IntelliAntSearch: Machine Learning Enhanced Reliability and Relevance in Unstructured P2P Networks

Abdulaziz Houtari

Computer Science and Engineering
Michigan State University
East Lansing, USA
houtaria@msu.edu

Deeksha Mohanty

Computer Science and Engineering
Michigan State University
East Lansing, USA
mohant11@msu.edu

Karen Suzue

Computer Science and Engineering
Michigan State University
East Lansing, USA
suzuekar@msu.edu

Abstract— **IntelliAntSearch proposes a new addition to enhancing ant-colony search algorithms within unstructured peer- to-peer (P2P) networks. Building upon the foundations of AntP2PR and DLAntP2P, IntelliAntSearch extends the capabilities of these original algorithms by integrating machine learning techniques. Specifically, our approach leverages Long Short-Term Memory (LSTM) networks to dynamically adjust parameters overlooked in both AntP2PR and its successor, DLAntP2P. By learning from historical data, LSTM networks allow IntelliAntSearch to fine-tune parameters related to pheromone and time-to-live (TTL) updates in real time, thereby optimizing search strategies adaptively. Results from simulations on PeerSIM show that IntelliAntSearch, when adapted on AntP2PR, performs slightly better than the original algorithm in terms of query success rate, but increases the message overhead in the network. Meanwhile, IntelliAntSearch applied to DLAntP2P leads to a greater improvement in success rate while retaining almost the same query count. Overall, IntelliAntSearch was largely able to achieve its goal of enhancing search efficiency, scalability, and adaptability in search for dynamic P2P environments.**

Keywords—unstructured peer-to-peer networks, ant colony, ant search, time-series analysis, Long Short Term Memory (LSTM), TTL (Time to Live)

I. INTRODUCTION AND MOTIVATION

Unstructured peer-to-peer (P2P) networks have gained significant attention in recent years due to their decentralized nature, which enables robust sharing and distribution of resources without the need for centralized servers. Unlike structured P2P networks, which rely on predefined organizational structures for efficient data lookup and routing, unstructured P2P networks offer a more flexible approach, where peers connect directly to one another without strict guidelines. However, this flexibility comes with its own set of challenges and advantages. As unstructured P2P networks expand to encompass thousands to millions of nodes, they are accompanied by regular fluctuations in peer availability and resource distribution. Traditional search algorithms, which include flooding or random-based techniques, are not scalable nor adaptable enough to accommodate such dynamicity or network size. Given this fact, it is crucial to develop novel search techniques that can flexibly adjust to these changes in real time. The ability to adapt allows search algorithms to maintain optimal query efficiency and resource discovery, thereby simultaneously reducing network load while increasing query success rate – two variables that are often thought to be tradeoffs.

Ant colony algorithms are a class of biologically-inspired algorithms that promise to offer robustness and adaptability in changing environments. They are based on the real-life phenomenon of stigmergy and swarms, in which ants follow each other's pheromones to trace the best path to a food source. As the pheromones accumulate, the most efficient and shortest paths are reinforced and become more apparent over time. These principles have been applied to many problems in Computer Science, including search in unstructured P2P algorithms as ant colony search does not require the ants to have global knowledge about the network they are traversing [5]. Many ant colony search algorithms, or ant-based routing methods, have been proposed over the years, including AntSearch [6], SemAnt [5], AntP2PR [1], and DLAntP2P [4], many of which have shown promising results in improving both network traffic and query hits when compared to traditional techniques.

AntP2PR, first proposed by Loukos et al. in 2010, laid the foundation for its successor DLAntP2P, which was proposed by Ahmadi et al. in 2016. Both algorithms were designed to address the challenge of free-riders in the network and to minimize the amount of traffic dedicated to resource discovery. While these algorithms have demonstrated adaptability and achieved relatively high success rates, they rely on static parameters manually set for updating pheromone values of peers and determining the time-to-live of messages. Such static use of parameters could lead to suboptimal routing decisions and reduced query efficiency, especially in dynamic network environments. To harness the overlooked potential for further dynamicity in these algorithms, IntelliAntSearch proposes to integrate machine learning techniques into the original AntP2PR and DLAntP2P algorithms. Specifically, IntelliAntSearch employs Long Short-Term Memory (LSTM) networks, which utilize historical data on query successes and failures to dynamically adapt the aforementioned parameters in response to real-time network conditions. Within the P2P network, neighbor selection and thus the movement of virtual, abstracted "ants" are then drawn on learned patterns from past data, which should lead to enhanced accuracy and efficiency in resource discovery. Using simulations conducted on the platform PeerSIM, our results mostly substantiate this assertion. IntelliAntSearch, when integrated into AntP2PR, leads to a slight improvement in query success rate over the original algorithm. However, this also leads to an increase in messages generated in the network, therefore increasing

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

network load. Meanwhile, we see a greater overall improvement when integrating IntelliAntSearch into DLAntP2P. Our results show that IntelliAntSearch leads to a higher success rate than that of the original algorithm, while still retaining an almost exact query count. Furthermore, this improvement is greater in comparison to that of the IntelliAntSearch-integrated AntP2PR algorithm.

The subsequent sections of this paper are structured as follows. Section II provides an overview of previous ant colony algorithms, including AntP2PR and DLAntP2P, as well as other swarm intelligence-based methods we have surveyed in preparation for this study. Section III describes the implementation details of IntelliAntSearch, which includes background information on the workings of LSTM and the PeerSim platform. Section IV describes the methodology and metrics used for testing and evaluating the algorithm against previous methods such as K-Walker and Flooding. Finally, section V includes the results of our simulations as well as their analysis.

II. Related work

A. Ant Colony Algorithms

Surveying various existing studies, we have explored both biologically inspired and non-biologically inspired approaches to querying in unstructured P2P networks while determining the project's direction. Biologically inspired methods, especially ant colony algorithms, are of particular interest as they demonstrate resilience and flexibility in changing environmental conditions.

Ant colony algorithms in unstructured P2P networks typically operate as follows. Initially, virtual "ants", which are represented by queries or messages, traverse the network to search for resources. They can either be released and forwarded en masse, meaning they are sent out collectively or in large groups, akin to a flooding-like approach, or be guided individually through the network like a 1-random walker algorithm, moving to one neighboring node at a time. "Ants" make their decision to visit a neighboring node based on the pheromone value of such node or the edge leading to it. Pheromone values serve as a form of distributed information about the quality of paths in the network, where nodes with higher pheromone values, for example, are more likely to be visited by the ants. Upon reaching a query hit, the "ants" communicate their findings back to the initiator of the query, updating pheromone values of each node or edge along the successful path in the process. This update reinforces the path that led to the successful query hit, making it more attractive for future queries. Over time, the most efficient paths are reinforced solely through local and indirect communication among the ants, leading to the emergence of optimized routing paths in the network.

Within the realm of ant colony algorithms, we have reviewed SemAnt [5], AntSearch [6], AntP2PR [1], as well as DLAntP2P and LantP2P [4], which are extensions of the previous AntP2PR algorithm.

1. *SemAnt [5]*

SemAnt was first proposed by Michlmayr in 2006. An earlier ant colony search algorithm, SemAnt was specifically developed with applications for distributed search engines in mind [5]. Distributed search engines are systems designed to retrieve and index information from multiple sources across a network. Users query for documents using keywords or phrases that describe what they are looking for. Upon receiving a query, distributed search engines distribute the query across the network using various search algorithms.

SemAnt operates on a few assumptions. In SemAnt, a controlled vocabulary of keywords are known as "concepts", and queries are assumed to be a series of concepts stringed together by a Boolean OR operator. This means that documents are retrieved if they are associated with at least one of the concepts in the query. Furthermore, each concept is associated with a unique pheromone type. This requires each peer in the network to maintain a routing table that records the amount of pheromone associated with every type, and do so for each edge leading to its immediate neighbors.

A "forward ant" is created by an initiator peer upon receiving a query. The ant travels the network until it is able to retrieve the document or its time-to-live (TTL) runs out. As the forward ant traverses the network, it decides, based on a probability, whether it wants to explore edges with lower pheromones, or exploit edges with the strongest pheromones. Upon reaching a hit, a "backward ant" is created to bring the document back to the initiator, updating pheromone values at each intermediate edge in the process. The strength of pheromone updates depends on the number of documents retrieved as well as the length of the path. This effectively leaves a pheromone trail for future ants to follow and encourages the formation of short optimized routing paths. True to real ant colonies, pheromone trails in SemAnt evaporate over a set time interval and by a predetermined amount of pheromones evaporated per time unit.

While results show that SemAnt outperforms k-random walk in hit rate and resource usage, that is, by traversing fewer edges, Michlmayr did not test SemAnt on a dynamic network, stating that "a static network topology and document distribution are assumed" [5, p.4] while intending to "evaluate and improve the performance of the algorithm in a dynamic setting" [5, p.4] in a future study. Furthermore, while pheromones are implemented with a degree of dynamicity in SemAnt, there is still potential for adaptive pheromone trails to limit routing towards leaving peers [5, p.4].

2. *AntSearch [6]*

AntSearch was developed by Wu et al. in 2006. It was proposed as a solution to avoiding free-riders in the network, which are defined as peers sharing less than 100 files [6]. Such peers take advantage of resources in the network but rarely contribute to it. Therefore, a significant amount of network traffic can be reduced by preventing messages from being sent to free-riders. In essence, AntSearch is a controlled flooding approach. Through AntSearch, each peer in the network has knowledge of its own pheromone value as well as that of its immediate neighbors. Pheromone values in AntSearch are effectively query hit rates. In comparison to other ant colony approaches surveyed for this study, AntSearch follows the ant colony

protocol more loosely. AntSearch operates in two distinct phases – the "Probe Phase" and the "Flooding Phase".

In the Probe Phase, the query initiator floods a few neighbors with a small TTL value, which is recommended to be a TTL of 2. The number of neighbors are determined using a range of k values, which denotes the percentage of peers with the highest pheromones to flood to. The purpose of this phase is to effectively approximate the number of results that can be retrieved for a query, as well as to predict the number of peers needed to reach for enough results to be obtained. Using the results obtained from this phase, a suitable k value is then chosen for the next Flooding Phase. During the Flooding Phase, the k value is fixed. AntSearch floods the algorithm towards peers with k% highest pheromone values in the network until they are all visited or enough results are retrieved.

The original study by Wu et al. demonstrate an average reduction of 50% in network traffic while still achieving sufficient results and maintaining comparable search latency to DQ+, an algorithm that floods to all peers irrespective of their reliability. While promising, this algorithm has the potential for further dynamicity using an adaptive k parameter during the Flooding Phase. Currently, the static k parameter means that AntSearch floods to the same proportion of neighbors from every node. By dynamically adjusting degrees of selectiveness at different areas of the network, the algorithm will further reduce network traffic. Furthermore, AntSearch can also benefit from dynamically adjusting TTL values.

3. *AntP2PR [1]*

AntP2PR, introduced by Loukos et al. in 2011, shares the common objective of mitigating free-riders in the network and minimizing network traffic. Its focus lies specifically in decreasing the traffic allocated for resource discovery, diverting it instead towards content transfer. Like AntSearch, peers in AntP2PR know of and utilize the pheromone values of their immediate neighbors to make their decisions. However, unlike AntSearch, peers in AntP2PR do not have information regarding their own pheromone values, as these values are relative to those of immediate neighbors. Also similar to AntSearch is AntP2PR's use of controlled flooding, however with varying TTL values determined dynamically based on pheromones instead.

Peers in AntP2PR maintain two tables: one for storing the pheromone values of their immediate neighbors, and another for recording those neighbors' query hits. When a query arrives at a node, the peer check for matches to its local repository. If no match occurs, the peer forwards the query to immediate neighbors whose pheromone values match a certain condition and if TTL is above zero. The condition, which involves using low and high pheromone thresholds predefined by the researcher, allows the peer to perform a dynamic query TTL update. The dynamic TTL update process is as follows: TTL is decremented if the neighbor's pheromones are less than the low bound, and is incremented if higher than the high bound; otherwise, it remains unchanged. This mechanism restricts the number of neighbors to which a query can be forwarded, making it less likely for the query to travel on suboptimal paths. Meanwhile, pheromones are updated when a query hit occurs. The process first involves visiting nodes on the successful path and incrementing appropriate entries in their hit count tables. Following an update in their hit count tables, each node then uses the new query hit values to update and normalize their pheromone tables. Alongside to query hit count, two predetermined parameters q_1 and q_2 are involved in determining the strength of these updates.

Results for AntP2PR demonstrate a typical tradeoff between network load and query hit or failure rate, that is, AntP2PR's reduction in the number of query packets generated comes at the cost of an increased rate of queries failing to return a result. Furthermore, while sensitivity analysis for parameters and pheromone thresholds was conducted in the original study, these values nonetheless remain static throughout the search process and must be manually chosen by the researcher. This makes AntP2PR difficult to implement in real-world networks. Additionally, AntP2PR was simulated on a network with non-dynamic topology consisting of 200 nodes, which is contrary to Loukos et al.'s claim that the algorithm "enables the network to adjust to the dynamic nature of Peer-to-Peer networks" [1, p.655].

4. *DLAntP2P & LAntP2P [4]*

Both DLAntP2P and LAntP2P were proposed by Ahmadi et al. in 2016 as an extension and improvement to the previous AntP2PR. Instead of a controlled flooding approach, DLAntP2P and LAntP2P employ adaptive decision-making capabilities to explicitly determine the number of ants to create and select which neighbor(s) to forward to. Although DLAntP2P and LAntP2P adopt the same pheromone and TTL update protocols of AntP2PR, the pheromones in these algorithms are also treated as probabilities for selecting ant actions using learning automata.

In DLAntP2P, each peer in the network hosts a variable learning automaton that employs the P-LRP learning algorithm. Similar to AntP2PR, each peer maintains a table recording the pheromone values associated with each of their immediate neighbors. Specifically, these pheromone values are assigned to the edges leading to those immediate neighbors. Initially, when a query is initiated, the initiator peer broadcasts the query to neighbors with the highest pheromone values, based on a randomly generated minimum pheromone threshold "r". Following this initial phase, the query or "ant" progresses through the network one neighbor at a time, rather than being replicated across several neighbors. This is accomplished by having the learning automaton at each peer select the edge with the highest pheromone value whenever an ant arrives at it.

In DLAntP2P, the query initiator doesn't utilize its learning automaton. Therefore, LAntP2P

proposes an alternative approach to enable all peers in the network to benefit from adaptive decision-making. In LAntP2P, each edge leading to a neighboring node is associated with two actions of varying probabilities: "creating the ant" and "not creating an ant". When a query is generated and cannot be resolved by the initiator, its learning automaton selects neighbors with high probabilities for "creating an ant" to forward the query. Intermediate nodes in LAntP2P function similarly to that of DLAntP2P, where the ant is replicated to progress through the network one neighbor at a time.

Results in the Ahmadi et al. study show that DLAntP2P and LAntP2P outperform AntP2PR and k-random walk in both success rate and message overhead [4]. DLAntP2P performs better than LAntP2P in generating fewer messages overall, but also having a higher success rate. In terms of query hits, AntP2PR trumps both algorithms simply because it is a controlled flooding approach and visits more nodes in the network at a time. Although DLAntP2P and LAntP2P demonstrate significantly greater adaptiveness and performance compared to AntP2PR, both algorithms still rely on the same pheromone and TTL update protocols as the original algorithm. These protocols continue to utilize static parameters that remain constant throughout the querying process and must be manually chosen by the researcher.

Overall, several limitations were identified in previous studies, prompting the development of IntelliAntSearch. Firstly, SemAnt and AntP2PR were evaluated in experiments conducted on static networks, which is a significant drawback due to the unrealistic nature of static network conditions. Additionally, none of the surveyed algorithms, including those simulated under dynamic conditions, were tested under different churn rates to assess their robustness and viability. Moreover, all of the surveyed algorithms except for SemAnt utilize static parameters for updating pheromone values and pheromone-based decision-making throughout the querying process.

B. Swarm Intelligence Algorithms [7]

In addition to ant colony algorithms, we have also surveyed two swarm intelligence algorithms based on the mechanisms of the Physarum Polycephalum Slime Mold and the Bark Beetle [7]. These algorithms both employ a form of controlled flooding, similar to that of AntP2PR.

1. The Phyasrum Polycephalum

There are two distinct phases to the Physarum Polycephalum Slime Mold algorithm: a forward phase and a backward phase. In summary, the forward phase occurs when the queries, or agents, are searching for resources. Once a query is resolved, the backward phase begins and agents retrace their steps back to the initiator peer. Similar to AntP2PR, each peer in the Slime Mold algorithm maintains a routing table that records flow values, serving as probabilities for selecting neighboring edges. This mechanism is meant to mimic how slime molds dynamically adjust their growth patterns in response to environmental cues, retracting and extending their pseudopodia in multiple directions depending on the concentration of nutrients. Upon a successful query hit and during the backward phase, the routing tables of peers on the successful path are retraced and updated using a delta function (1). In addition to probability-based decision-making, the Slime Mold Algorithm allows for agents to revisit peers, unlike AntP2PR. If an agent during the forward phase has visited all neighbors of a node, it chooses its next hop uniformly.

2. Bark Beetles

The Bark Beetles algorithm works similarly to the ant colony and slime mold algorithms. In nature, bark beetles detect and infest weakened trees through pheromones emitted by other beetles. They do this in order to reproduce within the tree. Over time, the aggregation of beetle pheromones on trees allows beetle swarms to select the most suitable host for infestation. Adapting this to P2P routing, the process begins when a source peer initiates a search query, spawning a configurable number of beetles. Each beetle is uniquely identified by an identification number. Within the network, each node maintains a pheromone matrix, initially empty, which tracks pheromone concentrations for discovered resources among neighboring nodes. Several configurable parameters, such as Time-To-Live (TTL), beetle count (akin to ant count), and pheromone types, can be adjusted prior to the querying process by the experimenter.

To navigate the network, beetles use a decision rule, selecting the neighbor with the highest pheromone concentration. Upon locating the resource, a beetle disseminates a notification message throughout the network, conveying information about the resource. If a beetle's TTL expires or it visits all neighboring nodes without finding a resource, it returns to the query initiator. Additionally, this algorithm offers the flexibility to prioritize either exploitation or exploration through a single parameter known as "sufficient pheromone concentration."

III. PROPOSED WORK AND TECHNICAL APPROACH

A. PeerSIM: Simulating P2P Network Dynamics [8]

To model and evaluate the performance of IntelliAntSearch, we utilized PeerSIM, a robust P2P network simulator written in Java [8]. PeerSIM is especially well-suited for our needs due to its flexibility and the extensible simulation framework it provides. The platform supports simulations with a large number of nodes, making it suitable for testing scalability in networks that may expand dynamically in size. This simulator has allowed us to construct an environment that mirrors the complexities of real-world P2P networks, enabling us to test and refine our LSTM-based routing algorithm under varied conditions and scenarios.

To implement a search protocol in PeerSim, there are six types of objects to consider:

1. The Node object, which represents peers in the P2P network
2. The Protocol object, which defines an operation to be performed at each node during a timestep
3. The Linkable object, which provides an interface for other Protocols to access neighboring nodes

4. The Transport object, which is a Protocol responsible for forwarding queries in the network and simulating drops or delays
5. The Control object, which can be scheduled for execution at different points during the simulation, either to schedule queries, modify the simulation, or observe and collect statistical data.
6. The Initializer, which is of type Control. They form the overlay network by instantiating edges between nodes according to different configurable mechanisms (for example, random joining)

Simulations in PeerSim can be easily set up using configuration files. Declaring a new element or component in the configuration file must adhere to the following scheme:

structure.stringID className

where:

1. *structure* is *protocol*, *init*, or *control*
2. *stringID* is the name assigned to the element. This name is referenced by other elements in the file.
3. *className* is the name of the object class. This class should be of type Protocol or Control.

If an element belongs to a class that requires parameters to be inputted, this must be done after the element is declared. Following element declaration, parameters can be defined as follows:

structure.stringID.parameterName VALUE

where:

1. *parameterName* is the name of the parameter defined within the class file
2. *VALUE* holds the value given for the corresponding parameter.

PeerSim offers two types of simulations: cycle-driven and event-driven. In cycle-driven mode, simulations progress at fixed discrete timesteps or cycles. Meanwhile, event-driven mode offers a more realistic and flexible progression of time. Events in event-driven simulation can be scheduled, in chronological order, to occur at any specific points in simulated time, without needing to adhere to regular, periodic timesteps. This mechanism allows the user to create dynamic and unpredictable systems, which is exactly what was needed for this study. In the context of IntelliAntSearch, we used the event-driven simulation engine, which allowed us to simulate asynchronous events as well as dropped packets and message delays to try to be as close as possible to real P2P networks. In addition, it allowed us to dynamize the network even more.

In addition to base PeerSim package, our implementation also utilizes the "iSearch" extension created by Gian Paolo Jesi and Simon Patarin from University of Bologna [9]. The iSearch extension provides additional services for building search protocols and collecting information from simulations. Search algorithms can be more conveniently implemented and standardized by extending their SearchProtocol class, which is connected to a SearchObserver class that collects statistics on the simulation. While iSearch is purely cycle-driven, we have modified its capabilities to accommodate event-driven simulations as well.

B. Equations

In our IntelliAntSearch model, we use a pheromone update equation derived from AntP2PR [1], which is what the algorithm uses for adjusting the routing paths based on past successes. The equation used is:

$$\Delta(qh) = q_1 \cdot e^{q_h q_2} \quad (1)$$

1. $\Delta(qh)$: Represents the change in pheromone level associated with a given routing path. This update is applied to the pheromone values stored for each path in the network, influencing future decisions on which paths to prioritize.
2. q_1: A scaling factor that adjusts the magnitude of the pheromone update. This parameter can be configured to increase or decrease the pheromone increment, thus affecting the aggressiveness of the routing updates. A higher q_1 leads to more pronounced changes in pheromone levels, making successful routes more attractive more quickly.
3. q_h: Represents the total number of successful query hits for a particular path. This metric serves as a feedback mechanism, indicating the effectiveness of the path based on past interactions.
4. q_2: Modulates the influence of q_h on the pheromone update. It acts as an exponent in the pheromone update equation, determining how rapidly the pheromone value increases with each additional query hit. A higher q_2 value results in more dramatic increases in pheromone levels with successful queries, enhancing the positive feedback loop.

C. Enhancing Query Efficiency with LSTM

In our proposed solution, we leverage Long Short-Term Memory (LSTM) neural networks to enhance the efficiency and accuracy of query handling in unstructured peer-to-peer (P2P) network environments. LSTM networks are a type of recurrent neural network (RNN) capable of learning order dependence in sequence prediction problems. This feature makes LSTMs ideal for predicting time-series data or any data where the current state is dependent on previous states. In the context of P2P networks, LSTMs have the potential to predict network conditions based on historical data, which includes traffic patterns, query success rates, and network load.

In our IntelliAntSearch model, the LSTM is trained to predict optimal values for *q1, q2,* and the pheromone thresholds (*hi*gh and *low* bounds), based on the evolving state of the network. This prediction is crucial for dynamically adjusting the parameters used in the pheromone update equation (1), ensuring that the search strategy remains optimal despite changes in network traffic and topology.

The LSTM model processes historical network performance data, learning patterns that lead to successful query resolutions and low network overload. This training allows the model to forecast the parameter values that are likely to achieve a balance between high success rates and manageable network load. For example, if the model detects an increasing trend in query success but also a rising network load, it would increase q_1 to boost success further while adjusting the high and low bounds to manage the load.

By integrating LSTM into our routing algorithm, we ensure that the search protocol is both reactive and predictive, adapting to anticipated changes in network conditions before

they can impact performance adversely. We aim for our approach to predict query traffic and optimize data routing methods, thereby reducing latency and improving throughput.

For training, we utilized Python with TensorFlow as it allowed us to efficiently train our model.

IV. Evaluation Methodology

For evaluating how IntelliAntSearch performed with respect to other models we've discussed earlier, we have chosen to focus on two primary metrics: Success Rate and Network Overload. These metrics were selected not only for their direct relevance to the performance of search algorithms in peer-to-peer (P2P) networks but also to highlight the inherent trade-offs that typically arise between them.

1. *Success Rate*

   Success Rate is defined as the ratio of successful query responses to the total number of queries initiated within the network. This metric measures the effectiveness of the search algorithm in locating and retrieving the desired information across the network. A higher success rate indicates a more efficient and reliable search process, where queries are more likely to find the required resources within fewer hops or less time. The success rate is particularly crucial in unstructured P2P networks, where the distribution of resources is not predetermined and can vary dynamically.

2. *Network Overload*

   Network Overload, quantified through the Number of Messages metric, refers to the total volume of network traffic generated by the search algorithm during the execution of queries. This includes all messages sent for initiating queries, routing information, and returning results. High network overload can lead to congestion, increased latency, and higher computational and bandwidth consumption, which can affect the network's overall performance and scalability. The goal is to minimize this metric to reduce the burden on the network while maintaining or improving the success rate.

Improving the success rate often requires more comprehensive search strategies, which can increase the number of messages transmitted and hence the network load. Conversely, strategies designed to minimize network traffic may reduce the scope or depth of the search, potentially lowering the success rate. Effective search algorithms need to balance these metrics to optimize both resource discovery and network utilization. This balance ensures that the network remains scalable and efficient, even as the number of nodes and the volume of queries increase.

V. Technical Results and Analysis

To analyze our simulations and approach, we ran three experiments in PeerSIM, all of which evaluated success rates (% of hit rate) and number of messages seen in the network (number of queries). For all, we used similar configurations with a simulated network delay between 20ms and 200ms, and a drop rate of 30% for packets. Each node in the network can have between 10 and 40 neighbors and all protocols had a TTL (time to live) of 3, meaning max number of hops.

1. In the first experiment, the already existing protocols are evaluated to validate our implementation and the simulator setup. According to the result of the experiment which is shown in Fig. 1 and Fig. 2, we conclude that our implementation matches the results of the other implementations by Loukos [1] and Ahmadi A [4] with DLAntP2P exceeding all other protocols in success rate and then AntP2PR and lastly K-walker. However, we introduced normal flooding and compared it to given protocols.

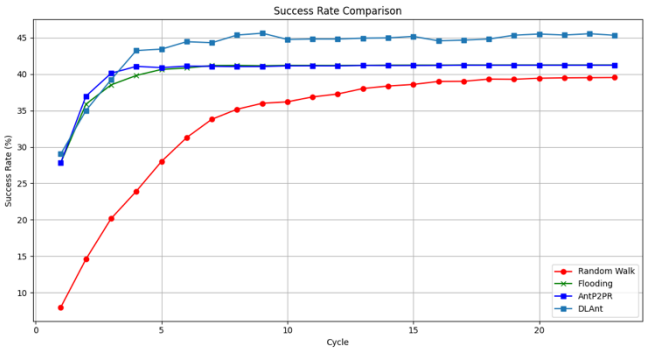

Figure 1.
Success rates (%) of original protocols

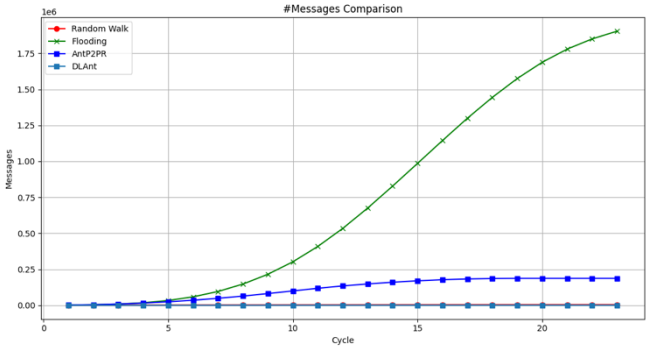

Figure 2.
Number of queries of original protocols

1. In the second experiment, we evaluated our IntelliAntSearch approach to AntP2PR [1] based on success rates and the number of queries made in the network. According to Fig. 3 and Fig. 4, we can see there is a slight improvement in the success rate by using our approach to choose the parameters, however, the number of queries increased.

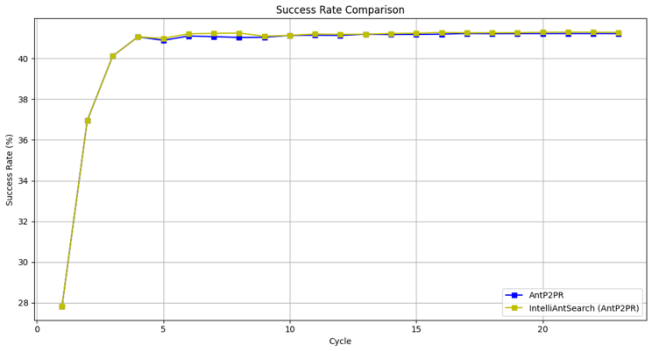

Figure 3.
Success rates (%) AntP2PR vs. IntelliAntSearch (AntP2PR)

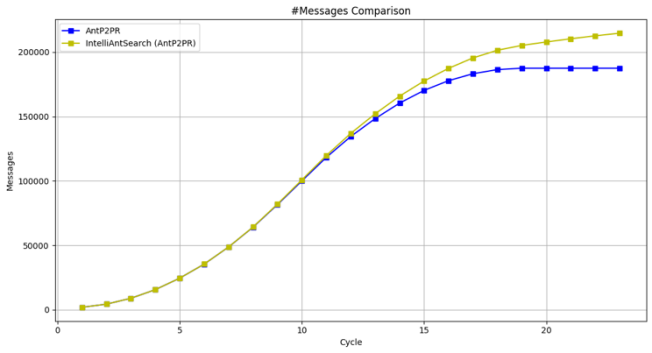

Figure 4.
Number of queries AntP2PR vs. IntelliAntSearch (AntP2PR)

2. In the third and last experiment, we evaluated IntelliAntSearch with DLAnt [4] by letting our approach choose the parameters for the given protocol. According to the result of the experiment which is shown in Fig. 5 and Fig. 6 we can see a bigger improvement than experiment 2 with AntP2PR in success rates and almost exactly similar query count.

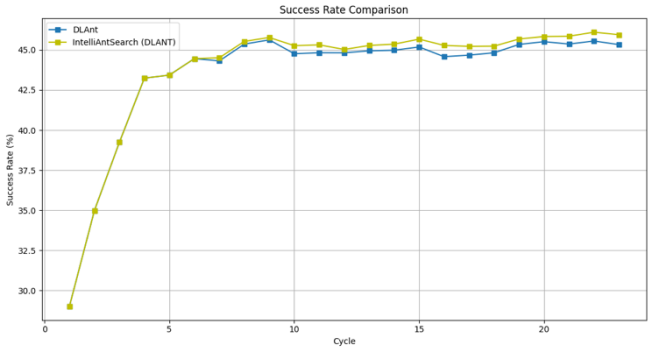

Figure 5.
Success rates (%) DLAnt vs. IntelliAntSearch (DLAnt)

Figure 6.
Number of queries DLAnt vs. IntelliAntSearch (DLAnt)

VI. Conclusion and Future Work

PeerSIM was difficult to start with, and until we found the "iSearch" approach, which helped us understand how to implement searching in the simulator, we couldn't really implement the algorithms. The documentation was poor, and the software is old, so it was difficult to fully dive into creating the simulations and fully understanding the process. Moreover, AntP2PR by Loukos [1] and DLAnt by Ahmad A. [4] do not cover the simulation environments very well, so we had to assume a lot of stuff (drop rate, message delay, node initialization with k neighbors). In addition to that, the proposed protocols by the given papers ran on a cycle-based engine where it is unrealistic to real-world event based P2P networks.

Due to time constraints, we were unable to test IntelliAntSearch against other algorithms for different churn rates or dynamic conditions. Among the previous methods we have surveyed, AntP2PR [1] and SemAnt [2] were evaluated on nondynamic networks, thus lacking realism. Furthermore, there has yet to be churn rate testing even among methods simulated with dynamic networks. In the context of unstructured P2P networks, testing search algorithms under dynamic conditions is essential as it demonstrates the full extent of an algorithm's effectiveness and viability. By subjecting the search techniques to various churn rates, including extreme dynamicity, we will be able to reveal the limits of their robustness.

VII. Workload Distribution Among Team Members

Overall: No strict delineation of roles, everyone contributed to each other's "tasks".

Abdulaziz –

- Suggested the 2010 paper we're extending
- Contributed papers for the topic
- Contributed to summarizing Ahmadi et al. 2016
- Migrating previous implementations (AntP2PR, Flooding, K-Walker) to iSearch
- Random K-Walker (Initial implementation, non-iSearch)
- A version of Flooding (non-iSearch)

Deeksha –

- Proposed extension to the AntP2PR algorithm using time-series analysis
- Contributed papers for the topic
- DLAnt (Initial implementation, non-iSearch)

- Visualizations for comparing implementations
- LSTM implementation

Karen –

- Contributed papers for the topic
- Literature Review for Ant Colony Routing algorithms
- AntP2PR (Initial implementation, non-iSearch)
- A version of Flooding (non-iSearch)
- DLAnt (iSearch implementation)

REFERENCES

[1] Loukos, F., Karatza, H., Mavromoustakis, C. (2010). AntP2PR: An ant intelligence inspired routing scheme for Peer-to-Peer networks. Simulation Modelling Practice and Theory. DOI:10.1016/j.simpat.2010.10.004

[2] Reyes, L.C., Santillán, C.G., Lam, M.A., Schaeffer, S.E., López, T.T., Izaguirre, R.O., & HéctorJ.Fraire, H.(2009). NAS Algorithm for Semantic Query Routing Systems in Complex Networks. International Symposium on Distributed Computing and Artificial Intelligence. DOI:10.1007/978-3-540-85863-8_34

[3] Toyoda F., Sakumot, Y., Ohsaki H.(2020). Proposal of an Efficient Blind Search Utilizing the Rendezvous of Random Walk Agents. Conference: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). DOI:10.1109/COMPSAC48688.2020.0-192

[4] Ahmadi A., Meybodi M., Saghiri A. (2016). Adaptive search in unstructured peer-to-peer networks based on ant colony and Learning Automata. Computer Networks, Conference: 2016 Artificial Intelligence and Robotics (IRANOPEN). DOI:10.1109/RIOS.2016.7529503

[5] Michlmayr E., "Ant Algorithms for Search in Unstructured Peer-to-Peer Networks," 22nd International Conference on Data Engineering Workshops (ICDEW'06), Atlanta, GA, USA, 2006, pp. x142-x142, DOI: 10.1109/ICDEW.2006.29

[6] Yang K., Wu, C., Ho J. (2006). AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks. Proceedings - International Symposium on Computers and Communications. DOI:10.1093/ietcom/e89-b.9.2300

[7] V. Šešum-Čavić, E. Kühn, L. Fleischhacker (2020). Efficient Search and Lookup in Unstructured P2P Overlay Networks Inspired by Swarm Intelligence. IEEE Transactions on Emerging Topics in Computational Intelligence. DOI:10.1109/TETCI.2019.2951813

[8] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," 2009 IEEE Ninth International Conference on Peer-to-Peer Computing, Seattle, WA, USA, 2009, pp. 99-100, DOI: 10.1109/P2P.2009.5284506

[9] Jesi, G. P., & Patarin, S. (24AD). Search Framework and algorithms on peersim. http://www.cs.unibo.it/~babaoglu/courses/csns/resources/tutorials/peersim3.pdf