The evolution of search algorithms over time

Renāts Jurševskis
Delft University of Technology
Delft, Netherlands
R.Jursevskis@student.tudelft.nl

Tudor Octavian Pocola
Delft University of Technology
Delft, Netherlands
T.O.Pocola@student.tudelft.nl

***Abstract*—The field of search algorithms has changed drastically in the past few decades. Modern search algorithms are solving completely different problems than just twenty years ago. This paper analyzes the factors that caused this shift and determines which innovations could affect the field in the future. To achieve this, some of the best-known search algorithms and techniques are compared and analyzed. The innovations in other industries have had a large impact on search algorithms by introducing new problems and expanding the scope of the field. Most notably, modern search engines have brought attention to the problem of optimally sorting search results and have inspired to apply innovations from various other fields like AI and big data. In addition, the future of search algorithms looks very encouraging, as artificial intelligence and quantum computing promise to improve the field even further. Therefore, we can conclude that instead of being caused by the exhaustion of classical methods, the evolution of search algorithms has originated from advancements in various other industries.**

***Index Terms*—search, search algorithm, search engine, pattern matching, search tree, informed search, uninformed search**

I. Introduction

We, as humans, have had the need for efficiently searching through an archive ever since we started keeping records in ancient Egypt. Think about how you would store hundreds of records in order to have easy access to any of them later. These records need to be stored and indexed to allow for efficiently searching through them. This problem was inherited by our computers when we started moving records onto them.

The field of search algorithms is a well-researched field with an extensive history [1] that started in mathematics but has grown rapidly since the invention of computers. It started modestly, with simple algorithms such as linear and binary search, but it grew swiftly and soon included such algorithms as depth-first search, iterative deepening search, steepest hill climbing, and A*.

This all changed with the advent of the internet when search algorithms became much more integrated into people's daily lives. So much so, that in today's world, most people think about a search engine when they hear the word "search", as this is the most relevant use of search algorithms at the moment. It is no wonder then, that the field has become much more relevant today than it ever was. This can be observed by looking at the interest tech giants like Google, Amazon and Microsoft are taking in the field.

Google was launched in 1998, so one would expect that they should have already found an optimal solution but, if we take a look at the history of search algorithms used in Google Search [3], we can clearly see that the algorithms that are being used are changing constantly. However, if we take a look at some of the recent updates, we can clearly see that the problem these algorithms are trying to solve strays further and further away from the original problem - that of finding records in an archive:

- "Hummingbird" - lays the groundwork for voice-enabled search;
- "RankBrain" - machine learning algorithm that can make guesses about unknown words, and replace them;
- "BERT" (Bidirectional Encoder Representations from Transformers) - a neural network for natural language processing that can figure out the context of a word by looking at the previous and following words.

Furthermore, it is hard to call these algorithms "search algorithms", since they solve a totally different task (speech recognition, natural language processing, etc.). So one might wonder if this shift in new algorithms might reflect the fact that we have exhausted all classical methods of search algorithms and we are now forced to look in directions other than pure algorithms or rather, the fact that new technological possibilities are opening new directions for this field.

The main goal of this paper is to analyze the extent to which we have exhausted the search algorithms available to us and determine which factors have contributed to the evolution of the field. To achieve this, we need to address the following questions:

1) Which important breakthroughs have modeled the field?
2) In what ways have search engines affected the development of search algorithms?
3) What developments can we expect in the future?

Each following section will tackle one of the aforementioned questions. Section "III. Classical search algorithms" will answer the first question by giving an overview of all the classical approaches of search algorithms that are already well defined in computer science. Section "IV. Modern Search Engines" will answer the second question by presenting the new search problems caused by the increasing popularity of the internet and how search engines have managed to solve them. Section "V. Future of Search" will answer the third and final question by presenting the extent to which there are new search algorithms to be discovered.

However, before answering these questions, we will first take a look at the materials and methodology used in our research.

TABLE I
A COMPARISON BETWEEN VARIOUS GRAPH SEARCH ALGORITHMS USING BIG-O NOTATION

Algorithm	Time Complexity	Space Complexity	Completeness	Optimality
Breadth-First Search	$O(\lvert V\rvert + \lvert E\rvert)$	$O(\lvert V\rvert)$	+	+
Depth-First Search	$O(\lvert V\rvert + \lvert E\rvert)$	$O(h)$	-	-
Iterative Deepening Depth-First Search	$O(b^d)$	$O(d)$	+	-
Uniform Cost Search	$O(b^{1+\frac{d}{\epsilon}})$	$O(\lvert V\rvert)$	+	+
Greedy Best First Search	$O(b^h)$	$O(b^h)$	-	-
Hill Climbing	N / A*	$O(1)$	-	-

The notation used: $|V|$ - the number of vertices in the tree, $|E|$ - the number of edges in the tree, h - maximum depth of the tree, b - branching factor, d - depth of the shallowest solution, ϵ - minimum cost of a node

**The notion of time complexity is not applicable to hill climbing, as it can be run for an infinite amount of time*

II. Materials and Methods

In order to perform the literature review, it was necessary to research the topic by finding relevant sources. To perform this search, we used the TU Delft Library, Google Scholar, and such bibliographic databases as Scopus and Mendeley.

Search terms like "Search Algorithm", "Graph Search", and "Pattern Matching" were used in order to get a general overview of the topic. Then, to get a deeper understanding of the algorithms used by search engines, we searched for articles using search terms "PageRank", "Topic-sensitive search", "Search personalization", "Mobile-friendly search", "Google Hummingbird". Finally, we explored the future of search algorithms by using such search terms as "Future of Search", "AI Search Algorithm", "Image search", and "Quantum Search Algorithm".

Initially, a list of candidate sources was created by searching for the aforementioned terms. After that, the list was reviewed in order to select the best and most relevant sources by analyzing the title and the abstract of the source. In order to choose reputable sources, we attempted to mainly use peer-reviewed articles. However, when no such sources could be found, book chapters and technical reports were used. In addition, we primarily chose sources that were cited in other articles. Sources that contained obvious inaccuracies or many spelling errors were excluded.

III. Classical search algorithms

As our computer memory grew larger and larger, we started storing more and more data on it, so the time it took to run a query on the data naturally increased, so, the need for more efficient search algorithms appeared. The way this was achieved at the beginning (and still is achieved today for some cases), was by representing the data as a search tree, where different pieces of information are modeled as nodes, and the relations between these pieces are modeled as edges [27].

One such tree is depicted in Fig. 1, where the start node is labeled with an S and the target node (i.e. the node containing the piece of information we want to retrieve) with T. The problem now becomes that of traversing the search tree efficiently using various algorithms. From the very beginning, scientists have identified two possible types of solutions to this problem: uninformed search and informed search [14].

Uninformed search is a brute force method that blindly considers all options or paths without knowing or assuming anything about the data. Uninformed searching techniques include algorithms such as Depth-First and Breadth-First Search, Iterative Deepening, Bidirectional Search, Uniform Cost Search but also many others.

On the other hand, informed search (sometimes called heuristic search) makes assumptions about the data using heuristics to model a heuristic function that gives some insight into the current state of the algorithm. This heuristic function can be used to choose the optimal path (as opposed to traversing all paths blindly). Informed search techniques include algorithms such as (steepest) hill climbing, A*, AO*, and Greedy Best First Algorithm, but not limited to.

Another category of search algorithms that exists is pattern matching. This class consists of algorithms for finding a pattern (usually text) inside a given sequence. The pattern-match algorithms are most commonly used to find a substring within another string. Some of the best-known pattern matching algorithms are Naive Pattern Matching, Knuth-Morris-Pratt algorithm, Rabin-Karp algorithm, Finite Automata, and Aho-Corasick.

In the following subsections, we will take a look at the following algorithms with the goal of understanding them better:

- *A. Uninformed search algorithms* - Breadth-First Search, Depth-First Search, Iterative Deepening Depth-first Search, and Uniform Cost Search;
- *B. Informed search algorithms* - Greedy Best First Search and Hill Climbing;
- *C. Pattern matching algorithms* - Naive Pattern Search, Knuth-Morris-Pratt and Aho-Corasick;

Note that we will not analyze the algorithms in-depth, as extensive research is already available [2], [14], [26], [27]. However, we include a quick comparison in Table I that mentions their time and space complexity, and also if the algorithm is complete (does it guarantee the correct answer) or optimal.

A. Uninformed Search Algorithms

*1) **Breadth-First Search**:* Holdsworth states that: "Breadth-First Search (BFS) is one of the oldest and most fundamental

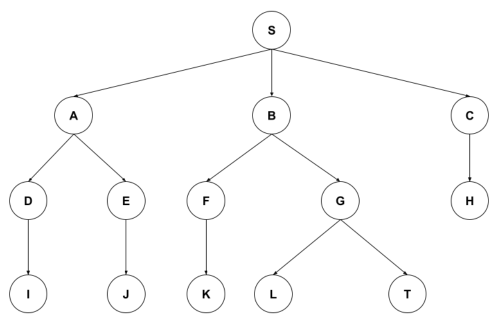

Fig. 1. A representation of a search tree, where node S denotes the starting node, and node T denotes the target node.

graph traversal algorithms, influencing many other graph algorithms" [15, p. 1]. It should not come as a surprise that BFS can also be used to search through a search tree.

BFS starts by adding the start node (root) into a first in first out (FIFO) queue and marking it as visited. Then, as long as the queue is not empty, the algorithm removes the first element in the queue, adds all its unvisited neighbors in the queue, and will mark them as visited. This will result in the whole traversal of the tree, where the nodes are traversed in the order of their level [23]. The order in which we traverse the search tree in figure 1 is S, A, B, C, D, E, F, G, H, I, J, K, L, T. As the name suggests, we traverse the tree in its width first. The time complexity of this algorithm is linear, as we visit each node at most once.

*2) **Depth-First Search**:* The only difference between BFS and Depth-First Search (DFS) is that DFS uses a first in last out (FILO) queue. This results in a different tree traversal, one in which we first traverse a path until the leaf node (the leaf nodes in figure 1 are: I, J, K, L, T, and H) before starting with another path. The order in which we traverse the search tree in figure 1 is S, A, D, I, E, J, B, F, K, G, L, T, C, H. As can be seen, we traverse the tree from top to bottom, instead of left to right like in BFS. The time complexity of this algorithm is also linear since we need to visit every node at most once.

These two techniques (BFS and DFS) blindly traverse the tree, without making any assumptions about the data. This is great if we do not know for sure if the desired piece of information is contained in our search graph. DFS always follows the first path it finds until the end, so it finds nodes that are not in the first sub-tree very late. BFS goes level by level but requires more space complexity for the queue. The space complexity of DFS is the maximum height of the tree and that of BFS is the total number of nodes (in the worse case, they are equal).

*3) **Iterative Deepening Depth-First search**:* Iterative deepening search (IDS), or more formally iterative deepening depth-first search (IDDFS) is a hybridization of the 2 algorithms discussed above, combining the space-efficiency of DFS and the early traversal of nodes close to the root of BFS. This algorithm works well on graphs that have infinite depth, but a finite branching factor (the number of child nodes per node). In addition, IDDFS is usually used on search trees that have a high branching factor (nodes have a lot of children, so the next level has much more nodes than the previous level).

The idea behind IDDFS is to set a maximum depth for DFS and not let it run beyond this given depth. If the goal node is not found, all the calculations are discarded and the maximum depth is increased. The order in which the nodes are visited is that of DFS, but the order in which the nodes are first visited is that of BFS. The order in which traverse the search tree in figure 1 is the following: S, S, A, B, C, S, A, D, E, B, F, G, C, H, S, A, D, I, E, J, B, F, K, G, L, T, C, H. As can be seen, the main drawback of IDDFS is that it repeats all the work that is done in a previous phase [30].

*4) **Uniform Cost Search**:* Uniform Cost Search (UCS) is an iteration of Dijkstra's algorithm that is used for very large (possibly infinite) graphs that can not be represented in memory. In UCS, we start by adding just the start node into the set of open nodes. This set contains the node as the key and its cost as the value. For each subsequent cycle, we expand the least expensive node. Furthermore, we do not insert duplicates into the queue, instead we decrease the value for a specific key if it is necessary (the new value is smaller than the old one). [29].

B. Informed Search Algorithms

*1) **Greedy Best First Search**:* Both DFS and BFS explore all possible nodes at any given state. The main idea behind Greedy Best First Search (GBFS) is to use a heuristic function to expand just the most desirable paths. The desirability of a path is measured using the heuristic function. The GBFS keeps track of two sets: closed nodes and open nodes. The set of closed nodes consists of all the nodes that have already been expanded, while the set of open nodes contains all the nodes that still need expansion. At every step of the algorithm, the node with the best value (based on the heuristic function) is chosen from the open set to be expanded. The algorithm finishes after expanding the target node. The heuristic function differs from case to case, so the order in which we traverse the tree differs based on the heuristic function that is used [26], [27].

*2) **Hill Climbing**:* Hill Climbing is a special kind of heuristic search used for optimizing mathematical problems, that would be otherwise impossible to solve. The main idea behind this algorithm is to try to find the global extreme of a function as fast as possible. This is achieved by setting an initial step. It is most often used in artificial intelligence. In the case of a search tree, Hill Climbing starts by expanding the start node first, and then it keeps expanding just the best node of the previously expanded nodes (modeled by a heuristic function). The Hill Climbing algorithm is very fast, but it can get stuck at a local extreme instead of finding the global one [26], [27].

C. Pattern Matching

*1) **Naive Pattern Search**:* Naive Pattern Search, as the name suggests, is the brute force idea that comes to mind when solving a pattern matching problem. That idea is to slide the pattern over the text and check for each position if the pattern matches the text at that given location. This is a basic approach, therefore, it creates a lot of overhead, as we need to compare every character from the text with every character of the pattern. The time complexity of this algorithm is given by the length of the text multiplied by the length of the pattern, as we need to start a comparison from every position in the text, each of which will require a traversal over the pattern. Due to its slow running time, this algorithm is often not used in applications where the text is long, but its simplicity is very appealing when the application needs to deal with only small strings [25].

*2) **Knuth-Morris-Pratt (KMP) algorithm**:* KMP builds on the idea of Naive Pattern Search by using the fact that when we find a mismatch between the pattern and the text, we have already gone over some of the characters in the next comparison, so we can skip some of the previously matched characters. The amount of characters we can skip at any given step is given by a heuristic function. This heuristic function is computed by calculating the longest proper prefix which is also a suffix. This needs to be pre-computed before we start the actual pattern matching as we need the whole heuristic function in order to skip characters. This was the first algorithm that solved the pattern matching problem in linear time. [25].

*3) **Aho-Corasick**:* There might be a case, where we want to find multiple patterns in a text. A real-world example of this would be to search words from a dictionary within any given text. If we use any of the algorithms mentioned above, we would need to run them independently for each pattern. This algorithm tackles this exact problem, as it enables us to search for multiple patterns in the same text efficiently. The main idea of this algorithm is to build a trie (prefix tree), and then to extend this trie to an automaton. Then we use this automaton to check the whole text [23]. The time complexity of this algorithm is given by the length of the text plus the number of characters in the dictionary plus the number of matches.

IV. Modern Search Engines

Due to the rising popularity of the internet, the need for efficient search algorithms has gained a lot of attention. Search engines have become an integral component of the internet, so they have motivated large-scale investments into the research of search algorithm optimization. However, the search algorithms used in search engines are not a direct match to the algorithms discussed before as they usually consist of two separate parts. To complete a search query, we need to not only find which web pages are relevant to the query, but also sort them based on their importance. Therefore, in order to develop efficient search engines, it has been necessary to discover new techniques for ordering web pages.

The backbone of most modern search engines is PageRank – an algorithm for ranking web pages based on their importance. It was initially developed by the founders of Google, L. Page and S. Brin, as a research project at Stanford University and later used as a foundation in the development of the Google search engine [4]. It models the internet as a graph consisting of nodes and links as can be seen in figure 2. Each web page can be thought of as a node, while each hyperlink that leads to another page is modeled as a link. Then, the importance of a page is directly dependent on the importance of other pages that link to it. Therefore, everything that is linked from an important page can also be considered important. This structure can also be compared to the concept of citations in scientific literature [5].

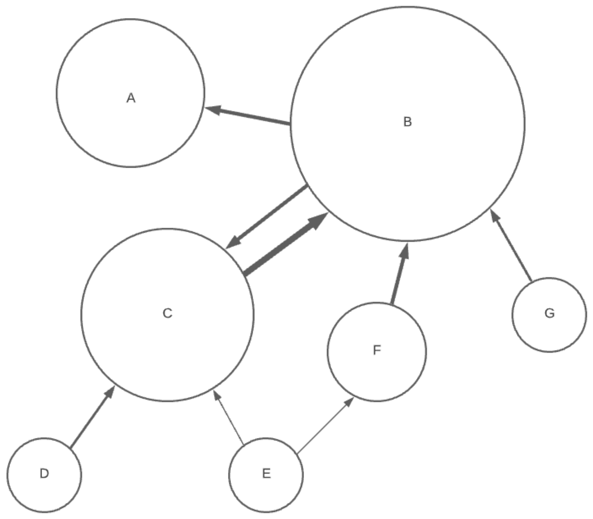

Fig. 2. A simplistic visualization of the PageRank graph. The size of the nodes represent the PageRank value of each website while the width of the links denotes the amount of PageRank score transferred to another page.

Once the PageRank scores have been computed, we still need to determine which pages will be included in the search results. A very basic approach directly matches the words given in the query with the content of each web page using pattern matching. This algorithm already works quite well, however, it also has its limitations. By directly looking at the content of the web page, this approach is susceptible to keyword stuffing – a practice of including irrelevant keywords in order to manipulate a website's ranking [6]. In addition, a direct interpretation of the query does not attempt to understand the context of the search.

Ever since the original creation of the PageRank algorithm, research has been focused on extending and improving it [3] [12]. Each modern search engine offers its own algorithmic solutions, however, such new features as topic-sensitive search, platform-specific ranking, and search personalization are almost always applied in order to improve the accuracy of the search results. For example, since the creation of PageRank, the Google search engine has experienced countless updates that incorporate new analytic metrics, reduce the transparency of the search process, and prioritize results based on personal data collected from the user [7]. One of the most significant updates named “Hummingbird” was introduced in 2013. In addition to simply matching the search terms with the title or content of the web page, this algorithmic update now allowed to measure the intent of the user [8]. This change not only improved the user experience by presenting more relevant search results, but also limited the usefulness of keyword stuffing.

The basic concept of PageRank can also be extended to support topic-sensitive ranking in order to increase the accuracy of the search results. Instead of directly finding the highest-ranked pages that contain the words given in the search query, this algorithm rates the importance of a page by calculating a set of scores with respect to various topics [12]. When the search query is made, we calculate the similarities between the query and each topic. Then, by combining the PageRank scores for the most similar topics, the final page ranking can be determined. By using a topic-sensitive approach, we can avoid the issue of highly-ranked web pages being recommended even when their content does not concern the topic of the search query [12]. In addition, this algorithm can be extended even further by memorizing the context of the search - information surrounding the search query. For example, the search for the word “Jaguar” could prioritize either animal-related or car-related search results based on the web page from which this query was made [16]. This approach can be considered one of the first forms of web personalization.

During the past few years, a lot of research has been focused on the use cases of personal user data. By analyzing the collected data and creating a user profile, modern search engines offer personalized search results [9]. A search engine using personalization can decide if a user is interested in a specific website based on user location data, search history, web browsing history, demographic data, and multiple other factors [11]. Although the usage of personalized search offers many benefits, it has also lead to growing concerns about the effects of such large-scale personal data collection. In addition, recently some research has been exploring the dangers of the filter bubble effect - the possibility for personalization to isolate people from interacting with a diversity of viewpoints or content [13].

With the rising popularity of mobile devices, search engines have also been forced to adapt to the changing landscape. As the fraction of search queries on mobile devices has grown rapidly, search algorithms have had to adjust based on the differences between various platforms. For mobile users it is extremely important that a web page is mobile-friendly - made to work well with much smaller screen sizes and a touchscreen interface. Therefore, most modern search engines analyze each web page and calculate its mobile-friendliness score that is later used in combination with the rest of the search algorithm to determine the final page ranking [17]. Solutions that calculate a special platform-friendliness score instead of just focusing on the mobile platform have also been proposed.

V. Future of Search

Although there has been extensive research on search algorithms for multiple decades, the search for better algorithms is not over. Many claim that the growth in the popularity of artificial intelligence can bring even more improvements to the industry. In addition, the inevitable arrival of quantum computers also promises to disrupt the current landscape.

While artificial intelligence is already used quite often in understanding the meaning of a search query, it can also be applied to other types of search. One area for future research is image search. Although some services already offer the option to find similar images or search for images matching the search query, current approaches are far from perfect [20].

One of the largest limitations for current implementations of image search is the lack of extensive ontologies. Compared to regular text-based search, images are much more difficult to categorize as a very deep understanding of the semantics of the images is required [21]. Image search is also limited by the difficult process of detecting similarities between two images and gathering information about their content. By utilizing new developments in machine learning and artificial intelligence, image recognition algorithms can be used to determine the contents of the image [22]. However, this field is still developing, so we can expect image search to become more accurate and precise in the near future.

The advancement of quantum computers is seen as very promising in the field of search of algorithms. One of the most famous applications of quantum computers is Grover’s search algorithm [18]. This algorithm is often called “database search” as it can theoretically be applied in order to search an unordered database in $O(\sqrt{n})$ time [19]. However, this interpretation has caused some debate, as this algorithm would require a very large amount of quantum hardware, so it would not be easily achievable.

VI. Conclusion

The goal of this review paper was to research the field of search algorithms and how it has developed over time due to factors like the invention of computers and the increasing popularity of search engines. In addition, this paper also analyzes existing search algorithms and provides an introduction to some of the most promising future innovations that could radically change the field in the near future. Although the paper does contain a broad overview of the topic, it does not go into much detail due to the vast amount of information concerning this topic.

The field of search algorithms originates from mathematics, however, the invention of computers changed it drastically. Initially the term “search algorithms” used to represent only pattern matching algorithms and graph algorithms like Depth-First Search, Breadth-First Search, and A*. However, the advancement in technology has caused a giant shift in the field of search algorithms. With the rising popularity of search engines, most search innovations have come in the form of optimizations for ranking and selecting matching web pages based on some search query. Since the original creation of the PageRank algorithm that can be considered a backbone of modern search engines, it has been extended countless times by such improvements as calculating topic-sensitive PageRank, incorporating search personalization, using AI to increase search accuracy, and adjusting to the increasing popularity of mobile platforms. However, the future of the field seems hopeful as the artificial intelligence and quantum computing industries promise to support new types of search and improve existing algorithms even further.

Since the formation of the search algorithm field, the type of problems that it contains has changed immensely. The advancements in technology have created new opportunities to apply search algorithms. Therefore, similarly to many other fields, the landscape of search algorithms has been shaped by innovations in other industries offering new possibilities rather than the exhaustion of classical methods.

Further research could be done on the major changes in the landscape of search algorithms. By analyzing the conception of search engines, it would be possible to get a deeper understanding of precisely what new problems were created, what initial solutions were proposed, and which of these techniques are still being used today. In addition, more research could also be done on the possible future of search algorithms and how this field might develop if the proposed quantum search algorithms could become a reality.

References

[1] M. Bullynck, “Histories of algorithms: Past, present and future,” *Historia Mathematica*, vol. 43, no. 3, pp. 332–341, 2016.

[2] N. Sultana, S. Paira, S. Chandra, and S. K. Alam, “A brief study and analysis of different searching algorithms”, *Second International Conference on Electrical, Computer and Communication Technologies*, pp. 944-948, 2017.

[3] “Google Algorithm Updates & Changes: A Complete History,” *Search Engine Journal*, 23-Feb-2021. [Online]. Available: https://www.searchenginejournal.com/google-algorithm-history/.

[4] L. Page, S. Brin, R. Motwani, T. Winograd, “The PageRank citation ranking: Bringing order to the web,” Stanford InfoLab, Stanford, United States, Tech. Rep. SIDL-WP-1999-0120, November 1999.

[5] M. Bianchini, M. Gori, and F. Scarselli, “Inside PageRank,” *ACM Transactions on Internet Technology*, vol. 5, no. 1, pp. 92–128, 2005.

[6] “Irrelevant keywords,” *Google*. [Online]. Available: https://developers.google.com/search/docs/advanced/guidelines/irrelevant-keywords.

[7] “8 major Google algorithm updates, explained,” *Search Engine Land*, 14-Oct-2020. [Online]. Available: https://searchengineland.com/8-major-google-algorithm-updates-explained-282627.

[8] C. O. Lin and R. Yazdanifard, “How Google’s New Algorithm, Hummingbird, Promotes Content and Inbound Marketing,” *American Journal of Industrial and Business Management*, vol. 04, no. 01, pp. 51–57, 2014.

[9] E. Bozdag, “Bias in algorithmic filtering and personalization,” *Ethics and Information Technology*, vol. 15, no. 3, pp. 209–227, 2013.

[10] A. Sieg, B. Mobasher, and R. Burke, “Web search personalization with ontological user profiles,” *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007.

[11] A. Hannak, P. Sapiezynski, A. Molavi Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson, “Measuring personalization of web search,” *Proceedings of the 22nd international conference on World Wide Web*, 2013.

[12] T. H. Haveliwala, “Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 784–796, 2003.

[13] T. T. Nguyen, P.-M. Hui, F. M. Harper, L. Terveen, and J. A. Konstan, “Exploring the filter bubble: the effect of using recommender systems on content diversity,” *Proceedings of the 23rd international conference on World wide web*, 2014.

[14] M. J. Pathak, R. L. Patel, Sonal P. Rami, ”Comparative Analysis of Search Algorithms”, *International Journal of Computer Applications*, vol. 179, no. 50, pp. 40-43, June 2018

[15] J. Holdsworth, ”The Nature of Breadth-First Search”, School of Computer Science, Mathematics, and Physics, James Cook University, Australia, Tech. Rep. 99-1, January 1999.

[16] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, “Placing search in context: the concept revisited,” *Proceedings of the tenth international conference on World Wide Web*, 2001.

[17] D. Schubert, “Influence of Mobile-friendly Design to Search Results on Google Search,” *Procedia - Social and Behavioral Sciences*, vol. 220, pp. 424–433, 2016.

[18] L. K. Grover, “A fast quantum mechanical algorithm for database search,” *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996.

[19] A. Ambainis, “Quantum search algorithms,” *ACM SIGACT News*, vol. 35, no. 2, pp. 22–35, 2004.

[20] K. Smelyakov, D. Sandrkin, I. Ruban, M. Vitalii, and Y. Romanenkov, “Search by Image. New Search Engine Service Model,” *International Scientific-Practical Conference Problems of Infocommunications. Science and Technology*, 2018.

[21] L. Zhang and Y. Rui, “Image search—from thousands to billions in 20 years,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1s, pp. 1–20, 2013.

[22] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, “Multi-Label Image Recognition With Graph Convolutional Networks,” *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[23] C. Y. Lee, “An Algorithm for Path Connections and Its Applications” *IRE Transactions on Electronic Computers*, vol. EC-10, issue: 3, pp. 346 - 365, 2019.

[24] A. V. Aho, M. J. Corasick, ”Efficient string matching: an aid to bibliographic search”, *Communications of the ACM*, vol. 18, issue: 6, pp. 333–340, 1975.

[25] D. E. Knuth, J. H. Morris, V. R. Pratt, ”Fast pattern matching in strings” *SIAM Journal on Computing*, vol. 6, no. 2, pp. 323–350, 1977.

[26] C. Wilt, J. Thayer, W. Ruml, ”A Comparison of Greedy Search Algorithms” *Association for the Advancement of Artificial Intelligence*, 2010.

[27] M. Sinthiya, M. Chidambaram, ”A Study on Best First Search” *International Research Journal of Engineering and Technology*, vol. 3, issue: 6, pp. 588-597, 2016.

[28] P. E. Hart, N. J. Nilsson, B. Raphael, ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, issue: 2, pp. 100–107, 1968.

[29] A. Felner, ”Position Paper: Dijkstra’s Algorithm versus Uniform Cost Search or a Case Against Dijkstra’s Algorithm” *Conference Proceedings of the Fourth Annual Symposium on Combinatorial Search*, pp.47-51, 2011.

[30] R.E. Korf, ”Depth-First Iterative-Deepening: An Optimal Admissible Tree Search” *Artificial Intelligence*, vol. 27, issue 1, pp.97-109, 1985.