DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch092

ASAP: an Ant resource Search Algorithm for swarm-like P2P networks

António Homem Ferreira[1], Carlos Martinho[1]

[1]INESC-ID / Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal
{antonio.h.ferreira,carlos.martinho}@ist.utl.pt

Abstract

The Ant Colony Optimization (ACO) meta-heuristic is a proven approach for solving complex distributed problems, being the routing problem one of such. By exploring the surroundings and indirect communication through pheromones, ants can find and follow the shortest path between a food source and its nest. Based on these characteristics, we present an ant-based algorithm for performing in-network resource search in a swarm-like Peer-to-Peer network. By marking connections that share same interests with a synthetic pheromone, a node can easily find a resource without having a significant impact on the network performance. Our approach focuses on decreasing the number of messages generated by each search, without having a negative impact on user experience. To achieve this, we present an algorithm that dynamically adapts based on the information a node has of its surroundings. The more information a node has of its neighbors, the higher the probability of choosing an exploitation strategy over an exploration one. Furthermore, the higher the number of nodes visited by an ant (and thus different paths followed), the lower the number of nodes explored in an exploration strategy. In order to decrease the number of messages sent to nodes that have already processed it, the parent ant informs each of its cloned ants about all the nodes to which each of these cloned ants will be sent to. Through simulation, we show the impact of these design choices in the algorithm's performance and discuss how it can be configured in order to adapt it to different networks.

Introduction

Peer-to-Peer (P2P) protocols rely on decentralized architectures for providing a large number of services like VoIP, video streaming, file sharing, etc. This architecture allows P2P networks to be extremely scalable, since every node can connect and exchange resources with every other node (being storage, processing power, content, etc). Due to these characteristics, P2P protocols account for a high percentage of Internet traffic (Cisco, 2011)(Sandvine, 2011). One of the most popular architecture in P2P networks are swarms, where the resource sharing is done by grouping peers sharing a same resource in a swarm. These swarms are isolated from each other, and for a peer to share different resources, it has to participate in several different swarms. Due to this swarm-like architecture, in-network search becomes difficult to implement. Although it is still possible to search for the resource identifier, in most cases, a peer needs to hold that resource in order to generate the identifier or search for it outside the network. This means that this identifier has to be either known at start or be generated with specific resource information that might not be known while performing a search. As for keywords search, these networks usually rely on outside services. However, there are several techniques that can be used to implement in-network search through keywords. Many of these techniques are nature-inspired algorithms that try to reproduce behaviors observed in nature. This work presents ASAP: an Ant Resource Search Algorithm for swarm-like P2P networks based on the Ant Colony Optimization (ACO) meta-heuristic (Dorigo and Caro, 1999)(Dorigo and Gambardella, 1997). By sending messages (ants) over the network and marking with a synthetic pheromone the links between nodes that lead us to a search response, we can forward future searches through those same links. By carefully choosing to whom the search messages should be forwarded to, we minimize the impact on network performance since flooding is avoided and less data is exchanged, resulting in lower bandwidth consumption in order to perform a search. As for user experience, it can also be improved since more and faster results can be achieved. Our approach focuses on dynamically adapting to the query results obtained. The algorithm was designed to evolve gradually from an exploration strategy, where a search is forwarded to many nodes in order to gather network information, to an exploitation strategy, where only the ones with the highest pheromone value receive and process the search message. However, the strategy choice also depends on the number of nodes already visited by the ant. If an ant knows that it or some clone already visited a high number of nodes, the possible number of paths the search is following increases and thus we can focus on forwarding the ant to the nodes that have a higher pheromone value. This way, we avoid flooding the network until the time-to-live (TTL) of the ant is reached. Furthermore, we also focus on minimizing the total number of messages sent between two

nodes that have already processed that same search. Discarding these messages has a significant impact on the network performance but none on the user experience.

Through simulation, we evaluate the algorithm's performance and show how the parameter configuration should be done based on some network properties. The simulated environment ran on peersim(Montresor and Jelasity, 2009) in a simulated swarm-like content sharing P2P network with a total number of 10000 nodes distributed over 50 different swarms, with each of the swarms sharing different content. As in real networks, nodes enter and leave the network throughout the experiment. Our results show the impact of the algorithm in both network performance and user experience. We show how the parameter configuration and variation affects the algorithm's performance and also show the importance and impact of some choices made during the development of the algorithm.

This paper is organized as follows: Section II gives a brief explanation on swarm-like P2P networks and Section III describes an overview of some of the related work done in this area. Then Section IV presents the algorithm, design and choices made during development, and Section V describes the simulation environment and obtained results. Finally, Section VI presents the final conclusions.

Swarm-like P2P networks

Swarm-like P2P networks aggregate nodes that share the same resource in a swarm. These swarms are isolated and independent from each other, where the nodes that belong to it only share a single resource among themselves. However, nodes can participate in as many different swarms as they please, as long as they are willing to share different resources. These multiswarm nodes can operate as a bridge between different swarms, connecting the isolated swarms with each other. In these P2P networks, despite being aggregated into swarms, nodes are not necessarily connected to all others that participate in the same swarm. Figure 1 illustrates a swarm-like P2P network, where the lines represent the connection between two nodes.

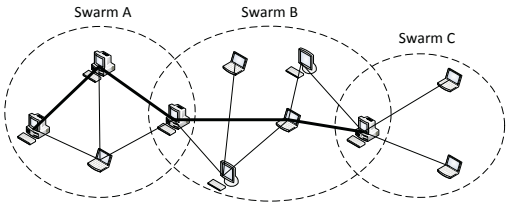

Figure 1: Swarm-like P2P network architecture.

Although proving to be a successful architecture for resource sharing, these networks usually do not provide an in-network keyword search mechanism due to their characteristics. In a swarm-like P2P network, in order for a node to make a specific in-network keyword search, it needs to send a message with the desired keyword to all or some of its neighbors, who will then forward it until a result is found or the nodes stop forwarding the message. One of the biggest problems with this approach is its impact on network performance as it can consume a great amount of the available bandwidth if the number of messages generated by each search is too high. On the other hand, if the number of messages is too low the search algorithm might have difficulty in finding resources that the node's closest neighbors don't have. In order for the algorithm to achieve best results, it needs to have as little impact on the network as possible and, at the same time still be able to find resources in the network. Furthermore, the search algorithm also needs to be able to adapt as nodes enter and leave the network. For this reason, in-network search is usually made through the resource identifier that, in most cases, is generated based on specific resource information that a node might not have if it does not hold that resource. As for keyword search, it is usually provided by services outside the network.

Figure 1 shows the message flow for a keyword search initiated by Node 1 in swarm A. The message is forwarded from swarm to swarm by the bridge nodes (the ones that participate in multiple swarms) until it reaches a node that has that resource, then a response message is sent back through the same path to Node 1.

Related work

The Ant Colony Optimization (ACO) meta-heuristic has been adopted by many to address complex distributed problems like routing problems in computer networks. One such work is SemAnt (Michlmayr, 2006), an ant-based algorithm to provide a distributed search engine in an unstructured P2P content sharing network. In SemAnt, each peer holds a table with a pheromone value associated with a specific keyword and a node. These keywords are predefined for the whole network. When an ant finds a desired content, it generates a backward ant that will follow the same path backward and update each corresponding pheromone value. From time to time, there is an evaporation rule that decreases the pheromone values in the tables. Though simulation, the author shows how the SemAnt outperforms the k-random walker approach (Lv et al., 2002). However, the simulated environment was based on a static network. One other work that focused on using the ACO for search in unstructured P2P networks is ERAntBudget (Wu et al., 2008). This work combined the Budget mechanism (Gkantsidis et al., 2005) and the ACO principle in an attempt to avoid generating a large amount of network traffic when searching for popular content, since there is a high probability of a query returning too many results. One other objective was to improve the query hit ratio for unpopular objects. Though simulation they compare ERAntBudget with other used techniques and showed that it can achieve a higher success rate using

less bandwidth. Later, the same authors presented a modified version called AEAntBudget (Chen et al., 2010). This new version inherited the characteristics of ERAntBudget and also provided mechanisms for progressively expanding the search scope based on previous results. With this modification, AEAntBudget can achieve better results than ERAntBudget. AntSearch (Wu et al., 2006) also adopts an ACO technique for query flooding in a P2P network. However, the authors' focus is in identifying free-riders, peers that consume but do not share their resources with the network. To do so, AntSearch uses pheromone values to identify these free-riders and avoid sending messages to the same, generating a smaller number of messages and getting query hit from peers that have a higher probability of sharing their resources with others.

Model

ANT colony's foraging behavior

Every living organism shows a foraging behavior which can be defined as: (1) search for a food source, (2) catch the food source and (3) ingest food source. Although it can be defined by the previous three steps, every organism has adopted and developed its own way of performing each action, optimizing it as needed. A well-known example is the ant colony. Ants by themselves are animals that do not appear to have much of an intelligent behavior, however, when in colony, they show a remarkable intelligent behavior. Ants always follow the shortest path between their nest and a food source. When searching for a food source, ants drop small amounts of pheromones so others can follow. When a food source is found, ants drop a significant larger amount of pheromones. When other ants leave the nest, the paths with larger amounts of pheromone have higher probability of being chosen. However, pheromone evaporates over time which results in larger amounts of pheromones for the shortest paths, as they are traveled by more ants.

The study of this complex behavior has resulted in many intelligent algorithms that have been successfully used for resolving complex and time consuming problems. The Ant Algorithm has been used to solve many different problems. This work will focus on the usage of an ant algorithm for resource searching in swarm-like P2P networks. By marking desired links with a synthetic pheromone, message forwarding will be done through the shortest path with higher probability, and thus save network resources such as bandwidth. Peers processing power can also be saved by processing less undesirable messages.

ASAP

Although P2P networks are based on decentralized architectures, many still depend on a centralized server for resource search within the network. Despite data exchange being done between nodes, resource search is done outside the network and depends on centralized services. The objective of this work is to study how this dependency could be broken by implementing in-network resource search through an ANT algorithm for query routing. Based on the ant foraging behavior, a distributed search engine can be implemented inside a P2P network. By marking, with synthetic pheromones nodes and connections that share same interests, a node can easily find the resource it is looking for, without having significant impact on the network's performance.

In order for a node to persist the routing information, each node needs three tables:

1. A table with pheromone values for each category, for each node. Each node can have different resources or know others with same, similar or very different resources. For this reason it is important to differentiate the nodes based on the category of the resources.
2. A table with an association between each category and its strategy weight value. Each node needs different strategies for processing different category searches, depending on the amount of information it has already gathered from previous searches for that category.
3. A table with the TTL value for each category. In order to improve network performance, the TTL value can be different for each category based on the information gathered by previous searches.

When a node searches for a specific resource in the network, it creates a query with a keyword and its respective category. The keyword can be any sequence of characters. As for the categories, they are predefined for the whole network and cannot be changed for a single node. Every resource has to be assigned to a category, otherwise it cannot be found.

When a node receives a query message, it checks if it holds any resource for the searched category. If the node is sharing resources for that category and it matches the searched keyword, then the node sends a reply message back through the same path. The reply message is responsible for updating the routing tables for each node it visits on the way to the one that started the query. This information will then be used by future searches for message routing optimization.

The algorithm is divided into three stages:

1. State transition: When ants choose the path to follow. Depends on the amount of pheromone the path has and the number of resources the end node has.
2. Pheromone update: When an ant finds what it is looking for, it drops an amount of pheromone that depends on the cost of the path.
3. Pheromone evaporation: Over time, pheromone evaporates and decreases for each of the marked paths.

The first step, state transition, is defined by two strategies: (1) exploration, where the ant explores the surrounding nodes and (2) exploitation, where the ant exploits the

existing information collected by its predecessors. The exploitation strategy selects the node, to which the message is sent, by choosing the one that has the highest combination of both pheromone value and number of resources. As for the exploration strategy, instead of choosing only the one with the highest combination of both values, it selects a set of nodes which have not yet been visited by the ant and calculates the probability distribution for all of them, based on both pheromone and number of resources values. If their probability is above a pre-determined threshold, the ant is cloned and sent to each one of these nodes. In order to select one of the strategies, the node uses the following equation:

$$\begin{cases} Exploration, & \text{if } rand > w_c \\ Exploitation, & \text{if } rand \leq w_c \end{cases} \quad (1)$$

where $rand \in [0, 1]$ is a random generated number and $w_c \in [0.2, 0.8]$, a threshold to define the probability of each strategy being chosen for category c. This parameter starts at its minimum value of 0.2 and is incremented each time there is an update to the pheromone tables for a given category, to a maximum value of 0.8. When there is a pheromone evaporation, this value is decremented. This forces the algorithm to use an exploration strategy during the startup phase and, as the node gathers more and more information about its sorroundings, the exploitation strategy is preferred, in an attempt to increase network performance. Equation 2 shows how this value is modified, where τ_{update} and $\tau_{evaporate}$ are weight values that determine how much the w_c parameter should increment and decrement over time.

$$\begin{cases} w_c = \tau_{update} \cdot w_c\,, & \text{for pheromone update} \\ w_c = \tau_{evaporate} \cdot w_c\,, & \text{for pheromone evaporation} \end{cases} \quad (2)$$

After choosing the strategy, if exploitation, the ant chooses its next node s through Equation 3.

$$s = MAX_{u \in U \cap u \notin s(F_q)} \left(\tau_{cu} \frac{\gamma_u}{10} \right), \quad (3)$$

where U is the set of all known and active nodes, $s\,(F_q)$ is the set of nodes already visited by this ant and others that it might know of, τ_{cu} is the amount of pheromone for category c in node u, γ_u represents the total number of different resources node u has.

If the choice was exploration, then the node chooses a set of neighbors to send the ant to, instead of sending just to the one with the highest pheromone concentration and highest number of resources. First, the node calculates the probability distribution for all neighbor nodes, through Equation 4.

$$p_j = \frac{\tau_{cu} \frac{\gamma_u}{10}}{\sum_{u \in U \cap u \notin s(F_q)} \tau_{cu} \frac{\gamma_u}{10}}, \quad (4)$$

then, with the number of nodes already visited by the ant, a percentile q is calculated using Equation 5.

$$q = \begin{cases} 1, & \text{if } \beta * Count(F_q) \geq 1 \\ \beta, & \text{if } \beta * Count(F_q) = 0 \\ \beta * Count(F_q), & Otherwise \end{cases} \quad (5)$$

where β is a constant, such that $\beta > 0$ and $Count(F_q)$ is the total number of nodes that have been visited by this ant and others that it might know of. After this, all the probability distribution values, calculated in Equation 4, are sorted in ascending order where the value k corresponds to the percentile q for that list of values. For every node j that satisfies the condition $p_j \geq k$, the ant forwards its clone to that node.

The exploration strategy will gradually evolve into an exploitation strategy as the ant visites more and more nodes, being β the value that defines the speed at which this happens. Furthermore, in an exploration strategy, when the ant clones itself it passes all its information to its clones, including the nodes that will be visited by each of its clones. For example, node A send two cloned ants to node B and node C. The cloned ant that reached node B will know that there was another ant that was sent to node C and thus will not send new ants to that node as it has already processed the query. These methods are mainly used for improving network performance, since it is expected to result in a lower number of messages sent and processed by each node. These methods decrease the number of messages between two nodes that have already processed the message, thus decreasing redundant cross-communication. For identifying a message as corresponding to an already processed one, each message is associated with a unique identifier and nodes keep a record of all the messages they already processed. This way, nodes discard messages that have already been processed. In both strategies, the total number of different resources, the neighbor node has, is also used to calculate if an ant should be forwarded to that same neighbor node. The nodes with a higher number of resources are more likely to have a resource a node is looking for. Furthermore, since they participate in multiple swarms, the query will be sent to a higher number of swarms, increasing the probability of finding an answer. This is also increasing the speed of convergence of the algorithm to find the shortest path.

A message is also associated with a time-to-live (TTL). This parameter is used so that the query does not continue endlessly. If the TTL is reached during a search, the ant stops and dies and no result is found. However, if the ant finds a result, it stops and goes back the same path. In swarm-like networks, where nodes are grouped into swarms that share the same resources, you only need to find one node to be able to join the swarm(s) that node is participating in. For this reason, the ant does not need to continue after finding a result. When returning after finding a result, the ant updates all the pheromone values for the searched category in each node it passes through. The pheromone value is updated as shown in Equation 6.

$$\begin{cases} \tau_{cu} = \tau_{cu} + Z, \\ Z = \gamma_q \frac{TTL_{max}}{2 \cdot h_{qr}} \end{cases} \quad (6)$$

The pheromone update equation depends on: the number of nodes visited by the ant from the response node to the current node (h_{qr}), the maximum TTL for the ant (TTL_{max}) and the value γ_q which represents the total number of swarms the response node participates in. This equation was designed to differenciate the responses to a query based on the total amount of different resources the response node holds as well as the number of nodes between origin and destination. This means that a path to a node that has a higher number of resources will have a higher increment in the pheromone values than another node that, despite also having the desired resource, has a lower number of resources. The pheromone increment is also higher for smaller size paths, that is paths with a lower number of nodes between origin and destination.

As for the pheromone evaporation, a global update rule was used. For every predefined interval T, each node applies the pheromone evaporation rule shown in Equation 7 for every row in its routing table. In this rule, $\tau \in [0, 1]$, is the amount of pheromone that should evaporate in this interval.

$$\eta_u = \eta_u \cdot (1 - \tau) \quad (7)$$

Many of the parameters used in the algorithm should be tuned based on network properties. For example, parameters such as TTL_{max} or β need to take into account the network size and average swarm size, so that the search does not have a negative impact on both network performance and user experience.

Use case

In order to fully understand the algorithm's workflow, we present a use case scenario where a node participating in a P2P network queries for a given resource. Figure 2 represents the network for this use case, where the lines represent the connection between two nodes. There are three swarms: A, B and C. While there are some nodes that only participate in a single swarm, nodes B1 and C1 participate in two swarms and work as a bridge between swarms A and B and swarms B and C, respectively. Each of the swarms represents a different resource. Node A1 just joined the network and has no information on its neighbors.

In this scenario, Node A1 initiates a query for the resource being shared in swarm C by Nodes C1, C2 and C3. First, the algorithm calculates $rand$ value and determines the usage of exploration and, since it has no information on its neighbors, it sends an ant to both Node A2 and A3. Since both ants know that they were sent to all of Node A1's neighbors, no messages are exchanged between Node A2 and A3, since they are already processing the search query. Node A2

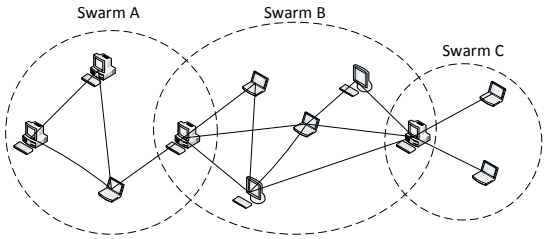

Figure 2: Swarm-like P2P network for use case.

does not have the content and does not have any other neighbors so it discards the message. As for Node A3, it uses the exploitation strategy and forwards the ant to Node B1. After this Node determined the use of exploration strategy, it calculated the following probability distribution values for its neighbors: [B3 - 90 ; B2 - 103 ; B4 - 250]. Then it calculates the percentil $q = 0.5$ that, when applied to the probability distribution values, results in $k = 103$. Since both Nodes B2 and B4 have a value equal or greater than k, the ant is forwarded to these nodes and not to Node B3. In Node B1, the ant only passed through one node thus the percentil q is low. However, if the ant had already passed through a high number of nodes, the percentil value would be much higher and the ant would only be sent to Node B4. After receiving the ants, Nodes B2 and B4 don't exchange messages between each other since each one knows that the other is already processing that same search, avoiding sending redundant messages. After receiving the ant, Node B4 forwards it to Node B5 which forwards the ant to Node C1. Node B2 also forwards the ant to Node C1. In this case, Node C1 can actually receive the query two times since Node B2 and Node B5 don't share the same parent ant so they cannot know that the other one also sent an ant to Node C1. Since Node C1 has the desired resource, the search ants are not forwarded anymore and a response ant is sent through both paths. The response ants will increase the pheromone values for every link they pass through, following the paths: (1) C1-B2-B1-A3-A1 and (2) C1-B5-B4-B1-A3-A1. Despite the pheromone increase in both paths, path (1) will have a higher increase than path (2) since the ant passes through a fewer number of nodes. The paths A1-A3 and A3-B1 will have their pheromone value increased twice since both responses pass through these paths. This pheromone value is only increased for the category to which the desired resource belongs to. Future queries initiated by Node A1, for a resource with that same category, will have a higher probability of following the previously discovered paths.

Simulation and Results

The simulated environment consisted of a content sharing swarm-like network topology with a total number of 10000

nodes randomly distributed over 50 swarms, according to the values in Table 1. Each swarm had a minimum size of 200 nodes and each node was connected to at least 10 other nodes. Each swarm was also associated with a different keyword that identified the content being shared on that same swarm. Since swarms are independent and unrelated with each other, some nodes participated in multiple swarms, working as a bridge between the different swarms. Table 1 shows the number of nodes that participated in multiple swarms.

Number of swarms	1	2	3	4	5	6
Number of nodes	2000	4000	2500	700	500	300

Table 1: Number of nodes that participate in one or more different swarms.

The experiment ran on a dynamic network, where a random number of nodes entered and left the network at any given time. However, no more than 5% of the network size (in this case, 500 nodes) could leave the network at the same time. Whenever the nodes reentered the network, they joined the same swarms they were previously participating in, since the node already had or was interested in those resources. However, the node established new connections different than the previous ones. The pheromone tables and other dynamic parameters, such as the strategy weight value were also set to default values upon reentering the network. As for the categories, we defined five: movie, music, application, tv show and book. During the experiment, 12000 queries were made.

Table 2 shows the initial parameter values. Since β defines the speed at which the algorithm evolves an ongoing search from an exploration strategy to an exploitation strategy, we varied this parameter in order to show how important its correct configuration is for the algorithm to achieve best performance.

Parameter	Value
Initial weight strategy for each category (w_c)	0.2
Weight strategy update (τ_{update})	1.25
Weight strategy evaporate ($\tau_{evaporate}$)	0.85
Initial pheromone value (τ_{cu})	0.009
Pheromone evaporation (τ)	0.03
TTL	24
β	1; 0.1; 0.01; 0.006

Table 2: Algorithm parameters value.

As explained in Section IV-B, the strategy weight value changed throughout the experiment, from a minimum value of 0.2 to a maximum value of 0.8. This allows a node to explore with higher probability its surroundings during the startup phase, when it enters the network and has no information whatsoever. This will lead to a higher number of messages exchanged during the startup phase. However, as the node gathers information about its neighbors, the preferred strategy changes to exploitation, which in turn will reduce the number of exchanged messages, as well as the paths an ant follows that do not lead to an answer, thus improving network performance. Figures 3 and 4 show this behavior.

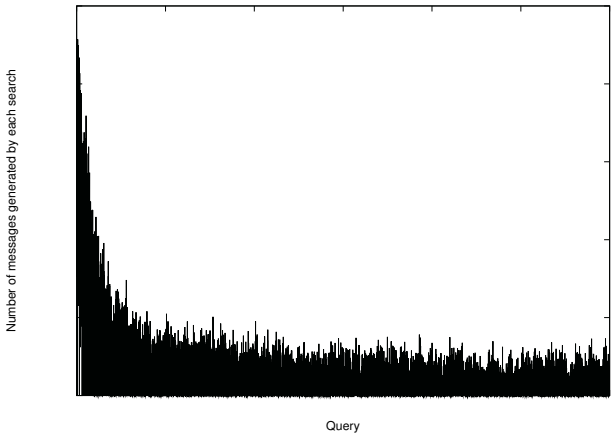

Figure 3: Number of messages generated by each query, with $\beta = 0.01$.

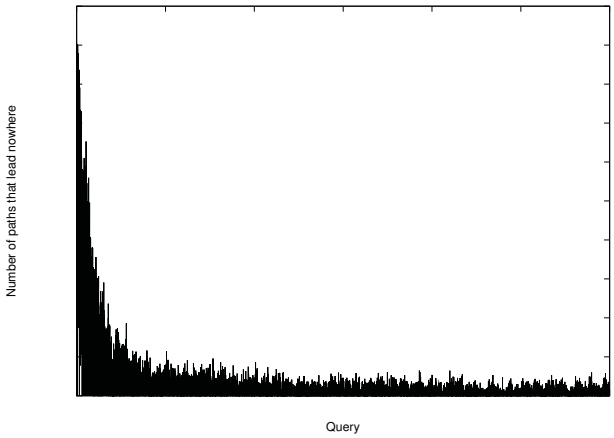

Figure 4: Number of paths that do not lead to an answer per query, with $\beta = 0.01$.

The first experiment ran with a β value of 0.01 and the algorithm's performance was compared to two modified versions of it, in order to show the importance of two choices made during the algorithm's development. In the first modified version (No parent memory), we modified the way the

cloned ant keeps the memory from its parent ant. When an ant cloned itself, it did not pass to its clone the information referring to all the node's neighbors that it would also be sent to. This way, an ant only knows the nodes that it visited and not the nodes visited by its first clones. As for the second modified version (No number of resources), the parameters $\frac{\gamma_u}{10}$ (the total number of different resources node u has) were removed from the algorithm. This way, the exploration/exploitation strategy equations were modified to only take into account the pheromone value. The first measurement made was the average number of messages processed by each node. Table 3 shows the results.

Number of swarms node participates in	Original	No number of resources	No parent memory
One	49,56	1119,17	3212,34
Two	224,05	1750,16	6253,71
Three	1365,15	2603,32	9132,34
Four	3438,31	3636,62	12959,42
Five	6625,93	4875,91	17700,18
Six	11272,8	6398,05	23340,49

Table 3: Average number of messages processed by each node that participates in one or more swarms.

Network performance is achieved by reducing the number of messages generated by each search to a minimum value so that a resource is still discovered. Table 3 shows just that. By passing to the cloned ant the information about all the node's neighbor the other cloned ants are also going to be sent to, the ant has more information about which nodes have already processed the search, avoiding visiting again these nodes. This way, the number of messages in the network decreases without affecting the search algorithm performance. As for the removal of the $\frac{\gamma_u}{10}$ parameter, it allows the algorithm to explore more the nodes with a fewer number of resources, thus distributing the workload more evenly in the network. However, the nodes with more resources are the ones that have a higher number of connections and have the highest probability of holding a resource or knowing someone that holds a resource that is desired. Although the usage of this parameter might have an impact on the network performance, as it does not distribute the workload evenly, it increases the probability of a node finding a resource in the network. Figure 5 shows the importance of the memory passing mechanism and differentiating nodes based on the number of different swarms they participate in, and how it can affect user experience when searching for resources in the network.

Both Table 3 and Figure 5 also show the tradeoff there needs to be between network performance and search performance. When distributing more evenly the workload in the network, we also decrement the number of queries that find

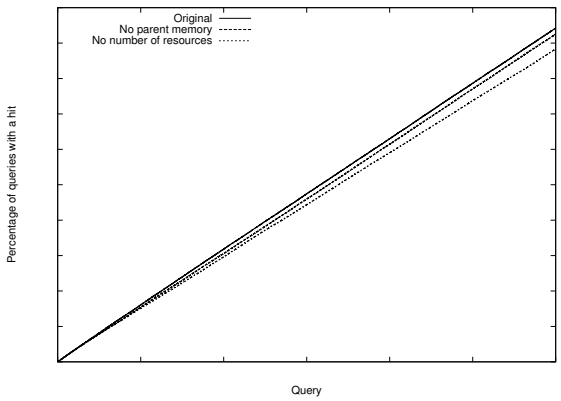

Figure 5: Cumulative distribution function for the queries with a hit.

at least one result. When focusing the workload in the nodes that have a higher probability of having a resource, we increment the number of queries that find the search resource, at the expense of network performance. Figure 5 also shows an unexpected behavior. When removing the memory mechanism, the algorithm generates a much higher number of messages (Table 3), however it achieves a lower hit rate than the original version. This can be explained by the large number of generated messages that are sent to nodes that have already processed the message, and thus are discarded.

After showing the importance of these algorithm's behaviors, we ran the simulation with four different β values: 1, 0.1, 0.01 and 0.006[1]. The β parameter defines how the exploration strategy evolves into an exploitation one, having direct impact on both network and algorithm performance. First, we compared how this value affects both the distribution of the workload in the network and the average number of messages generated by each search. Table 4 shows the results.

Number of swarms node participates in	β			
	1	0.1	0.01	0.006
One	1,63	29,29	49,56	78,76
Two	1,88	51,2	224,05	720,54
Three	2,25	64,67	1365,15	3853,47
Four	2,7	96,32	3438,31	7322,91
Five	63,25	656,66	6625,93	11798,08
Six	781,49	4954,7	11272,8	16987,04

Table 4: Average number of messages processed by each node that participates in one or more swarms.

As expected, by increasing the β value, the exploration strategy evolves more rapidly into an exploitation strategy,

[1]Simulator limitations forced the lowest value to be 0.006

generating a lower number of messages and focusing most of these messages in the nodes with the higher number of resources. As the value decreases, the algorithm makes a larger use of the exploration strategy, thus generating more messages and exploring more the nodes with a lower number of resources. This exploration strategy is very important in gathering information from the surrounding network so that more paths can be explored and the desired resource found. This has a direct impact on the search results and consequently the algorithm's performance. However, having a β value too low will result in a network flooding and consequently have a negative impact on network performance. Figure 6 shows a CDF with the percentage of queries that find the searched resource.

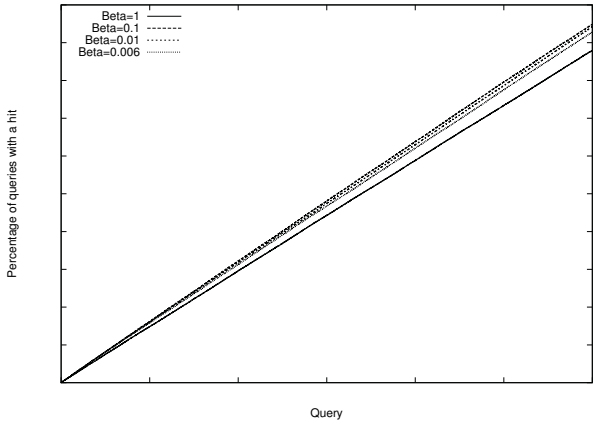

Figure 6: CDF for the queries with a hit, with different values for the β parameter.

In order to configure the β parameter, the average swarm size in the network and the percentage of nodes that actually participate in multiple swarms need to be taken into account. If a network has a low percentage of multi-swarm nodes, the β value should be low enough so that the algorithm does not focus all its messages on these nodes and distributes its load throughout all other nodes. This behavior can be observed in Table 6 for $\beta = 1$. On the other hand, if the network has a high percentage of multi-swarm nodes, the β value should be high enough so that it does not flood the network with query messages. The β parameter can have a significant impact, either positive or negative, on network and algorithm performance and thus should be configured accordingly to each network it is used in.

Conclusions

This work presents an ant-based algorithm capable of providing a search mechanism inside swarm-like P2P networks. The algorithm focuses on minimizing the impact it has on network performance without affecting the user experience. This is done through pheromone marking, strategy and message forwarding adaptation, based on the total number of nodes already visited by the search ant. This way, the number of messages generated by each search query decreases as the node has more information about the surrounding network. Our simulations show how the algorithm behaves and reacts to different parameter configuration and how it can be configured to achieve best network performance and results. The simulation results also show the reason for some design and development choices made upon creation.

Acknowledgements

This work was supported by national funds through FCT - Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013. The authors would also like to thank Professors Ricardo Pereira and Fernando Mira da Silva for their support.

References

Chen, Z., Liu, J., and Li, J. (2010). An adaptive expanding antbudget search algorithm for unstructured p2p networks. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 2, pages V2–777 –V2–781.

Cisco (2011). Cisco visual networking index: Forecast and methodology, 2010-2015.

Dorigo, M. and Caro, G. D. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the IEEE Congress on Evolutionary Computation*.

Dorigo, M. and Gambardella, L. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. In *IEEE Transactions on Evolutionary Computation*.

Gkantsidis, C., Mihail, M., and Saberi, A. (2005). Hybrid search schemes for unstructured peer-to-peer networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1526 – 1537 vol. 3.

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, ICS '02, pages 84–95, New York, NY, USA. ACM.

Michlmayr, E. (2006). Ant algorithms for search in unstructured peer-to-peer networks. In *IN PROCEEDINGS OF THE PH.D. WORKSHOP, 22ND INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE*.

Montresor, A. and Jelasity, M. (2009). PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*.

Sandvine (2011). Global internet phenomena report.

Wu, C.-J., Yang, K.-H., and Ho, J.-M. (2006). Antsearch: An ant search algorithm in unstructured peer-to-peer networks. *2012 IEEE Symposium on Computers and Communications (ISCC)*, 0:429–434.

Wu, G., Liu, J., Shen, X., Gao, L., Xu, J., and Xi, K. (2008). Erantbudget: A search algorithm in unstructured p2p networks. In *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, volume 2, pages 765 –769.