Shrinking the Network: The BlåtAnt Algorithm

Amos Brocco, Fulvio Frapolli, and Béat Hirsbrunner

Technical Report 08-04,
Department of Informatics, University of Fribourg, Switzerland
{amos.brocco,fulvio.frapolli,beat.hirsbrunner}@unifr.ch

BLÅTANT[1]

Abstract. BlåtAnt is a distributed and adaptive algorithm inspired by Ant Colony Optimization (ACO) to create overlay networks with small diameter by adding and removing logical links. In contrast to most existing methods, our algorithm is dynamic and can be used in evolving networks, where dynamic connections and disconnections are possible. Simulation results show that our approach produces and maintains networks with bounded diameter. A formal proof of the logic behind the algorithm is also provided.

1 Introduction

In the 1960's, social psychologist Stanley Milgram brought to the light the fact that people are commonly linked by short chains of acquaintances of an average length of six. This surprising observation, which was then supported by empirical experiments, initiated the so called "six degrees of separation" myth and a wide spread of research on the "small-world" phenomenon. In particular, the scientific community saw a growing interest for analytical models reproducing those social community properties, as a mean to predict and analyze social dynamics.

Beside from networks of individuals, the small-world phenomenon can also emerge from other kinds of networks; as pointed out in [9], networks of plane routes, or power distribution grids, also exhibit short paths between entities.

[1] From Harald Blåtand, the king thought to have reunited Denmark, Norway, and Sweden under a unique kingdom.

Early attempts at modeling small-worlds were based on random graph augmentation [2]. As subsequently proved in [3], random graphs have indeed small diameters, but fail to reproduce another important characteristic of real small-worlds: a high clustering coefficient. Roughly speaking, the clustering coefficient of a graph represents the average number of neighbors of a vertex which are connected to each others: for networks of individuals, it is commonly observed that friends know each others. Later models [14, 15] fixed this issue by augmenting the graph according to a probability distribution and achieved a better similarity to real social networks.

Unfortunately, although being able to display the same observable topological characteristics of real networks, probabilistic models still failed to explain how Milgram's experiment succeeded given the very little information available to each peer. This question was answered by Jon Kleinberg [16], who introduced a decentralized "greedy-routing" algorithm where routing choices are based only on local information available on the current node. A small-world network where such an algorithm is able to produce short paths between nodes is said to be navigable. Practical situations where small-diameter networks are interesting is undoubtfully resource discovery in distributed systems. By bounding the number of hops required to reach every other node, it is possible to design efficient query broadcast algorithms even in unstructured virtual networks.

This technical report describes a collaborative distributed algorithm that reorganizes an existing network by adding and removing logical links in order to reduce its diameter. The algorithm is based on swarm intelligence and ant colony optimization, and makes use of different ant species to collect and gather information across the network. Our approach is completely distributed and decentralized and does not require any kind of global knowledge; thus it is suitable for P2P and grid networks.

Swarm intelligence concerns algorithms inspired by the coordinated behavior of insect swarms in the real world. Ant colony optimization is a field of swarm intelligence that replicates the interaction between ants. Research in both fields has lead to algorithms and methods that are able to solve a variety of problems, such as TSP, network load balancing, and routing. Ants are mobile agents with limited capabilities, nevertheless global optimal solution emerges from their collaborative behavior.

This paper is organized as follows: Section 2 discusses related work in the field of network ant algorithms, graph augmentation algorithms, and distributed resource discovery approaches. Section 3 presents the basic rules proposed in our approach, along with an analytical proof of their correctness. Section 4 provides a detailed description of the BlåtAnt algorithm. Section 5 presents the results of empirical evaluations in static and dynamic scenarios; finally, Section 6 contains a conclusion on the work done and provides some insights on possible future research directions.

2 Related Work

Ant algorithms are bio-inspired methods imitating the behavior of real ants. These algorithms belong to a branch of artificial intelligence called Ant Colony Optimization (ACO)[4, 5]. The ACO metaheuristic has already been proven successful for solving different problems such as the Traveling Salesman Problem (TSP) [13], load balancing problems in distributed computing systems [6, 11], or routing [8, 10]. ACO itself is part of a larger branch of artificial intelligence called swarm intelligence, which groups all methods inspired by the behavior of swarm of insects. One of the advantages of ant algorithms is their intrinsic distributed nature, which permits a quasi-immediate application to network related problems where global knowledge is not available.

By mean of different collaborating species of ants, the BlåtAnt algorithm aims at bounding the diameter of an existing network by intelligently adding additional logical links. Such rewiring of the network will allow the implementation of other network algorithms exploiting the short distance between any node. In particular, we target resource discovery in dynamic grid systems.

By looking at existing graph augmentation methods, we can find various approaches based on random augmentation graphs. These algorithms, for example [14, 2, 15], generate networks with Small-World properties using a global knowledge of the system. A first example of decentralized approach can be found in [17], and an improved version is detailed in [18]. This last example shows that distributed augmentation of a graph is possible, nonetheless the algorithm proposed in that paper still requires some kind of *a priori* knowledge of the network, and is not adaptive with respect of the topology of the network. Our approach overcomes this problem by exploiting the adaptivity of ant algorithms.

One of the goals of our research is also to support efficient resource discovery in distributed systems, namely computing grids. In the field of P2P and Grid networks, there exist many solutions that use small-diameter networks to improve resource discovery and optimize routing of queries.

Resource discovery in distributed systems is a challenging task: as there is no global knowledge of the network, the worst case for locating a resource requires querying every other node. Ideally, a resource discovery algorithm must be able to find every available resource, in finite time, and with the lowest communication cost.

Existing solutions are commonly based on two approaches: distributed hashtables (DHTs) and flooding. Many examples of distributed hashtable protocols ([19–21]) enforce strict topologies to keep predictable distances and deterministic routing of queries. Unfortunately, this requires a global knowledge of the network or the partitioning of the search space in order to correctly assign links. For this reason, such solutions are of little or no interest in unstructured networks without centralized information. An exception is represented by Symphony [22], which is based on a completely randomized topology, where nodes have both a short and long distance links: construction of this network starts with a ring, and long distance links are added according to a probability distribution as in small-worlds models. Unfortunately even such solution has some drawbacks when it comes to grid systems. DHT applications typically manage resources such as files, whose content does not change so often. In grid systems, resource availability and type changes frequently, and computing resources are not relocatable. It is thus impossible to partition the search space or assign resources to nodes according to their position. In general, as pointed out in [29], hierarchical solutions would not adapt to the volatile and heterogeneous nature of resources shared in a grids. For these reasons, decentralized, unstructured, self-configuring architectures are a better alternative for grids.

Discovering resources in unstructured and dynamically evolving networks is best accomplished using flooding algorithms, which involves querying as many nodes as possible. Flooding does not typically require special topologies, but to avoid large network overheads it is necessary to limit the search space and avoid forwarding multiple copies of the same query [31]. This can be accomplished by setting a TTL (Time-To-Live) for each query, in order to limit the maximum number of forwardings. By knowing the maximum diameter of the network it is possible to determine the worst case distance, and adapt the TTL accordingly.

Cutting down distances in the network to improve flooding methods has been already used in [32]. A small-world overlay graph is created by differentiating between nodes that produce jobs, and nodes who execute jobs. The particular ring topology used by the algorithm ensures that queries sent by producer nodes only travel for short distances before reaching a consumer node.

Further examples of decentralized search in small-world networks are discussed in [23], and some examples of construction of overlay networks with small-world characteristics are presented. Other projects [25–27], propose a peer clustering based on the information shared, thus reducing the distance between nodes with similar offer. Nonetheless these solutions are not geared toward grid environments.

Existing research and solutions show that resource discovery in unstructured networks can be optimized by augmenting the existing network and reduce the distance between nodes. The BlåtAnt algorithm supports this idea by providing a way to bound the diameter of an overlay network through the addition of logical links. In contrast to most graph augmentation algorithms, BlåtAnt is completely decentralized, and does not require a global knowledge of the network. Finally, thanks to the continuous work done by ants, the algorithm is able to control even dynamic networks without supervision.

3 Centralized Algorithm

BlåtAnt is a distributed algorithm executed on a network which can be represented as a finite graph G. The goal of the algorithm is to rewire the network in order to bound its diameter into a certain interval determined by an user-defined parameter D. The rewiring process consists in adding and removing logical links between nodes. This sections details the rules used by the algorithm and provides a formal proof valid in the case of a centralized version with global knowledge of the network.

A detailed description of the actual distributed version along with an empirical evaluation follows in Sections 4 and 5.

3.1 Definitions

For the sake of clarity, we present here the recurring terms used throughout the rest of this paper.

Definition 1. *A* ***graph*** *is a set $\langle V, E\rangle$ of nodes and edges. In the considered scenario, a graph is a computer* ***network*** *consisting of* ***nodes*** *and un-directed* ***links****.*

Definition 2. *A node n_i is* ***adjacent*** *to another node n_j if there exist an edge $(n_i, n_j) \in E$. The* ***neighborhood*** *set N_i of node n_i is the set of nodes adjacent to n_i. In undirected graphs, adjacency is commutative: $n_j \in N_i \Leftrightarrow n_i \in N_j$.*

Definition 3. *We consider a node n_i as* ***connected*** *to another node n_j iff n_i is adjacent to n_j.*

3.2 Connection and Disconnection Rules

The rewiring process is based on two rules that are applied iteratively on the graph G representing the network. These rules only depend on a single integer parameter $D > 0$, which is adjustable according to the desired result. A first rule determines whether the distance between two nodes is too large, thus a shortcut connection is desirable. The second rule does the opposite by detecting and removing redundant links.

Rule 1 (Connection Rule). *Let n_i and n_j be two non-connected nodes in the network graph G, and $d_G(n_i, n_j)$ the minimal distance from n_i to n_j in G. We connect n_i to n_j if:*

$$d_G(n_i, n_j) \geq 2D - 1 \tag{1}$$

Rule 2 (Disconnection Rule). *Let n_i and n_j be two connected nodes in the network graph G, $i \neq j$. Let $G' \leftarrow G \setminus \{n_i\}$, and N_i be the set of all nodes adjacent to n_i. Node n_i is disconnected from $n_j \in N_i$ if:*

$$\exists\ n_k \in N_i, k \neq j\ :\ d_{G'}(n_j, n_k) + 1 \leq D \tag{2}$$

3.3 Proof of Correctness

We now prove that, for any undirected finite graph G, with a global knowledge and in a finite number of steps, it is possible to create a graph with a diameter less than $2D - 1$ by applying Rule 1 and 2 in any order.

Lemma 1 (Convergence of Rule 1). *For a given graph G, let be $G_0 = G$ and G_{n+1} the graph obtained by applying Rule 1 to G_n for two randomly chosen nodes. Then $\exists\ k \geq 0,\ \forall\ i \geq 1\ :\ G_k = G_{k+i}$, i.e. $\forall\ n_i, n_j \in G_k\ :\ d_{G_k}(n_i, n_j) < 2D - 1$.*

Proof. Since the function on the distances resulting from the application of Rule 1 is monotone decreasing, for k big enough, result follows. □

Lemma 2 (Monotonicity of Rule 1). *Let G' be a graph obtained by applying Rule 1 to G. Then*

$$\nexists\ n_i, n_j \in G\ :\ \textit{Rule 2 applies} \Longrightarrow \nexists\ n_i, n_j \in G'\ :\ \textit{Rule 2 applies}$$

Proof. Let (n_i, n_j), $n_i, n_j \in G$, the edge added by Rule 1. Because of the minimality of $d_G(n_i, n_j)$, the smallest cycle in G' has length $l \geq 2D-1+1 = 2D > D$. □

Lemma 3 (Safeness). *Applying Rule 2 cannot disconnect a connected graph.*

Proof. Consider a connected graph G, and three nodes $n_i, n_j, n_k \in G$, such that $n_j, n_k \in N_i$. If there exists a path between n_j and n_k satisfying (2), there are at least two disjoint paths from n_i to both n_j and n_k in G. Thus, removing a single edge cannot disconnect G. □

Corollary 1 (Convergence of Rule 2). *Given a graph G, let be $G_0 = G$, and G_{n+1} the graph obtained by applying Rule 2 to G_n for two randomly chosen nodes. Then $\exists\ k \geq 0,\ \forall\ l \geq 1 : G_k = G_{k+l}$.*

Theorem 1. *Given a graph G, let be $G_{0,0} = G$, and $G_{n,m}$ the graph obtained by applying, in any order, n times Rule 1 and m times Rule 2 on $G_{0,0}$ for randomly chosen nodes. Then $\exists\ k \geq 0, \exists\ l \geq 0$, and $\nexists\ n_i, n_j \in G_{k,l}$ such that Rule 1 or Rule 2 apply.*

Proof. Follows directly from Lemmas 1, 2, 3, and Corollary 1. □

Because of Rule 2, the resulting graph has a clustering coefficient equal to zero. In other words, graphs created with our algorithm will not not have full small-world characteristics. Figure 1 shows an example path graph consisting of 20 nodes; the initial diameter is 19, and $D = 3$. By applying both rewiring rules, we obtaing a graph (Figure 2) with a diameter of 4 ($< 2D - 1 = 5$).

4 BlåtAnt Algorithm Description

This section describes the logic behind the BlåtAnt algorithm, along with its data structures. The algorithm is completely distributed and asynchronous, and makes use of Rules 1 and 2 presented in the previous section. For the rest of this paper we will refer to these rules simply as *rewiring rules.*

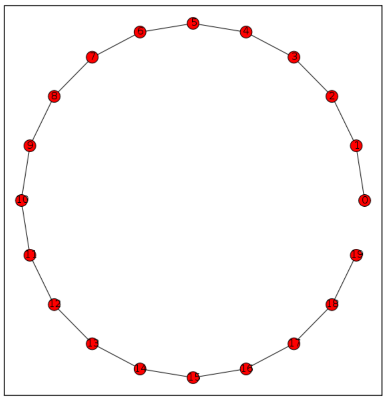


Fig. 1. Initial graph

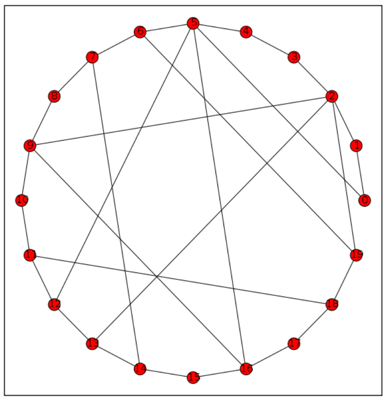


Fig. 2. Resulting graph

Each node in the network (or vertex in the graph) executes independently using local partial information about the network. In order to propagate and update that local information, we propose the use of ants. An ant is a lightweight mobile software agent that is executed on nodes. During execution, ants can migrate from node to node to access and manipulate local data structures. Ants wander across the network to collect and spread available information, and a global optimal solution of the problem of reducing network diameter is produced through successive local optimizations done by single nodes.

Since the algorithm is distributed and runs without a global view of the network, Rule 2 becomes extremely important to remove redundant links erroneously created by Rule 1, thus avoiding the creation of cliques. Because of the continuous work of ants, the algorithm is able to keep minimal diameters even in dynamic scenarios and evolving networks.

4.1 Node Data Structures

Each node n_i in the network maintains some data structures to store partial network information and data produced during the execution.

Alpha table $\boldsymbol{\alpha_i}$ The α_i table stores local information about the network. Data in the table is constantly updated by mean of the information gathered by ants. Each entry contains information about a node $n_j, j \neq i$:

- *distance*, the estimated distance d from n_i to n_j,
- *neighbors*, neighbors' identifiers $\{n_k\}$ with $n_k \in N_i$,
- $timestamp_i$, local time (local time of the last update),
- $timestamp_j$, remote time (local time at the remote node).

The local time value is used to clean the oldest entries in the table when its size reaches a user-defined maximum; the remote time is used to determine if an incoming information is newer than the existing one. Because the quality of the solution found by the algorithm depends on the information available on each node, the size of the alpha table should be limited only be the storage and computational capacity of the node.

Beta Pheromone Trail In order to detect the departure of a node from the network, a pheromone trail is used to keep track of the liveness of neighbors. Ants coming from n_j increase the pheromone concentration $\beta_i[n_j]$ on n_i, so that the trail effectively keeps track of the incoming traffic from that particular link. If the pheromone evaporates completely, the neighbor is assumed to be dead, and a disconnection procedure is started. Because connections and disconnections are performed asynchronously, the β pheromone is also used to remove stale connections resulting from unsuccessful rewiring actions.

Gamma Pheromone Trail To force a complete coverage of the network, a pheromone trail is used to direct ants to underexploited paths. For each $n_j \in N_i$ there exist a trail $\gamma_i[n_j] \geq 0$. When an ant moves from n_i to a neighbor n_j, it lays some pheromone that increases the concentration of $\gamma_i[n_j]$. At each wandering step, with probability κ, an ant can chose to follow an existing trail instead of proceeding at random. If multiple trails are encountered, the trail with the lowest concentration is chosen.

4.2 Ant Species

The rewiring of the network is assisted by different species of ants accomplishing specific tasks. Regardless of their species, all ants share a common set of features:

– ants have only access to information on the current node;
– ants can only sense or reinforce local trails;
– every ant remembers the node n_i where it comes from;
– every ant remembers the neighbors N_i of n_i.

As said before, when an ant moves from a n_i to n_j, a small concentration of gamma pheromone is deposed on $\gamma_i[n_j]$. Inversely, each time an ants arrives on a node n_j it hands out its information about the previous node n_i, and increases the beta pheromone concentration $\beta_j[n_i]$. Ants have a maximum lifetime ι, which is the maximum number of migration steps it can perform before being killed by the system.

Discovery Ant Discovery ants wander across the network, collecting and spreading information about visited nodes. This information is stored in a fixed-size circular buffer V of length l_V carried by the ant. When the ant arrives on node n_k, a triple containing the identifier of the node n_k (in real networks, typically the IP address), the identifiers its neighbors N_k, and the local timestamp t_k, is added to V. Discovery ants on node n_i are only allowed to migrate to a node contained in a restricted neighborhood set N_i^*, which contains all nodes in N_i without already visited nodes. If $N_i^* = \emptyset$, the circular buffer is emptied ($V \leftarrow \emptyset$) and N_i^* is re-initialized to N_i ($N_i^* \leftarrow N_i$). The choice of the migration target node is done as follows:

– with probability κ the target is chosen at random from N_i^*;
– with probability $1 - \kappa$, the neighbor with the lowest corresponding γ trail is chosen (i.e n_j such that $\gamma_i[n_j] = min_k\{\gamma_i[n_k] \mid n_k \in N_i^*\}$).

The behavior of a Discovery ant is illustrated in Algorithm A.1. At regular discrete intervals of length ι, each nodes generates, with probability μ, a new Discovery ant. This prevents the complete extinction of the ant population in the event of node or network crashes. The value of l_V determines the lookup capacity of each node, thus higher values help lowering the errors derived from the use of a local information about the network. As the transmission cost increases with the size of the buffer, a compromise must be made.

Depending on the network topology, it could be desirable to favor random exploration instead of following the lowest gamma trail concentration by adjusting the value of κ to higher values.

Link Ant Link ants are instantiated by a node aiming to connect to another node. The ant migrates from the node requesting the connection n_i, to a target node n_j. There, the ant checks if the distance estimation from n_j to n_i satisfies Condition 1 from Rule 1, and eventually calls the connection procedure on n_i for n_j Finally, the ant migrates back to n_i, the distance to n_j is checked once again, and eventually the connection procedure for n_j on n_i is executed. The whole process is detailed in Algorithm A.2.

Unlink Ant Unlink ants are used to disconnect from a node either because the link is in a cycle satisfying Rule 2, or because the neighbor is dead ($\beta_i[n_j] \to 0$). During the first phase, the ant migrates from n_i to n_j; if migration fails, n_j is immediately removed from the neighbors set N_i and both the beta and gamma trails corresponding to n_j are removed. On n_j, n_i is removed from N_j, and both $\beta_j[n_i]$ and $\gamma_j[n_i]$ are cleared. The ant then proceeds by moving back to n_i where the final steps are performed: removal of n_j from N_i and clearing of $\beta_i[n_j]$ and $\gamma_i[n_j]$. The behavior of Unlink ants is described in Algorithm A.3.

4.3 Frozen Connections

In Section 3 we proved that the centralized version of the algorithm cannot disconnect a connected network. Unfortunately, because in the decentralized version all actions are started independently and concurrently by nodes, Lemma 3 does not apply. Mechanisms to recover from network disconnection have not yet been implemented in the algorithm, thus to avoid an accidental disconnection of the network, we *freeze* initial and user-created links. The algorithm is not allowed to remove *frozen* connections, thus we suppose that the information about the status of a link is available on the node.

4.4 Timing

Each node n_i maintains a logical time t_i which is updated by events of incoming and outgoing ants. By using a logical time instead of real time it is possible to adjust pheromone evaporation according to the traffic passing through the node, thus preventing nodes with limited ant flows from clearing their information too rapidly.

4.5 Pheromone Reinforcement and Evaporation

Each pheromone trail τ is reinforced according to the following formula:

$$\tau \leftarrow \tau + \delta$$

where δ is the quantity of pheromone to be laid on the trail. The evaporation process is executed at regular intervals ω, and updates the concentration of a pheromone trail as follows:

$$\tau \leftarrow \tau * \psi$$

for a decay value $\psi < 1$. If $\tau < \epsilon$, for a small value $\epsilon \simeq 0$, the trail is completely removed and its value set to zero.

4.6 Algorithm Phases

There are four phases executed by the algorithm: inform, evaluate, connect, and disconnect. During the inform phase, discovery ants gather information about the network, and pass it to visited nodes. At regular intervals, nodes enter the evaluate phase and determine if it is necessary to connect or disconnect to other peers. Connection and disconnection phases are triggered by decisions made during the evaluation phase.

Inform Phase Discovery ants wandering on the network, collect information into their buffer V and pass it to the node n_i. This data is used to update the alpha table $alpha_i$. For each triple (n'_k, N'_k, t'_k), if $t'_k > \alpha_i[n_k][t_k]$, the entry corresponding to n_k is updated as follows:
$\alpha_i[n_k][t_k] \leftarrow t'_k$
$\alpha_i[n_k][N_k] \leftarrow N'_k$
$\alpha_i[n_k][t_i] \leftarrow t_i$
If α_i does not contain an entry for n_k, a new entry is created.

Evaluate phase At regular intervals ω, each node determines if new connections need to be made and if existing connections are redundant, and thus they can be removed. For the rest of this section, we describe the algorithm from perspective of a node n_i.

Evaluating a Connection To reduce the diameter the network, each node has to compute the distances separating him from other nodes, and check if Rule 1 applies. In the first step, a graph $\tilde{G}$ is constructed using the local information available in the α_i table and the neighbor set N_i. Then, the distance $d_{\tilde{G}}(n_i, n_j)$ $\forall n_j \in \tilde{G} \setminus \{n_i\}$ is computed. For each node n_j satisfying condition (1), a connection procedure is initiated by sending a LinkAnt from node n_i to n_j. Since $d_{\tilde{G}}(n_i, n_j) \geq d_G(n_i, n_j)$, this the rule is valid also with partial data.

Because the maximum observable distance in $\tilde{G}$ depends on the size of the circular buffer carried by the Discovery ant, the length of V l_V needs to be greater or equal to $2D - 1$. The pseudo-code of this evaluation phase is detailed in Algorithm A.8.

Evaluating a Disconnection A graph $\tilde{G}$ based on α_i and N_i is constructed. Because frozen connections cannot be removed, evaluation is based on a restricted neighbor set N_i' , $N_i' \leftarrow N_i \cap \Lambda \setminus \{n_j \in N_i \mid \text{link from } n_i \text{ to } n_j \text{ is frozen}\}$. The set Λ contains all valid keys found in the alpha table, thus N_i' is the set of all neighbors with a non-null entry in the alpha table, and whose connection with n_i is not frozen. For node in $n_j \in N_i'$, the distance $d_{\tilde{G}}(n_j, n_k)$ $\forall n_k \in N_i \setminus \{n_j\}$, is computed.

Definition 4. *Function max'_{id} takes a list of nodes on a path and returns the node with the greatest identifier which has at least a non-frozen link with either its successor or its predecessor on the path. For that, we suppose that in the alpha table not only the neighbors of a node are available, but also the status of each link.*

Because more than one node can detect the same cycle, we avoid the disconnection of the graph by only allowing one node to proceed. Hence, node n_i first checks is 2 holds, and then if $n_i \geq max'_{id}\{p, \ldots, q\}$, where $\{p, \ldots, q\}$ is the detected cycle. If all requirements are satisfied, a UnlinkAnt is sent from node n_i to n_j. As $d_{\tilde{G}}(n_j, n_k) \leq d_G(n_j, n_k)$, we ensure that after disconnection $d(n_i, n_j) \leq D$. Algorithm A.9 shows the pseudo-code of the who le procedure.

Connection A nodes connects n_i to another node n_j by first adding n_j to N_i and then updating the information in the α_i table. Updating the alpha table consists in re-evaluating the distances of all entries knowing that distance to n_j has become 1. As a connection is not performed atomically on both end points, at any moment the network could represent a directed graph.

Disconnection Disconnection from a node n_j is performed by removing n_j from the local neighbor set N_i, thus preventing any ant from migrating to n_j. Additionally, the information about n_j in α_i and of all nodes close to it are updated:

for n_j, the distance is set to the maximum possible distance according to the rewiring rules, i.e. D.

5 Evaluation

In order to evaluate the behavior of the algorithm and the quality of the results, different initial network topologies have been considered; for each case, 1280 iterations have been executed (an iteration represents a complete migration of the whole ant population). Complete tests up to 15000 iterations have not shown significant improvements or changes in the solution. The considered scenarios included a path graph of 1024 nodes, a 2D grid of size 32x32, a hypercube of 1024 nodes, and a LAN of 1281 nodes [2].

To ensure statistical validity of our results, we repeated each simulation run 42 times. Algorithm parameters used during 1all runs is shown in Table 1. As the evaluation took place on reliable networks, the ι parameter (ant respawn interval) was not used, thus ant colonies were created only at the beginning of each simulation run. Along with results, both maximum standard deviation σ_{max} of all topologies at the 1280^{th} iteration, and the maximum mean standard deviation σ'_{max} over all iterations are presented.

Convergence The first goal of the algorithm is to minimize the diameter of the network according to the value of D. As shown in Figure 3, the diameter converges exponentially in all four considered topologies. All results are below the upper bound $2D - 1 = 11$ with values around $D = 6$, and $\sigma_{max} = 0.32$ and $\sigma'_{max} = 1.16$. Because of its topology, *LAN 1281* takes more iterations: this phenomenon can be explained by the low connectivity in the original graph, which requires a longer exploration phase in the initial iterations. It is interesting to note that the hypercube diameter is also lowered, although being already below $2D - 1$: this behavior is due to overestimated distances resulting from local information on each node.

Graph Complexity In order to reduce the diameter, the algorithm should only add the necessary number of logical links, without creating a clique. For this purpose, we measured the number of edges in the resulting graph (Figure 4). The number of edges grows up until the diameter reaches the $2D - 1$ limit; at this point, the number of new edges decreases and the graph becomes stable.

Mean Node Degree and Variance Another parameter that has been measured is the distribution of the number of connections for each node. If the algorithm creates large hubs the overall fault tolerance is lowered, and a single node failure can compromise the whole network. Additionally, hubs concentrate communication and computational loads on a small number of nodes, which is

[2] https://networkx.lanl.gov/browser/networkx/trunk/doc/examples/lanl.edges

unoptimal in a completely distributed environment. Figure shows the evolution of mean degree obtained in each topology, whereas Figure depicts the degree variance. In our simulations, we obtained the smallest standard deviation of 1.44 with the *Hypercube 1024* topology, with an average degree of 12, and a maximum standard deviation of 3.23 in *LAN 1281* with an average degree of 6.93.

Network Load The overall network load is proportional to the number of ants exchanged by nodes during the execution. The initial population of ants is determined by the μ parameter, which was set to 0.15 in our experiments. Because the number of nodes is the same for all scenarios (1024 nodes, except for *LAN 1281*), the size of the initial colony is roughly the same in all topologies. As the algorithms starts to create new connections, new ants of the Link and Unlink species are created, and the population increases. When the optimal diameter is reached, the population starts to decrease, down to ≈ 400 for *LAN 1281* and ≈ 320 for the other three topologies. Such results are compatible with the initial difference in the size of the colonies, as the ratio between the size of the initial and final populations is ≈ 2 for all four topologies. As the number of Discovery ants is constant, half of the final population is composed of Link and Unlink ants instantiated because of over or under estimations of distances in the network.

Dynamic Networks Dynamic scenarios involve networks where new nodes can connect and existing ones quit the system. Even though the BlåtAnt algorithm does not yet implement a mechanism to prevent the partitioning of the network when a node disconnects or crashes, we conducted some initial tests of its behavior in such situations. The dynamic scenario consisted an initial path topology of 100 nodes: every 250 iterations a chain of 25 nodes is added to a random node in the graph, and every 50 iterations a node is removed. In order not to create a disconnected network, the node being removed is carefully chosen. Figure 6 shows the diameter of the network during the simulation: when a new chain of nodes is added, the diameter grows, but the algorithm is able to restore a minimal value after about 100 iterations. Removal of nodes does not seem to significantly affect the distances in the network. Additional tests would allow us to fully understand the dynamics of the algorithm in evolving scenarios.

Parameter	D	α max size	α max age	ψ	δ	l_V	ϵ	ι	μ	ω	κ
Value	6	20	20	0.9	0.1	15	0.005	∞	0.15	10	0.5

Table 1. Simulation Parameters

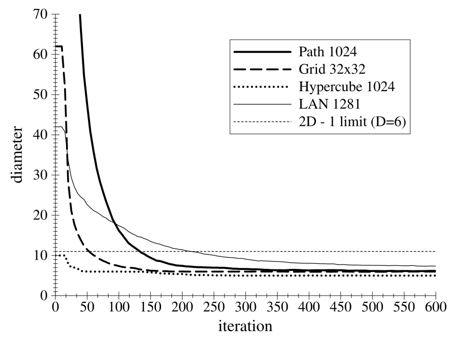


Fig. 3. Convergence of the diameter

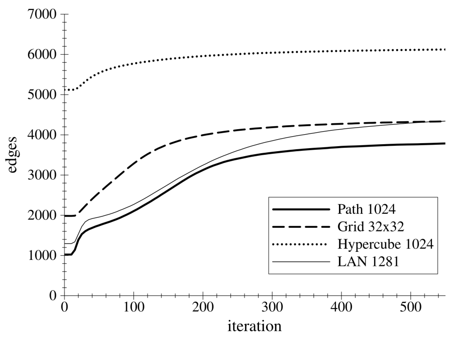


Fig. 4. Number of edges

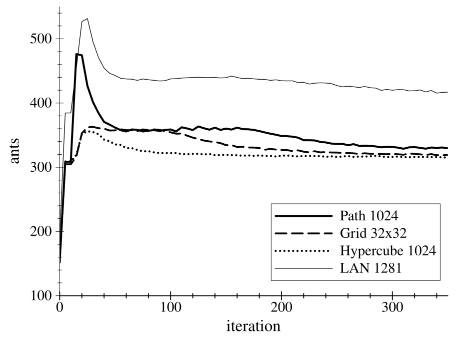


Fig. 5. Ant population size

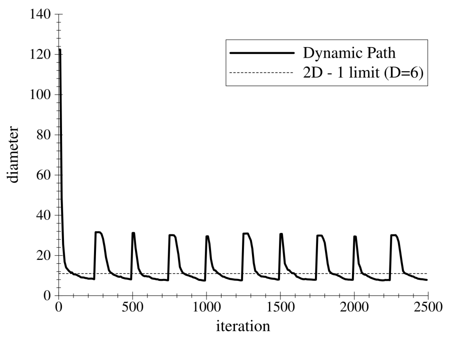
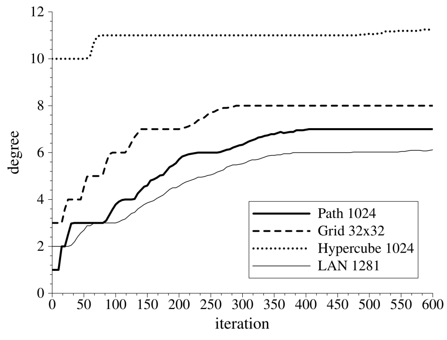


Fig. 6. Dynamic scenario

6 Conclusion

In this paper we presented BlåtAnt, a collaborative and distributed algorithm inspired by ACO, to bound the diameter of a network without requiring a global knowledge. Different species of ants wandering on the network collect and propagate the information that is used to create new logical links. Pheromone trails followed by ants ensure a even coverage of the network, by forcing ants toward underexploited paths. Preliminary support to detect the departure of adjacent nodes is also implemented. The two rules used to create and delete links have been formally proved in a scenario with global knowledge has been shown.

Simulations on different topologies have validated the behavior of the algorithm in a fully distributed environment: the solution converges to an optimal diameter, and the number of edges in the resulting network is also bounded. Evaluation has also determined the average communication cost in term of deployed ants. Through a simple dynamic scenario, the adaptive nature of the algorithm was illustrated, showing its ability to control the diameter of an evolving network.



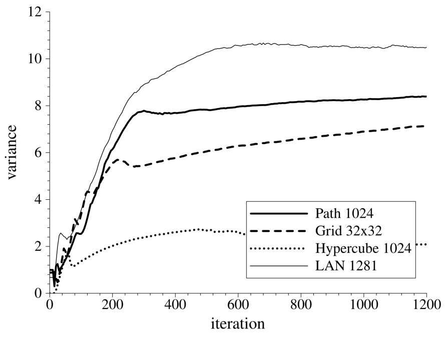


Fig. 7. Mean degree

Fig. 8. Degree variance

Initials results are encouraging, and we believe that BlåtAnt can be the foundation of other algorithms that can exploit the short diameter property of the network. Using an a-priori knowledge of the maximum number of hops separating any two nodes of the network, it is possible to devise optimized resource discovery algorithms, or improve load balancing in grid environments.

Nonetheless there are some issues that are still worth further investigation. The algorithm performed well in the simulated dynamic network, nevertheless a fault-tolerance mechanism is required. Although trying to increase the clustering coefficient may provide some degree of fault-tolerance, a specific recovery method is needed.

Improvement of the algorithm itself will require a better understanding of the influence of each parameter on the solution: fine tuning of each parameter will allow to increase the results and the robustness of the algorithm. In particular, an evaluation of the arrangement of links would allow us to improve the degree distribution of the resulting networks.

Finally, it could be interesting to adapt the algorithm to other kind of network-related optimization problems, such as minimization of roundtrip times, and in a more general way to network problems using weighted connections.

6.1 Acknowledgements

The BlåtAnt algorithm is part of the SmartGrid [7] project, a research supported by the Swiss Hasler Foundation [33] ("ManCom Initiative", project Nr. 2122). The authors would like to thank Michèle Courant for her proof reading of the final initial versions of this technical report.

A Algorithm Pseudo-Code

This appendix presents the pseudo-code of the algorithm. A detailed description is provided in Section 4.

Algorithm A.1: DiscoveryAnt(i)

```
while $True$
  do
    $tell(n_i)$
    $V \leftarrow V + (n_i, N_i, time_i)$
    $p \leftarrow uniform()$
    $N_i^* \leftarrow N_i \setminus \{n_k \mid (n_k, *, *) \in V\}$
    if $p < \omega$
      then $m \leftarrow uniform(N_i^*)$
      else $m \leftarrow minGamma(N_i^*)$
    $migrate(m)$
```

Algorithm A.2: LinkAnt(i, j)

```
if $migrate(n_j)$
  then if $(\alpha_j[n_i][distance] >= 2D - 1)$
  then
    CONNECT$(j, i)$
    if $migrate(n_i)$
      then
        if $(\alpha_j[n_i][distance] >= 2D - 1)$
          then
            CONNECT$(i, j)$
      else
        DISCONNECT$(j, i)$
        $end$
```

Algorithm A.3: UnlinkAnt(i, j)

```
if $migrate(n_j)$
  then
    $N_j \leftarrow N_j \setminus \{n_i\}$
    $\gamma_j[n_i] \leftarrow 0$
    if not $migrate(n_i)$
      then $end$
$N_i \leftarrow N_i \setminus \{n_j\}$
$\gamma_i[n_j] \leftarrow 0$
```

Algorithm A.4: CONNECT(i, j)

```
if $n_j \notin N(n_i)$
  then { $N(n_i) \leftarrow N(n_i) \cup \{n_j\}$
       { alphaUpdateOnConnection$(i, j)$
```

Algorithm A.5: DISCONNECT(i, j)

```
if $(n_j \in N_i) \wedge$ (link from $n_i$ to $n_j$ is not frozen)
  then { $N_i \leftarrow N_i \setminus \{n_j\}$
       { $\beta_i[n_j] \leftarrow 0$
       { $\gamma_i[n_j] \leftarrow 0$
       { alphaUpdateOnDisconnection$(i, j)$
```

Algorithm A.6: alphaUpdateOnConnection(i, j)

```
$\alpha_i[n_j][distance] \leftarrow 1$
$\alpha_i[n_j][neighbors] \leftarrow \alpha_i[n_j][neighbors] \cup \{n_i\}$
re-estimate distances of nodes related to $n_i$
```

Algorithm A.7: alphaUpdateOnDisconnection(i, j)

```
$\alpha_i[n_j][distance] \leftarrow D$
$\alpha_i[n_j][neighbors] \leftarrow \alpha_i[n_j][neighbors] \setminus \{n_i\}$
```

Algorithm A.8: evaluateConnection(i)

```
$G = \langle V, E \rangle \leftarrow$ graph constructed from $\alpha_i$ and $N_i$

for each $p \in V$
  do { $d \leftarrow \alpha_i[node][distance]$
     { $e \leftarrow d_G(n_i, p)$
     { if $(d >= 2D - 1) \wedge (e >= 2D - 1) \wedge (\gamma_i[p] = 0)$
     {   then new $LinkAnt(p)$
```

Algorithm A.9: EVALUATEDISCONNECTION(i)

```
$G = \langle V, E \rangle \leftarrow$ graph constructed from $\alpha_i$ and $N_i$
$G \leftarrow G \setminus \{n_i\}$
$\Lambda \leftarrow \{n_k \mid \alpha_i[n_k] \neq \emptyset\}$
$N'_i \leftarrow N_i \cap \Lambda \setminus \{n_j \in N_i \mid$ link from $n_i$ to $n_j$ is frozen$\}$
$C \leftarrow \{\}$

for each $p \in N'_i$
  do for each $q \in N_i$
    do $d \leftarrow d_G(p, q) + 1$
       if $(d < D) \wedge (i \geq max'_{id}\{p, \ldots, q\})$
         then $C \leftarrow C \cup (p, q, d)$

sort $C$ by ascending $d$
$X \leftarrow \{\}$

for each $(p, q, d) \in C$
  do if $(p \notin X)$
       then new $UnlinkAnt(p)$
            $X \leftarrow X \cup \{p, q\}$
```

References

1. S. Milgram. *The small world problem.* Psychology Today, Vol. 2, pp. 60-67, 1967.
2. P. Erdõs and A. Rényi. *On the evolution of random graphs.* Magyar Tud. Akad. Mat. Kutató Int. Közl. 5, pp. 17-61, 1960.
3. B. Bollobás. *The Diameter of Random Graphs.* Transactions of the American Mathematical Society, Volume 267, Number 1, September 1981.
4. M. Dorigo, T. Stützle, *Ant colony optimization.* The MIT Press, 2004.
5. M. Dorigo, G. Di Caro, and L.M. Gambardella, *Ant algorithms for discrete optimization.* Artificial Life 5 137-172, 1999
6. A. Montresor, H. Meling, and Ö. Babaoglu, *Messor: Load-Balancing through a Swarm of Autonomous Agents.* AP2PC 2002: 125-137, 2002
7. B. Hirsbrunner, M. Courant, A. Brocco, and P. Kuonen, *SmartGRID: Swarm Agent-Based Dynamic Scheduling for Robust, Reliable, and Reactive Grid Computing.* Working Paper 06-13, Department of Informatics, University of Fribourg, Switzerland, October 2006.
8. G. Di Caro, *Ant Colony Optimization and its application to adaptive routing in telecommunication networks* PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004.
9. Duncan J. Watts. *Six Degrees: The Science of a Connected Age*, Norton, W. W. & Company, Inc. Publisher, 2003.
10. G. Di Caro, F. Ducatelle, and L.M. Gambardella, *Swarm intelligence for routing in mobile ad hoc networks*, Proceedings of Swarm Intelligence Symposium (SIS 2005), IEEE, 2005.

11. M.A. Salehi, and H. Deldari, *Grid Load Balancing using an Echo System of Intelligent Ants*, Proceeding (517) Parallel and Distributed Computing and Networks - 2006, Innsbruck, Austria, 2006.
12. D. Wang, *A resource discovery model based on multi-agent technology in P2P system*, Proceedings of Intelligent Agent Technology (IAT 2004), IEEE, 2004.
13. M. Dorigo, L.M. Gambardella. *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.* IEEE Transactions on Evolutionary Computation, 1(1), pp. 53-66,1997.
14. D. J. Watts and S. H. Strogatz. *Collective dynamics of 'small-world' networks.* Nature 393, pp. 440-42, 1998.
15. R. Albert and A. Barabási. *Emergence of scaling in random networks.* Science, 286, pp. 509-512, 1999.
16. J. Kleinberg. *The Small-World Phenomenon: An Algorithmic Perspective.* Proceedings of the 32nd ACM Symposium on Thory of Computing, 2000.
17. P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. *Towards small world emergence.* Proceedings of 18th ACM Symposium on Parallelism in Algorithms and Architectures, pp. 225-232, 2006.
18. P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. *A fully distributed scheme to turn a network into a small world.* Technical Report RR2006-03, LIP - ENS Lyon, 2006.
19. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content-Addressable Network.* Proceedings of ACM SIGCOMM, 2001.
20. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. *Chord: A scalable peer-to-peer lookup service for internet applications.* In ACM SIGCOMM, 2001.
21. A. I. Rowstron and P. Druschel. *Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.* Proceedings of Middleware 2001, pp. 329-350, 2001.
22. G. S. Manku, M. Bawa, and Raghavan. *Symphony: distributed hashing in a small world.* In Proceedings of the 4th Conference on USENIX Symposium on internet Technologies and Systems, vol. 4, 2003.
23. J. Kleinberg. *Complex Networks and Decentralized Search Algorithms.* Proceedings of the International Congress of Mathematicians (ICM), 2006.
24. O. Sandberg. *Searching a Small World.* Lic. thesis, Chalmers University, 2005.
25. H. Zhang, A. Goel, and R. Govindan. *Using the small-world model to improve Freenet performance.* Proceedings of the 21st. Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1228-1237, 2002.
26. R. Akavipat, L. Wu, and F. Menczer. *Small World Peer Networks in Distributed Web Search.* Proc. 13th. WWW Conference, pp. 396397, 2004.
27. I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. *Freenet: A distributed anonymous information storage and retrieval system in designing privacy enhancing technologies.* International Workshop on Design Issues in Anonymity and Unobservability, 2001.
28. K. Y. Hui, J. C. Lui, and D. K. Yau. *Small-world overlay P2P networks: construction, management and handling of dynamic flash crowds.* Computer Networks, 50(15), pp. 2727-2746, October 2006.
29. A. Iamnitchi and I. Foster. *On Fully Decentralized Resource Discovery in Grid Environments*, GRID, 2001.
30. M. Ripeanu, I. Foster, and A. Iamnitchi. *Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design.* IEEE Internet Computing, 6(1), February 2002.

31. V. V. Dimakopoulos and E. Pitoura. *On the Performance of Flooding-Based Resource Discovery*. IEEE Transactions on Parallel and Distributed Systems, 17(11), November 2006.
32. K. Ali, S. Datta, and M. Aboelaze. *Grid resource discovery using small world overlay graphs*. CCECE '05, May 2005.
33. Swiss Hasler Foundation Website, http://www.haslerstiftung.ch/