{3}\epsilon')(\log \binom{n}{k} + l \log n + \log \log 2n)n}{\epsilon'^2} \tag{2.4}$$

Algorithm 5 IMM

procedure IMM(G, k, ϵ, l)
  $l = l(1 + \log 2/\log n)$
  $R = Sampling(G, k, \epsilon, l)$
  $S_k^* = NodeSelection(R, k)$
  return S_k^*

An implementation of this algorithm in Scala code has been developed and can

be found on our gitlab repository IMM [1]

2.6 Social Network Modelling

The datafication is not blocked to the measurable also relationships, experiences and moods, today, are catalogued. Many of the Web's social media companies have in the backbone the idea of datafication [45]. Social networking platforms by offering us a way to find and stay in touch with friends and colleagues, they take intangible elements of our everyday life and transform them into data that can be used to do new things. A really challenging problem in the literature is the characterization of the huge amount of heterogeneous data produced by social information networks [3].

Different approaches are used to model social networks depending to the application, the first models considering only interaction among users. But social networks consist of more than just a record of social network ties. Typically, communities contain additional interaction information that can be used for modeling relation among the users. Indeed, by facilitating communication and transfer of information almost every social network provides infrastructure for the formation and maintenance of communities over time. And naturally the system record those interactions and the created contents.

Successively, for modeling those multimedia content several approaches have been introduced. They can be properly grouped in four main categories.

In the first one, a social media network is defined as a graph composed by heterogeneous vertices, such as users, multimedia objects and tags. The second type of approaches is based on bipartite graphs, it propose a user-content bipartite graph model, composed by users' groups and objects. Tripartite graphs are used by the third category of approaches. A strategy that exploits users, tags and resources

[1] `https://gdelpuente@gitlab.com/gdelpuente/IMM.git`

for clustering purpose is presented. The last group encase the approaches based on hypergraph theory.

The approach we have follow belongs the second category. Starting from some definition will describe it to follow:

The graph is ultimately made up by three variety of entities:

- U - Users - the set of persons and organizations that benefit the specific social community. rather a lot of information about their profile, preferences, interests etc. can be utilized by the proposed model.

- T - Tags - the most meaningful terms or named entities - whose definition can be retrieved from dictionaries, ontologies etc. - of a given domain, or topics, utilized by users to mark down multimedia data and keywords, labels, tags, comments and so on, derived from the analysis of textual information such as

- M - Multimedia Objects The set of multimedia resources (i.e. images, video, audio, etc.) that within a social media networks community are shared. High level (metadata) and low level information (features) can be correctly used in the model.

$$V = U \cup T \cup M$$

Define P as $(U, T, M) \subseteq P$, the sets of all possible relationship between the defined vertices is $E \subseteq P^n$ where n=2, $\cdots$, M So, we can have relationships between two or more users, tags, images or mixed relationships always with two or more elements. In this way, we have the same power of representation of the third category but keeping slim model. Now we can define our bipartite graph as G (V, E)

Chapter 3

Bio inspired

The primary source of inspiration for humans is Nature. This holds true also for the developing of alternate algorithms for solving many real-world optimization problems. The beauty of nature inspired algorithms stands in the fact that they receive their sole inspiration from nature. Most of the nature inspired algorithms are constructed on some successful features of biological system. More in particular, these algorithms are founded on the selection of the fittest in biological systems which have evolved by natural selection over millions of years. In the group of bio-inspired algorithms, a special category of algorithms has been developed by taking inspiration from swarm intelligence. These are the most in vogue [5].

As defined by Bonabeau the swarm intelligence is "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies" [6].

Swarm intelligence (SI) relate to the collective, the behavior of multiple become visible, interacting agents who run some easy rules. While each agent may be considered as brainless, the entire system of multiple agents may present some self-organization behavior and therefore can act like some sort of collective intelligence. Based on the collective comportment of social insects like ants, bees, termites, and wasps, further from flocks of birds or fish or other animal societies. all SI-based

algorithms make use of multi-agents. In Table 3.1 a list of swarm intelligence algorithms is provided [24].

Therefore, to reach a swarm intelligent behavior there are two core concepts, that are necessary a sufficient: self-organization and division of labor. Self-organization can be defined as a set of dynamic procedures, which outcome in architecture, at the aggregate level, of a system by means of its low-level components that interact each other. The rules that drive each agent make use of only local information in the absence of cognition of the global pattern. The four ingredients on which the self-organization based on, described by Bonabeau are: Positive feedback, negative feedback, fluctuations and multiple interactions [6].

- Positive feedback (amplification) is simple behavioral empirical rules that contribute to the formation of structures. Activities like recruitment and reinforcement can be examples of this property.
- Negative feedback compensates for positive feedback and helps to stabilize the collective pattern. In order to avoid the saturation, exhaustion, or competition which might occur.
- Fluctuations an example can be: random walks, errors, random task switching among swarm individuals. Those activities are vital for creativity and innovation, can act as seeds from which structures nucleate and grow. Randomness is not a drawback but is often pivotal for emergent structures since it allows the finding of new solutions.
- Generally, self-organization lean on multiple interactions. Furthermore, is necessary a minimal density of mutually tolerant individuals, that should be able exploit of the results of their own activities as well as of others' activities.

The division of labor is the phenomenon in which different tasks are per-

formed simultaneously by specialized individuals. The latter is considered to be more efficient than the sequential task performance by unspecialized individuals [20] [73] [32] [51]. Division of labor also allows the swarm to respond to changed conditions in the search space.

Millonas defined some principles that must be satisfied by the swarm in order to call it intelligent: [46]

- The proximity principle: The swarm should be able to do simple space and time computations.
- The quality principle: The swarm should be able to respond to quality factors in the environment.
- The principle of diverse response: The swarm should not commit its activities along excessively narrow channels.
- The stability principle: The swarm should not change its mode of behavior upon every fluctuation of the environment.
- The adaptability principle: The swarm must be able to change behavior mode when needed.

SI-based algorithms are among the most common and largely used. One of the cases for such popularity, among the many, is that SI-based algorithms usually dispense information among multiple agents, so that self-organization, co-evolution and learning through the iterations may help to supply the high efficiency of most SI-based algorithms. Another reason is that can be parallelized easily, distributing multiple agent so that, from the implementation point of view, large-scale optimization becomes more practical and is easily mapped to the MapReduce model.

In 1999 the development of ant algorithms for discrete optimization is been a memorable part in swarm intelligence domain [85]. This was used for many problems: transportation [71], water distribution [44], disaster relief operations [94].

Table 3.1: Bio-Inspired Algorithms

Algorithm	**Author**	**ref**
Accelerated PSO	Yang et al.	[90], [88]
Ant colony optimization	Dorigo	[22]
Artificial bee colony	Karaboga and Basturk	[36]
Bacterial foraging	Passino	[53]
Bacterial-GA Foraging	Chen et al.	[12]
Bat algorithm	Yang	[89]
Bee colony optimization	Teodorovict and Dell'Orco	[72]
Bee system	Lucic and Teodorovic	[43]
BeeHive	Wedde et al.	[80]
Wolf search	Tang et al.	[68]
Bees algorithms	Pham et al.	[54]
Bees swarm optimization	Drias et al.	[23]
Bumblebees	Comellas and Martinez	[18]
Cat swarm	Chu et al.	[16]
Consultant-guided search	Iordache	[30]
Cuckoo search	Yang and Deb	[91]
Eagle strategy	Yang and Deb	[92]
Fast bacterial swarming algorithm	Chu et al.	[17]
Firefly algorithm	Yang	[87]
Fish swarm/school	Li et al.	[41]
Good lattice swarm optimization	Su et al.	[67]
Glowworm swarm optimization	Krishnanand and Ghose	[39], [40]
Hierarchical swarm model	Chen et al.	[11]
Krill Herd	Gandomi and Alavi	[26]
Monkey search	Mucherino and Seref	[47]
Particle swarm algorithm	Kennedy and Eberhart	[31]
Virtual ant algorithm	Yang	[93]
Virtual bees	Yang	[86]
Weightless Swarm Algorithm	Ting et al.	[74]

An adaptation of ant colony optimization algorithm done by Rivero et al. to use it for path search in social networks [58].

An ACO algorithm have as principal element a parameterized probabilistic model, which is called the pheromone model. Using this model are generated solution to the problem at hand. A solution is set up by a finite set of elements. At run-time, the pheromone values are updated by the ACO algorithm using above generated solutions. The update target is to focus the search within areas of the search space containing high-quality solutions [85].

Algorithm 6 ACO's algorithm

```
1: procedure ACO-InfluenceMaximization
2:     InitializePheromoneValue()
3:     while termination condition is not met do
4:         GenerateSolution()
5:         EvaluateSolution()
6:         UpdatePheromoneValue()
7:     Return best solution
```

Yang et al. [85]propose Application of the Ant Colony Optimization Algorithm to theInfluence-Maximization Problem, the operation of which is described below.

At the start the pheromone value of all node is very small $\epsilon \neq 0$. For each node is generated a possible solution picking a node from the neighborhood and then from the neighbors of the neighborhood, and so on. The selection can be totally random or giving to each node a probability p of being selected. Subsequently the influence of the solution is evaluated, consisting to the numbers of nodes expected at the end of the diffusion process. Only the top-m solutions, ordered by influence value, release the pheromone. So, the value of pheromone is updated with the value of influence of the solution. If a node is an element of more solutions the values are summed.

Then begin the three phases of the iterative process. In the GenerateSolutions() function, new set of nodes are composed by the artificial ants. K random number uniformly distributed in [0,1] are generated. By using each of these num-

bers, via the following rule are selected the nodes of the new set.

$$X(m,n) = \begin{cases} arg_{u \in V-r} max\{[\tau(u)]^{\alpha} * [\eta(u)]^{\beta}\} & if q \leq q_0 \\ S & otherwise \end{cases}$$

where q is the generated random number, q0 is a parameter ($0 \leq$ q0 ≤ 1), τ is the pheromone value, η is the heuristic value, and S is a random variable selected according to the probability distribution given by

$$p(u) = \frac{[\tau(u)]^{\alpha} * [\eta(u)]^{\beta}}{\sum [\tau(u)]^{\alpha} * [\eta(u)]^{\beta}}$$

Parameter q0 determines the relative importance of exploitation versus exploration. For determining the heuristic values of nodes can be used many different methods depending to the aim of the study, an example can be the degree of centrality. The EvaluateSolutions() function is then used to measure the quality of each new set. Analogous to the InitializePheromoneValue() function, UpdatePheromoneValue() according to

$$\tau(u) = (1-\rho) * \tau(u) + \sum_{k=1}^{m} \tau(u)^k$$

the pheromone is updated on all nodes. The ρ parameter is the evaporation rate and is implemented to avoid the algorithm converging too rapidly toward a suboptimal region. Zhang et al. too proposed a bio-inspired algorithm to find the most influential users of Sina Weibo, the popular microblogging service in China, Particle Swarm Optimization (PSO) [96], their algorithm utilizes social interaction pattern to find an optimal solution.

Chapter 4

Artificial Bee Colony

Among many intelligent swarms the algorithms related to the bee SI are the one which has been most widely studied and applied to solve the real-world problems. The survey prepared by Karaboga and Akay [35] and updated in 2014 [37], shows that the studies are mainly based on the dance and communication, task allocation, collective decision, mating, marriage, reproduction, nest site selection, foraging, pheromone and floral laying and navigation behaviors of the swarm.

Some known algorithms based on bee SI and their applications are: Yang proposed Virtual bee algorithm to solve the numerical function optimizations. BeeHive algorithm, was presented by Wedde et al. (2004) [80]). It has been inspired by the communication in the hive of honey bees, and applied to the routing in networks. In the same application area described by by Wedde and Farooq (2005) [79], BeeAdHoc algorithm, is a routing algorithm for energy efficient routing in mobile ad-hoc networks, also for the ride-matching problem, for the routing and wavelength assignment in all-optical networks, Bee colony optimization was presented by Teodorovic and Dellorco (2005) [72].Abbass (2001) [1] propose a unified model for the marriage in honey-bees within an optimization context. The model simulates the evolution of honey-bees starting with a solitary colony (single queen without a family) to the emergence of a eusocial colony (one or more queens with a

family). The bees algorithm was proposed by Pham et al. (2005) [54] and imitates the foraging behavior of honey bees. Bee system was defined for solving difficult combinatorial optimization problems by Lucic and Teodorovic (2001) [43].

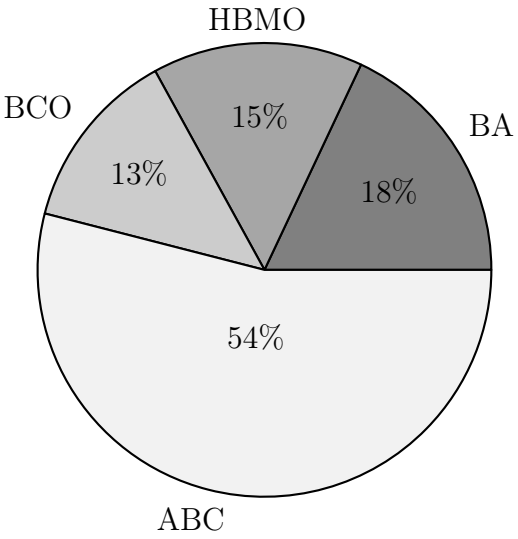


Figure 4.1: Percentages of publications regarding to some algorithms based on bee swarms

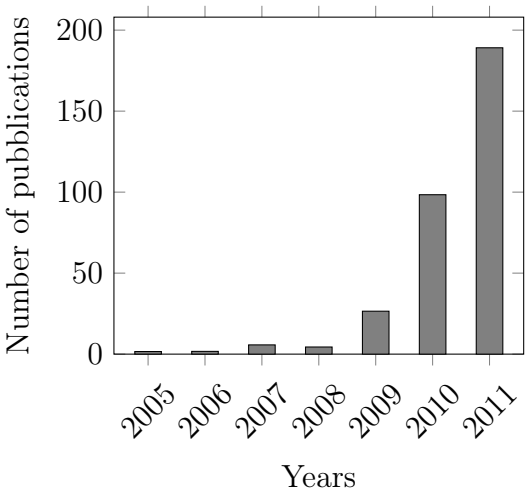


Figure 4.2: Number of pubblications through years

The Artificial Bee Colony (ABC) algorithm inspired by the natural foraging behavior of a honey bee, pursue a new meta-heuristic method, to discover the optimal food resources via a waggle dance, it was introduced by Karaboga et al. in 2007 [33] [36] [34]. In the group of the algorithms mentioned above, ABC is the one which has been most widely studied and applied to solve the real-world problems, so far. More than half (58%) of the publications related to bee swarm intelligence with respect to the algorithms, belongs to ABC.

4.1 Foraging behavior of honey bees

A colony of honey bees continually scour the territory searching for new flower patches, this problem leads to the emergence of collective intelligence of honey bee

swarms. The minimal model of forage selection includes three crucial elements: food sources, employed foragers and unemployed foragers. The two principal ways of the comportment defined by the model are: the recruitment to a rich nectar source and the abandonment of a poor source.

Food Sources is characterized by many factors such as its closeness to the nest, its richness or concentration of its energy, and the ease of pull out this energy. Foragers bees summarize those numerous variables in a single quantity that is "nectar source profitability" [62]

The honey bees are indistinguishable in shape and size but we can classify these bees in terms of their working pattern or responsibility during the process they perform.

The bees those are currently working on a food source may called as 'employer bees', employed foragers. The employer bees also bring the details about the sources and after going back to the hive they share the information with other bees standing in the hive.

Unemployed foragers can be divided in two types: scouts, searching the environment surrounding the nest for new food sources and onlookers waiting in the nest and establishing a food source through the details shared by employed foragers. The mean number of scouts averaged over conditions is about 5-10% [62]

The most significant occurrence in the training of the swarm knowledge is the exchange of information among bees

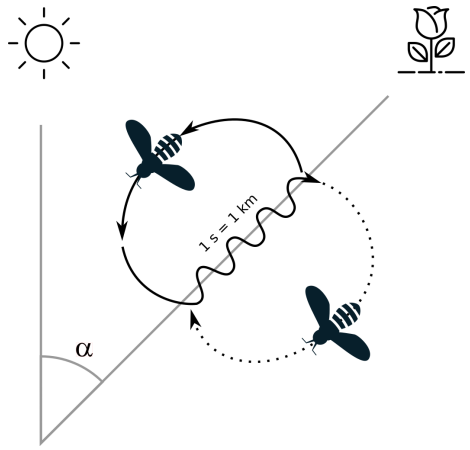
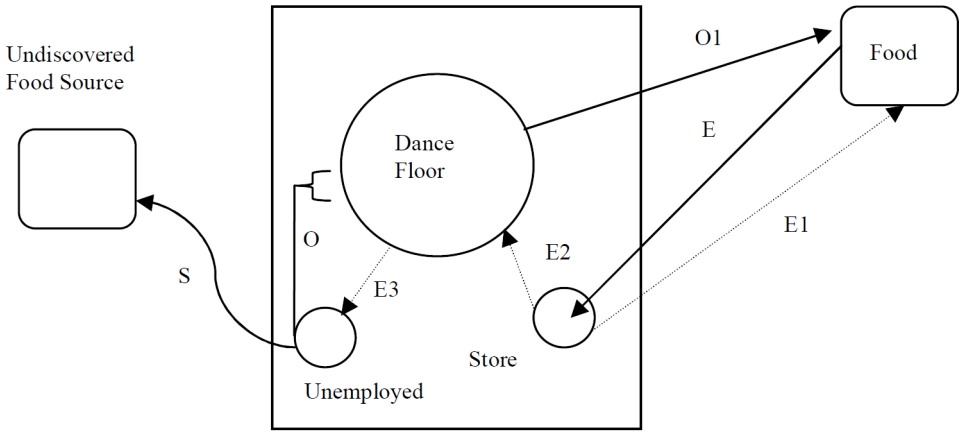


Figure 4.3: The bee waggle dance communicating information

While examining the entire hive it is possible to distinguish between some parts that commonly exist in all hives. With respect to exchanging information the most principal part of the hive is the dancing area. To spread the information an employer, make use of a peculiar technique known as 'waggle dance'. The waggle dance is a physical motion of the bees and is crucial for the foragers as it exhibit some important details about the food sources. Indeed, the duration of the dance is proportional to the scout evaluation of the food source profitability. Since details about all the discovered flower patches are accessible to an onlooker on the dance floor, the most accredited theory is that, she observes numerous dances and choose to employ herself at the most profitable source. More bees get recruited to harvest the best-rated food sources.

To help the comprehension of the basic of foraging process let's look at the figure.



Let examine an employer bee currently utilizing a food source. After his going back to the nest across E he might have three options to select. Path E1 that bring him to keep continue to exploiting, path E2 if he chooses to spread the details by dancing and E3 if abandonment of the flower patch has done. Besides, an onlooker expects in the nest and watch (position O) the waggle dance to get recruited. The path S indicates the random examination of undiscovered food source and explored by scout bees.

The four properties defined in the Section 3 on which self-organization is based in the case of honey bees foraging behavior can be expressed as follows:

- Positive feedback: The exploitation process of rich food sources is increased by bees.
- Negative feedback: As the nectar amount of a food source reduce, the number of onlookers visiting it decrease proportionally.
- Fluctuations: The scouts carry out a random search process for discovering new food sources.
- Multiple interactions: Employed bees spread their details about flowers patches with their hive buddy (onlookers) expecting on the dance area.

So, we can agree that the Millonas's principles are fully satisfied [46].

4.2 Algorithmic structure of ABC

The basic ABC algorithm like the minimal model can be separated into three part: Employed bee phase, Onlooker bee phase and Scout bee phase. Initially, all flower patches positions are located by scout bees. Afterwards, employed bees and onlooker bees took advantages of the nectar of food sources, and this repeated exploitation will finally bring them to become consumed. Hence, the employed bee which has exhausted the food resource becomes a scout bee, which is again in search of food sources. In ABC, the location of a flower patch act for a possible solution to the problem and the nectar amount (profitability) of a food source correlates with the quality (fitness) of the related solution. In the basic form, the number of employed bees is the same of the number of flower patches (solutions) considering that each employed bee is occupied with one and only one food source. The rough algorithmic skeleton of the ABC optimization approach is indicated as follows:

Algorithm 7 ABC's rough algorithmic skeleton

```
1: procedure BASICABC
2:     Initialization Phase
3:     repeat
4:         Employed Bees Phase
5:         Onlooker Bees Phase
6:         Scout Bees Phase
7:         Memorize the best solution achieved so far
8:     until Cycle = Maximum Cycle Number
```

In the initialization phase, control parameters are set and the ensemble of food sources (solutions) is initialized by artificial scout bees.

In the employed bees phase, artificial employed bees look for new food sources having more profitability inside the neighborhood of the food source which knows at the moment. They discover a neighbor food source and then calculate its quality. Next, a greedy choice is applied among its fitness and the old ones. Whereupon, employed bees provide their food source details with onlooker bees expecting in

the nest through the waggle dance on the dance floor.

In the onlooker bees phase, artificial onlooker bees, starting from the information shared by the employed bees, through a predefined law select their food sources. When a food source for an onlooker bee is picked, the sources in the neighborhood are considered, and the relative fitness values calculated. In the same manner of the employed bees phase, a greedy selection is applied.

In the scout bees phase, solutions of employed bees that cannot be enhanced through a number of trials established in advance, are abandoned and bees become scouts. After, randomly the scouts begin to find new solutions. So, the negative feedback behavior arises to balance the positive feedback sources which are poor are abandoned.

These three steps are iterated until a termination criteria is satisfied, for instance a maximum cycle number

4.2.1 Versions and expansion of the ABC optimization algorithm

Artificial bee colony algorithm was extended by Singh in 2009 for solving a constrained optimization problem in minimum spanning tree [65]. In 2011, Pan et al. proposed a discrete version of ABC for the lot-streaming flow shop scheduling problem [52]. A generic model based on the ABC for multi-objective design optimization in the same year was described by Omkar et al. [50]. Akay and Karaboga presented a modified versions of ABC algorithm to apply them for solving real-parameter optimization problems efficiently [2]. To improve the global search potential of basic ABC Wu et al. submit an enhanced ABC algorithm [83]. the application for constrained optimization problems led to a new upgrade of the ABC algorithm [8]. A military use to path planning of uninhabited combat air vehicle (UCAV) in various combat fields was described by Xu et al., a chaotic

ABC approach [84]. To promote the convergence rate Zhang et al. modified ABC algorithm by editing the steps of employed and onlooker bees [97].

A new approach is presented by Sankar et al. in 2016 [60], for modelling of influence diffusion in viral campaigns inside the online micro-blogging platform Twitter for pick out a set of seed nodes which lead to maximum propagation of influence. However, there are hardly few bioinspired algorithms dealing with approximation problems in social network scenarios, the influence maximization problem in particular.

4.3 ABC Algorithm for influence Maximization

We identify the analogy between individuals interacting on social media network and bees perform waggle dance in a bee colony. The dynamic personalities in social media influence their neighbors to a particular idea, subject.

We can use in this way the ABC algorithm to inspect the network to discover the key influential vertices though waggle dance. Each node in the given social context figured as food source. The employer bees, who operate to detect influential opinion leaders in the network, are at first allocated to top k vertices from the pre-estimated node ranking method. The scout bees worked to analyze the nearest neighbor vertices of employer bees for improved solutions. The onlooker bees represent the followers of influential opinion leaders. During the propagation process, they are marked with the status influenced. In each loop the local fitness value is evaluated by the maximum number of unique nodes that are influenced by a single node. The global fitness value is maximum unique influenced vertices by a set of k vertices.

4.3.1 Proposed Method

We have adopted the IC model of diffusion and the algorithm 9 was developed for calculating fitness on the bipartite graph. The function work either for a single node either for a set of nodes in input. The fitt value represent the numbers of hop in the net where the spread operates. In this process, if a node is already influenced by another then it is avoided.

Input to Algorithm 8 is the network G = (V, E) with node reputation rank value r_i of each node i and a positive integer k indicating the dimension of the influential set another integer $Nsout$ that represent the minimum number of bee scout for the scout phase. The output is a subset $A \in V$ with $|A| = k$ such that influence A is maximum. The proposed algorithm start initializing the subset $E \in V$ of employer bees with nodes with top k ranks as the initial solution. Evaluate the local fitness value of each bee in E by counting the nodes that can be reached within $fitt$ steps.

The set $S \in V$ of scout bees is initialized with the nearest neighbor nodes of initial employer bees. But if the neighborhood is empty or the number of scouts is less then $Nscout$, then are added others scout picked randomly in the network.

Now we lead the local search for optimal solutions to the problem by finding the neighboring nodes in each loop. An iterative process then starts, with choosing a scout bee with the highest reputation rank from S. If the local fitness value of a scout bee is greater than that of an employer bee then the employer bee is substituted with the respective scout bee. The substitution operation updates the onlooker and scout bee mood of the bees. The iterative process ends when some termination condition is met, such as exceeding the execution time limit or a certain ratio of the vertices being influenced. The result, which is the best k node set for influence maximization, is then returned. The final set of employer bees is identified as the set of k influential nodes in the network which can maximize

diffusion in the given context

Table 4.1: Used notation

Notation	Description
RankedGraph	A bipartite graph with ranked vertices *:G(V,E)*
k	The number of the most influential nodes
fitt	The number of step for fitting evaluation
nScout	the minimum number of scouts to be observed for each cycle
employers	The employers bees vertices
fitEmployers	The fittnes value of each employers bees vertices
neighborhood	The vertices reachable in a single step
scouts	The scouts bees vertices
fitSc	The fittnes value of scout bee vertex

Algorithm 8 ABC's algorithm

1: **procedure** ABC($RankedGraph, k, fitt, Nscout$)
2: $employers \leftarrow$ top k rank nodes
3: (***fitEmployers***, ***neighborhood***) $\leftarrow fitness(RankedGraph,$ ***employers***$, fitt)$
4: **while** termination condition is not met **do**
5: ***scouts*** $\leftarrow$ ***neighborhood***
6: **if** ***scouts*** is Empty $\vee$ scout numbers $<$ nScout **then**
7: take random scouts to reach the number nScout
8: **for** each scout sc in ***scouts*** **do**
9: ***fitSc*** $\leftarrow$ fitness($RankedGraph, sc, fitt$)
10: **if** ***fitSc*** $>$ fitness on any e in fitEmployers **then**
11: promote sc as an employer
12: remove the e with the minor fitness
13: update ***scouts***
14: return the ***employers***

Table 4.2: Used notation

Notation	Description
FIT	the number of steps for fitting evaluation
nodes	The nodes reached at each step
Vertices	The vertices of which have to calculate the fittnes
result	The fittnes of each vertex of ***Vertices***

Algorithm 9 Fitness algorithm

```
1: procedure FITTNES(RankedGraph, Vertices, fitt)
2:     FIT = fitt * 2                    ▷ for bipartite graph 1 step correspond 2 step
3:     nodes(0) = Vertices
4:     for i ← 1 to FIT do
5:         calculate neighborhood of nodes(i − 1)
6:         result = count number of vertices reached by Vertices
7:         nodes(i) = neighborhood
8:     return (result,nodes(1))
```

An implementation of this algorithm, others variants and utils functions, in Scala code has been developed and can be found on our gitlab repository [1]

4.3.2 Example of algorithm execution

Starting from an hypergraph of knowledge we represent it with the bipartite graph. Each hyperedge is been transformed to a node and each node included is connected to this one.

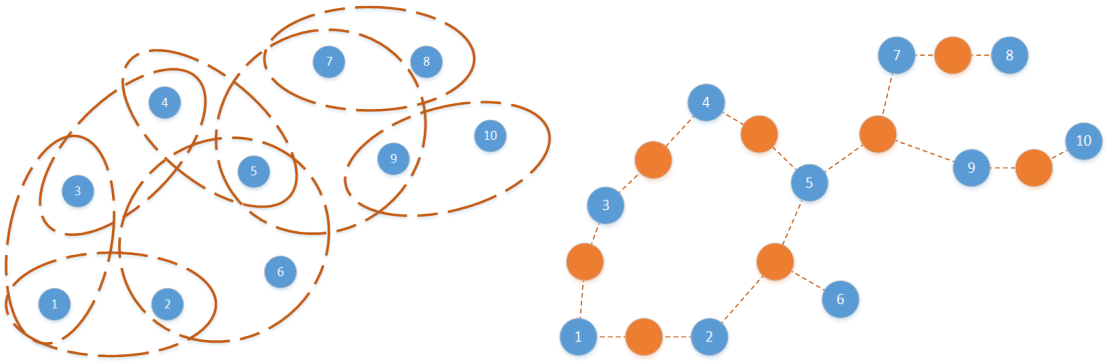


Figure 4.4: Hypergraph to Bipartite graph

To start the algorithm is necessary to do a ranking of the nodes of the hypergraph.

[1] https://gdelpuente@gitlab.com/gdelpuente/influence.git

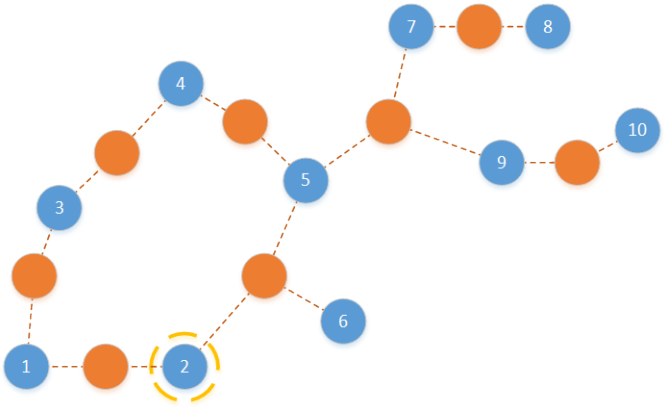


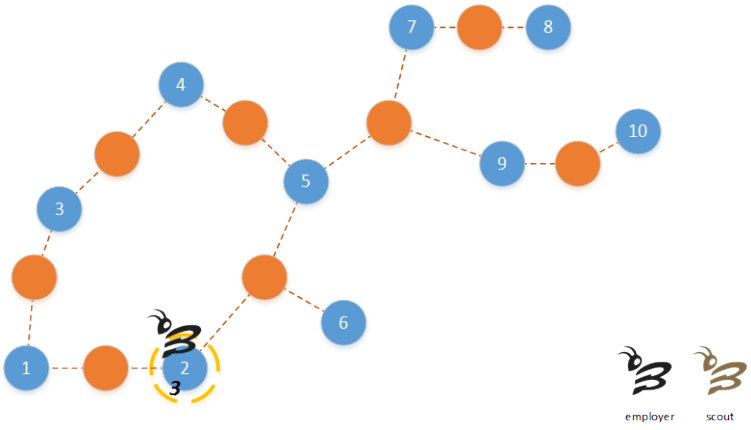
Figure 4.5: figure
top ranked node

id	rank
2	1,335
9	1,010
5	1,010
7	0,982
8	0,980
3	0,961
1	0,958
4	0,618
10	0,618
6	0,545

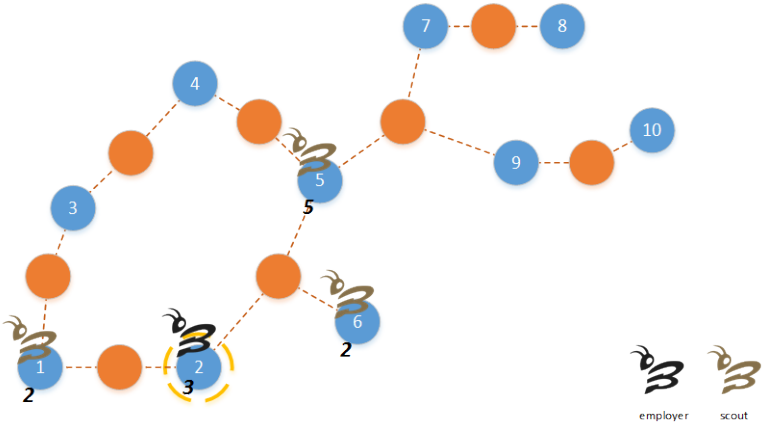
Figure 4.6: table
Ranking results

The ranking results are an example to allow a good representation of the algorithm

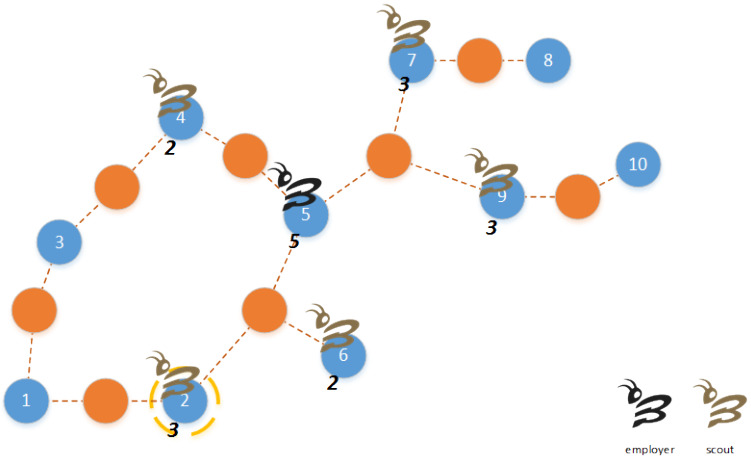
The proposed algorithm accept this input and initialize subset E $\subset$ V of employer bees with nodes with top k ranks as the initial solution. Evaluate the fitness value of each bee in E using the algorithm 9. This latter count the nodes that can be reached within $fitt$ steps.



The set S $\subset$ V of scout bees is initialized with the nearest neighbors nodes of initial employer bees. An iterative process then starts, if a scout bee has a fitness value greater then each employers bee, the status of the bee is updated to employer and the employer is downgraded to an outlooker.



The neighborhood is recalculated and next loop begin until condition is not met.



Chapter 5

Experimentation

5.1 Dataset

Our experimentation is focused on the significance of the parameters in input to the algorithm and the net topology against execution time. To do these, two main types of net have been artificially constructed: the star network and the complete network. These two have been chosen because they represent the two extremes based on the complexity of a network topology.

In **Star topology**, every node is connected to a central node. All traffic that traverses the network passes through the central vertex. The star topology is considered the easiest topology to design and implement. An advantage of the star topology is the simplicity of adding additional nodes. The primary disadvantage of the star topology is that the center node represents a single point of failure.

In the mathematical field of graph theory, a **complete graph** is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge. A complete digraph is a directed graph in which every pair of distinct vertices is connected by a pair of unique edges (one in each direction). It represents the most complex topology, everyone is connected with everyone. On the other hand, the neighborhood of each node is composed of all the nodes except it itself.

These two topologies seen in bipartite graphical optics are transformed by adding the nodes that characterize the network links. In the figure5.1 can be seen examples of this transformation for a star network

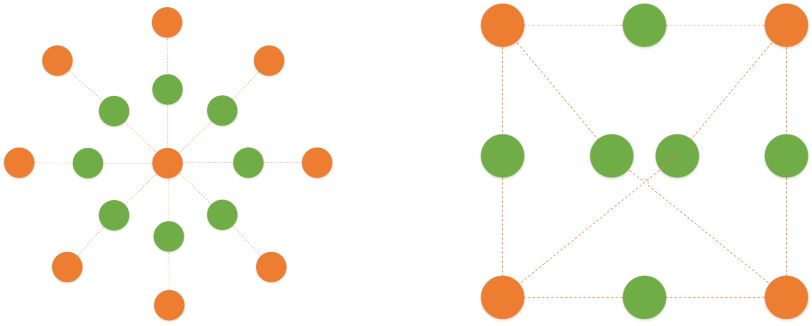


Figure 5.1: Topologies seen in bipartite graphical optics

We conducted the experiments on a PC with Windows 10, with Intel Core i5-6400 2.70GHz CPU and 4GB memory. All algorithms were implemented in Scala and compiled using scala-compiler v2.11.8, running on Spark 2.1.1.

5.2 Parameter Settings

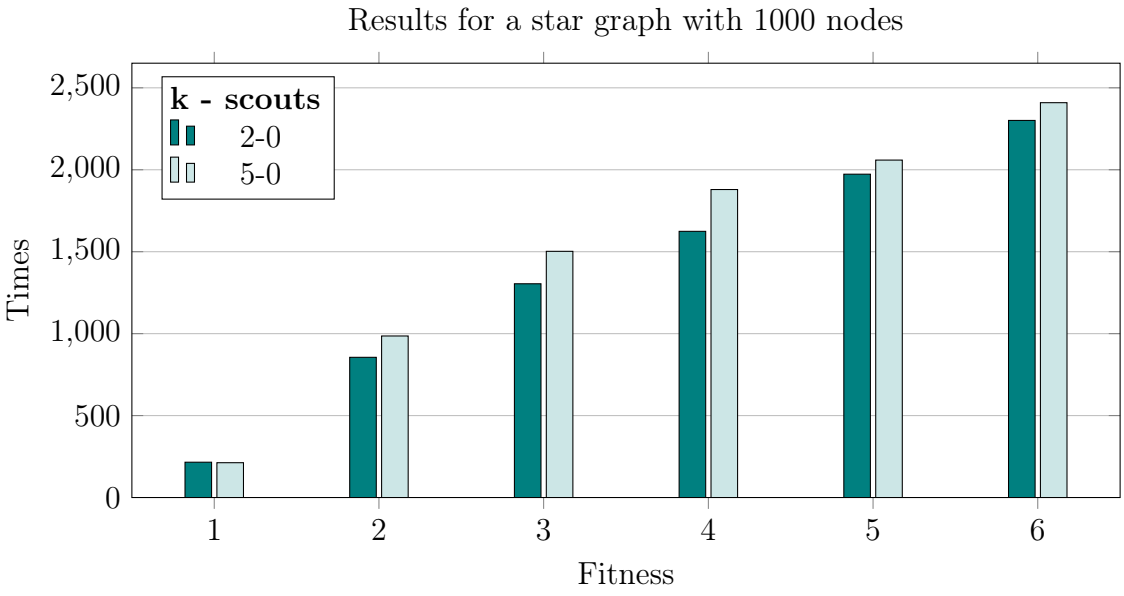
For the ranking of the generated network has been used the PageRank algorithm provided by Scala. The tolerance is set at 0.001 with the default reset probability at 0.15. We just launched an iteration of the ABC algorithm, as this parameter was not taken into account for the observations. In our experimentation, we varied the input parameters and the network topology, more specifically:

- Star graph, Complete graph
- Numbers of nodes in the networks 10 - 100 - 1000
- Minimum number of scouts 0 - 2 - 5 - 10
- Size of the seed set to find 2 - 5
- Number of step for fitting evaluation 1 - 2 - 3 - 4 - 5 - 6

In each of our experiments, we run each method 30 times and report the average measurements. The complete details of the measurements can be found in the sheet on our gitlab repository [1]

5.3 Results

The experimentation has been going on with increasing complexity of the experiments and has stopped at the saturation of the computer's capacity in use. For this reason the results with 1000 nodes that we can observe for the star network topology are:



For the complete graph topology just this observation:

topology	nodes	scouts	k	fitt	times(s)
complete	1000	0	2	1	15175,89

Table 5.1: Results for a complete graph with 1000 nodes

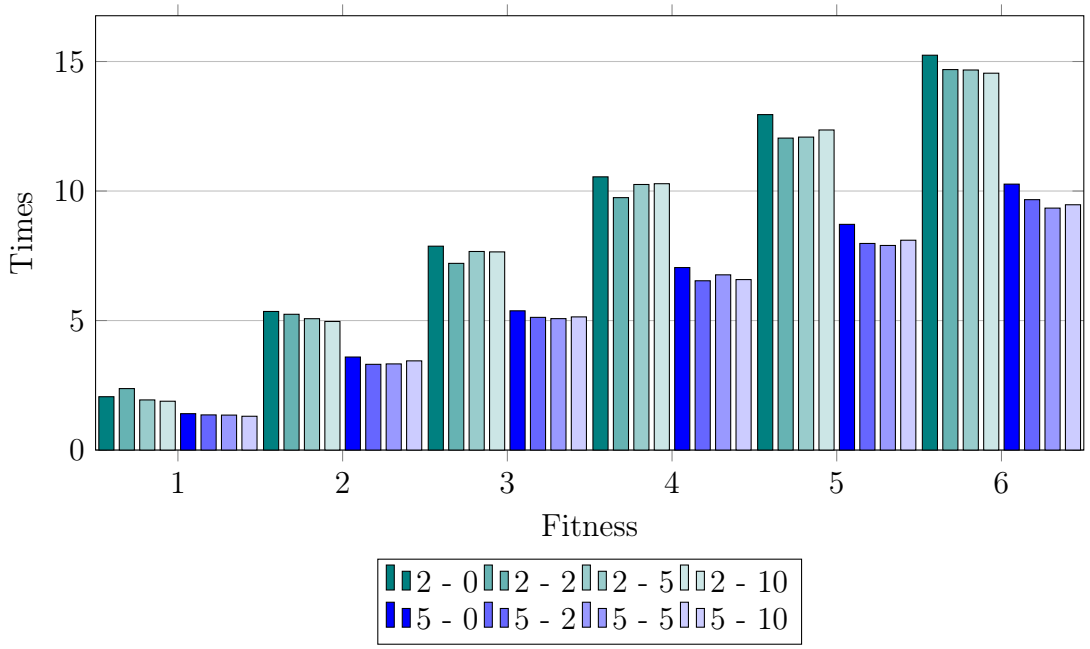
By this we can see that the growth of the *fitt* value to determine longer execution times, which is due to the greater exploration of the graph. The times

[1] `https://gdelpuente@gitlab.com/gdelpuente/influence.git`

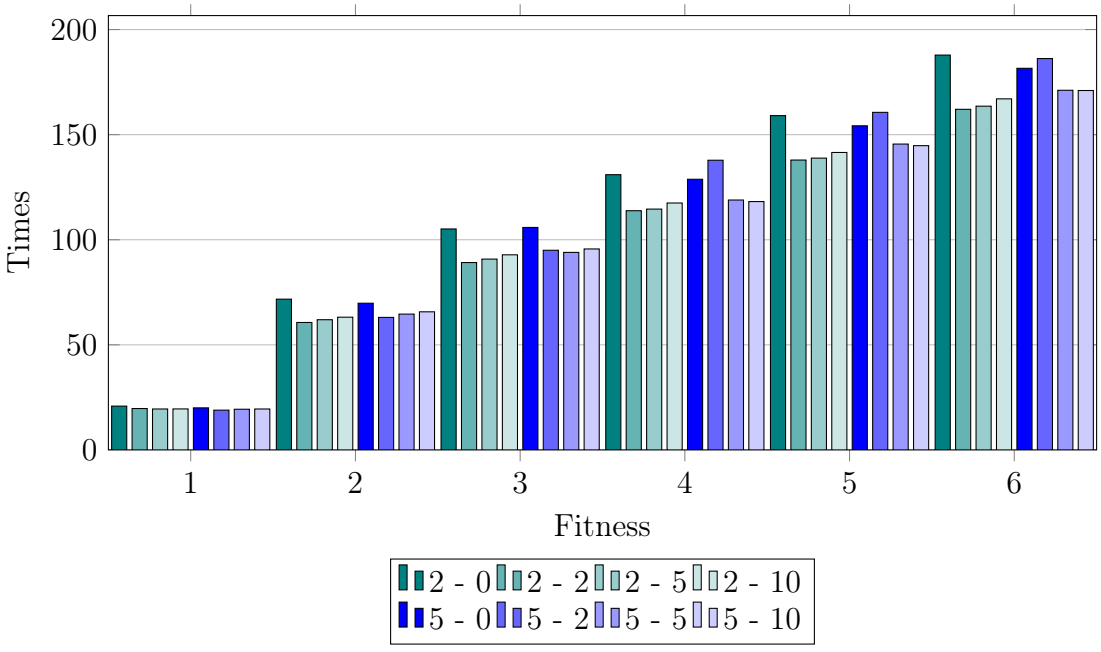
are already in the order of hours, but we must remember that the Scala language and all the rest of the architecture perform better on clusters. In fact, on a single machine, the only possible parallelization is between the core of the processor.

For a graph with 10 and 100 nodes We managed to run all experiments. First, we observe the variation of k and scout parameters in a star graph.

Results for a star graph with 10 nodes $k - scouts$ variation

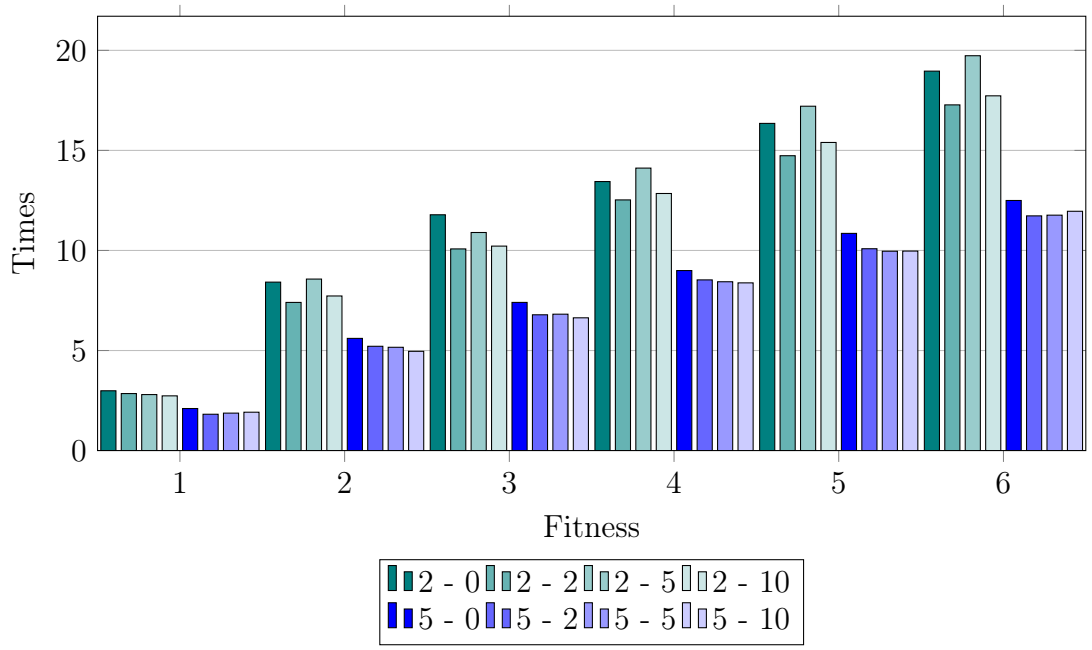


Results for a star graph with 100 nodes $k - scouts$ variation

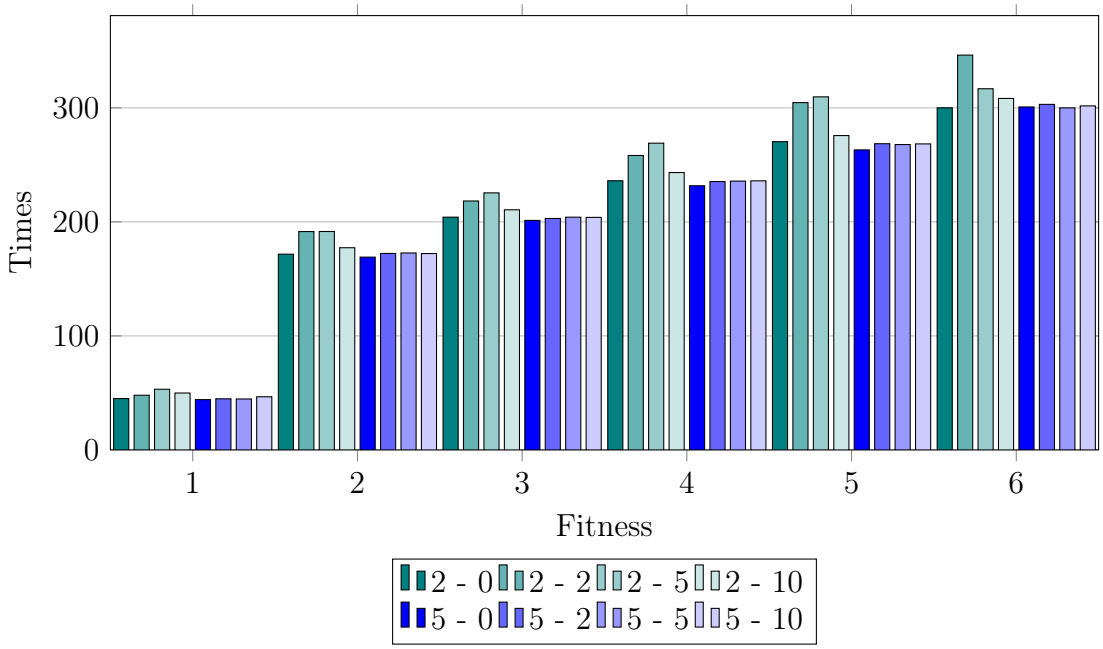


The variation of the *scout* parameter does not lead to significant variations in the duration of the execution, but for small graphs the growth of the parameter k improves the performance. This does not apply when the size of the graph increases.

Results for a complete graph with 10 nodes $k - scouts$ variation



Results for a complete graph with 100 nodes $k - scouts$ variation

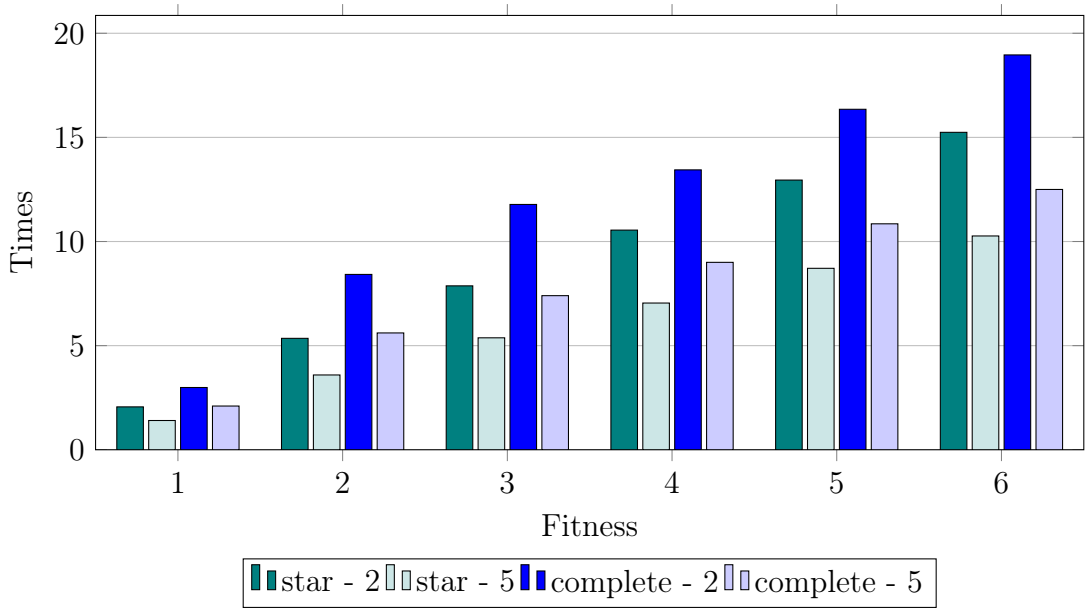


The same can be said for a complete graph, but moreover it can be obscured

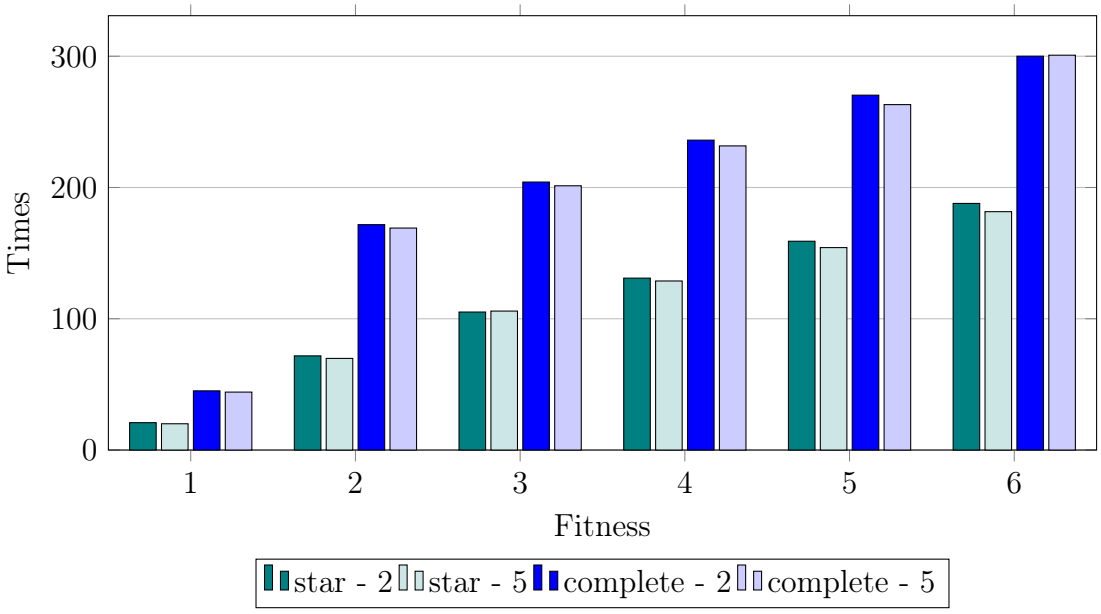
that the scout parameter introduces more randomness with a lower k value. The growth of parameter $fitt$ is always very influential, but for a complete graph with a high number of nodes after a big variation from 1 to 2 there is a gentle growth, so it seems to stabilize.

Now we locked the scout value at 0 to compare the result for the two topologies.

Results for a graph with 10 nodes 0 scouts $topology - k$ variation



Results for a graph with 100 nodes 0 scouts $topology - k$ variation



As we can see, the previous evaluation about parameter k is confirmed. In

addition, the complexity of the graph will in any case lead to a longer running time. In fact, for a complex graph the times are in any case longer.

Conclusion

In this thesis we have described the information society we live in. We are flooded with data that become new material to create new knowledge. One of the methods to enhance knowledge with a wide field of interest is the Social Network Analysis, the study of the graph generated by human relationships. Our work refers to this area of study. As seen in the previous chapters we have defined a bipartite-based model with which you can represent users, tags, multimedia, and ties between them.

Using this generic model any multimedia social network can be studied. In particular, as seen, we focused on one of the most discussed issues in Social Network Analysis, the Influence Maximization: the problem of selecting a small subset of nodes that could maximize the influence propagation.

Addressing this problem with a bio inspired approach brings considerable advantages over methods in literature. In fact, these algorithms by definition have qualities such as information distribution or the ease of process parallelization. This can be noticed by the implementation of the algorithm that is able to use in many cases the potential of MapReduce by avoiding cycles and also language-provided (Scala) operators for message exchange between vertices. Moreover, from the data obtained, it is noteworthy that input parameters with large networks will have less impact on the running time.

This experimentation is a good starting point for new studies in this direction. The algorithm can be tailored to the specific need or can be verticalized the model on

the analyzed social network. This way you can answer new questions. The question becomes so, embedded in topology.

Therefore, by paraphrasing Frank Lloyd Wright - Nature is the inspiration for all the optimization.

Bibliography

[1] Hussein A Abbass. Mbo: Marriage in honey bees optimization-a haplometrosis polygynous swarming approach. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 207–214. IEEE, 2001.

[2] Bahriye Akay and Dervis Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, 192:120–142, 2012.

[3] Flora Amato, Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperlì. Multimedia social network modeling: A proposal. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, 2016.

[4] Mark A Beyer and Douglas Laney. The importance of big data: a definition. *Stamford, CT: Gartner*, pages 2014–2018, 2012.

[5] S Binitha, S Siva Sathya, et al. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, 2(2):137–151, 2012.

[6] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[7] Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[8] Ivona Brajevic and Milan Tuba. An upgraded artificial bee colony (abc) algorithm for constrained optimization problems. *Journal of Intelligent Manufacturing*, pages 1–12, 2013.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[10] Edward Capriolo, Dean Wampler, and Jason Rutherglen. *Programming hive*. " O'Reilly Media, Inc.", 2012.

[11] Hanning Chen, Yunlong Zhu, Kunyuan Hu, and Xiaoxian He. Hierarchical swarm model: a new approach to optimization. *Discrete Dynamics in Nature and Society*, 2010, 2010.

[12] Tai-Chen Chen, Pei-Wei Tsai, Shu-Chuan Chu, and Jeng-Shyang Pan. A novel optimization approach: bacterial-ga foraging. In *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, pages 391–391. IEEE, 2007.

[13] Wei Chen, Laks VS Lakshmanan, and Carlos Castillo. Information and influence propagation in social networks. *Synthesis Lectures on Data Management*, 5(4):1–177, 2013.

[14] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 88–97. IEEE, 2010.

[15] Nicholas A Christakis and James H Fowler. The spread of obesity in a large social network over 32 years. *New England journal of medicine*, 357(4):370–379, 2007.

[16] Shu-Chuan Chu, Pei-Wei Tsai, and Jeng-Shyang Pan. Cat swarm optimization. In *Pacific Rim International Conference on Artificial Intelligence*, pages 854–858. Springer, 2006.

[17] Ying Chu, Hua Mi, Huilian Liao, Zhen Ji, and QH Wu. A fast bacterial swarming algorithm for high-dimensional function optimization. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3135–3140. IEEE, 2008.

[18] Francesc Comellas and Jesus Martinez-Navarro. Bumblebees: a multiagent combinatorial optimization algorithm inspired by social insect behaviour. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 811–814. ACM, 2009.

[19] World Wide Web Consortium et al. Internet live stats. *Retrieved June*, 10, 2016.

[20] Leandro Nunes De Castro and Fernando José Von Zuben. Artificial immune systems: Part i–basic theory and applications. *Universidade Estadual de Campinas, Dezembro de, Tech. Rep*, 210(1), 1999.

[21] Peter J Denning. The science of computing: Saving all the bits. *American Scientist*, 78(5):402–405, 1990.

[22] Marco Dorigo. Optimization, learning and natural algorithms. *Italian PhD dissertationPolitecnico di MilanoMilan*, 1992.

[23] Habiba Drias, Souhila Sadeg, and Safa Yahi. Cooperative bees swarm for solving the maximum weighted satisfiability problem. *Computational intelligence and bioinspired systems*, pages 417–448, 2005.

[24] Iztok Fister Jr, Xin-She Yang, Iztok Fister, Janez Brest, and Dušan Fister. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*, 2013.

[25] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[26] Amir Hossein Gandomi and Amir Hossein Alavi. Krill herd: a new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 17(12):4831–4845, 2012.

[27] Jennifer Golbeck. *Analyzing the social web*. Newnes, 2013.

[28] Apache Hadoop. Hadoop map-reduce tutorial, 2013.

[29] Oliver Hugo and Elizabeth Garnsey. The emergence of electronic messaging and the growth of four entrepreneurial entrants. *New Technology Based Firms in the New Millenium*, 2:97–123, 2001.

[30] Serban Iordache. Consultant-guided search: a new metaheuristic for combinatorial optimization problems. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 225–232. ACM, 2010.

[31] Kennedy James. Russell c.: Eberhart particle swarm optimization. *Neural Netw*, 4:1942–1948, 1995.

[32] Robert L Jeanne. The evolution of the organization of work in social insects. *Monitore Zoologico Italiano-Italian Journal of Zoology*, 20(2):119–133, 1986.

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

[34] Dervis Karaboga and Bahriye Akay. A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*, 214(1):108–132, 2009.

[35] Dervis Karaboga and Bahriye Akay. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, 31(1-4):61–85, 2009.

[36] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[37] Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.

[38] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[39] KN Krishnanand and Debasish Ghose. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In *Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 84–91. IEEE, 2005.

[40] KN Krishnanand and Debasish Ghose. Glowworm swarm optimisation: a new method for optimising multi-modal functions. *International Journal of Computational Intelligence Studies*, 1(1):93–119, 2009.

[41] Xiao-lei Li, Zhi-jiang Shao, Ji-xin Qian, et al. An optimizing method based on autonomous animats: fish-swarm algorithm. *System Engineering Theory and Practice*, 22(11):32–38, 2002.

[42] David Loshin. *Big data analytics: from strategic planning to enterprise integration with tools, techniques, NoSQL, and graph*. Elsevier, 2013.

[43] Panta Lucic and Dusan Teodorovic. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In *Preprints of the TRISTAN IV triennial symposium on transportation analysis*, pages 441–445, 2001.

[44] Holger R Maier, Angus R Simpson, Aaron C Zecchin, Wai Kuan Foong, Kuang Yeow Phang, Hsin Yeow Seah, and Chan Lim Tan. Ant colony optimization for design of water distribution systems. *Journal of water resources planning and management*, 129(3):200–209, 2003.

[45] V. Mayer-Schönberger and K. Cukier. *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. An Eamon Dolan book. Houghton Mifflin Harcourt, 2013.

[46] Mark M Millonas. Swarms, phase transitions, and collective intelligence. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 17, pages 417–417. ADDISON-WESLEY PUBLISHING CO, 1994.

[47] Antonio Mucherino, Onur Seref, Onur Seref, O Erhun Kundakcioglu, and Panos Pardalos. Monkey search: a novel metaheuristic search for global optimization. In *AIP conference proceedings*, volume 953, pages 162–173. AIP, 2007.

[48] Arun C Murthy, Vinod Kumar Vavilapalli, Doug Eadline, Joseph Niemiec, and Jeffrey Markham. *Apache Hadoop YARN: moving beyond MapReduce and batch processing with Apache Hadoop 2*. Pearson Education, 2013.

[49] Martin Odersky, Philippe Altherr, Vincent Cremet, Burak Emir, Stphane Micheloud, Nikolay Mihaylov, Michel Schinz, Erik Stenman, and Matthias Zenger. The scala language specification, 2004.

[50] SN Omkar, J Senthilnath, Rahul Khandelwal, G Narayana Naik, and S Gopalakrishnan. Artificial bee colony (abc) for multi-objective design optimization of composite structures. *Applied Soft Computing*, 11(1):489–499, 2011.

[51] George F Oster and Edward O Wilson. *Caste and ecology in the social insects*. Princeton University Press, 1978.

[52] Quan-Ke Pan, M Fatih Tasgetiren, Ponnuthurai N Suganthan, and Tay Jin Chua. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information sciences*, 181(12):2455–2468, 2011.

[53] Kevin M Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems*, 22(3):52–67, 2002.

[54] DT Pham, A Ghanbarzadeh, E Koc, S Otri, S Rahim, and M Zaidi. The bees algorithm-technical report. *Cardiff: Manufacturing Engineering Centre, Cardiff University*, 2005.

[55] Ioannis Pitas. *Graph-based social media analysis*. Chapman and Hall/CRC, 2015.

[56] Gil Press. A very short history of big data. *Forbes Tech Magazine, May*, 9, 2013.

[57] Chiara Pulice. *Social networks: Influence maximization and efficient graph management*. PhD thesis, Università della Calabria, 2015. unpublished thesis.

[58] Jessica Rivero, Dolores Cuadra, Javier Calle, and Pedro Isasi. Using the aco algorithm for path searches in social networks. *Applied Intelligence*, 36(4):899–917, 2012.

[59] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[60] C Prem Sankar, S Asharaf, and K Satheesh Kumar. Learning from bees: An approach for influence maximization on viral campaigns. *PloS one*, 11(12):e0168125, 2016.

[61] John R Seeley. The net of reciprocal influence. a problem in treating sociometric data. *Canadian Journal of Experimental Psychology*, 3:234, 1949.

[62] TD Seeley. The wisdom of the hive cambridge, 1995.

[63] Toby Segaran. *Programming collective intelligence: building smart web 2.0 applications*. " O'Reilly Media, Inc.", 2007.

[64] S Sicular. Gartners big data definition consists of three parts, not to be confused with three v s. forbes, 2013.

[65] Alok Singh. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, 9(2):625–631, 2009.

[66] Jean-Marc Spaggiari and Kevin O'Dell. *Architecting HBase Applications: A Guidebook for Successful Development and Design*. " O'Reilly Media, Inc.", 2016.

[67] Shoubao Su, Jiwen Wang, Wangkang Fan, and Xibing Yin. Good lattice swarm algorithm for constrained engineering design optimization. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 6421–6424. IEEE, 2007.

[68] Rui Tang, Simon Fong, Xin-She Yang, and Suash Deb. Wolf search algorithm with ephemeral memory. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pages 165–172. IEEE, 2012.

[69] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1539–1554. ACM, 2015.

[70] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 75–86. ACM, 2014.

[71] Dusan Teodorovic. Transport modeling by multi-agent systems: a swarm intelligence approach. *Transportation planning and Technology*, 26(4):289–312, 2003.

[72] Dušan Teodorovic and Mauro DellOrco. Bee colony optimization–a cooperative learning approach to complex transportation problems. *Advanced OR and AI methods in transportation*, pages 51–60, 2005.

[73] Valery Tereshko. Reaction-diffusion model of a honeybee colonys foraging behaviour. In *International Conference on Parallel Problem Solving from Nature*, pages 807–816. Springer, 2000.

[74] TO Ting, Ka Lok Man, Sheng-Uei Guan, Mohamed Nayel, and Kaiyu Wan. Weightless swarm algorithm (wsa) for dynamic optimization problems. In *IFIP International Conference on Network and Parallel Computing*, pages 508–515. Springer, 2012.

[75] Jeffrey Travers and Stanley Milgram. The small world problem. *Phychology Today*, 1:61–67, 1967.

[76] Maksim Tsvetovat and Alexander Kouznetsov. *Social Network Analysis for Startups: Finding connections on the social web.* " O'Reilly Media, Inc.", 2011.

[77] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.

[78] Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545, 2012.

[79] Horst F Wedde, Muddassar Farooq, Thorsten Pannenbaecker, Bjoern Vogel, Christian Mueller, Johannes Meth, and Rene Jeruschkat. Beeadhoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 153–160. ACM, 2005.

[80] Horst F Wedde, Muddassar Farooq, and Yue Zhang. Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 83–94. Springer, 2004.

[81] Tom White. Hadoop-the definitive guide: Storage and analysis at internet scale (revised and updated)., 2012.

[82] David Williams. *Probability with martingales*. Cambridge university press, 1991.

[83] Xin Jie Wu, Duo Hao, and Chao Xu. An improved method of artificial bee colony algorithm. In *Applied Mechanics and Materials*, volume 101, pages 315–319. Trans Tech Publ, 2012.

[84] Chunfang Xu, Haibin Duan, and Fang Liu. Chaotic artificial bee colony approach to uninhabited combat air vehicle (ucav) path planning. *Aerospace Science and Technology*, 14(8):535–541, 2010.

[85] Wan-Shiou Yang, Shi-Xin Weng, C Guestrin, C Faloutsos, J VanBriesen, and N Glance. Application of the ant colony optimization algorithm to the influence-maximization problem. *International Journal of Swarm Intelligence and Evolutionary Computation*, 1(1), 2012.

[86] Xin-She Yang. Engineering optimizations via nature-inspired virtual bee algorithms. *Artificial intelligence and knowledge engineering applications: a bioinspired approach*, pages 317–323, 2005.

[87] Xin-She Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2(2):78–84, 2010.

[88] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[89] Xin-She Yang. A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74, 2010.

[90] Xin-She Yang and Nature-Inspired Metaheuristic Algorithms. Luniver press. *Beckington, UK*, 2008.

[91] Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.

[92] Xin-She Yang and Suash Deb. Eagle strategy using lévy walk and firefly algorithms for stochastic optimization. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 101–111, 2010.

[93] Xin-She Yang, Janet M Lees, and Chris T Morley. Application of virtual ant algorithms in the optimization of cfrp shear strengthened precracked structures. In *International Conference on Computational Science*, pages 834–837. Springer, 2006.

[94] Wei Yi and Arun Kumar. Ant colony optimization for disaster relief operations. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):660–672, 2007.

[95] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[96] Biao Zhang, Shuai Zhong, Kunmei Wen, Ruixuan Li, and Xiwu Gu. Finding high-influence microblog users with an improved pso algorithm. *International Journal of Modelling, Identification and Control*, 18(4):349–356, 2013.

[97] Xiu Zhang, Xin Zhang, SL Ho, and WN Fu. A modification of artificial bee colony algorithm applied to loudspeaker design problem. *IEEE Transactions on Magnetics*, 50(2):737–740, 2014.