Paper published at Applied Soft Computing Volume 13, Issue 12, December 2013, Pages 4813–4831 DOI: 10.1016/j.asoc.2013.07.022. This is a pre-editorial draft.

Please cite as: Krynicki, K., Jaen, J., Mocholí, J.A. On the performance of ACO-based methods in p2p resource discovery. Appied Soft Computing 13(12): 4813-4831 (2013)

On the performance of ACO–based methods in P2P resource discovery

Kamil Krynicki, Javier Jaen, Jose A. Mocholi

ISSI – Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain
{kkrynicki, fjaen, jmocholi}@dsic.upv.es

Corresponding author:
Kamil Krynicki
Camino de Vera, s/n, 46022 Valencia, Spain
Email: kkrynicki@dsic.upv.es
Phone: +34 96 387 35 69
Fax: +34 96 387 73 59

Abstract

Over the recent years peer-to-peer (p2p) systems have become increasingly popular. As of today most of the internet IP traffic is already transmitted in this format and still it is said to double in volume till 2014. Most p2p systems, however, are not pure serverless solutions, nor is the searching in those networks highly efficient, usually achieved by simple flooding. In order to confront with the growing traffic we must consider more elaborate search mechanisms and far less centralized environments. An effective proposal to this problem is to solve it in the domain of Ant Colony Optimization metaheuristics. In this paper we present an overview of ACO algorithms that offer the best potential in this field, under the strict requirements and limitations of a pure p2p network. We design several experiments to serve as an evaluation platform for the mentioned algorithms to conclude the features of a high quality approach. Finally, we consider two hybrid extensions to the classical algorithms, in order to examine their contribution to the overall quality robustness.

Keywords

Semantic Search, Ant Colony Optimization, Peer–to–peer

1 Introduction

Since the introduction of email, one of the most successful examples of a large–scale distributed application in its time, the research field of distributed computing [1] has experienced an enormous development. The reasons for using such systems are, firstly, because the nature of the application requires that several computing nodes produce and share data across a communication network and, secondly, because of practical reasons with respect to a centralized system in terms of scalability, reliability or expandability.

Distributed environments have drawbacks however, namely, it is quite difficult to propose a resource discovery mechanism that would be as efficient as the optimum of a centralized depository. It is also more of a challenge to obtain a complete answer, as well as estimate the completeness of it. Even so – the benefits of a distributed environment overweigh the drawbacks and that is why the search for efficient resource discovery and search algorithms is crucial.

These drawbacks are specially challenging in p2p distributed environments. Abstractly speaking, p2p systems are networks of interconnected peers, where some provide resources of any nature whereas others wish to obtain them. The roles of peers (sometimes referred to as nodes) are variable. One example of a real life use of p2p might be en–masse concurrent calculations which have been done with great success. Distributed computing projects, such as SETI@home [2] or Collatz Conjecture [3], peaked at hundreds of teraflops of computing power with relatively low costs. What stimulates the development of distributed techniques is the comparison of these results with super–computers such as ORNL Supercomputer [4], which involve immense investments and oscillate at about ten–fifteen thousand teraflops. Other examples of p2p application are: sharing of storage and content distribution [5] [6] [7], where the desired content is treated as the resource, sharing of bandwidth, streaming or even anonymous communication solutions, both text and VoIP [8].

Nodes in p2p systems are in possession of resources. A query in such a system is, in short, a process of demanding resources from a subset of peers and then returning to the sender peer with results. There are several degrees of distribution in p2p environment to be considered. The most extreme one is the case of p2p systems with a very high distribution degree which implies the following:

1. The content, information or process of global scope is very highly undesirable.
2. The cost of the exchange of data between two peers of the system is considerable. So much so, the system would rather obtain no data than obtain low relevancy data.

In this respect, a remarkable computing strategy to address the problem of effective searching in highly distributed p2p systems has been Ant Colony Optimization (ACO, introduced in [9]). With the query masquerading itself as an ant in search of food and depositing chemical substance as trails, which can be read by other ants, one can achieve a very good implementation of p2p search. The biggest benefits of the use of ACO are: no (or a very small amount of) global information, generic nature, quick convergence to near–optimal solution and robustness in terms of system load.

There has been several ACO proposals addressing this issue (Ant Colony System – ACS [10], Min–Max [11], Neighboring–Ant Search – NAS [12], Semant [13]), albeit there are factors whose impact has not been thoroughly studied such as the existence of long distance connections between peers in unstructured environments and the consideration of hybrid strategies that take advantage of underlying structured topologies. Therefore, in this paper we will show that there is still room for improvement in the area of ACO based p2p search systems and we will propose an implementation of a p2p version of ACS which is competitive in both unstructured network topologies with varying number of long distance connections and structured hypercube ones if a hybrid strategy is defined.

In section 2 we will analyze the problem in detail and describe the use of ACO in p2p search. It will contain the mathematical base and the concepts to consider; we also present motivations for choosing specific algorithms for the comparative study. In section 4 we will propose the experimental dimensions to be analyzed, the designed experiments and present their results with comments. Finally, in section 5 we will formulate the final discussion.

2 Ant colony optimization

Ant Colony Optimization is a swarm intelligence approach to problem–solving introduced by Marco Dorigo in his work on distributed optimization in 1991 [14]. The core idea of ACO is twofold, firstly – as properly named – it uses a swarm of simple and stochastic automaton to solve complex problems and, secondly, the communication between these is through stigmergy and therefore indirect. Such a communication method has shown to provide interesting results, especially with the emphasis on finding the shortest path [15] or paths optimizing a given function [16] [17]. The automaton, or agents, in ACO are called ants. Each ant has the simple task of finding the required resource (search phase) and bringing it back to its nest (returning phase); without the loss of generality one can limit the world, in which ants live, to a bidirectional graph of finite size with nodes representing possible locations of resources and edges representing trails.

The ants in the search phase are referred to as Forward Ants, while ants in the returning phase are referred to as Backward Ants. All ants follow a simple and non–deterministic search algorithm that can be summarized in the following (the micro–scale algorithm – ant's behavior):

Algorithm

1. Consider the Ant A_1 that finds itself in a node n_e (emitting node) of graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, with the task of finding the set of resources $\{r \mid r \in q(G)\}$, where $q(G)$ is a perfect response of the graph G to the query q.
2. A_1 checks the node in which it currently resides for the presence of resource r.
3. If a resource r is found it is added to the ant's basket.
4. If A_1 has found enough resources to fill its basket or any other pre–established condition or set of conditions $D(A)$ holds true – A_1 proceeds to step 8.
5. A_1 performs the step transition based on available, local knowledge: $ST(A_1, n_j, q)$. Being in node n_j, it chooses node $n \in \{n_i | (n_j, n_i) \in E\}$ as the next destination. Adds n to the stack of nodes visited $n(A_1)$.
6. A_1 performs Local Pheromone Update – metaphore of pheromone evaporation.
7. Proceed to step 1.
8. A_1 converts from being a Forward Ant to being a Backward Ant.
9. (A_1 performs optimization of the stack $n(A_1)$) – optional, indication of a hybrid ACO.
10. A_1 performs an evaluation of the trail found based on a quality measure function $QM(A_1)$.
11. A_1 returns to the emitting node following the stack $n(A_1)$, at every step performing an update of the locally stored pheromone trails using Global Pheromone Update rule – metaphore of pheromone deposition.
12. When A_1returns to the emitting node, it deposits found resources from the basket and the algorithm concludes.

End Algorithm

As it can be deduced from the micro–scale algorithm, there are several key factors that define ACO algorithms:

1. Graph topology (or the lack of it).
2. State Transition function $ST(A, n, q)$, where A stands for ant, n is the current node and q is the carried query.
3. Local Pheromone Update function – the model of pheromone evaporation.
4. Global Pheromone Update function – the model of depositing pheromones after the search concludes.
5. Quality Measure function $QM(A)$, where A stands for ant. Here, in fact, the ant is the evaluated element – seeing how it represents the query, the route over the graph and the resources found.
6. Query completion requirement $D(A)$, where A stands for ant.
7. Post–processing algorithms – such as route optimization, loop detection and removal, graph topology exploitation, etc.

Consequently, the macro–scale algorithm (the system's large scale behavior) for ACO p2p search could be defined as follows:

Algorithm

1. Query q is requested upon a node n.
2. Bring to life a Forward Ant FA_q in the node n and supply it with q.
3. Let FA_q perform the micro–scale algorithm.
4. Until δt time units have passed, consider the q resolution pending.
5. If Backward Ant BA_q is received in the node n after less than δt time units have passed, consider the basket of BA_q the graph's response to q and dispose of BA_q.
6. If BA_q is not received within δt time units, consider the q resolved with no results.

End Algorithm

Taking as basis the previous generic algorithmic schema, the definition of an ACO–based query resolution algorithm in p2p environments must conform to the following additional query–resource (q–r) principles for it to be considered p2p compliant:

1. Every node may have any amount of resources – including zero resources.
2. Every node may issue a query – that is, a request for a set of resources of any nature; one that may be constructed of resources residing in one or many nodes within the network.
3. Every node may not be aware of the content of any other node but itself.
4. Every node must be connected to a set of nodes via bidirectional links of high traveling cost. A Degenerated (disconnected) node may be connected to zero other nodes.
5. Every query is propagated among nodes, collecting resources that correspond to the request issued.
6. The destination (the final) node of a query is never known a priori nor is it deterministic.
7. The trail of a query is never known a priori nor is it deterministic.

We formulate the above list a priori, based on common-sense and general requirements. It will serve to filter out algorithms that have no applicability in the field of p2p. Omitting of any or all of these principles is possible. Such a system would, however, suffer from lower generality and it would be incomparable to the real world p2p networks. Once the generic approach for ACO p2p searching has been introduced we can discuss, within the dimensions mentioned above, some of the more prominent ACO algorithms proposed in the literature. In the following subsections we will describe in moderate detail some of the principal ACO and ACO-like algorithms and then formulate the subset of those best applicable for p2p and proceed to in–detail study.

2.1 Ant Colony System

Ant Colony System [9] is one of the most popular implementations of the ACO metaheuristic. It is an extension and improvement over the Ant System (AS) [14]. It has been chosen as the principle candidate for p2p search.

The state transition function of ACS is a very elaborate mechanism that consists of two phases: exploitation (2.1) and exploration (2.2). A randomly chosen value q is compared against q_0 – a fixed parameter.

If $q < q_0$, the exploitation phase is executed,

$$s = \begin{cases} argmax_{u \in J_{k(r)}}\{|\tau(r,u)| \times |\eta(r,u)|^{\beta}\}, & if\ q < q_0 \\ S, & otherwise \end{cases}, \qquad (2.1)$$

where $\tau(r,u)$ is the pheromone level deposited between nodes r and u, $\eta(r,u)$ is the cost of the transition between the nodes r and u, $J_{k(r)}$ is the neighborhood of the node r and β is a parameter that adjusts the weight of the cost versus pheromone ($\beta > 0$). In this case s is the next node to visit and it is deterministic.

If $q \geq q_0$, the exploration phase is executed,

$$p_k(r,s) = \begin{cases} \dfrac{|\tau(r,s)| \times |\eta(r,s)|^{\beta}}{\sum_{v \epsilon J_k(r)} |\tau(r,v)| \times |\eta(r,v)|^{\beta}}, & if\ s\ \epsilon\ J_k(r) \\ 0 & otherwise \end{cases} \qquad (2.2)$$

Every r outgoing link is assigned a weight p_k expressed by (2.2) and the result (the next node to visit) is chosen randomly with the weights taken in consideration.

Equations (2.3) and (2.4) express the concept of pheromone deposition and pheromone evaporation respectively.

$$\tau(r,u) \leftarrow (1-\alpha)\cdot \tau(r,u) + \alpha \cdot \delta\tau(r,u,ant), \tag{2.3}$$

$$\tau(r,u) \leftarrow (1-\rho)\cdot \tau(r,u) + \rho \cdot \Delta\tau(r,u), \tag{2.4}$$

where α and ρ are parameters that reflect the influence of the new pheromone value ($\alpha\epsilon\langle 0,1\rangle$ and $\rho\epsilon\langle 0,1\rangle$), $\delta\tau(r,u,ant)$ is a function that measures the quality of the route covered by the FA ant in order to adjust the amount of pheromones to be deposited, while $\Delta\tau(r,u)$ is calculated by:

$$\Delta\tau(r,u) = \gamma \cdot max_{z\epsilon J_k(r)}\tau(u,z), \tag{2.5}$$

where γ is a parameter with which one can tune the value of $\Delta\tau(r,u)$.

The quality measure ($QM(A)$) is expressed by:

$$\delta\tau(r,u) = \begin{cases} \frac{1}{|ant.h|}, & if\ |ant.h| > 0 \\ 0, & otherwise \end{cases}, \tag{2.6}$$

where, $ant.h$ is the length of the route covered by the FA ant. At the moment of deposition and evaporation the pheromone values can be capped by ph_{max} – maximum value, as well as ph_{min} – the minimum value. Every trail has the ph_{init} pheromone value before any ant affects it.

The query completion is achieved by either collecting between R_{min} and R_{max} resources or making ttl steps (time to live – the maximum amount of state transitions); formally:

$$D(A) = \begin{cases} 1, & if\ r\ \epsilon\langle R_{min,}R_{max}\rangle or\ h \geq ttl_{max} \\ 0, & otherwise \end{cases}, \tag{2.7}$$

where r is the amount of resources found and h is the amount of steps taken.

If no resources were found the ant has a choice whether to finish the algorithm empty or perform the route back to the emitting node, without any pheromone updates and inform the node about the failure. For our purposes we chose the latter solution.

2.2 Among other extensions of Ant System the Min–Max [11] must be named. It uses explicit minimum and maximum values for pheromones on each edge as well as several minor tweaks in the State Transition and Quality Measure functions.Semant

The Semant algorithm [18] is our second candidate for p2p search, it uses a very similar approach to the classical ACS, however it adds several extensions. One of the most prominent is the use of a 2–dimensional pheromone table stored in every node, that is a ($keyword, outgoing\ link$) pair, rather than the typical 1–dimensional pheromone per outgoing link. This can be understood as an additional layer (overlay) of pheromones per every taxonomy entity used in the query routing. For more details on the concept, and our variation of it, see section 3.1.

Semant maintains the exploration–exploitation dilemma approach from the ACS. The exploitation is now expressed as in (2.8)

$$s = argmax_{u\in J_{k(r)}}\{|\tau_{cu}| \times |\eta_u|^{\beta}\}, \tag{2.8}$$

where τ_{cu} is the pheromone level from the current node to the u node within the c concept overlay and η_u is the cost of traveling to u. The result s is the next node to be visited, and, as in ACS, it is deterministic. The major change is the exploration phase – every edge that origins in the current node is assigned a probability $p_{cr}\epsilon\langle 0,1\rangle$ according to (2.9) but in this case a resolution of probability is done per link, which means that p_{cr} is the probability that the r destination node in the c concept overlay will be returned as the next step:

$$p_{cr} = \frac{|\tau_{cr}| \times |\eta_r|^{\beta}}{\sum_{v\epsilon J_k(r)}|\tau_{cv}| \times |\eta_v|^{\beta}} \tag{2.9}$$

The consequences of such an approach are twofold: firstly, there might be more than one link as a result of this, and secondly, there might be no links. In the first case, the original FA is sent to one of the chosen links and a clone of the FA, called FA^{ci}, will be sent to every i–th link, i>1. In the second case, a result will be obtained by falling back to the exploitation phase. This behavior is formally described by equation (2.10) and constrained by (2.11):

$$GO_j = \begin{cases} 1, & if\ p \leq p_{cj} \\ 0, & otherwise \end{cases}, \tag{2.10}$$

$$\sum\nolimits_{j \epsilon J_k(r)} p_j = 1, \tag{2.11}$$

where GO_j is a function that expresses the fact of an ant (or its clone) choosing to go to the j node (value 1) or not (value 0) and p is a random variable.

The pheromone management is also different to the ACS approach. The pheromone deposition is a linearly growing function, that is, the act of dropping n units of pheromone will increase the value by n (compare equation (2.3) to (2.12)). Hence the maximum value of $\tau(r,u)$ is more of an issue to consider.

$$\tau(r,u) \leftarrow \tau(r,u) + \delta\tau(r,u) \tag{2.12}$$

In (2.12) $\delta\tau(r,u)$ represents the quality measure and is calculated as:

$$\delta\tau(r,u) = \mathrm{w_d} \cdot \frac{|R|}{R_{max}} + (1 - w_d) \cdot \frac{ttl_{max}}{2 \cdot h}, \tag{2.13}$$

where w_d is a parameter that expresses the balance between both components of the equation ($w_d \epsilon \langle 0,1 \rangle$), R is the amount of resources found, R_{max} is the maximum resources allowed, ttl_{max} is the maximum number of steps allowed and h is the number of steps taken. Here, as in ACS, the pheromone levels can be limited by ph_{max} and ph_{min}. Since Semant uses linear growth of pheromone, instead of weighted growth, the ph_{max} must be set to a very high value, in order to avoid issues with having all the paths at its maximum value –thus not providing any information.

The evaporation process is very similar to ACS and somewhat simplified:

$$\tau(r,u) \leftarrow (1 - \rho) \cdot \tau(r,u) \tag{2.14}$$

The query completion is achieved in an identical manner to ACS (2.1).

2.3 Neighboring–Ant Search

Another proposition of an extension of the basic AS was proposed by C. Gómez Santillán et al. [12]. It is based on exploiting the node distribution and several look–ahead heuristics. For in depth look consult the work [12]. Here we will focus on the pseudocode governing the behavior of Neighboring–Ant Search (NAS), provided by the authors of NAS:

Algorithm

```
1.  for each query in $r_k$ create a search agent k with $TTL_k$= $max_{TTL}$ and $Hit_{sk}$=0
2.  WHILE $Hit_{sk}$ < maxResults and $TTL_k$ > 0
3.  // Phase 1: The evaluation of results
4.  IF the unvisited $s_k \in \{r_k \cup \Gamma(r_k)\}$ has the searched resource
5.  $r_k = append\ s_k\ to\ path_k$
6.  $Hit_{sk} = Hit_{sk} + 1$
7.  Local Pheromone Update
8.  Global Pheromone Update
9.  else // Phase 2: The state transition
10. IF $r_k$ is a leaf node or does not have an unvisited neighbor
11. remove the last node from $path_k$
12. ELSE $s_k$ = apply the transition rule with the DDC function
```

```
13. $r_k = append\ s_k\ to\ path_k$
14. Local Pheromone Update
15. $TTL_k = TTL_k - 1$
16. kill the search agent
```

End Algorithm

At step 5 clearly the NAS algorithm takes advantage of basing its routing decisions on the content of the neighboring nodes. This, yet again, violates the required 3 of the q–r principles. Furthermore at steps 11 and 12 it permits removing nodes from the path hence improving the overall quality measure by simulating the path shorter than it actually was. And finally, NAS generates a BA (called retrieval agent) at every occurrence of a resource, putting a great additional load on the system. A remark is made in [19], where Michlmayr notices that the less resources a single agent carries (the R_{min} variant of Semant, described in 2.2) the better overall score of the results obtained, but the smaller the value of a single query. In other words: there is an increase of measured quality (which will be defined formally in the section 4.2) but at the cost of the real value for the user – less results at a time; and for the system – more load. All these factors contribute to the fact that in [12] NAS achieves results better by approximately one order of magnitude and simply it is not comparable with an ACO algorithm of a more pure nature. The fact of examining the content of neighboring nodes should improve the results by a factor of average node degree, that is, an average of the degrees of all the nodes in the system. This is because within, what is calculated as one step, they analyze all the neighbors, therefore making several steps in one; in the terms of means this translates into making $n = average\ node\ grade$ steps and later reporting it as one step. Worth mentioning is the fact that NAS guides the FA (search agents) towards nodes with high degrees – further exploiting the proposed approach.

2.4 Random k–walker

The most straightforward algorithm is the k–walker explored in detail in [20]. It has to be emphasized that it is not an ACO algorithm, but it serves as a good benchmark, a reference in a given test; it is also proposed as such in the experimental study of Semant [19], which we follow closely in order to maximize the fidelity of its result recreation. Moreover, the random behavior is a firm minimum performance expectance; a way to discard an algorithm if it fails to surpass it in every measure. The reason why it has been included in this section is that it can be easily expressed in the ACO's terms, degenerated, but valid and follow the general flow of ACO. By doing this we also show how we implemented it using our ACO testing middleware.

The state transition consists of one phase – the dilemma of exploration versus exploration is removed. On the first step k–walker generates k forward ants and each makes a random decision on how to continue– on every other it simply takes a random decision (2.25). The probability of choosing to go from the node r to the node s is $p_k(r,s)$ (2.26).

$$s = \begin{cases} \{S\} & , \quad if\ |h| > 0 \\ \{S_1, \dots, S_k\}, & if\ |h| = 0' \end{cases} \tag{2.15}$$

where h is the number of steps taken.

$$p_k(r,s) = \begin{cases} \dfrac{1}{|J_k(r)|}, & if\ s \in J_k(r) \\ 0 \quad , & otherwise \end{cases} \tag{2.16}$$

The pheromone management is not relevant because $\tau(r,u)$ does not appear in (2.26). Therefore for evaporation, as well as deposition are:

$$\tau(r,u) \leftarrow \tau(r,u), \tag{2.17}$$

which indicates that there is no pheromone evolution.

The query completion is achieved in an identical manner to ACS (2.1).

2.5 Other algorithms

Other ACO algorithms that are highly worth mentioning are the following:

1. AntNet [21]
2. AntHocNet [22]

3. Ant Based Control [23]

Their core ideas however do not fit our established q–r principles. The AntNet and AntHocNet are mainly used for packet routing where the destination is well known and only the path is to be discovered. This stands in high contrast to the q–r principles (point 6), in which the destination in not known but merely described by the combination of query, resources and the algorithm. Ant Based Control, on the other hand, is used in circuit switching environments.

For the sake of completeness another approach deserves a mention – it is the most straightforward and most redundant approach of all – namely: the k–flooding [24]. It has been used for many years in the Gnutella protocol [5] and it is basically sending the query to all the neighbor nodes until the k–th depth. Flooding has a somewhat limited variant; called t–top k–flooding, in which case the flood is sent to only the t best neighbors. As shown in the work by Jun–qing Li et al. [25] they are vastly inefficient compared to ACO and will not be considered in this work.

2.6 Summary of the algorithms

Once the main algorithmic ACO–based approaches for solving the proposed problem have been presented, the following comparative table relates them according to some dimensions that are important for selecting the candidate algorithms that will be included in our experimental study.

	Evolutive	Deterministic	Semantic Overlay	Look Ahead	Hybrid	q–r compliant
ACS	yes	no	no	no	no	Yes
Semant	yes	no	yes (taxonomical)	no	no	Yes
NAS	yes	no	no	yes (various)	no	no (violates pt. 3)
k–Random Walks	no	no	no	no	no	Yes
AntNet	yes	no	no	no	no	no (violates pt. 6)
AntHocNet	yes	no	no	no	no	no (violates pt. 6)
Ant Based Control	yes	no	no	no	no	no (violates pt. 6)
k–flood	no	yes	no	no	no	no (violates pt. 7)
t–top flood	no	yes	no	no	no	no (violates pt. 7)

Table 1 Comparison of classical algorithms

From Table 3 we can easily conclude that in the pure form only three algorithms qualify, in terms of q–r principles, for further study. This was the reason why we chose to increase the test sample by introducing a set of extensions to the classical approach. In the section 3 we present the mentioned extensions in detail.

3 Proposed ACO extensions

In order to adapt ACS closely to the requirements of p2p environments we have decided to introduce two extensions that can be used separately or combined at will: a semantic and a hybrid extension. The semantic extension is based on that proposed by Semant's authors and the hybrid extension is aimed at exploiting hypercube topology.

3.1 Semantic Extension: Routing Concept

The first extension is the notion of the Routing Concept, mentioned already while discussing Semant in section 2.2. Its theoretical base was presented in [26], where the idea of several Overlay Networks superposed over the Physical Node Network is introduced.

Every node n keeps a 2–dimensional matrix $\Omega\colon N_n \times R_n$, with real, positive values, where N_n is the space of outgoing links from the node n and R_n is the space of routing concepts maintained by this particular node n. This matrix is referred to as routing table, or routing matrix. The n'–th, r–th element of Ω corresponds to the pheromone value of the n'–th outgoing ling for the r–th routing concept, which can be written as $\Omega_{n'r} = \tau$. There are two functions defined:

1) $ph_U(n', r, \tau)$, that establishes the value $\Omega_{n'r}$ for the $n' \in N_n$ and $r \in R_n$ as τ
2) $ph_R(n', r)$, that returns the value $\Omega_{n'r}$ for the $n' \in N_n$ and $r \in R_n$.

In both cases, if $r \notin R_n$, the matrix is redefined as $\Omega^{\mathsf{T}}\colon N_n \times \ {R_n}^{\mathsf{T}}$, the space of routing concepts is redefined as ${R_n}^{\mathsf{T}}\colon R_n \cup r$, and $\forall n' \in N_n \ \{\Omega_{n'r} = ph_{init}\}$ where ph_{init} is the initial value of the pheromone. Both functions are undefined for $n' \notin N_n$. Node n, at initialization, has $R_n = \{r_{def}\}$, and $\forall n' \in N_n \ \{\Omega_{n' r_{def}} = ph_{init}\}$, where r_{def} is the default routing concept. In other words: if the requested routing concept is not present in the routing table the table is extended by adding a new row for the missing routing concept all with initial values. Additionally, the routing table maintains the pheromone value only for the immediate neighbors.

Notice that it stands in contrast to the AntNet [21] algorithm, where the second dimension in the routing table is also used, but it monitors all the accessible nodes from a given node n, both directly and indirectly: $(node, outgoing\ link)$. Such an approach would be of not good, both memory– and efficiency–wise, in a p2p environment due to the typical size range and high dynamism.

Routing concept is a generalization of the pheromone–per–keyword approach. Firstly, the routing concept does not have to be a keyword but any type of distinction will serve, such as a taxonomy of entities, numerical values, etc. Additionally, it allows the query to choose, at every step, into which overlay network it should be injected, rather than have one fixed at query's creation. The routing table dimension in each node can grow and shrink at will, without any a priori limitations. If no routing concept is chosen the default routing concept will be used – eliminating the Overlay Network concept for a given query. If the routing concept remains unchanged during the querying process the approach is reduced to the one described in Semant. ACS can be extended easily by adding the routing concept functionality.

3.2 Hybrid Extension: Hybrid Route Optimization

In its most general approach hybrid setups are very complex and elaborate systems of coordinated algorithms of mutual interaction. In order to properly apply hybrid extension we, firstly, faced design a decision, as there are several large classes of hybrid techniques. According to [27] the most suitable hybrid class for evolutionary–class metaheuristic is HRH (high–level relay hybrid). In such a setup algorithms that form components of the hybrid are self–contained and sequentially executed, forming processing (initial phase) and postprocessing (intermediate and final phases). As the author mentions, coarse grain evolutionary components, such as ACO, are not suitable for finding near–optimal solutions under difficult conditions. Local search technique is a good complement in this case.

Another view of Hybrid taxonomy is provided in [28]. There the author establishes a slightly more in–depth, yet similar, taxonomy based around four key questions: the class hybridized components (metaheuristics, search techniques, seeding techniques etc.), the level of hybridization (weak, strong), the order of execution (parallel, interleaved, batch) and control strategy. In this light ACO–applicable hybrid systems would be denominated weak–coupled, batch, integrative solution, similar to the previous suggestion of HRH. The author also decomposes the hybrid strategies into level–by–level processing approaches, where four elements are extracted: OF – output function, IM – improvement method, SCM – solution combination method and IF – input function, which can be depicted as

OF+IM+SCM+IF. The ACO itself is fully expressed in the above terms; we, however, add the TRO (Taboo Route Optimization, see below) in the SCM phase.

Our attempted solution is akin to the one by Duan et al. [29], which propose the following structure: DE+HS+HJ, where DE is the Differential Evolution algorithm representing the evolutive component (ACO in our case), HS is the Harmony Search – which has been omitted in our approach, and finally HJ is the Hooke and Jeeves direct search method, a Local Search Algorithm that performs the final refinement. For HJ we use a domain–and–topology–bound solution which we named TRO.

The TRO consists of path shortening with the assumption of an underlying hypercube topology and exploits the fact that optimal paths in hypercube–based networks are well–known and solved problems. During the process of converting a forward ant into a backward ant the route optimization will be executed. The Taboo Route Optimization draws its name somewhat from the Taboo Search [30], as they have several concepts in common. In our case the taboos, however, are not solutions but components of the solutions to be maintained in the final result. Also they are not established within the search process, but injected in the initial step.

TRO consists of two phases:

Algorithm

1. Nodes within the path p

$$p\colon [n_1, n_2, \dots, n_N] \tag{2.1}$$

   that was covered by a forward ant FA, will be analyzed and all that have provided required resources will be marked as *taboo*. The first and the last node will be marked as well. As a result we obtain the path p^t

$$p^t\colon [n_1^t, \dots, n_{nr_i}, \dots, n_{r_i}^t, \dots, n_N^t], \tag{2.2}$$

   where:
   - n_{nr_i} is read as the i–th node that has not provided any resources, and
   - n_{r_i} is read as the i–th node that has provided resources.

   A subpath sp^t will be composed of only marked nodes n_i^t of the path p^t and nodes will be renamed n_{sp_1}, n_{sp_2}, etc.

$$sp^t : [n_{sp_1}, n_{sp_2}, \dots, n_{sp_M}] \tag{2.3}$$

2. For every pair $n_{sp_i}, n_{sp_{i+1}}$ of nodes form the path sp^t, a topology–based optimal path between them will be found, named p_i^{i+1} and expressed as a sequence of nodes. The path resolving is performed according to the standard deterministic routing approach [31]. If p_i^{i+1} is shorter than the number of nodes interposed between $n_{sp_i}, n_{sp_{i+1}}$ in the original path p, the appropriate section within p will be replaced by p_i^{i+1}; in other case, p_i^{i+1} will be discarded.
3. Once the process is complete the newly created path will be named p_{optim} and will replace the original path p provided to the backward ant BA by the forward ant FA; thus BA will follow the path p_{optim} on its way to the emitting node.

Algorithm End

All the backward ant pheromone duties will be performed on the way as shown in the section 2.

3.3 Summary of the extended algorithms

With the use of the extension we created four new algorithms, based on those selected in Section 2.6. All of which compliant with our q–r principles.

	Evolutive	Deterministic	Semantic Overlay	Look Ahead	Hybrid	q–r compliant
Semantic ACS	yes	no	yes (routing	no	no	Yes

			concept)			
Hybrid–Semantic ACS	yes	no	yes (routing concept)	no	yes	Yes
Hybrid Semant	yes	no	yes (taxonomical)	no	yes	Yes
Hybrid k–Random Walks	no	no	no	no	yes	Yes

Table 2 Comparison of extended algorithms

4 Experimental Study

Based on our previous comparative study, we decided to choose for further testing only those algorithms that were compliant with our established q–r principles (see section 2). Of those that remain, the pure ACS was also rejected due to the fact that it would inevitably score less than the Semantic ACS, this way we place Semant and Semantic ACS on equal footing. In the following section, we will describe the network topologies, the quality metrics used and the test setup that were used in the experimental study.

4.1 Network topologies

In its purest form the ACO algorithms do not use any additional path processing. Nevertheless the idea of local path optimization was introduced early, in [9] to improve both: the speed of the path convergence, as well as the quality of the solution obtained. One of the approaches is to use advantages the topology may provide; some node topologies include ring [32], toroid [33], hypercube [34] and others in which corresponding local path optimizations apply. In this respect, we will verify whether the topology has no impact on either the speed or quality of convergence unless followed by a local optimization algorithm that uses it explicitly as it has been stated in [19] [12] by taking into consideration the three topologies described next.

4.1.1 Semant topology

The world consists of n nodes, where n is an even number. The nodes are organized in a fully connected grid with a toroidal topology [35], where both dimensions d_1, d_2 of the creating rectangle are chosen to fulfill $|d_2 - d_1| \cong 0$ to minimize the medium distance. Additionally, every node n_1 has one long distance connection (LDC) that connects it directly to another node n_2 with the toroidial distance $\|n_2, n_1\| > 2$. The probability of a node n_1 having LDC of length *len* is proportional to len^{-1}. The above description is taken directly from the guidelines in [18]. This kind of world will be named *sem–n,* where n stands for the number of nodes. In the [19] and [12] the *sem–n* world is approached as if unstructured. The average degree of a node is

$$av(sem - n\,) = 5 \tag{3.1}$$

4.1.2 LDC topology

This kind of world resembles the *sem–n* world in every detail with the exception of LDC connections. In the *sem–n* world every node has exactly one LDC connection, while in this world there will be extra *m* LDC connections distributed randomly and evenly among all the nodes. The length–wise distribution of LDC connections from *sem–n* applies. This kind of world will be named *ldc–n–m* where *n* stands for the number of nodes and *m* for the number of additional LDC links. The average degree of a node in *ldc–n–m* is:

$$av(ldc - n - m\,) = \frac{5n + 2m}{n} = 5 + 2\left(\frac{m}{n}\right) \tag{3.2}$$

Note that:

$$sem - n \equiv ldc - n - 0, \tag{3.3}$$

and consequently:

$$av(sem - n) = av(ldc - n - 0\,) \tag{3.4}$$

4.1.3 Hypercube topology

The hypercube world is a hypercube manifold of degree *d* [36]. Therefore it will have $n = 2^d$ nodes. This kind of world will be named *hc–d*, where *d* stands for the degree of the world. In this case the average degree of a node is, unsurprisingly:

$$av(hc - d\,) = d \tag{3.5}$$

Additionally note that:

$$av(hc - 10) \cong av(ldc - 1024 - 2400) \quad (3.6)$$

4.2 Quality measures

We have decided to adopt a common efficiency quality measure which is widely used by several authors such as in [18] and [10], where it is defined as a Hop per Hit (dimensionless) ratio; hop is the number of steps taken by an agent and hit is the number of resources found and it reflects rather well the quality of a resolution of queries. It is, yet again, measure taken from the Semant study [19], which we must use in order to remain comparable. What is irrelevant in our testing is the absolute execution time. In the real setups the evolution takes place in the span of days, even weeks. So in our case it is, for all practical purposes, highly accelerated and the execution time transmits no information. One might argue the importance of the time factor in an attempt to solve a local problem by applying ACO algorithm; it is, however, not our case. In the field of p2p query routing the true value is how quickly the system evolves in terms of iterations, rather than time units. Moreover, it is important to add supplemetary views in order to fully understand the undergoing processes. These will be provided by two additional metrics: Hit per Ant (dimensionless) and Ants (dimensionless).

Hit per Ant reflects the amount of resources found by a single agent. This permits to compare easily multi–ant algorithms with single–ant ones. There is a question to consider here – namely: with comparable Hop per Hit values, could low Hit per Ant be considered inferior to high Hit per Ant? We argue that the answer to this question is affirmative, because a low Hit per Ant value reflects the fact that each agent finds a small amount of resources. In consequence, in order to achieve a comparable Hop per Hit value an algorithm with a low Hit per Ant ratio will have to use more ants in the process – which will directly affect the system load in a p2p environment. To better understand compare ten ants, each performing one step with one ant performing ten steps.

On the other hand, the amount of ants used, denoted as Ant, which could be read as Ants per Query, shows how many ants are created by a single request. This measure can be used to analyze whether the system evolves towards using less ants, which is a favorable situation if a scalable p2p system is to be capable of processing a vast number of queries per unit of time. In this respect, we will consider a forward and a backward ant as separate beings so the absolute minimum for this measure is two ants if a backward ant's creation is forced and one, if it is not.

4.3 Experiment Setup

Every test will consist of an amount n of queries of random nature, with a given taxonomy (see 4.3.1), released from random nodes within the given world. The query will be propagated following the rules of the algorithm that is being tested (see 2). For every query a set of data will be stored: the birth (creation) nanosecond, the death nanosecond, the query as text, the number of hops made, the number of resources found and the location of resources found. Based on this we will sort the full data, collected over the n iterations, by birth nanosecond, calculate the quality measures (see 4.2) and present the results as graphs. The amount of queries will be fixed at $n = 100000$. Each test run will be repeated three times to assure consistency. The decision to limit the execution repetition at three was taken due to time and disk space constrains. The full set of crude data, as it is, occupies more than 60 Gb of disk space and is a result of more than 250 hours of pure processing time. Additional limiting factor was a high consistency of independent executions leading us to believe that more repetitions would not improve accuracy.

Our testing platform is a highly configurable Java–based engine that supports all the above algorithms. Tests will be run on Intel Pentium 4 630 at 3.00GHz with 4 GB of ram on a 32bit Windows 7 machine.

The scalability of the solution is not an issue to be addressed in our case, as all the real-life implementation will be very highly distributed and slow-evolving. While testing the limits of our software in the mentioned machine we managed to easily generate graphs of up to 60000 nodes and release onto them more than one million ants. A typical user would own not more than several nodes and process only singular ants at a time.

4.3.1 Taxonomy and resource distribution

As in [18], the ACM Computing Classification System [34] will be the taxonomical vocabulary used. Every resource in the network N is described by one, and only one leaf taxonomical concept t (referred to as the taxonomical entity) of the ACM classification. A resource has therefore only two properties – its owner node n and a taxonomical label t. It is depictured as $r(n,t)$. It must be pointed out that two resources $r_1(n_1,t_1)$ and $r_2(n_2,t_2)$, unless explicitly $r_1 = r_2$, are not considered equal, even if $n_1 = n_2$ and $t_1 = t_2$. The consequence of such an approach leads to valuing higher those nodes that provide many resources of the same t.

The distribution of resources within the network follows strictly the approach by Semant test setup [18]. The resources are evenly distributed among the nodes, as well as among the entities in the taxonomy tree. Additionally, every node is a designated expert in a given field (there can be multiple experts in each field) which is expressed by the composition of resources in it. Of all the resources units in a node, 60% is labeled with the field in which the node is considered an “expert”, further 20% is labeled with another field that is closely related in the taxonomical tree to the expert field, and the last 20% is purely random, but with the restriction to be outside the expert field. This is said to resemble real–world distribution more, reflecting the fact that people have specific interests and hobbies [35].

4.3.2 Query and query resolution

Every query q will only carry one of the ACM classification leaf entities and it will be fully defined by it. In this case, however, $q_1(t_1) = q_2(t_2)$ iff $t_1 = t_2$; the benefit of such an approach is to be able to compare results of two queries released at different time points in the testing process and to show relative improvement between them. Routing concept overlay (explained in section 2.6.1) for a query $q_1(t_1)$ will be t_1 and will remain so during the entire querying process.

The resolution of a query $q_r(t_r)$ in a node n_i consists of finding all the resources that have been labeled with t_r, that is, all the resources $r_r \in \{ r \mid \exists r(n_i, t_r)\}$.

4.3.3 Query distribution

During the evaluation process every node of the network N of size n has a probability of being chosen to generate a query q with the probability of 1/N. In the Semant [17] evaluation setup every node has a probability of 0.1 to generate a query at every time unit. This leads to the conclusion that in order to have an equivalent test to the Semant test of T time units one must execute $n \times 0.1 \times$ T sequential iterations. A scale factor of x10 will be used in order to convert between time units of Semant and iterations used in this work. Every query q will carry a randomly chosen taxonomy entity t with the guarantee that there exists a resource within the network N that is described by t.

4.3.4 Execution Parameters

In Table 1 we summarize the recommended parameters for ACS algorithm.

Parameter	Interpretation	Value
TTL	Time to Live	25
q_0	Weight of exploiting vs. exploring strategy	0.80
R_{max}	Maximum number of resources to fetch	10
R_{min}	Minimum number of resources to fetch	5
α	Weight of newly deposited pheromone	0.07
β	Weight of link costs	1
γ	Factor in pheromone evaporation	0.02
ρ	Weight of evaporation	0.10
ph_{min}	Minimum pheromone level	0.001
ph_{max}	Maximum pheromone level	1
ph_{init}	Initial pheromone level	0.009

Table 3 ACS recommended parameters

In Table 2 we summarize the recommended parameters for Semant algorithm.

Parameter	Interpretation	Value
TTL	Time to Live	25
q_0	Weight of exploiting vs. exploring strategy	0.85
R_{max}	Maximum number of resources to fetch	10
R_{min}	Minimum number of resources to fetch	5
w_d	Weight of resource	0.5

	quantity vs. link costs	
β	Weight of link costs	1
ρ	Evaporation factor	0.07
ph_{min}	Minimum pheromone level	0.001
ph_{max}	Maximum pheromone level	10000
ph_{init}	Initial pheromone level	0.009

Table 4 Semant recommended parameters

4.3.5 Experiment Evaluation

Within each experiment the obtained data will be processed and presented two-fold.

Firstly we will intend to simply plot the data points of all the three measures mentioned in the section 4.2. Due to the large amount of data and its high variability we chose to use simple rolling average of the size 64 as an impulse filter and a data-set compacting method. This will serve as a graphical confirmation of consistency between independent executions; as well as allowing us to formulate initial observations.

Secondly we choose to perform statistical analysis to back up the graphical observations. Here, again, we use rolling average in order to limit the amount of data involved in calculations. The process will be performed over all the independent executions within an experiment; having in mind that each configuration is executed three times. The statistical analysis will consist of stating the H_0 and H_1hypothesis as follows:

- H_0: There is no statistically relevant difference between the algorithms: ACS, SemAnt, RandomWalker k-2, in terms of Hop per Hit measure.
- H_1: There exists a statistically relevant difference between the algorithms: ACS, SemAnt, RandomWalker k-2 terms of Hop per Hit measure.

For the hypothesis' evaluation we will use the Friedman test, for the mutual comparison between the algorithms and the Wilcoxon Signed-Rank Test method with the Bonferroni correction applied. All the statistical tests will be performed at $\sigma < 0.05$. Evaluation techniques we apply have been proposed by Derrac et al. in [39] specifically for such cases of studies.The only exception to the above description is the Experiment 1, where we attempt to recreate the Semant's results in terms of Hop per Hit. There is only graphical data provided by the authors of Semant and therefore we must rely on graphical analysis solely.

We will use the T-Means test to express difference between any given pair of algorithms if necessary.

4.4 Performance in an unstructured environment

4.4.1 Experiment 1: sem–1024 world – recreating the results of Semant

In this section we will analyze the performance of the chosen algorithms in an unstructured world. Firstly we will show that we have managed to recreate the results of Semant using our testing platform and, then, we will extend the comparison. The topology used in the work [18] is always sem–1024 (see section 3.1.1 for more details) and all the execution details are described in Table 1 and Table 2.

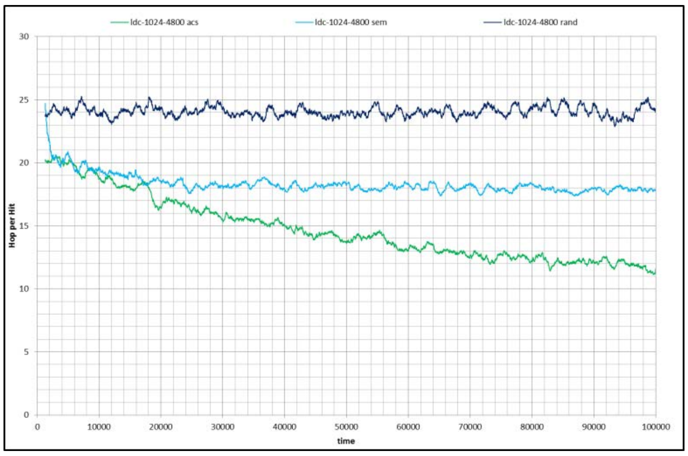
In order to make any subsequent results viable we must first demonstrate that the implementation of the environment of Semant is comparable to the results obtained in the original work. To achieve this we have developed a testing platform and applied the strictest details that are provided.

Being able to recreate results presented in [18] leads to the conclusion that the implementation we have created is a correct one albeit there is a slight and irrelevant difference in the results of random k–walker, most likely due to an implementation decisions taken by authors and not specified explicitly in their work. Additionally, in this experiment we have included a comparison between these two approaches and our extension of ACS taking advantage of the Routing Concept idea presented above.

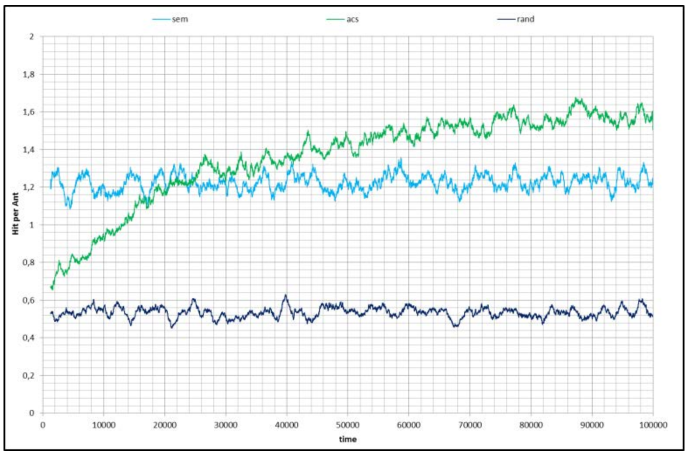
4.4.1.1 Results

The most notable difference between ACS and Semant is visible in the section of 0 – 10000 iterations; see: Figure 3–1 points a), b) and c). Semant, due to the fact of using many ants, seeds much more pheromone in a shorter period of time. It affects a greater part the system with singular iterations, especially in the very early phase, when the probability of generating multiple ants at each step is high; this fact will be referred to as the *iteration impact*. The upside to this is that paths appear quicker – the initial phase is much more intense. From the Figure 3–1 a) we can conclude that the relative improvement within the initial 10% of the iterations is large: in case of Semant the average number of Hop per Hit drops from 24 to 18, while in case of ACS only from 17 to 16 – nevertheless, ACS performs better still in absolute terms. The downside however is what happens after – the multi–ant approach continues to generate additional ants (albeit sporadically) and drags results into slightly worse values. And, not surprisingly, the overall convergence is better with the traditional ACS approach than the Semant one. Notice also the Figure 3–1 c) that reflects the efficiency of a single agent: clearly Semant makes no progress in this field, the improvement of Hop per Hit must stem out of shortening the paths rather than collecting more resources. In case of ACS we observe a continuous improvement throughout the entire test – asymptotically to the value of 1.7 Hit per Ant.

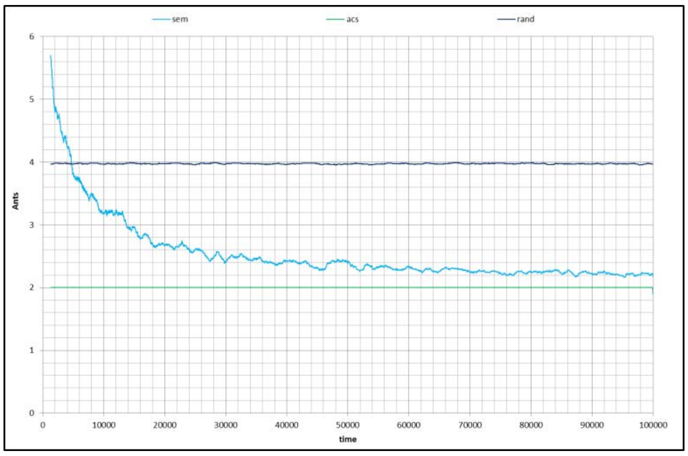
The *convergence* can be defined as the state of paths in the system after passing the horizon of iterations – a point after which there is no (or very little) improvement in measures. It can be observed that Semant's horizon is estimated at about 30000 iterations, while ACS' is at >100000 iterations; however, it is not as clear and one might argue a different point in time. This is due to the fact that the in *hc–d* tests (covered in section 3.5.1) we notice that the expected value for convergence seems to be 10 Hop per Hit, regardless of the setup. It has not been reached here due to the insufficient length of the test; nevertheless, not reaching the horizon is not as crucial as the mutual correlation between the algorithms in this setup.



a) Hop per Hit



c) Hit per Ant



b) Ant

Figure 3–1 Results in the world sem–1024 for a) Hop per Hit b) Ant c) Hit per Ant

4.4.2 Experiment 2: ldc–1024–m world

In this section we will analyze the impact of the number of randomly added long distance connections to the previous setup. The important question to be answered here is whether denser connections will provide benefits in terms of convergence. Although it has been suggested in [18] and [11] that the variation of long distance connections should not have any influence, there is no empirical evidence to support this claim. Consequently, we have considered a family of *ldc–n* topologies that will be subjected to the test; the chosen values are shown in the Table 4

World Name	Average Node Degree
ldc–1024–100	5.19
ldc–1024–300	5.59
ldc–1024–600	6.17
ldc–1024–1200	7.34
ldc–1024–2400	9.69
ldc–1024–4800	14.38

Table 5 Worlds chosen for ldc–1024–m tests

4.4.2.1 Results

The results shown in Figure 3–2 indicate that the influence of long distance connections is in almost all cases negligible. It is not as simple as stating that there is no influence; it is, however, is highly disproportional to the amount of links added. Theoretically, starting with the extreme cases, a fully connected graph and a graph organized into a ring, we must expect that in the first case all the possible queries will be answered in at most one step, while in the other they will be answered, on average, in $\frac{n}{4}$ steps, where n is the amount of nodes; which makes a considerable difference. Thus the statement that the appearance of new links has no influence has been proven false in the extreme cases.

Extreme cases aside, we see from figures Figure 3–2**,** Figure 3–3 and Figure 3–4, that effectively tripling the average degree of the node (Table 4) will not result in equivalent improvement in measurements. Still, if we consider the Hop per Hit measure (Figure 3–2) we can see that, even though Semant and Random Walks have hardly reacted to the amount of links, the densest network always results in the best convergence. ACS makes much more of the network size. This can be appreciated especially in the measure of Hit per Ant (Figure 3–3), where the difference reaches 28%, bearing in mind that the number of links is more than 5 times the original.

The Friedman Test results are presented in the Table 6 and Table 7. We can report that statistically significant difference was observed, $\chi^2(17) = 26865{,}258$, $\sigma = 0.00$. Further analysis with Wilcoson Signed Ranks Test of the values, presented in Table 8 proves that increasing the random links parameter improves the measured parameter significantly (with minor expections in Random Walker k-2), with the significance level at $\sigma < 0.05$, and after the Bonferroni correction, $\sigma < 0.0072$. Finally, in Table 9, we compare corresponding ACS and Semant results concluding that in all cases ACS is significantly better than Semant with $\sigma < 0.05$, and after the Bonferroni correction, $\sigma < 0.0035$.

It must be stated however that, in absolute values, the improvements are not very satisfying, so unless the cost of adding and maintaining an internode link is exceptionally low, it makes very little sense to blindly densify the unstructured network in hope of better convergence. Note that Semant is still unable to use just two ants (Figure 3–4) and only approaches this value asymptotically.

Test	acs-ldc-100	acs-ldc-300	acs-ldc-600	acs-ldc-1200	acs-ldc-2400	acs-ldc-4800
Mean Rank	8,25	6,19	5,78	3,65	2,63	2,03
Test	sem-ldc-100	sem-ldc-300	sem-ldc-600	sem-ldc-1200	sem-ldc-2400	sem-ldc-4800
Mean Rank	9,43	9,25	8,3	8,61	7,56	7,16
Test	ran-ldc-100	ran-ldc-300	ran-ldc-600	ran-ldc-1200	ran-ldc-2400	ran-ldc-4800
Mean Rank	15,98	15,62	15,59	15,41	15,32	14,23

Table 6 Experiment Ranks, Friedman Test for ldc-1024-m world

N	2000
Chi-square	26865,258
df	17
Asymp. Sig.	,000

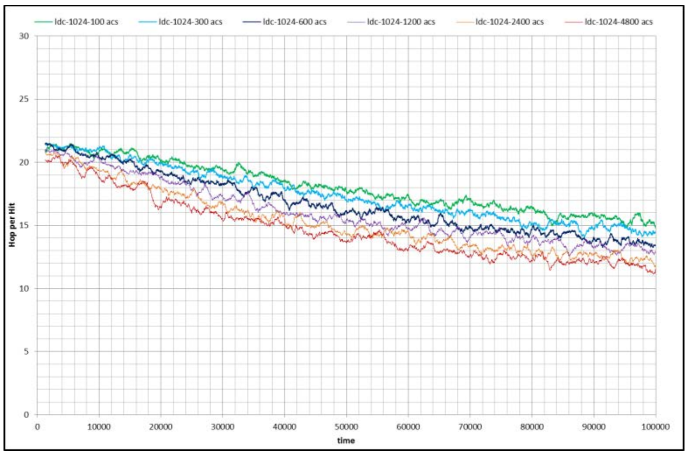
Table 7 Friedman Test Statistics for ldc-1024-m world

ACS	acs-ldc-300 - acs-ldc-100	acs-ldc-600 - acs-ldc-300	acs-ldc-1200 - acs-ldc-600	acs-ldc-2400 - acs-ldc-1200	acs-ldc-4800 - acs-ldc-2400
Z	-26,921	-6,348	-29,688	-19,039	-13,145
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000
Semant	sem-ldc-300 - sem-ldc-100	sem-ldc-600 - sem-ldc-300	sem-ldc-1200 - sem-ldc-600	sem-ldc-2400 - sem-ldc-1200	sem-ldc-4800 - sem-ldc-2400
Z	-1,713	-12,361	-3,589	-12,178	-5,765
Asymp. Sig. (2-tailed)	,087	,000	,000	,000	,000
Random	ran-ldc-300 - ran-ldc-100	ran-ldc-600 - ran-ldc-300	ran-ldc-1200 - ran-ldc-600	ran-ldc-2400 - ran-ldc-1200	ran-ldc-4800 - ran-ldc-2400
Z	-6,532	-1,008	-2,490	-1,778	-17,634
Asymp. Sig. (2-tailed)	,000	,313	,013	,075	,000

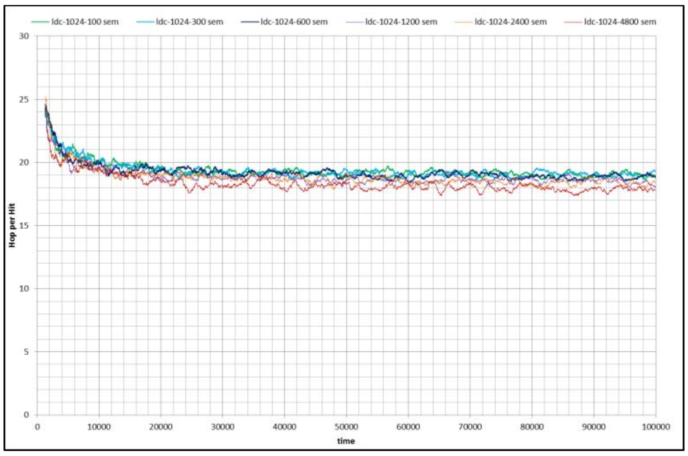
Table 8 Wilcoxon Signed Ranks Test statistics, based on positive ranks, for ACS, Semant and Random Walker k-2 in ldc-1024-m

	acs-ldc-100 - sem-ldc-100	acs-ldc-300 - em-ldc-300	acs-ldc-600 - sem-ldc-600	acs-ldc-1200 - sem-ldc-1200	acs-ldc-2400 - sem-ldc-2400	acs-ldc-4800 - sem-ldc-4800
Z	-15,424	-28,274	-26,339	-35,756	-36,203	-37,198
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000	,000

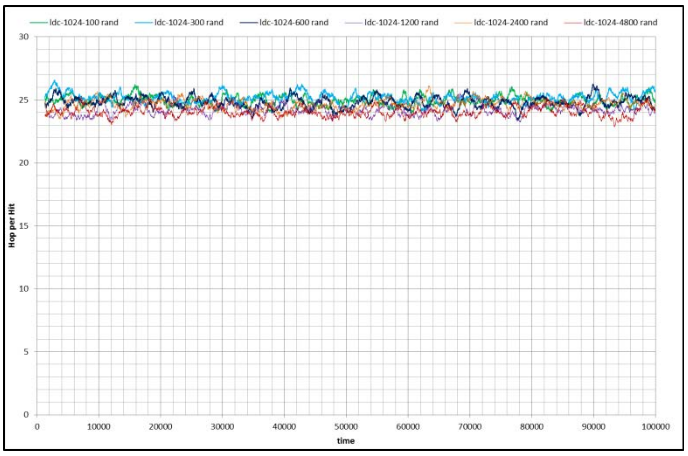
Table 9 Wilcoxon Signed Ranks Test statistics, based on positive ranks, ACS to Semant comparison in ldc-1024-m



a) ACS

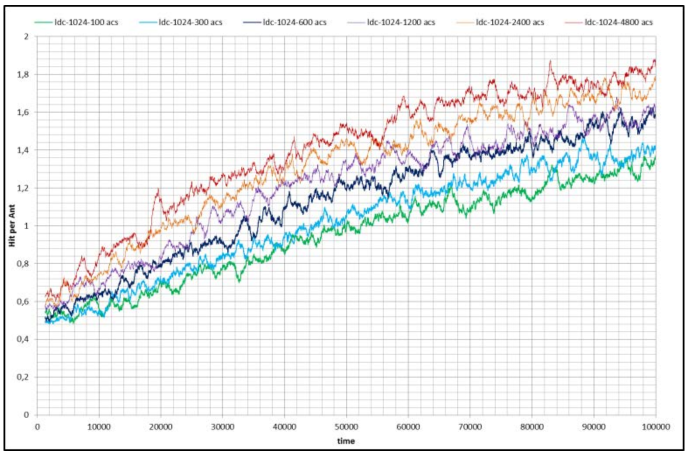


c) Semant

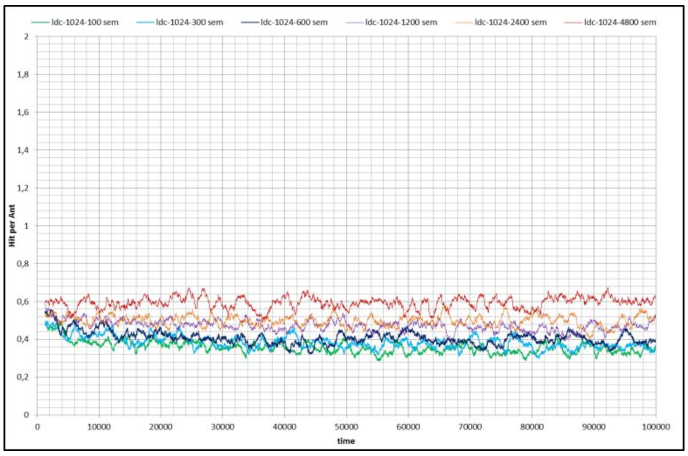


b) Random

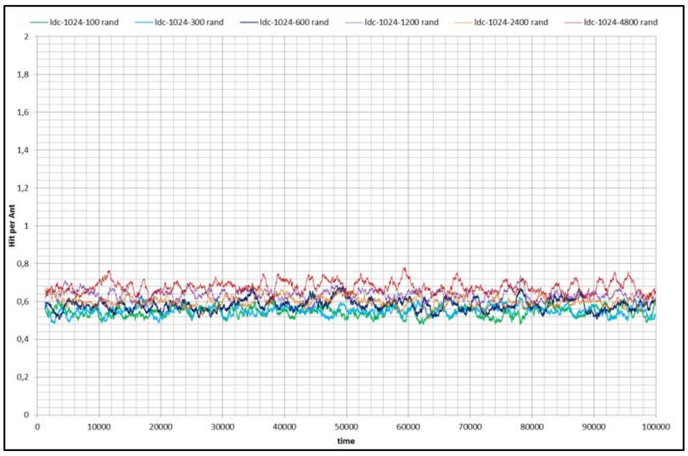
Figure 3–2 Hop per Hit comparison in ldc–1024–n for a) ACS b) Random c) Semant



a) ACS

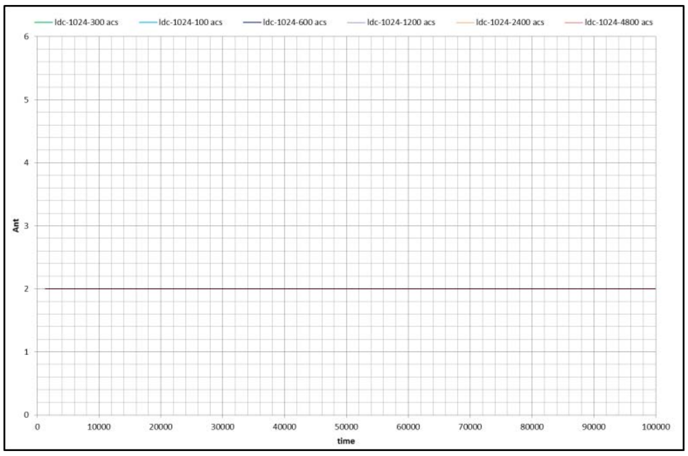


c) Semant

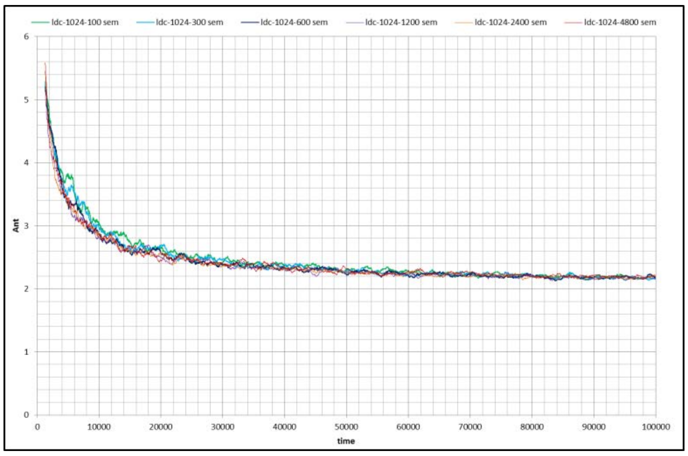


b) Random

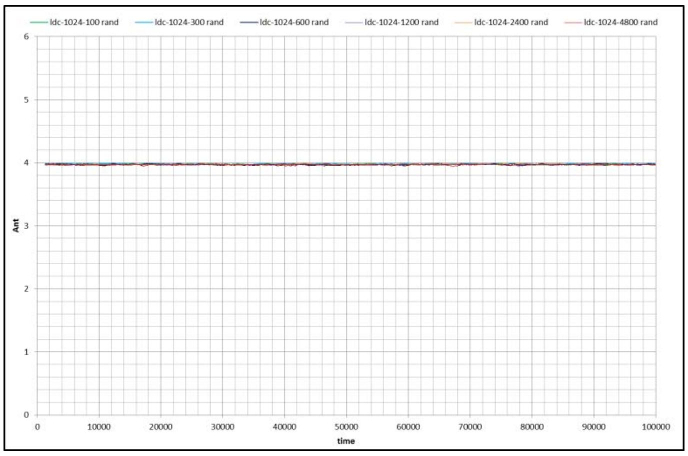
Figure 3–3 Hit per Ant comparison in ldc–1024–n of a) ACS b) Random c) Semant



a) ACS



c) Semant



b) Random

Figure 3–4 Ant comparison in ldc–1024–n of a) ACS b) Random c) Semant d) Best case comparison

4.5 Performance in an structured (hypercube–structured) environment

In this section we will show that the simple act of choosing a topology and, by consequence, structuring the world is not sufficient in order to achieve better results. Experiment 3 will display performance in *hc–d* worlds without any hybrid approach; that is: without taking any benefit of the underlying structure and compare them to the *ldc–n–m* worlds. In experiment 4 we will propose a way to exploit the hypercube topology and present adequate improvements. Experiment 5 further exploits the ideas of the experiment 4 and highlights some details of *hc–d* worlds that the previous failed to show.

4.5.1 Experiment 3: Hybridless approach

As a way to support the idea of the average node degree as the key component in the convergence quality we have decided to compare two topologically different manifolds, namely: a toroid (torus manifold) and a hypercube manifold. From Table 4 we can see that a ldc–1024–2400 world has approximately the same degree as a hc–10 world, as well as exactly the same world size of 1024 nodes; see equation (3.6). We have run an identical test over the above stated world and compiled the results in the next section. Apart from the comparison between these two specific topologies we have decided to show the behavior in different hypercube worlds all shown in the Table 5, similarly to the experiment 2 approach. This will help us to understand the relative behavior of Semant vs. ACS. Note that all are hybridless – the underlying topology is not exploited in any way.

World Name	Average Node Degree
hc–7	7
hc–8	8
hc–9	9
hc–10	10
hc–11	11
hc–12	12

Table 10 Worlds chosen for hc–d tests

4.5.1.1 Results / Comparison to ldc–1024–2400

It is interesting to see that a completely different organization of nodes (hc–10 and ldc–1024–2400) of similar node degree results in comparable outcomes. This only further confirms what was stated in 4.4.2.1 – the node degree has very little impact. In Figure 4–5 the graphs belonging to one world overlap nearly perfectly the graphs belonging to the other. Even though Friedman Test does detect a significant difference in terms of Hop per Hit, it is about 0.46 for ACS and 0.23 for Semant in absolute values, which is negligible, as shown in Table 13

Paired Differences	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference		df	Sig. (2-tailed)
				Lower	Upper		
acs-hc-10 – acs-ldc-2400	0,45926	1,40604	0,03144	0,3976	0,52092	1999	0,000
sem-hc-10 – sem-ldc-2400	0,23734	1,75627	0,03927	0,16032	0,31435	1999	0,000

Table 11 Comparison hc-10 with ldc-2400

Focusing on the results within the hc-d we can easily make a several very strong statements. It becomes very obvious that ACS outperforms Semant undisputedly in terms of convergence quality. This stands especially true considering that in every single figure (Figure 4–6, Figure 4–7, Figure 4–8) ACS comes out ahead. One thing needs to be pointed out: it is interesting to see how with the size of the world ACS struggles increasingly to converge – e.g. a small world such hc–7 is fully penetrated after less than 5000 iterations, whereas one of size hc–13 even after 100 000 iterations still has not reached its horizon. Semant suffers such a problem as well, however in slightly less impacting manner, in spite of the high iteration impact (as explained in 4.4.1.1), the hc–13 graph in Figure 4–8 b) does not reach the value of about 2, which is the indication of the horizon in Semant's case.

Last conclusion, which we arrive at, is that regardless of the size of the hc–d world the convergence is always within a range of similar values. Moreover they are not far off the results obtained in the Experiment 2 and it leads us to believe that a wise choice of topology could affect the convergence speed only; not improve the quality of the convergence obtained. See the section 4.4.1.1 for more details.

The statistical analysis resulted in similar conclusions to the ones obtained in Experiment 2. Table 14 and Table 15 present the confirmation of statistical differences, $\chi^2(20) = 34130{,}756$, $\sigma = 0.00$; in Table 14 we order all the tests in statistically significant ascending order along the hc-d values at $\sigma < 0.0035$, with minor exceptions for the Random Walker k-2 and Table 19 proves the superiority of Acs over Semant at $\sigma < 0.0017$. Bonferroni correction was applied in all the cases.

Test	acs-hc-7	acs-hc-8	acs-hc-9	acs-hc-10	acs-hc-11	acs-hc-12
Mean Rank	1,25	2,53	3,31	5,14	7,64	10,1
Test	ran-hc-7	ran-hc-8	ran-hc-9	ran-hc-10	ran-hc-11	ran-hc-12
Mean Rank	16,05	17,58	17,73	17,9	17,98	17,95
Test	sem-hc-7	sem-hc-8	sem-hc-9	sem-hc-10	sem-hc-11	sem-hc-12
Mean Rank	5,43	6,08	7,97	9,46	10,86	12,38

Table 12 Experiment Ranks, Friedman Test for hc-d world

N	2000
Chi-square	34130,756
df	20
Asymp. Sig.	,000

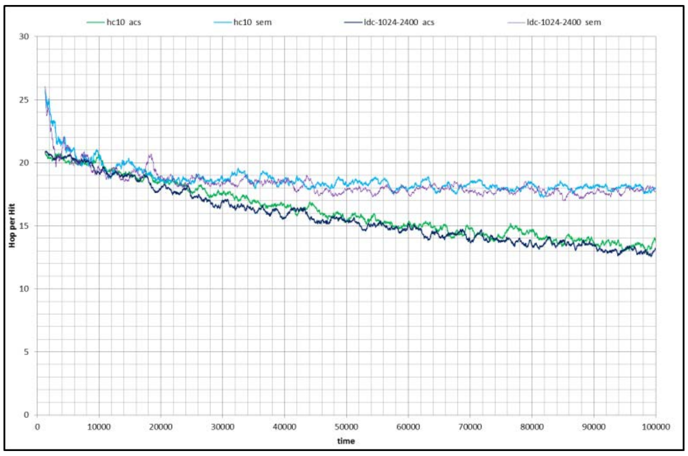
Table 13 Friedman Test Statistics for hc-d world

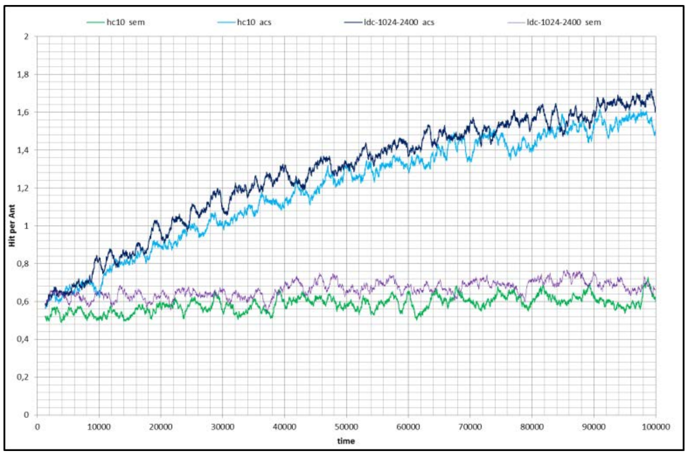
ACS	5 acs-hc-8 6 - acs-hc-7	7 acs-hc-9 8 - acs-hc-8	9 acs-hc-10 10 - acs-hc-9	11 acs-hc-11 12 - acs-hc-10	13 acs-hc-12 14 - acs-hc-11	15 acs-hc-13 16 - acs-hc-12
Z	-35,088	-18,649	-33,309	-34,594	-29,898	-23,934
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000	,000
Semant	sem-hc-8 - sem-hc-7	sem-hc-9 - sem-hc-8	sem-hc-10 - sem-hc-9	sem-hc-11 - sem-hc-10	sem-hc-12 - sem-hc-11	sem-hc-13 - sem-hc-12
Z	-10,956	-25,574	-18,510	-15,284	-16,240	-17,386
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000	,000
Random	ran-hc-8 - ran-hc-7	ran-hc-9 - ran-hc-8	ran-hc-10 - ran-hc-9	ran-hc-11 - ran-hc-10	ran-hc-12 - ran-hc-11	ran-hc-13 - ran-hc-12
Z	-20,402	-1,885	-2,888	-,191	-,282	-,829
Asymp. Sig. (2-tailed)	,000	,059	,004	,848	,778	,407

Table 14 Wilcoxon Signed Ranks Test statistics, based on positive ranks, for ACS, Semant and Random Walker k-2 in hc-d

	acs-hc-7 - sem-hc-7	acs-hc-8 - sem-hc-8	acs-hc-9 - sem-hc-9	acs-hc-10 - sem-hc-10	acs-hc-11 - sem-hc-11	acs-hc-12 - sem-hc-12	acs-hc-13 - sem-hc-13
Z	-38,570	-37,639	-37,307	-35,293	-29,590	-23,308	-21,946
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000	,000	,000

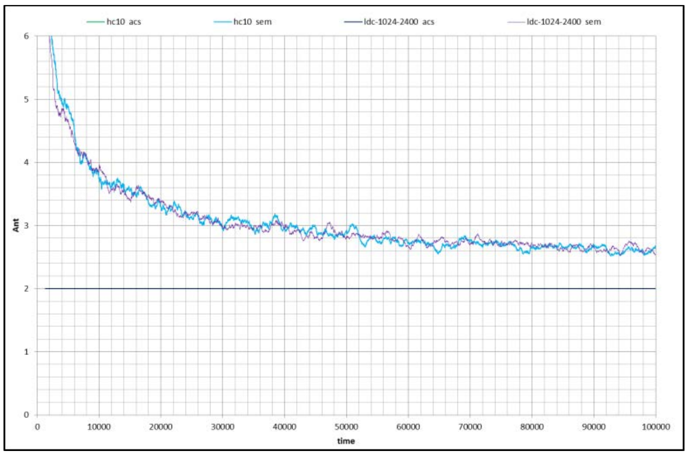
Table 15 Wilcoxon Signed Ranks Test statistics, based on positive ranks, ACS to Semant comparison in hc-d





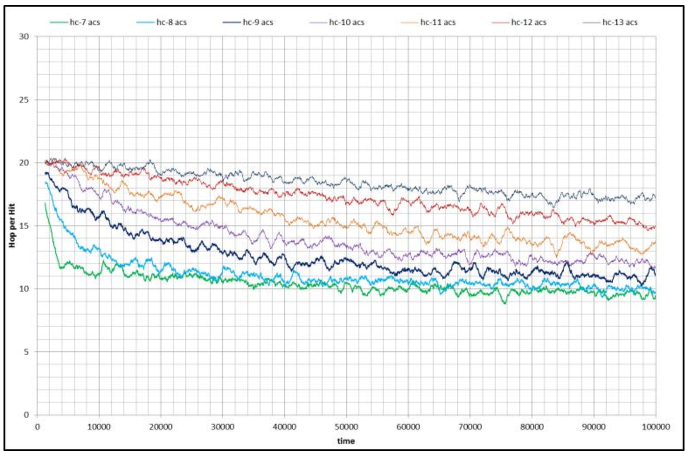
a) Hop per Hit

c) Hit per Ant

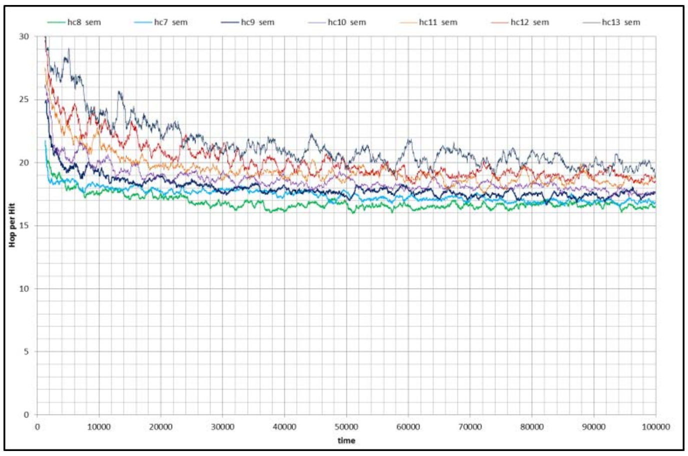


b) Ant

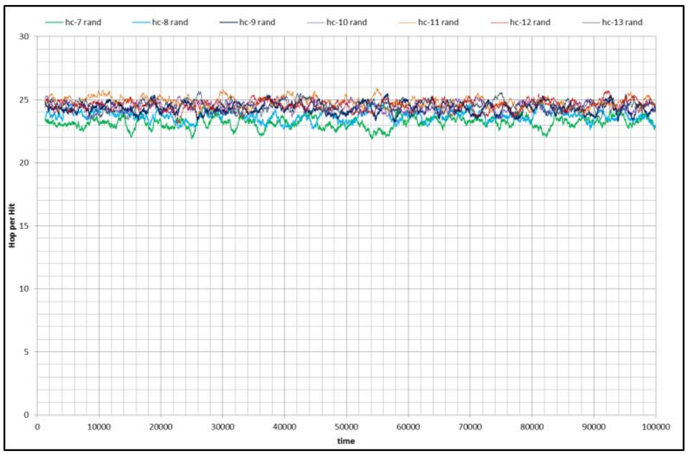
Figure 4–5 Intra world comparison: ldc–1024–2400 vs. hc–10 a) Hop per Hit b) Ant c) Hit per Ant



a) ACS

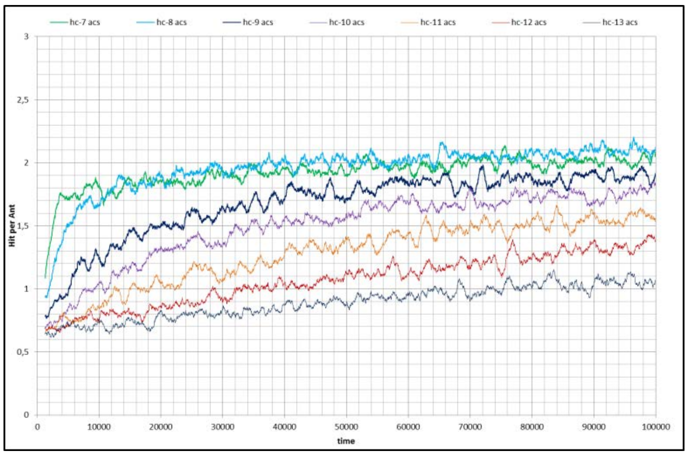


c) Semant

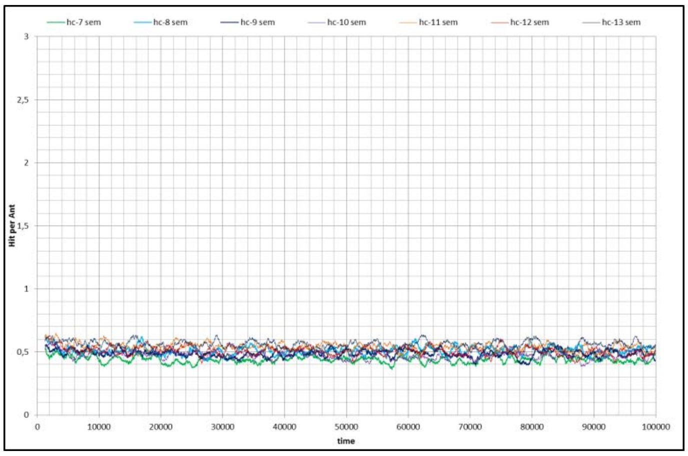


b) Random

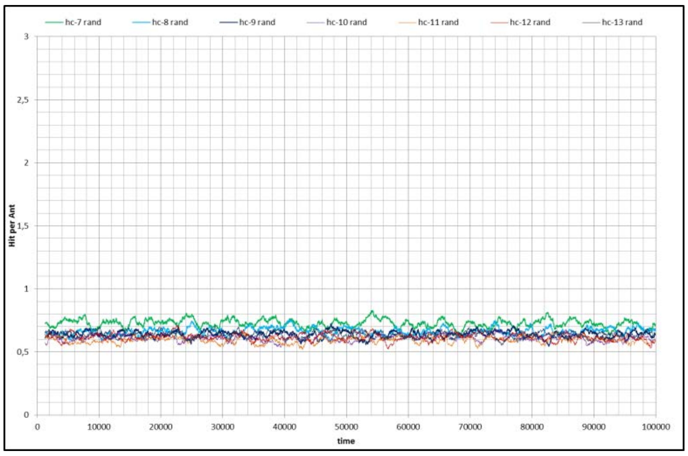
Figure 4–6 Hop per Hit comparison in hc–d non–hybrid approach for a) ACS b) Semant c) Random



a) ACS

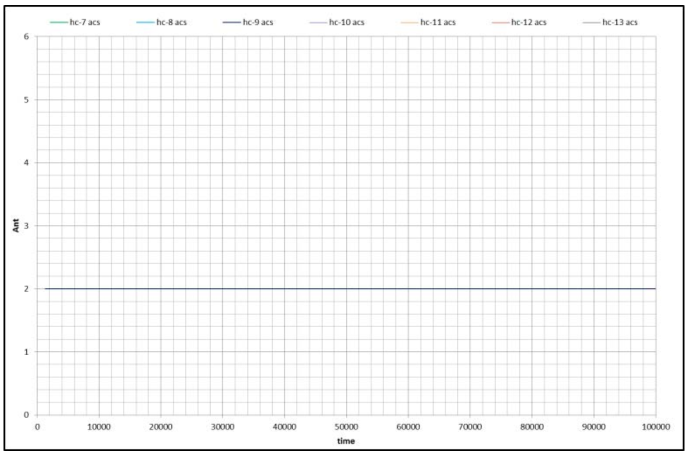


c) Semant

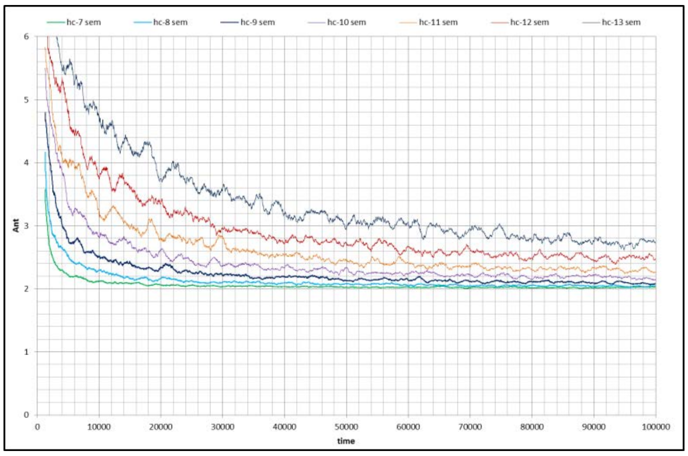


b) Random

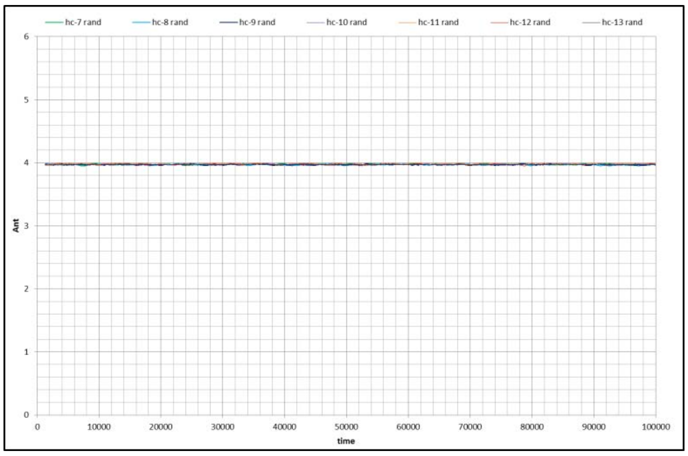
Figure 4–7 Hit per Ant comparison in hc–d non–hybrid approach for a) ACS b) Semant c) Random



a) ACS



c) Semant



b) Random

Figure 4–8 Ant comparison in hc–d non–hybrid approach for a) ACS b) Semant c) Random

16.1.1 Experiment 4: Hybrid approach

In this section we will show the impact of adding a hybrid path processing (explained in 2.6.2) to a known topology – in this case hc–d, as it was considered in the experiment 3. The worlds shown in Table 5 will be chosen. Apart from the usage of TRO there is no difference between this experiment and the experiment 3. We must stress that the hybrid approach is not highly general, as it requires the hypercube topology. It is based around exploiting the knowledge provided be this fact.

16.1.1.1 Results / Comparison to hc–10

There are several interesting results stemming out these tests. We need to emphasize that both Semant and ACS are inherently unstructured algorithms, that is, such ones that were designed to operate in unstructured worlds.

Firstly, in the comparison of hybrid hc–d against the non–hybrid we see that Semant has finally been able to make some progress in the Hit per Ant measurement, already reaching very considerable results (compare Figure 3–11 versus Figure 3–3 and Figure 3–7). Semant displayed another advantage as well – in the form of a dramatic reduction in the use of ants – as it can be concluded form the Figure 3–9 point b). It fell from about 6 in the initial phases to 3; still it does not solve the problem of asymptotically approaching the value 2. In Table 20 we can observe that the difference in Hop per Hit measure is about 8.44 for ACS and 10.64 for Semant, in absolute values. This is a very highly relevant difference and proves that it has been a quality leap from a hybridless hypercube to a hybrid hypercube.

Paired Differences	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference		df	Sig. (2-tailed)
				Lower	Upper		
acs-hc-10 – acs-hco-10	8,43763	2,74258	,06133	8,31736	8,55790	1999	,000
sem-hc-10 – sem-hco-10	10,63759	4,01519	,08978	10,46151	10,81366	1999	,000

Table 16 Comparison hc-10 with hco-10

The hybrid approach in itself is such a powerful tool that even the random algorithm has finally reported a slightly improved convergence values in Hop per Hit terms. Of course it is not quite suitable to name the state "convergence" as there is no pheromone, nor system evolution involved.

The convergence limits that have been mentioned in4.5.1.1 have changed from 10 Hop per Hit for ACS and 18 Hop per Hit for Semant to 2 Hop per Hit and 5 Hop per Hit respectively; see Figure 3–9 point a). That is a significant improvement of factor oscillating between 4x and 5x. Also a single ant is much more efficient now – being able to find the unprecedented 3 resources while it has never reached more than 2, see Figure 3–9 point c).

ACS, as earlier, outperforms Semant, but here the quick convergence of Semant plays a much more crucial role. In the large worlds the Hop per Hit measure seems to be better for Semant than ACS. It is not so: what actually occurs is that ACS is further from its corresponding horizon than Semant. Semant has always displayed such a quality but never has it ended in the state of Semant actually being higher evaluated than ACS (see hc–12, hc–13 and h–14 in graphs a) and c) of Figure 3–10). Consider, for instance, the point d) of Figure 3–2. There is a short section of 5000 – 7000 iterations where Semant actually performs better than ACS. Here, due to the size of the world, every convergence process is slowed down so much that the short section mentioned above is encompassed within our entire test of 100000 iterations. The horizon is never reached and the convergence does not occur (as discussed in 4.4.1.1). To further analyze this phenomenon we proposed another experiment (shown in the section 4.5.3).

Again the output of the Friedman Test confirms the graphical analysis, see Table 17 and Table 18. $\chi^2(20) = 33529{,}717$, $\sigma = 0.00$. In case of the Wilcoxon Sign Ranked comparison (Table 19, $\sigma < 0.0035$) we have noticed one deviation from the typical result, namely the acs-hco-9 does not appear to be better than acs-hco-8. It can be explained by a surprisingly high quality of acs-hco-8, which is always a possibility in a non-deterministic setup. . The most relevant results are presented in Table 20. We can clearly see that until the hc-10 world ACS is significantly better while hc-11 and above it is significantly worse. It is a statistical confirmation of the undergoing processes, explained in the previous paragraph.

The hybrid approach has allowed all the algorithms to reach convergence of values never achieved before. The costs of the hybrid approach are: slightly increased computation effort, further positioned convergence horizon and the requirement to establish a topology.

Test	acs-hco-7	acs-hco-8	acs-hco-9	acs-hco-10	acs-hco-11	acs-hco-12
Mean Rank	5,76	3,07	2,67	4,99	8,31	11,96
Test	ran-hco-7	ran-hco-8	ran-hco-9	ran-hco-10	ran-hco-11	ran-hco-12
Mean Rank	16,78	17,39	17,98	17,70	17,96	18,01
Test	sem-hco-7	sem-hco-8	sem-hco-9	sem-hco-10	sem-hco-11	sem-hco-12
Mean Rank	7,02	5,15	5,18	6,47	8,14	11,39

Table 17 Experiment Ranks, Friedman Test for hc-d world with hybrid path optimization

N	2000
Chi-square	33529,717
df	20
Asymp. Sig.	,000

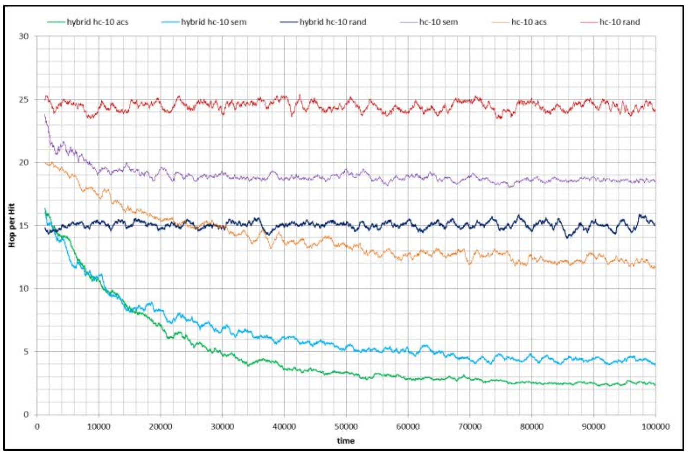
Table 18 Friedman Test Statistics for hc-d world with hybrid path optimization

ACS	acs-hco-8 - acs-hco-7	acs-hco-9 - acs-hco-8	acs-hco-10 - acs-hco-9	acs-hco-11 - acs-hco-10	acs-hco-12 - acs-hco-11	acs-hco-13 - acs-hco-12
Z	-24,130	-,698	-32,828	-37,134	-38,101	-33,411
Asymp. Sig. (2-tailed)	,000	,485	,000	,000	,000	,000
Semant	sem-hco-8 - sem-hco-7	sem-hco-9 - sem-hco-8	sem-hco-10 - sem-hco-9	sem-hco-11 - sem-hco-10	sem-hco-12 - sem-hco-11	sem-hco-13 - sem-hco-12
Z	-19,591	-5,183	-20,480	-24,662	-34,783	-34,470
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000	,000
Random	ran-hco-8 - ran-hco-7	ran-hco-9 - ran-hco-8	ran-hco-10 - ran-hco-9	ran-hco-11 - ran-hco-10	ran-hco-12 - ran-hco-11	ran-hco-13 - ran-hco-12
Z	-9,142	-8,679	-4,220	-4,185	-1,047	-1,671
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,295	,095

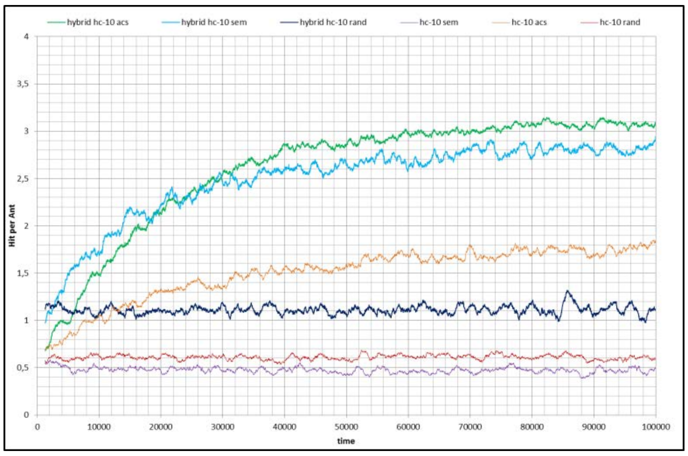
Table 19 Wilcoxon Signed Ranks Test statistics, based on positive ranks, for ACS, Semant and Random Walker k-2 in hc-d with hybrid path optimization

	acs-hco-7 - sem-hco-7	acs-hco-8 - sem-hco-8	acs-hco-9 - sem-hco-9	acs-hco-10 - sem-hco-10	acs-hco-11 - sem-hco-11	acs-hco-12 - sem-hco-12	acs-hco-13 - sem-hco-13
Z	-23,938	-28,946	-28,437	-13,053	12,639	21,502	10,599
Asymp. Sig. (2-tailed)	,000	,000	,000	,000	,000	,000	,000

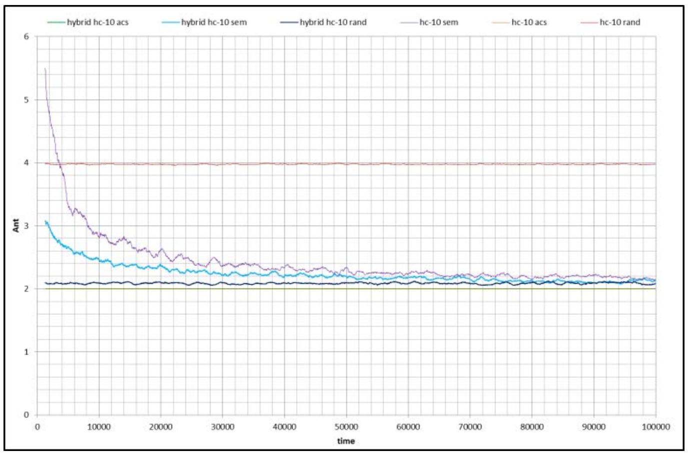
Table 20 Wilcoxon Signed Ranks Test statistics, based on positive ranks, ACS to Semant comparison in hc-d with hybrid path optimization



a) Hop per Hit

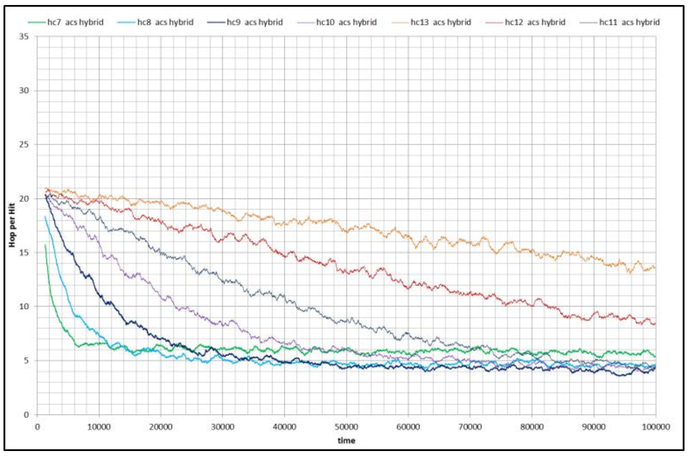


c) Hit per Ant

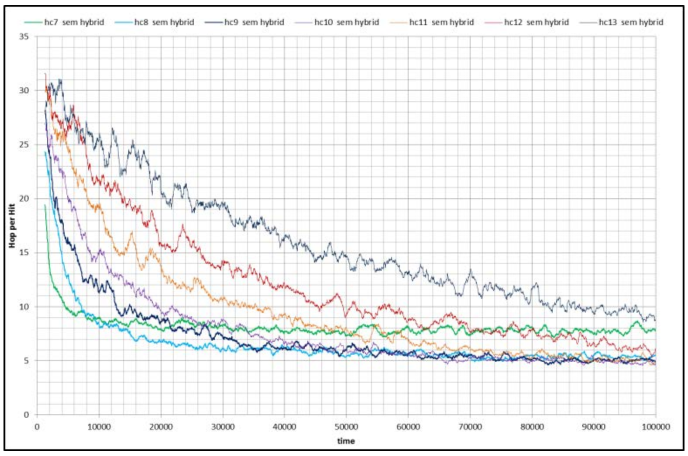


b) Ant

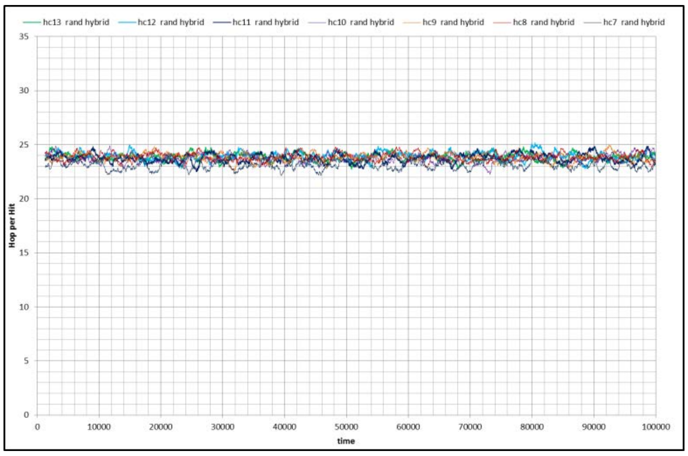
Figure 4–9 Intra world comparison: hc–10 – hybrid vs. hc–10 non–hybrid a) Hop per Hit b) Ant c) Hit per Ant



a) ACS

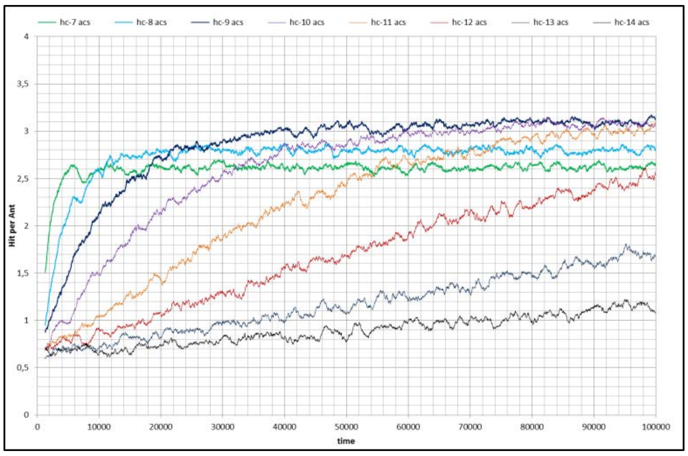


c) Semant

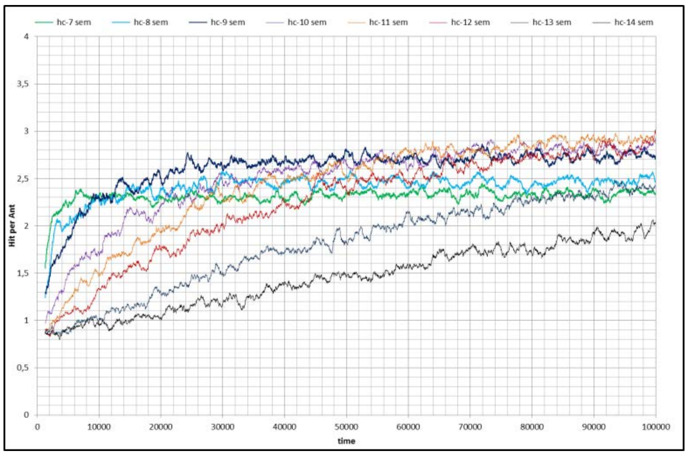


b) Random

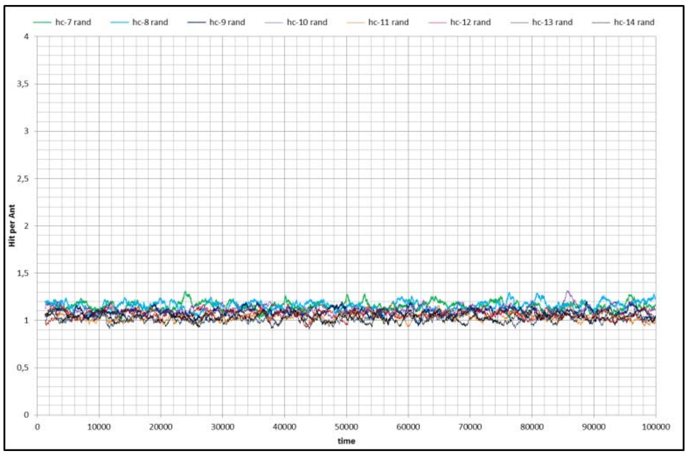
Figure 4–10 Hop per Hit comparison in hc–d with hybrid approach f a) ACS b) Random c) Semant



a) ACS

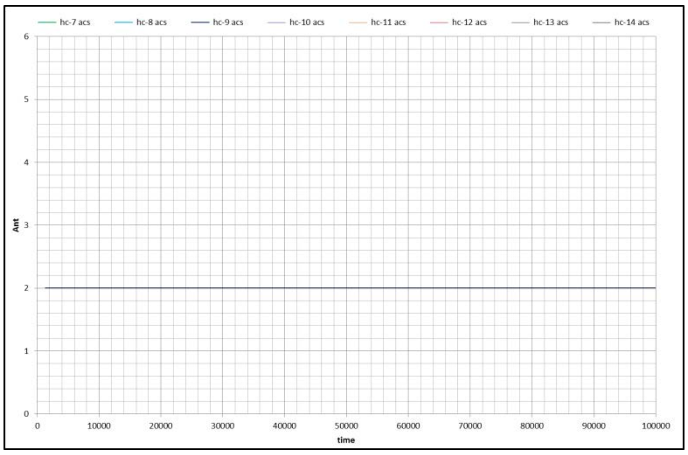


c) Semant

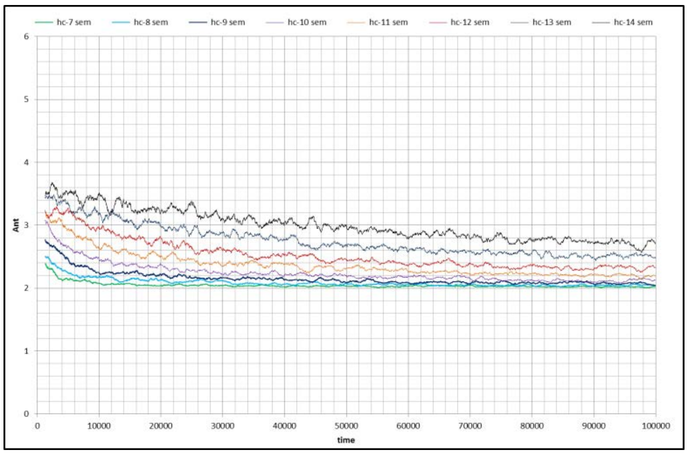


b) Random

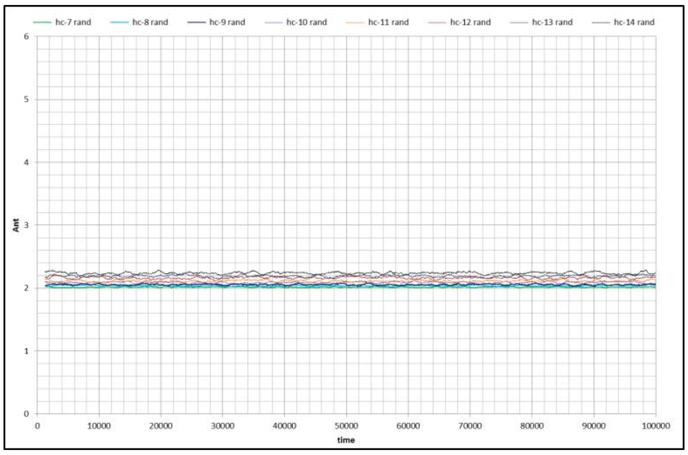
Figure 4–11 Hit per Ant comparison in hc–d with hybrid approach for a) ACS b) Random c) Semant



a) ACS



c) Semant



b) Random

Figure 4–12 Ant comparison in hc–d with hybrid approach for a) ACS b) Random c) Semant

16.1.2 Experiment 5: Hybrid approach 500k

The horizon problem (see section: 4.4.1.1) has caused results in higher dimension worlds of the previous test become unconverged due to the limitation on the number of iterations. Similarly, as Table 20 shows, it allowed Semant to outperform ACS. We decided to observe the behavior of the selected two algorithms in further stages, only basing ourselves on the Hop per Hit measure. In order to do so we increased the number of iterations from 100000 to 500000, which was deemed sufficient.

16.1.2.1 Results

The results are presented in Figure 3–13. It quickly became apparent that the limit of 500000 iterations is enough to observe the convergence horizon. As in smaller-degree optimized cases, both algorithms achieved comparable values. In this case however the window of Semant's superiority over ACS was so extended that both: the graphical and statistical reasoning prove that Semant is in fact better in this extreme case. As usual we applied the Freidman Test that concluded statistical differences (see Table 21 and Table 22) with values $\chi^2(3) = 16442{,}921$, $\sigma = 0.00$. Finally Table 23 confirms that Semant is better in this case, at $\sigma < 0.0125$.

Test	acs-hco-12 500k	acs-hco-13 500k	sem-hco-12 500k	sem-hco-13 500k
Mean Rank	2,14	3,75	1,48	2,64

Table 21 Experiment Ranks, Friedman Test for hc-d world with hybrid path optimization at 500k

N	10000
Chi-square	16442,921
df	3
Asymp. Sig.	,000

Table 22 Friedman Test Statistics for hc-d world with hybrid path optimization at 500k

	sem-hco-12 500k - acs-hco-12 500k	sem-hco-13 500k - acs-hco-13 500k
Z	-52,790	-75,607
Asymp. Sig. (2-tailed)	,000	,000

Table 23 Wilcoxon Signed Ranks Test statistics, based on positive ranks, ACS to Semant comparison in hc-d with hybrid path optimization at 500k

The comparison of the results in section 4.5.2.1 and the current ones raises the important issue of the iteration impact yet again. The more effect a single query has on the system (iteration impact – expressed as a ratio of nodes affected to all the nodes), the quicker the convergence occurs; which might erroneously cause a conclusion of the superiority of Semant over ACS in some large worlds in early phases of the test. The problem is the quality rather than the speed – the mechanisms that cause the higher impacts in early stages also cause an overhead in the later stages. Semant serves as a perfect example: intensified search in recently initialized world causes a boost in Hop per Hit measure but the very same process penalizes the results just after that – additional ants that are being sent aimlessly with no, or little chance of discovering any new resources. Whether this can be of any benefit in a world that has a dynamic resource distribution or structure will be the objective of future work.

What needs to be noted is that the introduction of hybrid path post processing mechanism has had much higher influence on the convergence process than the introduction of a multi–ant mechanism (compare the results in 4.5.2 vs. the results in 4.5.1).

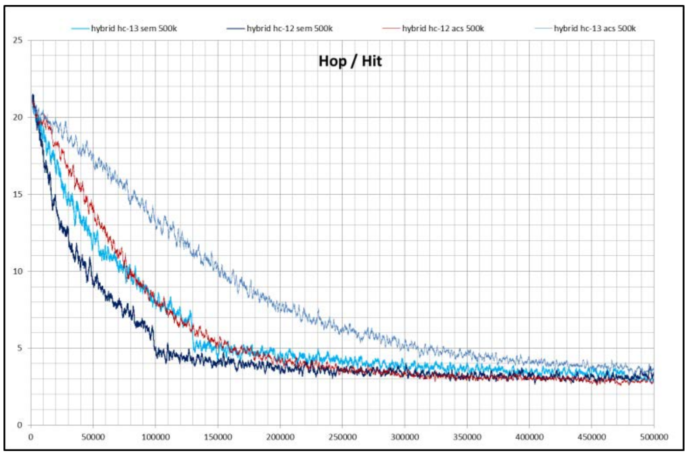


Figure 4–13 Hop per Hit in the world hc–12 and hc–13 under 500k conditions

17 Conclusions and Future Work

In this work we have proven several important facts about the use of ant–based methods in the p2p environments. We limited the scope of possibilities by choosing a set of reasonable prerequisites and picked two algorithms (in five variants) of eight taken in consideration. Random behavior was selected as the background and the base–line.

As underlying structures we have elected multidimensional hypercubes, toruses and toruses with additional links – in every case the only factor impacting the results was the average degree of the node, which translates directly into the link density. There was no perceptible difference between a hypercube of average node degree 10 and torus of average node degree ~10. This conclusion can be taken a step further. As the average node degree has a highly disproportional influence on the results, only very slightly demonstrating itself in extremes, we state that unexploited underlying topology is irrelevant to the results. On the other hand, the topology–aware hybrid route optimization makes a big difference – scoring results unobtainable in other approaches. Therefore, exploiting the underlying topology is highly relevant to the results and can have a very positive impact. Similar conclusions, with no empirical backup demonstrated, have been suggested in [19] and [12].

Another conclusion to notice is that the addition of the hybrid path optimization has by far more impact on the results than then original difference between ACS and Semant. One can understand this as a confirmation of the superiority of hybrid methods over slight tweaks in parameters of the classical algorithms. This is a practical implication to the question of ant–based p2p search and must be always taken in consideration when constructing a p2p search mechanism.

We have shown that the classical approach of ACS, extended with the Routing Concept notion scores better than the elaborate construction of Semant. Under all circumstances it has achieved superior convergence and lower use of system resources. The confrontation of a single ant (ACS) versus multi–ant (Semant) algorithms reveals a profound difference. Multi–ant algorithms, represented by Semant, have the tendency to quickly penetrate the world and seed pheromone values more rapidly. However, as was stated earlier, this process penalizes the results in the long run because the multi–ant mechanism continues to emit additional agents even when there is no need. These unnecessary agents return to their corresponding initial nodes and mostly find no new resources – therefore putting an additional strain on the system for no benefit.

It needs to be pointed out that the notion of quick convergence, as opposed to the quality convergence, might prove to be more useful in dynamic systems. The reasoning is that, even though one algorithm might theoretically reach a better state of convergence, it would never do so due to the environment changing constantly and spoiling what was established; whereas the quick one – although not as good – would keep the average convergence in a better state. This will form the bulk of our future work: examining the behavior of the mentioned algorithms under the strain of variability. It will include resources disappearing and reappearing, nodes reattaching themselves to other points in the system, nodes disconnecting from the system completely, etc.

18 Acknowledgements

This work was funded by the Spanish Ministry of Education and Science and Innovation under the National Strategic Program of Scientific Research, Development and Technological Innovation (I+D+i) and project TIN2010–20488. Kamil Krynicki is supported by the FPI fellowship from Universidad Politécnica de Valencia.

References

[1] A. S. Tanenbaum and M. van Steen, Distributed systems : principles and paradigms. Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 2002.

[2] Seti@Home. University of California. (1999-2011) http://setiathome.berkeley.edu/

[3] Collatz Conjecture. (2008) http://boinc.thesonntags.com/collatz/

[4] Top500. (2012, November) http://www.top500.org/list/2012/11/

[5] Gnutella 0.6. T. Klingberg. (2002, June) http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[6] L. Wang, SoFA: An expert-driven, self-organization peer-to-peer semantic communities for network resource management, *Expert Systems with Applications*, Vol. 38, No. 1, pp. 94–105, January 2011.

[7] G. Beydounb, G. Lowa, N. Trana, and P. Bogga, Development of a peer-to-peer information sharing system using ontologies, *Expert Systems with Applications*, Vol. 38, No. 8, pp. 9352–9364, August 2011.

[8] Voice over Internet Protocol (VoIP) Definition and Overview - Teliqo. Teliqo. (2012) http://www.teliqo.com/voip/

[9] M. Dorigo and G. Di Caro, The Ant Colony Optimization Meta-Heuristic, in *New Ideas in Optimization*.: McGraw-Hill, 1999, pp. 11-32.

[10] M. Dorigo and L. M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary*, No. 1, pp. 53-66, 1997.

[11] T. Stutzle and H. Hoos, Improvements on the Ant-System: Introducing the MAX-MIN Ant System, in *International Conference of Artificial Neural Networks and Genetic Algorithms*, Portoroz, Slovenia, 1997, pp. 245–249.

[12] C. Gómez Santillán, L. Cruz Reyes, E. Meza Conde, E. Schaeffer, and G. Castilla Valdez, A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks, *Computación y Sistemas*, Vol. 13, No. 4, pp. 433 - 448, 2010.

[13] E. Michlmayr, A. Pany, and G. Kappel, Using taxonomies for content-based routing with ants, *Computer Networks*, Vol. 51, No. 16, pp. 4514-4528, November 2007.

[14] A. Colorn, M. Dorigo, and V. Maniezzo, Distributed Optimization by Ant Colonies, in *Première Conférence Européenne Sur la Vie Artificielle*, Paris, France, 1991, pp. 134-142.

[15] S. Goss, S. Aron, JL. Deneubourg, and JM. Pasteels, Self-organized shortcuts in the Argentine ant, *Naturwissenschaften*, Vol. 76, pp. 579-581, 1989.

[16] J. Jaén, J. A. Mocholí, A. Catalá, and E. Navarro, Digital ants as the best cicerones for museum visitors, *Applied Soft Computing*, Vol. 11, pp. 111-119, 2011.

[17] J. A. Mocholí, V. Martinez, J. Jaen, and A. Catalá, A multicriteria ant colony algorithm for generating music playlists, *Expert Systems with Applications*, Vol. 39, pp. 2270-2278, February 2012.

[18] E. Michlmayr, Ant Algorithms for Search in Unstructured Peer-to-Peer Networks, in *22nd International Conference on Data Engineering Workshops*, Vienna, Austria, 2006, p. 142.

[19] E. Michlmayr, Ant Algorithms for Self-Organization in Social Networks. Vienna, Austria: Vienna University of Technology, 2006, PhD thesis.

[20] Q. Lv, P. Cao, E. Cohen, and S. Shenker, Search and Replication in Unstructured Peer-to-Peer Networks, in *Proceedings of the 16th ACM Conference on Supercomputing*, New York, USA, 2002, pp. 84-95.

[21] G. Di Caro and M. Dorigo, AntNet: Distributed Stigmergic Control for Communications, *Journal of Artificial Intelligence Research*, No. 9, pp. 317-365, July 1998.

[22] G. D. Caro, F. Ducatelle, and L. M. Gambardella, AntHocNet: An Ant-based Hybrid Routing Algorithm for Mobile and Ad Hoc Networks, in *Parallel Problem Solving from Nature (PPSN) VIII. Lecture Notes in Computer Science*, Birmingham, UK, 2004, pp. 461–470.

[23] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz, Antbased Load Balancing in Telecommunications Networks, *Journal on Adaptive Behavior*, Vol. 5, No. 2, pp. 169–207, 1996.

[24] An Efficient Scheme for Query Processing on Peer-to-Peer Networks. M. T. Prinkey. (2001) http://aeolusres.homestead.com/files/index.html

[25] L. Jun-qing, P. Quan-ke, and X. Sheng-xian, Research on Peer Selection in Peer-to-Peer Networks using Ant Colony Optimization, in *Fourth International Conference on Natural Computation*, Vol. 7, 2008, pp. 516-520.

[26] A. Crespo and H. Garcia-Molina, "Semantic Overlay Networks for P2P Systems",Computer Science Department, Stanford University, Stanford, Technical report 2002.

[27] EG. Talbi, A Taxonomy of Hybrid Metaheuristics, *Journal of Heuristics* , Vol. 8, No. 5, pp. 541-564, September 2002.

[28] G. Raidl, A Unified View on Hybrid Metaheuristics, in *3rd International Workshop on Hybrid Metaheuristics, HM 2006*, Gran Canaria, 2006, pp. 1 - 12.

[29] Q. Duan, TW. Liao, and HZ. Yi, A comparative study of different local search application strategies in hybrid metaheuristics, *Applied Soft Computing*, 2012, In Press, 10.1016/j.asoc.2012.05.016.

[30] F. Glover, Tabu Search - Part 1, *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190–206, June 1989.

[31] J. M. Gordon and Q. F. Stout, Hypercube message routing in the presence of faults, in *Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues*, New York, NY, USA, 1988 , pp. 318 - 327.

[32] I. Stoica, R. Morris, D. Krager, M. Frans Kaashoek, and H. Balakrishnan, Chord: A Scalable Peertopeer Lookup Service for Internet Applications, in *SIGCOMM'01*, San Diego, USA, 2001.

[33] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schneker, A scalable content-addressable network, in *SIGCOMM '01*, New York, USA, 2001, pp. 161-172.

[34] M. Dorigo and C. Blum, The hypercube framework for ant colony optimization, *IEEE Transactions on Systems, Man and Cybernetics part B*, Vol. 34, No. 2, pp. 1161-1172, 2004.

[35] "Torus" MathWorld--A Wolfram Web Resource. E. Weisstein. http://mathworld.wolfram.com/Torus.html

[36] "Hypercube" MathWorld--A Wolfram Web Resource. E. Weisstein. http://mathworld.wolfram.com/Hypercube.html

[37] Association for Computing Machinery. (1998) ACMComputing Classification System (ACM CSS).

[38] S. Zhao, D. Stutzbach, and R. Rejaie, Characterizing Files in the Modern Gnutella Network: A Measurement Study, in *In Proceedings of SPIE/ACM Multimedia Computing and Networking*, 2006.

[39] J. Derrac, S. García, D. Molina, and F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm and Evolutionary*

Computation, Vol. 1, No. 1, pp. 3-18, March 2011.

Vitae

Kamil Krynicki received the of Master of Science degree in Computer Science in the Wrocław University of Technology in 2010 for his thesis in the field of fuzzy computing and image processing. Awarded the first class honors for the best graduate of the year, the best student of the year and the winner of the IT–challenge in the category of non–doctoral thesis. Since September 2011 works as a researcher on the subject of ant colonies in the Universidad Politécnica de Valencia, after being granted the FPI scholarship

Javier Jaen is head of the FutureLab team, computer science professor at the Universitat Politècnica de València (Spain), Fulbright scholarship recipient and member of Upsilon Pi Epsilon: International Honor Society for the Computing and Information Disciplines. He has obtained a Master of Science degree from Virginia Tech (USA) and an Advanced Studies Degree from Institut National des Sciences Appliquées de Lyon (France). His research interests include grid computing, ant colony optimization, tangible interfaces and ubiquitous systems. He is the lead researcher of several research projects in collaboration with companies such as Microsoft Research, Telefónica R&D, Bancaja and America's Cup Management. He has been awarded the best PhD dissertation in computer science award at UPV, the award from the Society Commission at UPV 2007 and the eMobility 2003 award

Jose A. Mocholí received his Ph.D. in Computer Science from the Universidad Politécnica de Valencia in 2011. He is currently a researcher in the Department of Information Systems and Computation at the same university. His current research interests include affective computing, evolutionary computation, augmented reality, ubiquitous and distributed computing.

List of figures:

List of Tables: