UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Ant Colony Algorithms for the Resolution of Semantic Searches in P2P Networks

by

Kamil Krynicki

Supervised by

Dr. Javier Jaen

December 4, 2015

Ph.D Thesis

# Ant Colony Algorithms for the Resolution of Semantic Searches in P2P Networks

by

Kamil Krynicki

supervised by

Dr. Javier Jaen

*UNIVERSITAT POLITÈCNICA DE VALÈNCIA*

December 4, 2015

# Summary

The long-lasting trend in the field of computation of stress and resource distribution has found its way into computer networks via the concept of peer-to-peer (*P2P*) connectivity. P2P is a symmetrical model, where each network node is enabled a comparable range of capacities and resources. It stands in a stark contrast to the classical, strongly asymmetrical client-server approach. P2P, originally considered only a complimentary, server-side structure to the straightforward client-server model, has been shown to have the substantial potential on its own, with multiple, widely known benefits: good fault tolerance and recovery, satisfactory scalability and intrinsic load distribution. However, contrary to client-server, P2P networks require sophisticated solutions on all levels, ranging from network organization, to resource location and managing.

In this thesis we address one of the key issues of P2P networks: performing efficient resource searches of semantic nature under realistic, dynamic conditions. There have been numerous solutions to this matter, with evolutionary, stigmergy-based, and simple computational foci, but few attempt to resolve the full range of challenges this problem entails. To name a few: real-life P2P networks are rarely static, nodes disconnect, reconnect and change their content. In addition, a trivial incorporation of semantic searches into well-known algorithms causes significant decrease in search efficiency.

In our research we build a solution incrementally, starting with the classic Ant Colony System (*ACS*) within the Ant Colony Optimization metaheuristic (*ACO*). ACO is an algorithmic framework used for solving combinatorial optimization problems that fits contractually the problem very well, albeit not providing an immediate solution to any of the aforementioned problems.

First, we propose an efficient ACS variant in structured (hypercube structured) P2P networks, by enabling a path-post processing algorithm, which called Tabu Route Optimization (*TRO*). Next, we proceed to resolve the issue of network dynamism with an ACO-compatible information diffusion approach. Consequently, we attempt to incorporate the semantic component of the searches. This initial approximation to the problem was achieved by allowing ACS to differentiate between

search types with the pheromone-per-concept idea. We called the outcome of this merger Routing Concept ACS (*RC-ACS*). RC-ACS is a robust, static multipheromone implementation of ACS. However, we were able to conclude from it that the pheromone-per-concept approach offers only limited scalability and cannot be considered a global solution.

Thus, further progress was made in this respect when we introduced to RC-ACS our novel idea: dynamic pheromone creation, which replaces the static one-to-one assignment. We called the resulting algorithm Angry Ant Framework (*AAF*). In AAF new pheromone levels are created as needed and during the search, rather than prior to it. The final step was to enable AAF, not only to create pheromone levels, but to reassign them to optimize the pheromone usage. The resulting algorithm is called EntropicAAF and it has been evaluated as one of the top-performing algorithms for P2P semantic searches under all conditions.

# Resumen

La popular tendencia de distribución de carga y recursos en el ámbito de la computación se ha transmitido a las redes computacionales a través del concepto de la conectividad peer-to-peer (*P2P*). P2P es un modelo simétrico, en el cual a cada nodo de la red se le otorga un rango comparable de capacidades y recursos. Se trata de un fuerte contraste con el clásico y fuertemente asimétrico enfoque cliente-servidor. P2P, originalmente considerado solo como una estructura del lado del servidor complementaria al sencillo modelo cliente-servidor, ha demostrado tener un potencial considerable por sí mismo, con múltiples beneficios ampliamente conocidos: buena tolerancia a fallos y recuperación, escalabilidad satisfactoria y distribución de carga intrínseca. Sin embargo, al contrario que el modelo cliente-servidor, las redes P2P requieren de soluciones sofisticadas a todos los niveles, desde la organización de la red hasta la gestión y localización de recursos.

Esta tesis aborda uno de los problemas principales de las redes P2P: la búsqueda eficiente de recursos de naturaleza semántica bajo condiciones dinámicas y realistas. Ha habido numerosas soluciones a este problema basadas en enfoques evolucionarios, estigmérgicos y simples, pero pocas han tratado de resolver el abanico completo de desafíos. En primer lugar, las redes P2P reales son raramente estáticas: los nodos se desconectan, reconectan y cambian de contenido. Además, la incorporación trivial de búsquedas semánticas en algoritmos conocidos causa un decremento significativo de la eficiencia de la búsqueda.

En esta investigación se ha construido una solución de manera incremental, comenzando por el clásico Ant Colony System (*ACS*) basado en la metaheurística de Ant Colony Optimization (*ACO*). ACO es un framework algorítmico usado para búsquedas en grafos que encaja perfectamente con las condiciones del problema, aunque no provee una solución inmediata a las cuestiones mencionadas anteriormente.

En primer lugar, se propone una variante eficiente de ACS para redes P2P estructuradas (con estructura de hipercubo) permitiendo el postprocesamiento de las rutas, al que hemos denominado Tabu Route Optimization (*TRO*). A continuación,

se ha tratado de resolver el problema del dinamismo de la red mediante la difusión de la información a través de una estrategia compatible con ACO. En consecuencia, se ha tratado de incorporar el componente semántico de las búsquedas. Esta aproximación inicial al problema ha sido lograda permitiendo al ACS diferenciar entre tipos de búsquedas atravéś de la idea de pheromone-per-concept. El resultado de esta fusión se ha denominado Routing Concept ACS (*RC-ACS*). RC-ACS es una implementación multiferomona estática y robusta de ACS. Sin embargo, a partir de esta implementación se ha podido concluir que el enfoque pheromone-per-concept ofrece solo escalabilidad limitada y que no puede ser considerado una solución global.

Por lo tanto, para lograr una mejora a este respecto, se ha introducido al RC-ACS una novedosa idea: la creación dinámica de feromonas, que reemplaza la asignación estática uno a uno. En el algoritmo resultante, al que hemos denominado Angry Ant Framework (*AAF*), los nuevos niveles de feromona se crean conforme se necesitan y durante la búsqueda, en lugar de crearse antes de la misma. La mejora final se ha obtenido al permitir al AAF no solo crear niveles de feromona, sino también reasignarlos para optimizar el uso de la misma. El algoritmo resultante se denomina EntropicAAF y ha sido evaluado como uno de los algoritmos más exitosos para las búsquedas semánticas P2P bajo todas las condiciones.

# Resum

La popular tendència de distribuir càrrega i recursos en el camp de la computació s'ha estès cap a les xarxes d'ordinadors a través del concepte de connexions d'igual a igual (de l'anglès, peer to peer o *P2P*). P2P és un model simètric on cada node de la xarxa disposa del mateix nombre de capacitats i recursos. P2P, considerat originàriament només una estructura situada al servidor complementària al model client-servidor simple, ha provat tindre el suficient potencial per ella mateixa, amb múltiples beneficis ben coneguts: una bona tolerància a errades i recuperació, una satisfactòria escalabilitat i una intrínseca distribució de càrrega. No obstant, contràriament al client-servidor, les xarxes P2P requereixen solucions sofisticades a tots els nivells, que varien des de l'organització de la xarxa a la localització de recursos i la seua gestió.

En aquesta tesi s'adreça un dels problemes clau de les xarxes P2P: ser capaç de realitzar eficientment cerques de recursos de naturalesa semàntica sota condicions realistes i dinàmiques. Existeixen nombroses solucions a aquest tema basades en la computació simple, evolutiva i també basades en l'estimèrgia (de l'anglès, *stigmergy*), però pocs esforços s'han realitzat per intentar resoldre l'ampli conjunt de reptes existent. En primer lloc, les xarxes P2P reals són rarament estàtiques: els nodes es connecten, desconnecten i canvien els seus continguts. A més a més, la incorporació trivial de cerques semàntiques als algorismes existents causa una disminució significant de l'eficiència de la cerca.

En aquesta recerca s'ha construït una solució incremental, començant pel sistema clàssic de colònia de formigues (de l'anglés, Ant Colony System o *ACS*) dins de la metaheurística d'optimització de colònies de formigues (de l'anglès, Ant Colony Optimization o *ACO*). ACO és un entorn algorísmic utilitzat per cercar en grafs i que aborda el problema de forma satisfactòria, tot i que no proveeix d'una solució immediata a cap dels problemes anteriorment mencionats.

Primer, s'ha proposat una variant eficient d'ACS en xarxes P2P estructurades (en forma d'hipercub) a través d'un algorisme de processament post-camí el qual s'ha anomenat en anglès Tabu Route Optimization (*TRO*). A continuació, s'ha procedit a resoldre el problema del dinamisme de les xarxes amb un enfocament

de difusió d'informació compatible amb ACO. Com a conseqüència, s'ha intentat incorporar la component semàntica de les cerques. Aquest enfocament inicial al problema s'ha realitzat permetent a ACS diferenciar entre tipus de cerques amb la idea de "feromona per concepte", i s'ha anomenat a aquest producte Routing Concept ACS o *RC-ACS*. RC-ACS és una implementació multi-feromona robusta i estàtica d'ACS. No obstant, s'ha pogut concloure que l'enfocament de feromona per concepte ofereix només una escalabilitat limitada i no pot ser considerada una solució global.

En aquest respecte s'ha realitzat progrés posteriorment introduint una nova idea a RC-ACS: la creació dinàmica de feromones, la qual reemplaça a l'assignació un a un de les mateixes. A l'algorisme resultant se l'ha anomenat en anglès Angry Ant Framework (*AAF*). En AAF es creen nous nivells de feromones a mesura que es necessiten durant la cerca, i no abans d'aquesta. El progrés final s'ha aconseguit quan s'ha permès a AAF, no sols crear nivells de feromones, sinó reassignar-los per optimitzar la utilització de feromones. L'algorisme resultant s'ha anomenat *EntropicAAF* i ha sigut avaluat com un dels algorismes per a cerques semàntiques P2P amb millors prestacions.

# Acknowledgments

I would like to express my highest gratitude to my thesis director and mentor, Dr Javier Jaén, who made all of this possible, dedicated me an unmeasurable amount of effort and shared his wisdom and experience with me. I would like to offer my thanks to Dr Michael E. Houle, who devoted his time to my research, as well as guided me in art of science and scientific writing. I want to reserve my special thanks for two people who helped me out in what are the most difficult moments of any large endeavor, the beginning and the end: Jose Antonio Mocholí and Patricia Pons. I would like to thank my friends and colleagues, who showed interest in my work throughout these years: Javier Gonzalez Huerta, Alejandro Catalá, Priscilla Cedillo, Fernando Garcia, Vicente Nacher, Faizy Ahsan, Oussama Chelly and Jarek Kowalczyk. Most of all I want to thank my mother and my father, who helped out immensely, each in his own, unique way.

# Contents

# Chapter 1

# Introduction

In this initial section of the thesis we will present and motivate the main problem we tackle. We will discuss in general terms the elemental concepts, as well as the background of our research and the challenges put forward by the state of the knowledge. We will later continue to present a series of journal and conference articles that encompass the totality our findings.

## 1.1 Challenges

The rapid growth of Internet and data traffic has put forward a great challenge for engineers of diverse fields all over the globe. In order to match the demand for the data volume that is transmitted over the Internet every day, progress is needed in all aspects of technology: ranging from low-level hardware solutions, such as faster and more robust connectors and routers, all the way to logical and methodological changes, which revolve around the substitution of centralized, server-based models with distributed peer-to-peer ones. Some could argue that the conceptual changes are arguably more beneficial, especially short term, due to several reasons; the cost and time of implementation to name the leading ones. Good examples of this tendency are the implementations of tcp/ip v6 standard [1] and the very recent http/2 network protocol [2]. The diffusion of a new technology might take upwards of several years and new methodology, if exceptionally successful, propagates in a matter of months.

One of such models was the aforementioned peer-to-peer (P2P) architecture. A centralized client-server structure, in which one entity (server) possesses a disproportionally large amount of resources and provides them, upon request to other, underprivileged entities (clients), is without a doubt the most straightforward

network-related model. It is not fault-free however, as it presents significant drawbacks. A server is a single point of failure, it provides a hard-limit on the efficiency of the system and is vulnerable to intentional attacks and natural disasters. Naturally, there exists a number of sufficient, partial solutions that allow the client-server model to work under most conditions, but only the step towards a serverless P2P network guarantees a potentially unbound efficiency and robustness. In P2P networks all entities (called peers) are equal, they posses a portion of the resources of the system and the data traffic flows directly between the peers, which largely eliminates the need for centralized servers.

Unfortunately, P2P networks present a challenge from the extreme opposite end of the spectrum. With resources distributed among a large, often unknown, number of peers how can we guarantee efficient resource searches? A centralized system tracks precisely its resources and can answer instantly and correctly to a request from a client. A distributed system lacks this capacity, as any attempt to globally monitor peers and their content would forfeit the benefit of the decentralization. Therefore, if a peer of the network releases a request (a *query*) for a set of resources, it can only be resolved partially, against the neighbors of the emitter, or perhaps some well-known good resource providers. The problem becomes more difficult still, if we take in consideration the dynamic nature of P2P networks. Peers can leave and join the network at will, as well as expand, modify or eliminate their resource repositories.

The solution or potential solutions to this problem are multiple and vary greatly in applicability and cost [3]. Nevertheless, most of them, even the most successful, commercial implementations [4] are simple derivatives of the idea of an *information flood*. Flooding consists of propagating queries to most, or even all of the peers nearby to the emitter. These peers, in turn, can attempt to resolve the received request or continue to propagate it to their own neighbors. Such a cascading process is inherently difficult to control, to evaluate dynamically and to stop. See figure 1.1 for a visual example of an information flood. Note the explosion of traffic between the nodes, as well as the nodes called multiple times within a single search. In addition to this problem, in most existing solutions the peers tend to be organized in a predetermined structure, which reintroduces centralized concepts and global deamons into P2P networks. This is not, however, a necessity, as there are examples of search metaheuristics of purely distributed nature. A prime example of this approach is the Ant Colony Optimization ($ACO$) metaheuristic, which employs the concept of stigmergy - an indirect and completely decentralized communication technique.

ACO [5] is a metaheuristic used to solve optimization problems, often modeled as path and node searches in graphs, that mimics the behavior of ants in search for food. From a biologists' perspective ant behavior can be viewed as follows. Ants seek ways to benefit the colony. To achieve this they can set out on a random sweep of the nearby territory in search for goods or they can participate in

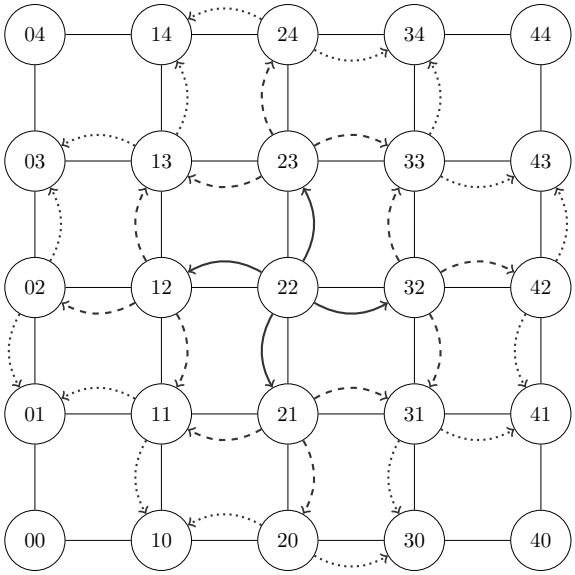



**Figure 1.1:** Information flood-like propagation in a small P2P network. Node 22 initiates the call. Continuous lines represent the first generation of calls, dashed lines represent the second generation and dotted lines represent the third generation.

the transportation of goods from the sources already located. These two broad types of behavior are known as exploration and exploitation respectively. An ant performing exploration might randomly stumble upon a valuable object. In such an event it carries out an evaluation and returns to the nest, secreting a trail of pheromone on its way back. The higher the estimated value, or *goodness*, of the newly found source, the stronger the pheromone trail will be. This, in turn, can entice more ants to choose to exploit that particular trail over performing exploration. A self-reinforcing process takes place, where more and more ants deposit pheromone, the trail becomes increasingly attractive and eventually results in a densely populated trail. On the other hand, the deposited pheromone undergoes constant evaporation, which is a natural way for ants to forget old trails that lead towards exhausted sources. However, the pheromone evaporation provides an additional benefit; namely, the shorter the trail, the less time it takes for an ant to walk the trail entirely. If two trails lead towards the same source the shorter will be less evaporated due to more frequent depositions of pheromone. The difference in the pheromone intensity causes a difference in ant traffic and, after a while, the longer route will be completely forgotten. This process was illustrated in the double bridge experiment [6] and later modeled mathematically by Goss et al [7].

ACO search agents, just like real-life ants, have no knowledge of the entire space they explore. Initially the amount of pheromone is low so agents survey the net-

work in a near-random manner. As the pheromone gets progressively deposited the agents' behavior develops into more deterministic and less stochastic. Two opposing processes take place: on one hand trails are being reinforced with new depositions, on the other the pheromone evaporation decreases the pheromone intensity. With time, some ways to construct the path are marked as progressively less and less attractive, until they eventually evaporate away completely. After a while all non-feasible or suboptimal trails get discarded and the solution path converges to its final shape; the overwhelming majority of agents travel exclusively on it and any deviation causes the goodness to decrease.

The use of ACO is a good first step towards efficient searches in P2P networks, as ACO appears to be compatible with the decentralized nature of P2P. P2P networks can be easily modeled as graphs, in its classical form ACO algorithms have no need for centralized entities, are not limited to any particular topologies and provide robust or even state-of-the-art solutions to many classical graph-related problems. See figure 1.2 for a visual representation of a sample ACO-guided search in a P2P network, compare with figure 1.1, most notably, the strain on the network and the network traffic. Nevertheless, the use of ACO in P2P networks is not a flawless idea. Due to the lengthy process of pheromone deposition and evaporation ACO algorithms cope only to an acceptable degree with dynamic graphs and dynamically distributed resources [8] [9] [10] [11]. In addition, and perhaps more importantly, they are not suitable for class-based or semantic searches [12] [13].

Semantic search [14] is not a simple search for any resources, but rather resources that fulfill a series of logical constraints. In its simplest, the constraint can be reduced to class adherence, which means that the search is performed and requesting resources of given classes only. In the context of P2P networks semantic searches are of high interest. First, the ability to recognize the request as belonging to a given class of requests is a way to increase the precision of the response, by improving the query routing decisions. Second, separating unrelated query traffic allows a specialized handling of each one. For instance, queries for scarce resources could be allocated higher time-to-live, and therefore, increasing the chance of yielding results.

As mentioned ACO search agents only work with a single pheromone value, that indicates if a given direction is of high or low quality. With a class-constrained search we are essentially unable to guarantee that the pheromone-reflected quality is informing the agent about the constrained objective of their search. Moreover, agents with different objectives might disagree about the quality of a certain path and change its evaluation, introducing chaos and limiting the usefulness of ACO substantially. The most trivial solution is to execute the simple ACO algorithm $C$ times ($C$ *layers*) [15], one for each one of the $C$ classes of the resources present in the network, and to use only the appropriate layer for the resource and query class in question. This solutions, unsurprisingly, is highly suboptimal and only suitable for very low $C$ [16].

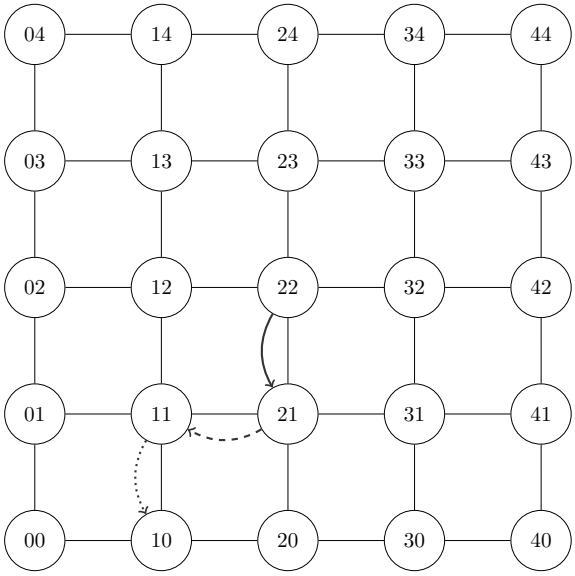



**Figure 1.2:** Information ACO-like propagation in a small P2P network. Node 22 initiates the call. Continuous lines represent the first generation of calls, dashed lines represent the second generation and dotted lines represent the third generation.

From a multitude of existing ACO algorithms we chose the, arguably, most classical and well known one: Ant Colony System (*ACS*, [17]). ACS was originally designed to solve the Traveling Salesman Problem (*TSP*) so it required some modifications, in order to work in a P2P environment. Above all, the authors of ACS, as they were tackling a centralized problem, forfeited the distributed nature of ACO to gain the efficiency of execution, by using a number of global operations and some centralized data. In order to revert this change we split the global pheromone matrix, which contained the totality of the pheromone state of the network, among the nodes, so that each node has only the view of the nodes directly connected to it. In addition, we defined the global pheromone evaporation as a local action, performed by returning agents, to completely eliminated the need for global clocks or messaging systems. Furthermore, we made some natural changes to the search end conditions. In TSP it is, naturally, visiting all the nodes and returning to the starting node. In our implementation it would not be sensible to maintain such a requirement, so it was replaced by a time-to-live mechanic (*TTL*) and minimum resources found threshold. Finally, ACS includes a mechanism of selective exploration. It consists of sending a number of agents, awaiting their return and selecting only the best one for pheromone deposition, discarding the remainder. This would be a very wasteful approach in a P2P network, where the main cost of executing a query is the inter-node agent transfer. In our approach even a very

inefficiently obtained resource should not be discarded, as the cost of obtaining it has already been incurred and is irrecoverable.

To summarize, the main objective of this thesis is to enable efficient Resolution of Semantic Searches in P2P networks using Ant Colony Algorithms. In order to complete it we must resolve the problem of the impact of dynamism of the network on ACO (challenge 1) and allow ACO to route agents based not only on the pheromone value, but also on the semantic content of the query, regardless of the value $C$ (challenge 2), while at the same time persevering all the desired properties of ACO (global constraint).

## 1.2 Contributions

To resolve the challenges presented in the previous section we decided to divide our efforts in six stages, each one concluding with a journal or a conference publication, validating our findings. First we focused on the problem of the dynamic graphs (contributions 1 - 3), which we concluded with a real-life implementation (contribution 4), and later we tackled the semantic issue (contributions 5 - 6), with a real-life solution still pending. Aside the 6 contributions we provide one additional of a technical nature, that summarizes our advances in ACO execution environments (contribution 7*).

1. In chapter 2 we present the paper *On the performance of ACO–based methods in P2P resource discovery* [12]. The main focus of this preliminary work was to formally define what prerequisites must an algorithm fulfill in order to be considered acceptable for P2P resources queries. We compare several techniques and propose a couple of partial solutions in the form of hybrid extensions of the classical approaches.

2. In chapter 3 we present the paper *Ant Colony Optimization for resource querying in dynamic peer-to-peer grids* [8]. Here we summarize the problems the dynamic graphs present for ACO algorithms. We examine 4 types of network dynamism that is expected to be experienced in P2P networks and demonstrate the effects they have on classical ACO algorithms.

3. Having formalized and demonstrated the problem we proceed to our first complete solution to the problem of dynamism. In chapter 4, in the paper *A Diffusion-Based ACO Resource Discovery Framework for Dynamic P2P Networks* [18], we show how to efficiently combat the network dynamism by a new mechanism called *information diffusion*, fully within the ACO metaheuristic.

4. Subsequently, we decided to apply our extended ACO algorithm to solve a real-life problem (see chapter 5). We opted to use it in the complex subject of

learning unit recommendation for relearning tasks for people with Acquired Brain Injuries (*ABI*). As of December 4, 2015 our journal paper *An ACO-based personalized learning technique in support of people with Acquired Brain Injury* [19] is undergoing its second reviews.

5. In chapter 6 we present a global solution to the semantic search in P2P networks with ACO. The paper, titled *A non–Hybrid Ant Colony Optimization Heuristic for Convergence Quality* [20] is the unveiling of our novel ACO algorithm called Angry Ant Framework (*AAF*). AAF was initially designed to simply enable semantic searches with ACO, but it has proven to be better in non-semantic context as well, which is demonstrated in this paper.

6. Finally, in chapter 7 we combine all the research to the date and study AAF in a fully semantic context. In addition, we present improvements over the original AAF design, with which we have achieved an unprecedented query efficiency in P2P networks, both with and without the semantic components. To the best of our knowledge, the algorithms presented in the paper *An Efficient ACO Strategy for the Resolution of Multi-Class Queries* [16] are the highest performing in the field of semantic ACO-based P2P searches and have the potential to be evaluated as the state-of-the-art. Our work is currently undergoing independent peer reviews.

7* We conclude our work with a publication of a technical nature. In order to put forward some of the most elaborate and extensive ACO experiments we were forced to write a highly configurable and expendable ACO middleware. In the end we decided to transfer our programing effort to the ACO community and we published our middleware at a workshop of a world-class conference. See chapter 8 for the paper *AntElements: An Extensible and Scalable Ant Colony Optimization Middleware* [21], which displays the highlights of our software.

In summary, throughout our research we have produced 7 scientific articles: 2 Q1 journal papers, 2 Q1 journal papers that are still under review, 2 CORE A conference papers and 1 CORE B conference paper.

# Chapter 2

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

Kamil Krynicki Javier Jaen Jose A. Mocholi

ISSI Research Group, Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

(2013), Applied Soft Computing Journal, 13(12), 4813–4831.
doi:10.1016/j.asoc.2013.07.022

**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.

## 2.1 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 [22] 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 [23] or Collatz Conjecture [24], 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 [25], which involve immense investments and oscillate at about ten–fifteen thousand teraflops. Other examples of P2P application are: sharing of storage and content distribution [4] [26] [27], where the desired content is treated as the resource, sharing of bandwidth, streaming or even anonymous communication solutions, both text and VoIP [28].

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 [29]). 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* [17], *Max–Min* [30], Neighboring–Ant Search, *NAS* [31], SemAnt [13] and various mixed solutions [32]), 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 2.2 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; in section 2.3 we present motivations for choosing specific algorithms for the comparative study. In section 2.4 we will propose the experimental dimensions to be analyzed, the designed experiments and present their results with comments. Finally, in section 2.5 we will formulate the final discussion.

## 2.2 Ant colony optimization for P2P searching

Ant Colony Optimization is a swarm intelligence approach to problem-solving introduced by Marco Dorigo in his work on distributed optimization in 1991 [33]. The core idea of ACO is twofold, firstly – as properly named – it uses a swarm of simple and stochastic automata 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 [7] or paths optimizing a given function [34] [35]. The automata, 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.

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:** Micro-scale 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 $\{r|r \in q(G)\}$, 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** resource $r$ is found **then**
4: $r$ is added to the ant's basket.
5: **end if**
6: **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, $A_1$ **then**
7: proceed to step 12.
8: **end if**
9: $A_1$ 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, $n(A_1)$.
10: $A_1$ performs Local Pheromone Update – metaphore of the natural process of pheromone evaporation.
11: Proceed to step 1.
12: $A_1$ converts from being a Forward Ant to being a Backward Ant.
13: (optional, indication of a hybrid ACO) $A_1$ performs optimization of the stack $n(A_1)$
14: $A_1$ performs an evaluation of the trail found based on a quality measure function $QM(A_1)$.
15: $A_1$ 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.
16: When $A_1$ returns to the emitting node, it deposits found resources from the basket and the algorithm concludes.

Several key factors that define ACO algorithms canbe deduced from the micro–scale algorithm:

1. Graph topology (or the lack thereof).
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.

One of the most popular implementations of the ACO metaheuristic is Ant Colony System [17]. It is an extension and improvement over the Ant System (AS) [36]. It has been chosen as the principle candidate for P2P search. In the particular case of ACS 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:

$$D(A) = \begin{cases} 1 & \text{if } ifr \in [R_{min}, R_{max}] \vee h \geq ttl_{max}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

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.

Consequently, the macro–scale algorithm (the system's large scale behavior) for ACO P2P search could be defined as follows:

**Algorithm 2:** Macro-scale algorithm

1: Query $q$ is requested upon the 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 $\delta t$ time units have passed, consider the $q$ resolution pending.
5: **if** Backward Ant $BA_q$ is received in the node $n$ after less than $\delta t$ time units have passed **then**
6: consider the basket of $BA_q$ the graph's response to $q$ and dispose of $BA_q$.
7: **else if** $BA_q$ is not received within $\delta t$ time units **then**
8: consider the $q$ resolved with no results.
9: **end if**

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.

The previous list of requirements 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 principle ACO and ACO-like algorithms, besides the well-known ACS strategy, and then formulate the subset of those best applicable for P2P and proceed to in–detail study.

### 2.2.1 Semant

The Semant algorithm [15] 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 2.3.1.

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

$$s = \text{argmax}_{u \in J_{k(r)}} \{\tau_\rho(cu) \times \eta_u^\beta\} \tag{2.2}$$

where:

- $\tau_\rho(cu) \in [0, 1]$ is the pheromone level from the current node to the $u$ node within the $c$ concept overlay
- $\eta_u$ is the cost of traveling to $u$
- $s$ 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 $p_{cr} \in [0, 1]$ 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:

$$p_{cr} = \frac{\tau_{cr} \times \eta_r^\beta}{\sum_{u \in J_{k(r)}} \tau_\rho(cu) \times \eta_u^\beta} \tag{2.3}$$

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 $FA^c i$, 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 2.4 and constrained by 2.5:

$$GO_j = \begin{cases} 1 & \text{if } p \leq p_{cr}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.4}$$

$$\sum_{j \in J_{k(r)}} p_j = 1 \tag{2.5}$$

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 $n$. Hence the maximum value of $\tau(r,u)$ is more of an issue to consider:

$$\tau_\rho(vu) \leftarrow \tau(r,u) + \delta\tau(r,u) \tag{2.6}$$

$$\tau_\rho(vu) \leftarrow \tau_\rho(vu) + w_d \times \frac{|S|}{S_{max}} + (1 - w_d) \times \frac{|L|}{2L_{max}} \tag{2.7}$$

where $w_d \in [0,1]$ is a parameter that expresses the balance between both components of the equation, $R$ 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:

$$\tau(r,u) \leftarrow (1 - \rho) \cdot \tau(r,u) \tag{2.8}$$

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

### 2.2.2 Neighboring–Ant Search

Another proposition of an extension of the basic AS was proposed by C. Gómez Santillán et al. [31]. It is based on exploiting the node distribution and several look–ahead heuristics. For in depth look consult the work [31]. Here we will focus on the pseudocode governing the behavior of Neighboring–Ant Search (NAS), provided by the authors of NAS:

**Algorithm 3:** NAS algorithm

1: **for all** query in $r_k$ create a search agent $k$ with $TTL_k = max_{TTL}$ and $Hit_{sk} = 0$ **do**
2:  **while** $Hit_{sk} < maxResultsandTTL_k > 0$ **do**
3:  **if** the unvisited $s_k \in \{r_k \cup \Gamma(r_k)\}$ has the searched resource **then**
4:  $r_k$ = append $s_k$ to $path_k$
5:  $Hit_{sk} = Hit_{sk} + 1$
6:  Local Pheromone Update
7:  Global Pheromone Update
8:  **else**
9:  **if** $r_k$ is a leaf node or does not have an unvisited neighbor **then**

10: remove the last node from $path_k$
11: **else**
12: $s_k$ = apply the transition rule with the DDC function
13: $r_k$ = append $s_k$ to $path_k$
14: Local Pheromone Update
15: **end if**
16: **end if**
17: **end while**
18: $TTL_k = TTL_k - 1$
19: **end for**
20: kill the search agent

At step 3 clearly the NAS algorithm takes advantage of basing its routing decisions on the content of the neighboring nodes. This, yet again, violates the third of the q–r principles. Furthermore at step 10 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 [15], where Michlmayr notices that the less resources a single agent carries (the $R_{min}$ variant of Semant), 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 2.4.2) 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 [31] 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.

### 2.2.3 Random k–walker

The most straightforward algorithm is the $k$–walker explored in detail in [37]. 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, 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 one of them makes a random decision on how to continue; on every other it simply takes a random decision 2.9. The probability of choosing to go from the node $r$ to the node s is $p_k(r, s)$ (2.10))

$$s = \begin{cases} \{S\} & \text{if } |h| > 0, \\ \{S_1, \ldots, S_k\} & \text{if } |h| = 0, \end{cases} \tag{2.9}$$

where $h$ is the number of steps taken.

$$p_k(r, s) = \begin{cases} \frac{1}{|J_k(r)|} & \text{if } s \in J_k(r), \\ 0 & \text{otherwise}, \end{cases} \tag{2.10}$$

The pheromone management is not relevant because $\tau(r, u)$ does not appear in 2.10. Therefore for evaporation, as well as deposition are:

$$\tau(r, u) \leftarrow \tau(r, u) \tag{2.11}$$

which indicates that there is no pheromone evolution.

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

### 2.2.4 Other algorithms

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

1. AntNet [38]
2. AntHocNet [39]
3. Ant Based Control [40]

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, in which the destination in not known but merely described by the combination of query, resources and the algorithm. The same problem occurs with Ant Based Control, which 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 [41]. It has been used for many years in the Gnutella protocol [4] 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. [42] they are vastly inefficient compared to ACO and will not be considered in this work.

### 2.2.5 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.

**Table 2.1:** Comparison of classical algorithms

| | Evolutive | Determ. | Semantic | Look Ahead | Hybrid | q–r compliant |
|---|---|---|---|---|---|---|
| ACS | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Semant | ✓ | ✗ | ✓(tax.) | ✗ | ✗ | ✓ |
| NAS | ✓ | ✗ | ✗ | ✓(var.) | ✗ | ✗, violates pt. 3 |
| k–Random Walks | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| AntNet | ✓ | ✗ | ✗ | ✗ | ✗ | ✗, violates pt. 6 |
| AntHocNet | ✓ | ✗ | ✗ | ✗ | ✗ | ✗, violates pt. 6 |
| Ant Based Control | ✓ | ✗ | ✗ | ✗ | ✗ | ✗, violates pt. 6 |
| k–flood | ✗ | ✓ | ✗ | ✗ | ✗ | ✗, violates pt. 7 |
| t–top flood | ✗ | ✓ | ✗ | ✗ | ✗ | ✗, violates pt. 7 |

From Table 2.1 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 2.3 we present the mentioned extensions in detail.

## 2.3 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.

Based on the algorithms selected in Section 2.2.5 and with the use of the extensions explained below we create four new algorithms, namely: Semantic ACS, Hybrid-Semantic ACS, Hybrid Semant and Hybrid k-Random Walks. All of them are compliant with our q–r principles and they are a combination of the corresponding classical algorithm and one or both extensions.

### 2.3.1 Semantic Extension: Routing Concept

The first extension is the notion of the Routing Concept, mentioned already while discussing Semant in section 2.2.1. Its theoretical base was presented in [43], where the idea of several Overlay Networks superposed over the Physical Node Network is introduced.

Every node $n$ keeps a 2–dimensional matrix $\Omega : N_n \times R_n$, 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 $\Omega$ corresponds to the pheromone value of the $n'$–th outgoing link for the $r$–th routing concept, which can be written as $\Omega_{n'r} = \tau$. There are two functions defined:

1. $ph_{Update}(n', r, \tau)$, that establishes as $\tau$ the value $\Omega_{n'r}$ for the $n' \in N_n$ and $r \in R_n$
2. $ph_{Retrieve}(n', r)$, 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 $\Omega^\dagger : N_n \times R_n^\dagger$, the space of routing concepts is redefined as $R_n^\dagger : R_n \cup r$, 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 [38] 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 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, introduced in [44]. 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.

### 2.3.2 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 a design decision, as there are several large classes of hybrid techniques. According to [45] 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 [46]. There the author establishes a slightly more in–depth, yet similar, taxonomy based around four key questions: the class of the 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: output function ($OF$), improvement method ($IM$), solution combination method ($SCM$) and input function ($IF$), 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. [47], 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 [48], 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.

**Algorithm 4:** TRO algorithm

1: Nodes within the path $p : [n_1, n_2, \ldots, n_N]$ that was covered by a forward ant FA, will be analyzed and all those 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 $p^t : [n_1^t, \ldots, n_{nr_i}, \ldots, n_{r_i}^t, \ldots, n_N^t]$, 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.

2: 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. $sp^t : [n_{sp_1}, n_{sp_2}, \ldots, n_{sp_M}]$

3: For every pair $n_{sp_i}, n_{sp_{i+1}}$ of nodes form the subpath $sp^t$, 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 [49]. 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 $p_i^{i+1}$.

4: Once the process is complete the newly created path will be named $p_o ptim$ and will replace the original path $p$, which was provided to the backward ant BA by the forward ant FA; thus BA will follow the path $p_o ptim$ on its way to the emitting node.

All the backward ant pheromone duties will be performed on the way as shown in the section 2.2.

## 2.4 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. 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 and the test setup that were used in the experimental study.

### 2.4.1 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 [29] 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 network topologies include ring [50], toroid [51], hypercube [52] 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 [15] [31] by taking into consideration the three topologies described next.

#### *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, where both dimensions $d_1$, $d_2$ of the creating rectangle are chosen to fulfill $|d_2 - d_1| \cong 0$ to minimize the average 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 [15]. This kind of world will be named *sem–n*, where $n$ stands for the number of nodes. In the [15] and [31] the sem–n world is approached as if unstructured. The average degree of a node is

$$av(sem - n) = 5 \tag{2.12}$$

***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 still 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:

$$av(ldc - n - m) = \frac{(5n + 2m)}{n} = 5 + 2\frac{m}{n} \tag{2.13}$$

Note that:

$$sem - n \equiv ldc - n - 0 \tag{2.14}$$

***Hypercube topology***

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

$$av(hc - d) = d \tag{2.15}$$

Additionally note that:

$$av(hc - 10) \cong av(ldc - 1024 - 2400) \tag{2.16}$$

### 2.4.2 Quality measures

We have decided to adopt a common efficiency quality measure which is widely used by several authors such as in [15] and [17], 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, a measure taken from the Semant study, 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 supplementary 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: 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 one 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.

### 2.4.3 Experiment Setup

Every test will consist of an amount $n$ of queries of random nature, with a given taxonomy (see 2.4.3), released from random nodes within the given world. The query will be propagated following the rules of the algorithm that is being tested. 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 section 2.4.2) and present the results as graphs. The amount of queries will be fixed at $n = 10^5$. 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 implementations 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 $10^6$ ants. A typical user would own not more than several nodes and process only singular ants at a time.

### *Taxonomy and resource distribution*

ACM Computing Classification System [53] will be the taxonomical vocabulary used. Every resource in the network $G'$ is described by one, and only one leaf taxonomical concept $t$ of the ACM classification. A resource has therefore only two properties: its owner vertex $v$ and a taxonomical label $t$. It is written as $r(v,t)$. Note that $v_1 = v_2 \wedge t_1 = t_2 \implies r_1(v_1,t_1) = r_2(v_2,t_2)$. Such an approach leads to valuing higher those nodes that provide many resources of the same $t$, which is the objective.

The distribution of resources within the network follows strictly the approach by the test setup [15]. 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 resource 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, reflecting the fact that people have specific interests and hobbies [54].

### *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.

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 $r_r \in \{r|\exists r(n_i,t_r)\}$.

**Table 2.2:** ACS and Semant recommended parameters

| **Parameter** | **Interpretation** | **Value (ACS)** | **Value (Semant)** |
|---|---|---|---|
| $TTL$ | Time to Live | 25 | 25 |
| $q_0$ | Weight of exploiting vs. exploring strategy | 0.80 | 0.85 |
| $R_{max}$ | Maximum number of resources to fetch | 10 | 10 |
| $R_{min}$ | Minimum number of resources to fetch | 5 | 5 |
| $\alpha$ | Weight of newly deposited pheromone | 0.07 | N/A |
| $w_d$ | Weight of resource quantity vs. link costs | N/A | 0.5 |
| $\beta$ | Weight of link costs | 1 | 1 |
| $\gamma$ | Factor in pheromone evaporation | 0.02 | N/A |
| $\rho$ | Weight of evaporation | 0.10 | 0.07 |
| $ph_{min}$ | Minimum pheromone level | 0.001 | 0.001 |
| $ph_{max}$ | Maximum pheromone level | 1 | 10000 |
| $ph_{init}$ | Initial pheromone level | 0.009 | 0.009 |

### *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 $\frac{1}{N}$. In the Semant's evaluation setup is time-based, 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 $\times 10$ 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 $t$.

### *Execution Parameters*

In Table 2.2 we summarize the recommended parameters for ACS and Semant algorithms.

***Experiment Evaluation***

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

First, we will intend to simply plot the data points of all the three measures mentioned in the section 2.4.2. 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.

Second, 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 $H_1$ hypothesis as follows:

1. $H_0$: There is no statistically relevant difference between the algorithms ACS, SemAnt and RandomWalker k-2, in terms of Hop per Hit measure.
2. $H_1$: There exists a statistically relevant difference between the algorithms ACS, SemAnt and RandomWalker k-2 in 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 [55] 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.

### 2.4.4 Performance in an unstructured environment

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

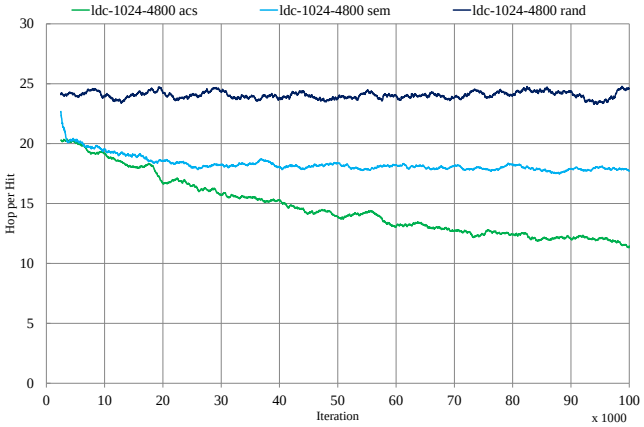
In this section we will analyze the performance of the chosen algorithms in an unstructured world. Firstly we show that we have managed to recreate the results of Semant using our testing platform and, then, we extend the comparison. The

topology used in the work [15] is always sem–1024, all the execution parameters are described in Table 2.2. 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 in the original Semant work.

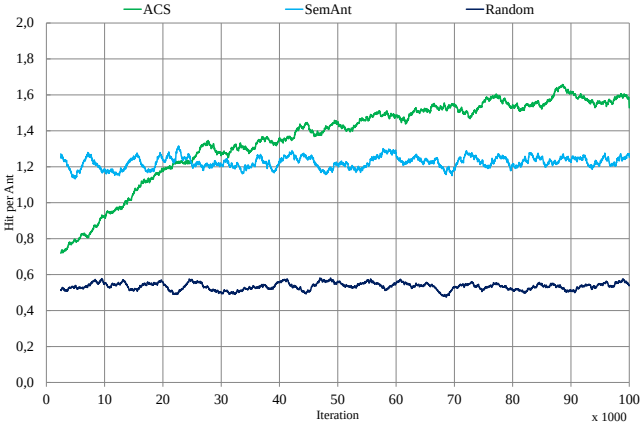
Being able to recreate results presented in [15] leads to the conclusion that the implementation we have created is a correct one albeit there is a slight and irrelevant difference in the results of random k–walker, most likely due to an implementation decision taken by authors and not specified explicitly 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.

The most notable difference between ACS and Semant is visible in the section of 0 – 10000 iterations; see Figure 2.1. 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, as the initial phase is much more intense. From the Figure 2.1a) we can conclude that the relative improvement within the initial 10% of the iterations is large: in case of Semant the average number of HpH 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 2.1b) that reflects the efficiency of a single agent: clearly Semant makes no progress in this field, the improvement of HpH 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 HpA.

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 in the hc–d tests we notice that the expected value for convergence seems to be 10 HpH, 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 relative correlation between the algorithms in this setup.




**(a)** Hop per Hit




**(b)** Hit per Ant

**Figure 2.1:** Results in sem–1024

***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 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: ldc–1024–100, ldc–1024–300, ldc–1024–600, ldc–1024–1200, ldc–1024–2400 and ldc–1024–4800 with average node degree of 5.19, 5.59, 6.17, 7.34, 9.69 and 14.38 respectively.

The results shown in the below figures and tables 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, 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 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 Figure 2.2 that effectively tripling the average degree of the nodes will not result in equivalent improvement in measurements. Still, if we consider the HpH measure 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, where the difference reaches 28%, bearing in mind that the number of links is more than 5 times the original.

The Friedman Test mean ranks are presented in Table 2.3. We can report that statistically significant difference was observed, $\chi^2 2(17) = 26865.258$, $\sigma = 0.00$. Further analysis with Wilcoson Signed Ranks Test of the values (presented in Table 2.4) proves that increasing the random links improves the measured parameter significantly (with minor exceptions in Random Walker k-2), with the significance level at $\sigma < 0.05$, and $\sigma < 0.0072$ after the Bonferroni correction. Finally, in Table 2.4, row ACS-Semant, we compare corresponding ACS and Semant results concluding that in all cases ACS is significantly better than Semant with $\sigma < 0.05$, and $\sigma < 0.0035$ after the Bonferroni correction.

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

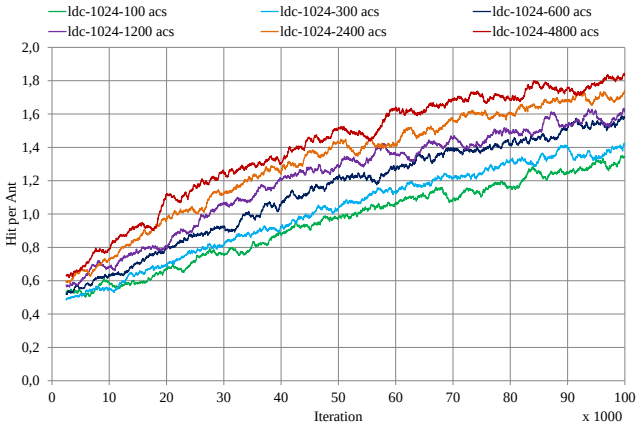
network in hope of better convergence. Note that Semant is still unable to use just two ants and only approaches this value asymptotically.

**Table 2.3:** Experiment Ranks, Friedman Test for ldc-1024-m world

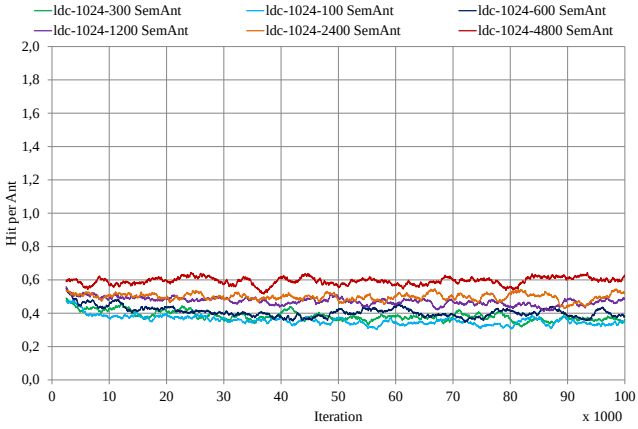
| | | | | | | |
|---|---|---|---|---|---|---|
| **Test** (acs-ldc-) | 100 | 300 | 600 | 1200 | 2400 | 4800 |
| **Mean Rank** | 8.25 | 6.19 | 5.78 | 3.65 | 2.63 | 2.03 |
| **Test** (sem-ldc-) | 100 | 300 | 600 | 1200 | 2400 | 4800 |
| **Mean Rank** | 9.43 | 9.25 | 8.3 | 8.61 | 7.56 | 7.16 |
| **Test** (ran-ldc-) | 100 | 300 | 600 | 1200 | 2400 | 4800 |
| **Mean Rank** | 15.98 | 15.62 | 15.59 | 15.41 | 15.32 | 14.23 |

**Table 2.4:** Wilcoxon Signed Ranks Test statistics, based on positive ranks, for intra-ACS, intra-Semant , intra-Random Walker k-2 and ACS to Semant inter-comparison in ldc-1024-m

| | | | | | | |
|---|---|---|---|---|---|---|
| **ACS** (acs-) | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 | -ldc-100 |
| | - | - | - | - | - | - |
| | -ldc-100 | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 |
| **Z** | -26.921 | -6.348 | -29.688 | -19.039 | -13.145 | 38.682 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Semant** (sem-) | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 | -ldc-100 |
| | - | - | - | - | - | - |
| | -ldc-100 | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 |
| **Z** | -1.713 | -12.361 | -3.589 | -12.178 | -5.765 | 22.930 |
| $\sigma$ | 0.087 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Rand.** (ran-) | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 | -ldc-100 |
| | - | - | - | - | - | - |
| | -ldc-100 | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 |
| **Z** | -6.532 | -10.008 | -2.490 | -1.778 | -17.634 | 26.960 |
| $\sigma$ | 0.000 | 0.313 | 0.013 | 0.075 | 0.000 | 0.000 |
| **ACS** (acs-) | -ldc-100 | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 |
| **-** | - | - | - | - | - | - |
| **Semant** (sem-) | -ldc-100 | -ldc-300 | -ldc-600 | -ldc-1200 | -ldc-2400 | -ldc-4800 |
| **Z** | -15.424 | -28.274 | -26.339 | -35.756 | -36.203 | -37.198 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |




**(a)** ACS




**(b)** Semant

**Figure 2.2:** Hit per Ant comparison in ldc–1024–n

### 2.4.5 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 displays performance in hc–d worlds without any hybrid approach; that is: without taking any benefit of the underlying structure, and compares 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 explores the ideas of experiment 4 and highlights some details of hc–d worlds that the previous failed to show.

***Experiment 3: Hybridless approach***

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 earlier: the node degree has very little impact. In Figure 2.3 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 2.5.

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 measure (Figure 2.4, Table 2.7) ACS comes out ahead. One thing needs to be pointed out: it is interesting to see how ACS struggles increasingly to converge with the size of the world – e.g. a small world such hc–7 is fully penetrated after less than 5000 iterations, whereas one of size hc–13 still has not reached its horizon even after $10^5$ iterations. Semant suffers such a problem as well, however in a slightly less impacting manner: in spite of the high iteration impact, it does not reach the value of about 2 Ant per Query, which would be 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.

The statistical analysis resulted in similar conclusions to the ones obtained in Experiment 2. Table 2.6 presents the confirmation of statistical differences, $\chi^2(20) = 34130.756$, $\sigma = 0.00$; in Table 2.7 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 row ACS – Semant of Table 2.7 proves the superiority of ACS over Semant at $\sigma < 0.0017$. Bonferroni correction was applied in all the cases.

**Table 2.5:** Comparison: hc-10 with ldc-2400

| **Paired** | **Mean** | **Std.** | | **95% Conf. Int.** | | **df** | **Sig.** |
|---|---|---|---|---|---|---|---|
| | | Dev | Error | Lower | Upper | | |
| **acs-hc-10** - **acs-ldc-2400** | 0.4592 | 1.4060 | 0.0314 | 0.3976 | 0.5209 | 1999 | 0.00 |
| **sem-hc-10** - **sem-ldc-2400** | 0.2373 | 1.7562 | 0.0392 | 0.1603 | 0.3143 | 1999 | 0.00 |

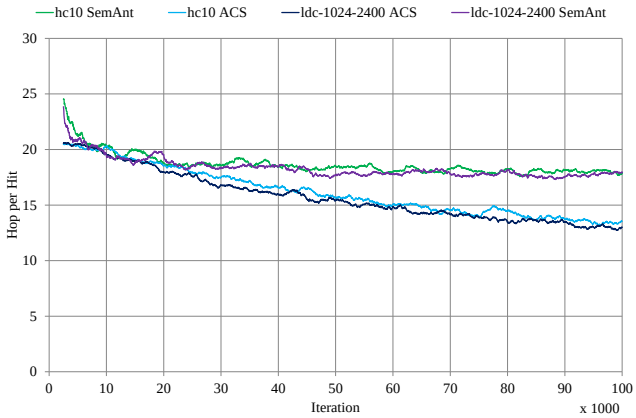
**Table 2.6:** Experiment Ranks, Friedman Test for ldc-1024-m world

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Test** (acs-hc-) | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Mean Rank** | 1.25 | 2.53 | 3.31 | 5.14 | 7.64 | 10.1 | 11.72 |
| **Test** (sem-hc-) | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Mean Rank** | 5.43 | 6.08 | 7.97 | 9.46 | 10.86 | 12.38 | 14.10 |
| **Test** (ran-hc-) | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Mean Rank** | 16.05 | 17.58 | 17.73 | 17.9 | 17.98 | 17.95 | 17.73 |

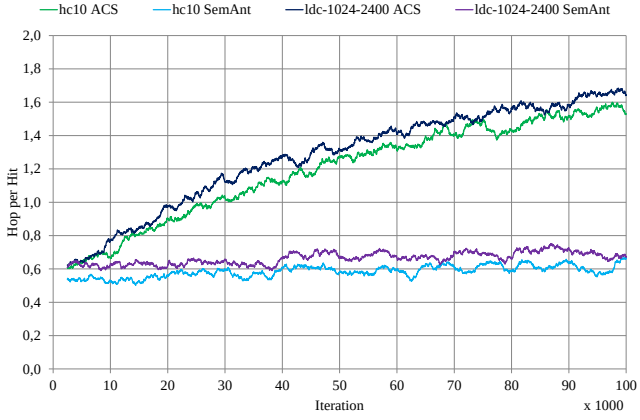
***Experiment 4: Hybrid approach***

In this section we will show the impact of adding a hybrid path processing to a known topology – in this case hc–d, as it was considered in the experiment 3. The worlds identical to experiment 3 will be chosen, so 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 general, as it requires the hypercube topology. It is based around exploiting the knowledge provided by this fact.

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 measure, reaching considerable results, compare 2.6 versus 2.2 and 2.5. In Table 2.8 we can observe that the difference in HpH measure is about 8.44 for ACS and 10.64 for Semant, in absolute values. This is a highly relevant difference and proves that it has been a quality leap from a hybridless hypercube to a hybrid hypercube.

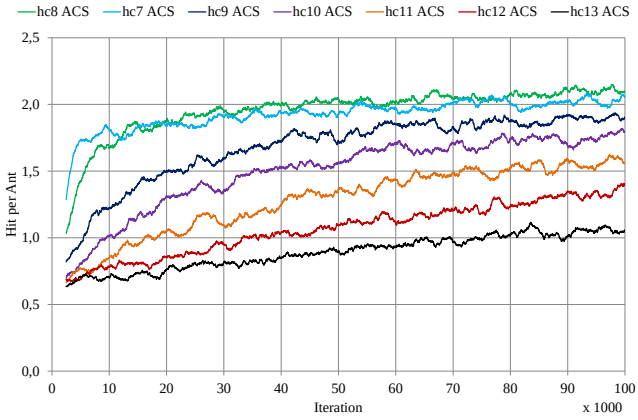



**(a)** Hop per Hit

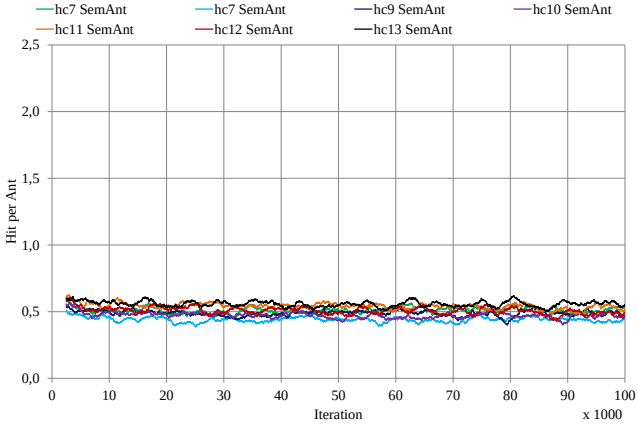



**(b)** Hit per ant

**Figure 2.3:** Intra world comparison: ldc–1024–2400 vs. hc–10




**(a)** ACS




**(b)** Semant

**Figure 2.4:** Hit per Ant comparison in hc–d non–hybrid approach (Random omitted as it is constant)

**Table 2.7:** Wilcoxon Signed Ranks Test statistics, based on positive ranks, for intra-ACS, intra-Semant , intra-Random Walker k-2 and ACS to Semant inter-comparison in ldc-1024-m

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **ACS** (acs-) | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 | -hc-7 |
| | - | - | - | - | - | - | - |
| | -hc-7 | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 |
| **Z** | -35.088 | -18.649 | -33.309 | -34.594 | -29.898 | -23.934 | 38.730 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Semant** (sem-) | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 | -hc-7 |
| | - | - | - | - | - | - | - |
| | -hc-7 | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 |
| **Z** | -10.956 | -25.574 | -18.510 | -15.284 | -16.240 | -17.386 | 38.725 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Rand.** (ran-) | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 | -hc-7 |
| | - | - | - | - | - | - | - |
| | -hc-7 | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 |
| **Z** | -20.402 | -1.885 | -2.888 | -0.191 | -0.282 | -0.829 | 22.838 |
| $\sigma$ | 0.000 | 0.059 | 0.004 | 0.848 | 0.778 | 0.407 | 0.000 |
| **ACS** (acs-) | -hc-7 | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 |
| **-** | - | - | - | - | - | - | - |
| **Semant** (sem-) | -hc-7 | -hc-8 | -hc-9 | -hc-10 | -hc-11 | -hc-12 | -hc-13 |
| **Z** | -38.570 | -37.639 | -37.307 | -35.293 | -29.590 | -23.308 | -21.946 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

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

The convergence limits 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 2.5a. That is a significant improvement of factor oscillating between $\times 4$ and $\times 5$. 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 2.5b.

ACS, as earlier, outperforms Semant, but here the quick convergence of Semant plays a much more crucial role. In 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 in Figure 2.6). Consider, for instance, the point 2.1b. There is a short section of 5000 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 $10^5$ iterations. The horizon is never reached and the convergence does not occur. To further analyze this phenomenon we proposed experiment 5.

Again the output of the Friedman Test confirms the graphical analysis (see Table 2.9). $\chi^2(20) = 33529.717$, $\sigma = 0.00$. In case of the Wilcoxon Sign Ranked comparison (Table 2.10, $\sigma < 0.0035$) 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 ACS – Semant row, in Table 2.10. 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.

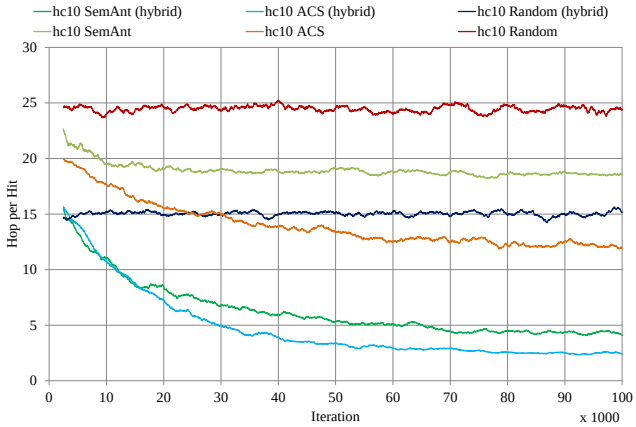
The hybrid approach has allowed all the algorithms to reach convergence of values never achieved before. The costs of the hybrid approach are: slightly increased computation effort, further positioned convergence horizon and the requirement to establish a topology.

**Table 2.8:** Comparison: hc-10 with ldc-2400

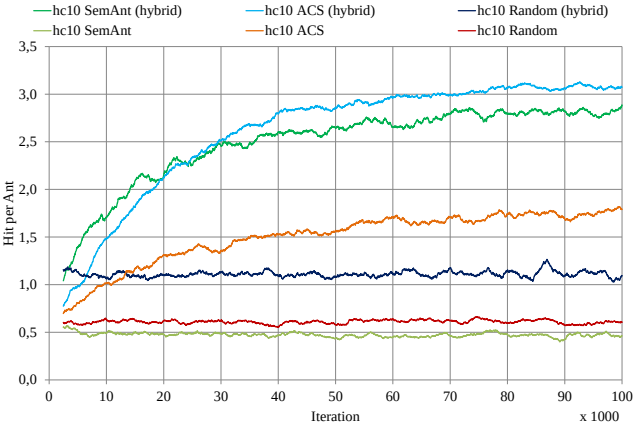
| **Paired** | **Mean** | **Std.** | | **95% Conf. Int.** | | **df** | **Sig.** |
|---|---|---|---|---|---|---|---|
| | | Dev | Error | Lower | Upper | | |
| **acs-hc-10** - **acs-hco-10** | 8.4376 | 2.7425 | 0.0613 | 8.3173 | 8.5579 | 1999 | 0.000 |
| **sem-hc-10** - **sem-hco-10** | 10.6375 | 4.00151 | 0.0897 | 10.4615 | 10.8136 | 1999 | 0.000 |

**Table 2.9:** Experiment Ranks, Friedman Test for hc-d world with hybrid path optimization

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Test** (acs-hco-) | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Mean Rank** | 5.76 | 3.07 | 2.67 | 4.99 | 8.31 | 11.96 | 13.04 |
| **Test** (sem-hco-) | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Mean Rank** | 7.02 | 5.15 | 5.18 | 6.47 | 8.14 | 11.39 | 13.95 |
| **Test** (ran-hco-) | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| **Mean Rank** | 16.78 | 17.39 | 17.98 | 17.70 | 17.96 | 18.01 | 18.09 |

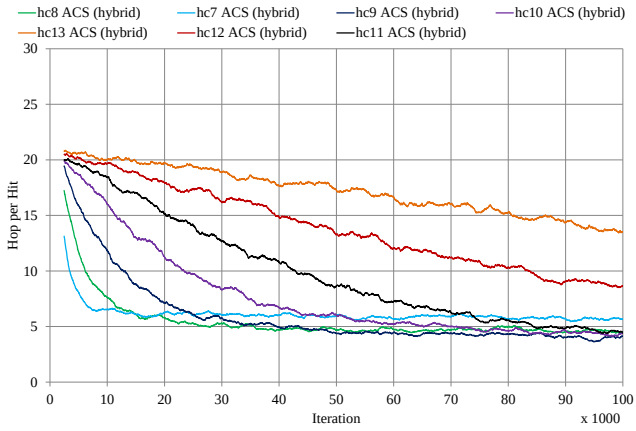



**(a)** Hop per Hit

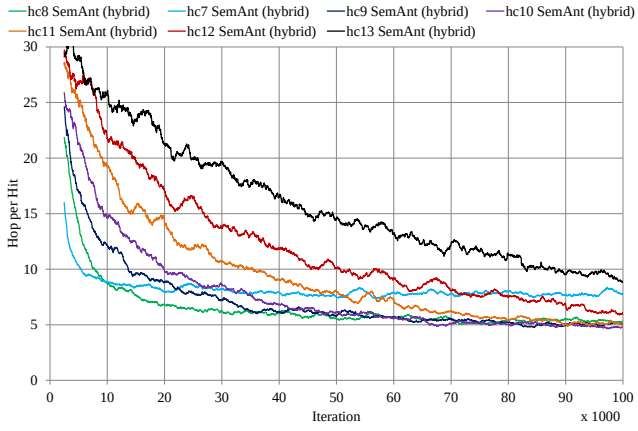



**(b)** Hit per ant

**Figure 2.5:** Intra world comparison: hc–10 – hybrid vs. hc–10 non–hybrid

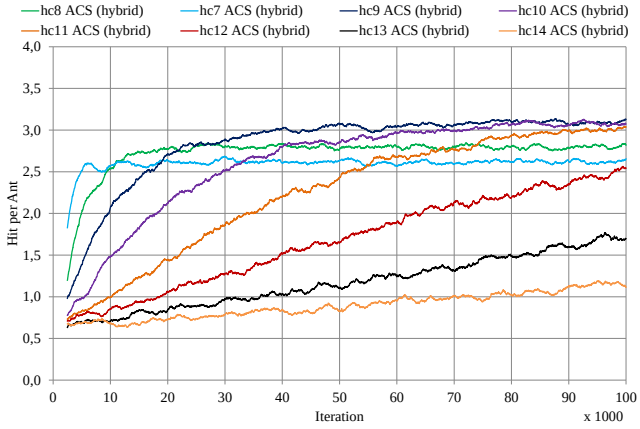



**(a)** ACS

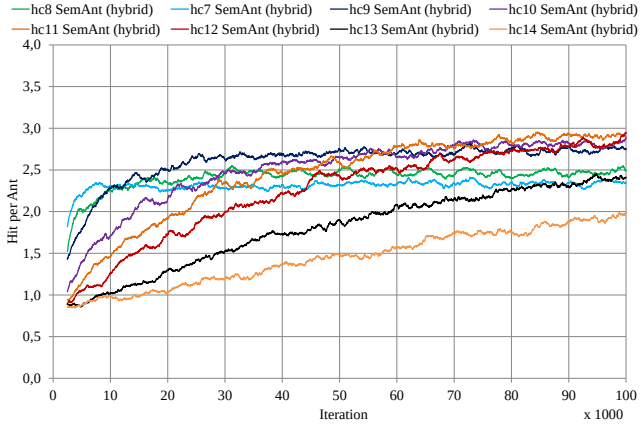



**(b)** Semant

**Figure 2.6:** Hit per Ant comparison in hc–d with hybrid approach (Random omitted as it is constant)




**(a)** ACS




**(b)** Semant

**Figure 2.7:** Hop per Hit comparison in hc–d with hybrid approach (Random omitted as it is constant)

**Table 2.10:** Wilcoxon Signed Ranks Test statistics, based on positive ranks, for intra-ACS, intra-Semant, intra-Random Walker k-2 and ACS to Semant inter-comparison in hc-d with hybrid path optimization

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **ACS** (acs-) | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 | -hco-7 |
| | - | - | - | - | - | - | - |
| | -hco-7 | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 |
| **Z** | -24.130 | -0.698 | -32.828 | -37.134 | -38.101 | -33.411 | 38.734 |
| $\sigma$ | 0.000 | 0.485 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Semant** (sem-) | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 | -hco-7 |
| | - | - | - | - | - | - | - |
| | -hco-7 | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 |
| **Z** | -19.591 | -5.183 | -20.480 | -24.662 | -34.783 | -34.470 | 38.724 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| **Rand.** (ran-) | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 | -hco-7 |
| | - | - | - | - | - | - | - |
| | -hco-7 | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 |
| **Z** | -9.142 | -8.679 | -4.220 | -4.185 | -10.047 | -1.671 | 19.031 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.295 | 0.095 | 0.000 |
| **ACS** (acs-) | -hco-7 | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 |
| **-** | - | - | - | - | - | - | - |
| **Semant** (sem-) | -hco-7 | -hco-8 | -hco-9 | -hco-10 | -hco-11 | -hco-12 | -hco-13 |
| **Z** | -23.938 | -28.946 | -28.437 | -130.053 | 12.639 | 21.502 | 10.599 |
| $\sigma$ | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

***Experiment 5: Hybrid approach 500k***

The horizon problem has caused results in higher dimension worlds of the previous test become unconverged due to the limitation on the number of iterations. 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 $10^5$ to $5 \times 10^5$, which was deemed sufficient.

The results are presented in Figure 2.8. It quickly became apparent that the limit of $5 \times 10^5$ 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 2.11) with values $\chi^2(3) = 16442,921$, $\sigma = 0.00$. Finally we determine that Semant is better in both inter-comparisons: sem-hco-12 500k – acs-hco-12 500k was ranked at $-52.790$, while sem-hco-13 500k – acs-hco-13 500k at $-75,607$, both with Asymp. Sig (2-tailed) below 0.0035.

The comparison of the results in the previous section 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 by a single iteration of ACO, 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, since additional ants 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.

**Table 2.11:** Experiment Ranks, Friedman Test for hc-d world with hybrid path optimization at 500k

| **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 |

### 2.4.6 Time-based analysis

In order to put the preceding results in perspective we chose to translate the iteration-based results into time-based ones. The most crucial discussion in this approach is the point in time at which ACS stats to score better results, in terms of Hop per Hit, than Semant. In Table 2.12 we present the compiled results. Iterations are grouped down to a unit of 1000, while time is measured with the precision of 0.1 [s].

It can be observed that in the case of the hypercube topology ACS is competitive against Semant even in early iterations. In the case of a toroidal network with long distance connections ACS becomes even more so as the number of long distance connections increases. Finally, in the case of a hybrid hypercube approach as the size of the network increases, the number of iterations needed for ACS to surpass Semant in terms of Hops per Hit increases. In this case ACS would be less recommended for applications requiring real-time P2P resource searching.

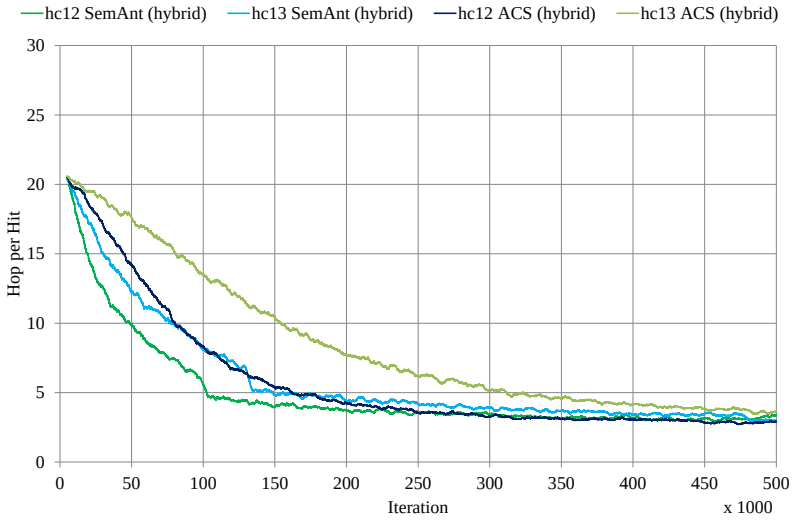



**Figure 2.8:** Hop per Hit in the world hc–12 and hc–13 with extended iteration horizon

## 2.5 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 network's 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 relevant to the results and can have a very positive impact. Similar

**Table 2.12:** Time-based analysis. Point in time, at which ACS surpasses SemAnt

| Test | Iteration - | Time [s] |
|---|---|---|
| ldc-100 | 40000 | 120.0 |
| ldc-300 | 18000 | 59.4 |
| ldc-600 | 10000 | 33.0 |
| ldc-1200 | 8000 | 24.0 |
| ldc-2400 | 2000 | 6.1 |
| ldc-4800 | 0 | 0 |
| **Test** | **Iteration -** | **Time [s]** |
| hc-7 | 0 | 0 |
| hc-8 | 0 | 0 |
| hc-9 | 0 | 0 |
| hc-10 | 0 | 0 |
| hc-11 | 0 | 0 |
| hc-12 | 0 | 0 |
| hc-13 | 0 | 0 |
| **Test** | **Iteration -** | **Time [s]** |
| hco-7 | 0 | 0 |
| hco-8 | 0 | 0 |
| hco-9 | 7000 | 21.0 |
| hco-10 | 21000 | 88.2 |
| hco-11 | 39000 | 105.3 |
| hco-12 | 81000 | 267.3 |
| hco-13 | >100000 | >318.0 |

conclusions, with no empirical backup demonstrated, have been suggested in [15] and [31].

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 into 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.

In summary, the above conclusions must be taken into account when attempting the construction of a high quality ACO adaptation in the field of P2P.

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.

## 2.6 Acknowledgments

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) project TIN2010–20488. Kamil Krynicki is supported by a FPI fellowship from Universidad Politécnica de Valencia.

# Chapter 3

# Ant Colony Optimization for resource querying in dynamic peer-to-peer grids

Kamil Krynicki Javier Jaen Jose A. Mocholi

ISSI Research Group, Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

(2014), International Journal of Bio-Inspired Computation, 6(3), 153.
doi:10.1504/IJBIC.2014.062634

## Abstract

The applicability of peer-to-peer (P2P) in the domain of grid computing has been an important subject over the past years. Nevertheless, the sole merger between P2P and the concept of grid is not sufficient to guarantee nontrivial efficiency. Some claim that ant colony optimization (ACO) algorithms might provide a definite answer to this question. However, the use of ACO in grid networks causes several problems. The first and foremost stems out of the fact that ACO algorithms usually perform well under the conditions of static networks, solving predetermined problems in a known and bound space. The question that remains to be answered is whether the evolutive component of these algorithms is able to cope with changing conditions; and by those we mean changes both in the positive sense, such as the appearance of new resources, but also in the negative sense, such

as the disappearance or failure of fragments of the network. In this paper we study these considerations in depth, bearing in mind the specificity of the peer-to-peer nature.

## 3.1 Introduction

The idea of the grid originated in the early nineties as a new metaphore for distributed computing with resources that are heterogeneous and dynamic in nature. The key idea behind it was to loosely couple computing nodes to allow them to collaborate in order to achieve a common goal. Due to an apparent ambiguity and similarity to other solutions such as parallel computing and computer clusters a proper definition was provided in [56]. A checklist of requirements that define grid computing consists of three primary attributes: computing resources are not administered centrally, open standards are used, nontrivial quality of service is achieved.

Over the past years many subgenres of grid computation have spawn. They include many concepts, just to name a few: Software as a Service , DataGrids [57], CPU scavenging – with its prime example SETI@Home [23]. The last mentioned is, interestingly, also a very good example of peer-to-peer (P2P) cooperation and a proof of these two computation architectures complementing each other. Trunfio et al. [58] formalize the split between the technologies stating that the grid used to be restricted to scientific and limited applications as opposed to P2P which was much more wide scale and accessible.

The key point of the grid computing model is the question of the underlying protocol and node organization. One of the most prominent approaches is the use of P2P architectures as a base building block (the communication layer) and authors agree that it is a suitable platform for grid computing [59], sometimes seeing it as a sister technology. This distributed infrastructure does fulfil automatically the first two of the three required points: the distributed resources and open standards. Also, it allows transferring all the knowledge developed in the field of P2P with all its benefits to the grid computing. The third point, however, which is the question of the efficient service remains to be answered; which becomes even more crucial if we take into consideration the growing scope and highly dynamic nature of P2P networks.

In this work we chose to examine the ant colony optimization metaheuristic as a possible solution to the effective searching of resources in dynamic P2P grids. ACO algorithms achieved very good results in problems that include routing and resource discovery as shown in [17], and there were even attempts, of limited extent, to apply them to the P2P grid itself [60], which proves the importance of this idea.

However, no previous work has analyzed the suitability of this algorithmic strategy under all the types of dynamism considered in our study.

In order to observe and analyze the applicability of ACO under those conditions we have designed several experiments to isolate a set of real-life resembling forms of variability that a network might experience and, consequently, subject selected algorithms to tests based around those forms of variability. The direct objective is to establish whether or not the algorithms designed to work in static environments can cope with various forms of dynamism.

## 3.2 Related Work

The related works, to the question of resource discovery in general, might be categorized as follows: the centric solutions, the Grid solutions, the P2P Grid solutions and the P2P Grid solutions with ACO as the search algorithm.

The centric solutions are feasible only in small-scale due to a strong limitation cap of processing power, storage space, etc. Another drawback is that servers are single points of failure, creating potentially vulnerable systems. Consequently, there have been steps taken in order to disperse the system among a grid of nodes, following the metaphor of an electric power grid [56], with many challenges of its own. Some try to achieve efficiency in such setups using proprietary or mixed approaches [61] [62], some base themselves on the P2P paradigm [63], while other use ACO algorithms or ACO-grid combinations [64] [65].

In the mixed approaches category it is worth pointing out the work of Brocco et al. [62] presenting an interesting approach to the subject of resource discovery in grid networks. They prepared a hybrid solution encompassing a wide range of algorithms: ACO, local flooding with caching and replication based on gossiping algorithms; they report satisfactory results, but limiting the size of the network to 1281 nodes and analyzing their algorithm under just one type of dynamism - defined in this paper as reconfiguration, which is just one of the types of dynamism considered in our work. A purer approach is presented in [65], where an ACO-like mechanism is the only one that governs the behaviour of search agents in a grid. Here, similarly, the concept of data descriptors' replication is present, which is outside our objectives and, as it occurs in [62], only reconfiguration operations are considered as a form of dynamism affecting the underlying network.

We must also mention here the work of Deng et al. [60]. They touched upon the subject of ACO in P2P Grids, however with slightly different emphasis. In our case the experimental study focus is placed on ACO; in their work it is on P2P Grids. Also, while we examine various existing ACO algorithms and observe their behaviour when applied to the question of P2P, they design a proprietary ACO algorithm (roughly based on ACS [17]) along with P2P Grid and test its

robustness against a non-ACO method, namely distributed hash table routing. The scope of our experiments is also larger, extending to thousands of nodes and tens of thousands of links over one hundred thousand iterations. Additionally, they do not take in consideration the fundamental concept of dynamism in the network as we do.

In summary, the main contribution of this work is twofold. First, it is a pure ACO solution in the field of P2P Grids, without any kind of hybridization in form of replication, flooding, etc. Second, we attempt to exhaust the subject of structural dynamism and its impact on ACO in general, a subject that is very seldom explored to this extent. We do so in networks, which are built to be as close as possible approximations of real-life systems: they consist of thousands of nodes and tens of thousands of links.

## 3.3 Ant Colony Optimization in P2P

Ant Colony Optimization [33] is a swarm intelligence approach to problem-solving introduced by Marco Dorigo in his work on distributed optimization in 1991. 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 [7], optimizing a given function [34] [66] or in other graph-related problems, such as T-Colouring [67]. 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 $G(V, E)$ of finite size with vertices $v \in V$ representing possible locations of resources and edges $e \in E$ representing trails.

In order to consider ACO P2P-compliant we need to formulate additional requisites. Every ACO-based query resolution algorithm in P2P environments must conform to the below query-resource *(q-r)* principles:

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 answered with resources residing in one or many nodes within the network.
3. Every node must not be aware of the content of any other node but itself.

4. Every node may 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.

ACO methods can be grouped into two broad groups, namely: single ant and multi ant, with the distinction revolving around the amount of ants generated for solving one act of search. For each group we chose a representative for our experimental evaluation: RC-ACS (Routing Concept ACS), our proposed semantic extension of ACS [17] and Semant [15] for single- and multi- ant approaches respectively. These two representatives have been selected after a prior examination of ACO algorithms based on two factors: firstly, compliance with the previous (q-r) principles (i.e., ACO strategies violating any of the previous q-r principles were discarded because they would not be applicable in the P2P domain) and, secondly, recognized performance under static conditions.

Our semantic extension of ACS, RC-ACS, which is based on the pheromone-per-keyword heuristic as in [15], is designed taking into account that in a P2P network there is a large quantity of traffic of very different natures. The pheromone left by queries for certain resources must not influence the search of other, which may be unrelated. In order to tackle this problem we introduce the notion of Routing Concept, noted as $\rho$. It requires that every edge maintains, not a single pheromone value $\tau(vu)$ between its source node $v$ and each neighbor node $u$, but rather an associative array $\tau_\rho(vu)$, where $\rho$ is obtained from the current request by means of $\rho = \rho(Q)$, where $\rho(Q)$ is a function that extracts the Routing Concept based on the Query $Q$ issued. In our case we defined the function $\rho(Q)$ as one that returns the taxonomical label of the parent concept of the requested one.

Ants follow a simple and non-deterministic search algorithm that we summarized in Fig 5

**Algorithm 5:** ACO-based query in graphs. ACS/SemAnt differences

1: **let** Request $R$ be generated by the vertex $v_0$ and routed by ant $A$. $i = 0$.
2: **repeat**
3: A attempts to satisfy R in the current vertex $v_i$.
4: **if** R satisfied **then**
5: end loop
6: **end if**

7: Choose randomly between exploration and exploitation
8: **if** exploitation **then**
9: Go to vertex $v_{i+1}$:

$$v_{i+1} = \text{argmax}_{u \in V} \{\tau_\rho(v_i u) \times \eta(v_i u)^\beta\} \tag{3.1}$$

where:

- $\tau_\rho(v_i u) \in [0,1]$ is the pheromone value on the edge $v_i u$, for the routing concept $\rho$. $\tau_\rho \in [0,1]$,if $v_i u \in E$; $\tau_\rho = 0$,if $v_i u \notin E$)
- $\eta(v_i u)$ is the cost value on the edge $v_i u$
- $\beta \in [0,1]$ is a parameter

10: **else if** exploration **then** Evaluate

$$p(v_i, u) = \frac{\tau_{v_i u} \times \eta^\beta_{v_i u}}{\sum_{z \in V} \tau_{v_i z} \times \eta^\beta_{v_i z}} \text{if } v_i u \in E \tag{3.2}$$

11: (*RC-ACS) Go to vertex $v_{i+1}$, chosen from all $u \in V$with probability $p(v_i, u)$
12: (*SemAnt) Send a clone of the original ant $A$ to every vertex $z$, each with probability $p(v_i, z)$
13: **end if**
14: Perform local pheromone update on the edge $v_i$ and $v_{i+1}$:

$$\tau_\rho(v_i v_{i+1}) \leftarrow (1-\rho) \cdot \tau_\rho(v_i v_{i+1}) + \rho \cdot \gamma \cdot \max_{z \in V} \tau_{v_i z} \tag{3.3}$$

where:

- $\rho \in [0,1]$ and $\gamma \in [0,1]$ are parameters

15: **until** R is not satisfied or A has not terminated
16: Perform global pheromone update for every edge $vu$ in the solution $S$ to the request $R$
17: (*RC-ACS)

$$\tau_\rho(vu) \leftarrow (1-\alpha) \cdot \tau_\rho(vu) + \alpha \times \frac{1}{|L|} \tag{3.4}$$

where:

- $|L|$ is the amount of edges in the solution $S$
- $\alpha \in [0,1]$ is a parameter

18: (*SemAnt)

$$\tau_\rho(vu) \leftarrow \tau_\rho(vu) + w_d \times \frac{|S|}{S_{max}} + (1-w_d) \times \frac{|L|}{2L_{max}} \tag{3.5}$$

where:

- $|S|$ is the amount of resources in the solution

- $S_{max}$ is the maximum amount of resources permitted
- $L_{max}$ is the maximum amount of edges in the solution permitted
- $w_d \in [0, 1]$ is a parameter

## 3.4 Experimental methodology

We define the graph topology as follows: the graph $G(V, E)$ consists of $|V|$ vertices, where $|V|$ is an even number. The edges $E$ are organized in a fully connected grid with a toroidal topology, 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. Consequently, we create the graph $G'(V, E \cup LDC)$, where $LDC$ is a set of all long distance connections ($LDC_{ij}$) in the graph $G$; every vertex $v_i$ has exactly one long distance edge that connects it directly to a random vertex $v_j$, with the pure toroidial distance $d_E(v_i v_j) > 2, \forall (v_i v_j) \in E$.

$$LDC \subset v_i v_j | (v_i v_j) \notin E \wedge d_E(v_i v_j) > 2 \tag{3.6}$$

The probability of a node $v_i$ having $LDC_{ij}$ of length $d_E(v_i v_j)$ is proportional to $d_E(v_i v_j)^( - 1)$. The above explanation is a strict reformulation of the approach from [15].

Although the execution of every experiment will begin with the $G'$ topology, this topology will be subject to changes according to the type of dynamism studied in each experiment. Therefore, each experiment will define two concepts:

1. The modification function of the given graph $\Delta : G \to G$, which transforms the graph $G$.
2. The modification function period $\Delta_\rho$ is an integer which expresses how often $\Delta$ function is executed.

Each experiment will be evaluated according to a common quality measure called Hop per Hit *(HpH)*, which is a dimensionless ratio that has been effectively used to measure performance in P2P systems in previous works [15] [18]. In this metric hop is the total amount of steps taken by an agent and hit is the amount of resources found.

See Table 3.1 for details on algorithm execution parameters.

**Table 3.1:** Execution parameters

| Parameter | Interpretation | Value (RC-ACS) | Value (SemAnt) |
|---|---|---|---|
| $TTL$ | Time to Live | 25 | 25 |
| $q_0$ | Weight of exploiting vs. exploring strategy | 0.80 | 0.85 |
| $R_{max}$ | Maximum number of resources to fetch | 10 | 10 |
| $R_{min}$ | Minimum number of resources to fetch | 5 | 5 |
| $\alpha$ | Weight of newly deposited pheromone | 0.07 | N/A |
| $w_d$ | Weight of resource quantity vs. link costs | N/A | 0.5 |
| $\beta$ | Weight of link costs | 1 | 1 |
| $\gamma$ | Factor in pheromone evaporation | 0.02 | N/A |
| $\rho$ | Weight of evaporation | 0.10 | 0.07 |
| $ph_{min}$ | Minimum pheromone level | 0.001 | 0.001 |
| $ph_{max}$ | Maximum pheromone level | 1 | 10000 |
| $ph_{init}$ | Initial pheromone level | 0.009 | 0.009 |

### 3.4.1 Test Setup

Every test will consist of an amount $it_{max}$ of iterations. Iteration is defined as a single act of querying, launched from a random node, routed according to the given algorithm. 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 and the location of resources found. Due to the large amount of data and its high variability, we chose to use simple rolling average of size 254 as an impulse filter and plot the function $HpH(it)$, where it is the iteration number. The amount of queries will be fixed at $it_{max} = 10^5$.

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.

***Taxonomy and resource distribution***

ACM Computing Classification System [53] will be the taxonomical vocabulary used. Every resource in the network $G'$ is described by one, and only one leaf taxonomical concept $t$ of the ACM classification. A resource has therefore only two properties: its owner vertex $v$ and a taxonomical label $t$. It is written as $r(v,t)$. Note that $v_1 = v_2 \wedge t_1 = t_2 \not\Longrightarrow r_1(v_1,t_1) = r_2(v_2,t_2)$. Such an approach leads to valuing higher those nodes that provide many resources of the same $t$, which is the objective.

The distribution of resources within the network follows strictly the approach by the test setup [15]. 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 resource 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, reflecting the fact that people have specific interests and hobbies [54].

***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.

The resolution of a query $q_r(t_r)$ in a vertex $v$ consists of finding all the resources that have been labeled with $t_r$, that is, all the resources $r_r \in \{r|\exists r(v,t_r)\}$. During the evaluation process every node of the graph G of size $|E|$ has a probability of being chosen to generate a query $q$ with the probability of $\frac{1}{|E|}$.

### 3.4.2 Types of dynamism

We decided to focus on the most common types of dynamism a real-life P2P network can experience. They are: expansion (the process of densification of the network), contraction (the process of rarefaction of the network), enrichment (the process of addition of new resources) and reconfiguration (the process of migration of nodes within the network) as it is summarized in Table 3.2.

The network expansion is one of the most straightforward ways to improve the robustness of algorithms within a P2P network. The subject of expanding the net-

work by adding links in both uninformed and informed manner has been examined many times; for instance, in the case of Overlay Networks [43] and Interest-Based Locality [68], as well as Acquaintance Links [69]. Here we will show the effect of random network growth specifically on ACO algorithms in order to express the fact that users might be allowed to create direct links, or shortcuts, to their favorite nodes, with the assumption that it is somewhat a random process.

The network contraction is a far less studied phenomenon. In this case the nodes and their corresponding resources remain unchanged, but the interconnecting network does shrink, leaving nodes disconnected and the system partitioned. Naturally this must hinder the algorithms ability to obtain good results therefore the response should be negative. This kind of behaviour might be seen as a situation in which the communication between the nodes is progressively worsened by, for instance, an on-going electrical storm.

By the network enrichment we understand the increase in the amount of resources that every node provides. This clearly must enable the algorithms to gather more resources, but the question of whether the newly provided resources can be effectively discovered remains to be answered. Note that it is much more likely for users to add resources than to remove them so the phenomenon of resources massively disappearing from the system is secondary. Hence the counter-part to this dynamism, might we call it *Network impoverishment*, is not included. This is a direct conclusion form a universal and long-lasting trend of ever-increasing traffic over file sharing services and similar [70]. The process of network enrichment from the perspective of the processing grids could be seen as an act of expanding the computer capabilities to provide more services.

The more mobile the network's users are, the more important the network reconfiguration becomes. If a P2P network provides location based resources one would suppose it would be beneficial to reattach the user to the nodes in the current neighborhood. This directly implies a disappearance from the original location and appearance in another. It is a very interesting concept to examine from the ACO's point of view, seeing how they take a period of time to stabilize a path and then to remove it. If a key resource provider in a given neighborhood disappears, it should lead to a sudden drop in the quality of results, therefore, we expect this type of dynamism hinder results severely.

**Table 3.2:** Comparison of dynamism types

| Dynamism | Experiments | Expected impact | Example of a real-life analogue |
|---|---|---|---|
| Expansion | NetGrow 10k | Positive | Users create direct links to favorite sources |
| Contraction | NetShrink 10k | Negative | Communication distortion due to interference |
| Enrichment | ResGrow 10k | Positive | Nodes are equipped with new devices; users share more files |
| Reconfiguration | NodeMigrate 10k | Negative | Mobile users travel to distant locations |

## 3.5 Experimental study

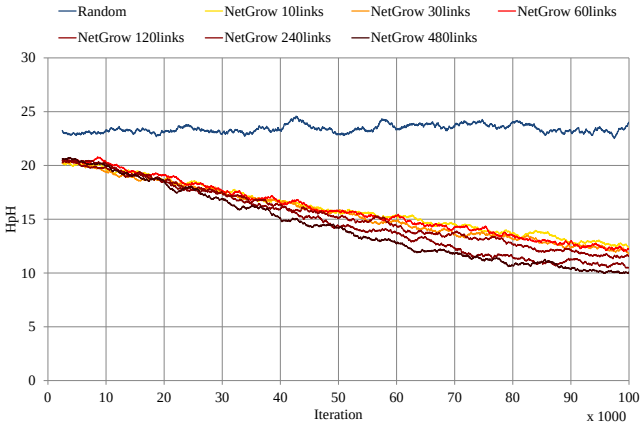
In this section we present all the experimental results. In experiments 1 - 4 we examine in detail all the types of dynamism mentioned in Section 3.4.2. In every experiment the random behaviour (RandomWalks k2) is shown as the baseline and the minimum HpH expectancy.

### 3.5.1 Experiment 1: NetGrow 10k

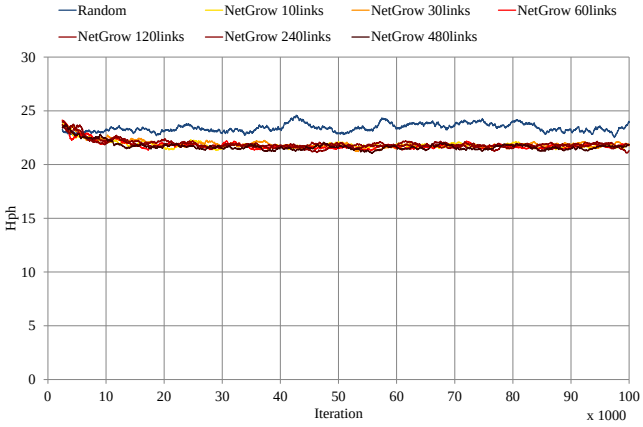
**Table 3.3:** NetGrow 10k experiments

| Experiment | Graph size $\|V\|$ | Growth Size $\Delta$ | Initial Edge Set Size $\|E\|_{init}$ | Final Edge Set Size $\|E\|_{fin}$ | Initial Average Vertex Degree $\overline{deg(v)}_{init}$ | Final Average Vertex Degree $\overline{deg(v)}_{fin}$ |
|---|---|---|---|---|---|---|
| NetGrow10k 10links | 1024 | 10 | 2560 | 2650 | 5.00 | 5.18 |
| NetGrow10k 30links | 1024 | 30 | 2560 | 2830 | 5.00 | 5.53 |
| NetGrow10k 60links | 1024 | 60 | 2560 | 3100 | 5.00 | 6.05 |
| NetGrow10k 120links | 1024 | 120 | 2560 | 3640 | 5.00 | 7.11 |
| NetGrow10k 240links | 1024 | 240 | 2560 | 4720 | 5.00 | 9.22 |
| NetGrow10k 480links | 1024 | 480 | 2560 | 6880 | 5.00 | 13.44 |

We have executed this experiment in 6 independent configurations; see Table 3.3 for the details. The most visible conclusion from the results presented in Figure 3.1 is that both algorithms struggle to appreciate the additional shortcuts added




**(a)** RC-ACS




**(b)** SemAnt

**Figure 3.1:** HpH in NetGrow 10k

to the system. In case of multi-ant behaviour it is particularly striking, as there is absolutely no improvement detected in terms of HpH. Moreover, SemAnt remains better than random behaviour only by a slight margin. There is improvement detected in case of RC-ACS, but not proportional to the amount of edges added, as in the variant NetGrow10k 480links we increase the edge set from 2560 to 6880, while the final average HpH reaches only 20% improvement.

### 3.5.2 Experiment 2: ResGrow 10k

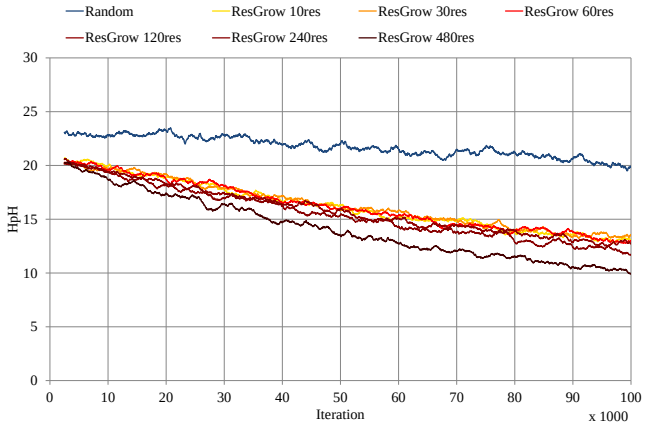
**Table 3.4:** ResGrow 10k experiments

| **Experiment** | **Graph size** $\lvert V\rvert$ | **Growth Size** $\Delta$ | **Initial Res. Count** $\lvert Res\rvert_{init}$ | **Final Res. Count** $\lvert Res\rvert_{fin}$ | **Initial Avg. Res. Count** | **Final Avg. Res. Count** |
|---|---|---|---|---|---|---|
| ResGrow10k 10res | 1024 | 10 | 30720 | 30810 | 30 | 30.09 |
| ResGrow10k 30res | 1024 | 30 | 30720 | 30990 | 30 | 30.26 |
| ResGrow10k 60res | 1024 | 60 | 30720 | 31260 | 30 | 30.53 |
| ResGrow10k 120res | 1024 | 120 | 30720 | 31800 | 30 | 31.05 |
| ResGrow10k 240res | 1024 | 240 | 30720 | 32880 | 30 | 32.11 |
| ResGrow10k 480res | 1024 | 480 | 30720 | 35040 | 30 | 34.22 |

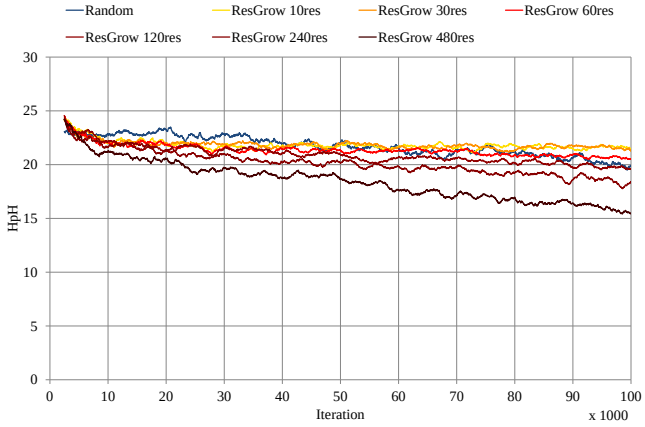
As in real life resources constantly appear and disappear from the system; if you consider data grid systems, replication services may constantly add new files to the distributed repositories. Therefore, the question to answer here is how effective are the proposed algorithms in finding emerging resources, while the already found ones are still in place. In order to simulate this dynamic we designed ResGrow experiment. See Table 3.4 for the independent experiment configurations used.

This experiment was designed as the one with the most positive impact expected and, also, the one that affects the given state of pheromone the least, as it only manipulates the resources. Consequently, it was expected that, in the worst case, the results obtained are at least as good as the alteration-free execution.

As it can be observed in Figure 3.2 this indeed was the case. Every algorithm reported a significant improvement, disproportionately large, when compared to the increase of the amount of resources per node. In the most extreme case the improvement on HpH measure reaches 28% as a result of the growth of the resource pool by only 13%. Note that in Figure 3.2 the HpH improves linearly with the total amount of resources distributed, as well as with the amount of resources per growth. This means that all the algorithms are capable of finding new resources; much better than new routes, as shown in the Experiment 1. Note that even the random baseline improves with time. Thanks to its multi-ant penetration




**(a)** RC-ACS




**(b)** SemAnt

**Figure 3.2:** HpH in ResGrow 10k

capabilities, SemAnt is the most efficient in the HpH measure in terms of relative improvement; still remains inferior in absolute terms, however.

### 3.5.3 Experiment 3: NodeMigrate 10k

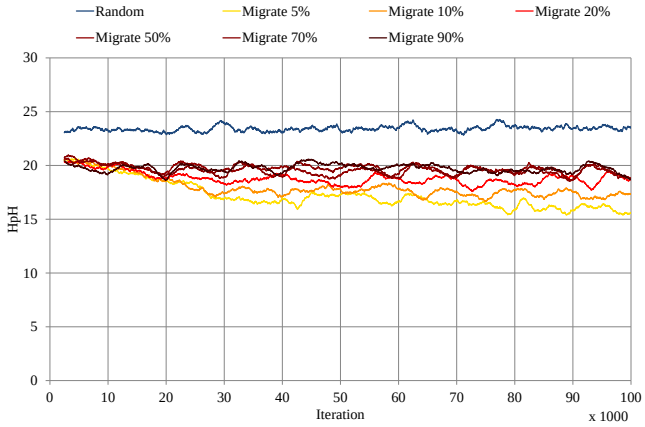
**Table 3.5:** NodeMigrate 10k experiments

| Experiment | Graph size $\|V\|$ | Migration Size $\Delta$ | Migrations total | Migrations per node |
|---|---|---|---|---|
| NodeMigrate 10node | 1024 | 5% (51 nodes) | 459 | 0.45 |
| NodeMigrate 30node | 1024 | 10% (102 nodes) | 918 | 0.90 |
| NodeMigrate 60node | 1024 | 20% (204 nodes) | 1836 | 1.80 |
| NodeMigrate 120node | 1024 | 50% (512 nodes) | 4608 | 4.50 |
| NodeMigrate 240node | 1024 | 70% (716 nodes) | 6444 | 6.30 |
| NodeMigrate 480node | 1024 | 90% (921 nodes) | 8289 | 8.10 |

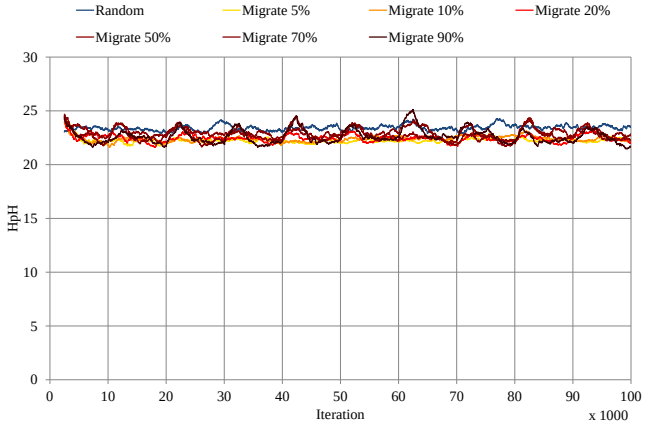
This experiment tries to recreate the conditions of a live and dynamic network of nodes. The real-life analogue of the NodeMigrate can be understood as, for instance, mobile phone based P2P network where users constantly change location. With the change of location comes the process of reattachment to the network, yet the owner's node will provide the same resources as before. Another example of this situation would be the migration of files in a data grid replication service as a consequence of changing QoS characteristics related to the network or certain computing nodes. The question here is whether the evaluated algorithms are capable of erasing the path that stopped providing resources.

The results obtained from this experiment confirmed precisely our forecasts. In this case the task before the algorithms was more difficult because in this situation they had to, not only, find a new source of resources, but also forget the already established paths. The alteration of the network is increasingly impacting until nearly full network reboot in the case of 90% migration.

For multi-ant ACO 3.3b is a perfect depiction of the actual struggle. After the transformation of the network there is an eruption of ants that try to seed new pheromone trails and, naturally, the larger the migration, the more ants appear. In terms of absolute efficiency (Figure 3.3) there is a noticeable drop, but interestingly, it is higher in case of RC-ACS than in case of SemAnt. RC-ACS still obtains better overall results: under 20 HpH, while SemAnt above 20 HpH. Due to the fact that the overall structure was maintained, as there was no link removal or addition as a result of the migration, the Random behaviour was largely unaffected by the entire process.




**(a)** RC-ACS




**(b)** SemAnt

**Figure 3.3:** HpH in NodeMigrate 10k

### 3.5.4 Experiment 4: NetShrink 10k

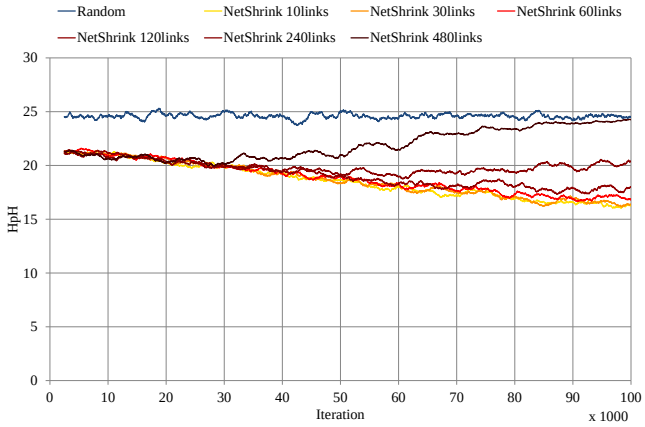
**Table 3.6:** NetShrink 10k experiments

| Experiment | Graph size $\lvert V\rvert$ | Growth Size $\Delta$ | Initial Edge Set Size $\lvert E\rvert_{init}$ | Final Edge Set Size $\lvert E\rvert_{fin}$ | Initial Average Vertex Degree $\overline{deg(v)_{init}}$ | Final Average Vertex Degree $\overline{deg(v)_{fin}}$ |
|---|---|---|---|---|---|---|
| NetShrink10k 10links | 1024 | 10 | 2560 | 2470 | 5.00 | 4.82 |
| NetShrink10k 30links | 1024 | 30 | 2560 | 2290 | 5.00 | 4.47 |
| NetShrink10k 60links | 1024 | 60 | 2560 | 2020 | 5.00 | 3.95 |
| NetShrink10k 120links | 1024 | 120 | 2560 | 1480 | 5.00 | 2.89 |
| NetShrink10k 240links | 1024 | 240 | 2560 | 400 | 5.00 | 0.78 |
| NetShrink10k 480links | 1024 | 480 | 2560 | 0 | 5.00 | 0.00 |

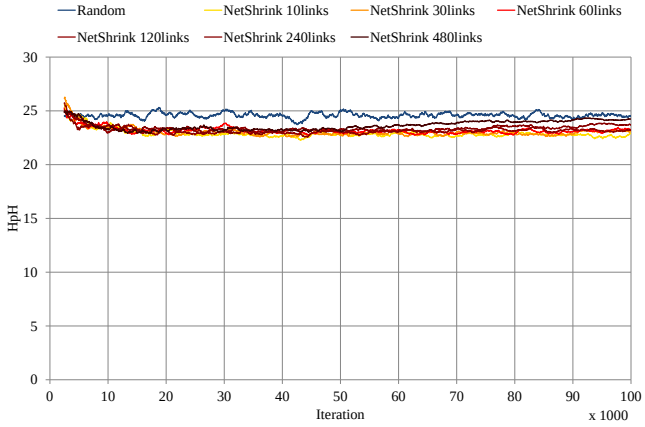
This experiment is the most impacting of the two negative experiments due to the fact that here links, and consequently the resources, disappear irreversibly. In Table 3.6 we placed all the execution variants, note how in the most extreme case, NetShrink10k 480links, there will be no links in the system after the experiments concludes. Bear in mind that we expected an increasingly large portion of queries to become unresolved ($hit = 0$) in this setup which will simply result in lack of data points and somewhat reduced possibilities of comparison of the HpH measure. To address this issue we decided, instead of dropping the unresolved data point, to penalize it with the value 25 HpH. This caused the HpH graph for this particular experiment to be incomparable with the other, form the experiments 1 - 3.

The first observation that stands out is the relatively low impact the amount of links has. Even though it has been stated on several occasions ([18] [31]) it still is interesting to see that reaching $\overline{deg(v)} < 3.00$ does not impair the HpH value much. RC-ACS in the network with $\overline{deg(v)} = 0.78$, which indicates isolated nodes, is better than SemAnt with $\overline{deg(v)} = 4.82$. Note that the final sections of the graph for NetShrink10k 480links, when there are no links at all, still do not converge to the value 25 due to node's capability of answering themselves the asked query.

As shown in Figure 3.4 all the algorithms are mostly unaffected with the exception of the two most impacting variants: NetShrink10k 240links and NetShrink10k 480links; the former however only slightly. As a result of this experiment we may conclude that the design of a lean network structure is important, with shortcuts added only in an educated way, in order to optimize the benefit. Moreover, links can be dropped from the system with ongoing ACO convergence process without much harm.




**(a)** RC-ACS




**(b)** SemAnt

**Figure 3.4:** HpH in NetShrink10k

## 3.6 Conclusions and future works

In the introduction we have stated a question in regard to the applicability of ACO algorithms to the problem of effective resource searching in dynamic grid computing environments and the concept of peer-to-peer connectivity in general. Our main focus was the dynamism of networks and its implications to the convergence process of ACO algorithms.

In the NetGrow experiment our semantic extension of ACS, RC-ACS, behaved the best, being able to use the newly added links most effectively. In the ResGrow experiment RC-ACS maintained the best overall effectiveness. In negative impact experiments (NetShrink, NodeMigrate) RC-ACS lost the least of its initial effectiveness and came out ahead yet again. It is the most telling in case of NodeMigrate, which is, at once, the most natural systems' behaviour and the most negatively impacting.

The main question asked here was whether ACO could be successfully applied in the field of grid computing and P2P in general and the answer is: it can, but only when chosen carefully while following several restrictions, as defined in section 3.3. Moreover, our proposed single ant semantic extension of ACS has exhibited better performance than Semant, the existing multi-ant reference work in the field of ACO semantic search. The future work will consist of further analysis of various dimensions on the robustness and efficiency and the creation of a new ACO algorithm, which would encompass all the desirable aspects to cope with different forms of dynamism.

Chapter 4

# A Diffusion-Based ACO Resource Discovery Framework for Dynamic P2P Networks

Kamil Krynicki Javier Jaen Alejandro Catala

ISSI Research Group, Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

(2013), In 2013 IEEE Congress on Evolutionary Computation, CEC 2013 (pp. 860–867). IEEE. doi:10.1109/CEC.2013.6557658

**Abstract**

The Ant Colony Optimization (ACO) has been a very resourceful metaheuristic over the past decade and it has been successfully used to approximately solve many static NP-Hard problems. There is a limit, however, of its applicability in the field of P2P networks; derived from the fact that such networks have the potential to evolve constantly and at a high pace, rendering the already-established results useless. In this paper we approach the problem by proposing a generic knowledge diffusion mechanism that extends the classical ACO paradigm to better deal with the P2P's dynamic nature. Focusing initially on the appearance of new resources in the network we have shown that it is possible to increase the efficiency of ant routing by a significant margin.

## 4.1 Introduction

Since the introduction of the ACO metaheuristic [36] it has been considered a dynamic algorithm, one that is capable of successfully adapting in case a change in the system occurs [71], very much resembling the actual behavior of ants. One can indicate however a pace of the evolution of the underlying structure at which the classical ACO will not be able to keep up and might achieve highly suboptimal results during the entirety of its execution; we can view it as a certain level of inertia of the system. Naturally, ACO thrives at solving static problems; both: classical, such as the Traveling Salesman Problem [72] and the Assignment Problem [73], but also more uncommon ones, as shown in our several works [35] [66] [34]. Nevertheless, the dynamic issue has remained largely unaddressed. In various papers [12] [29] [15] it has also been shown that ACO-based algorithms are more than an adequate base for a P2P search engine. However, P2P networks are innately and unpredictably dynamic which makes the mentioned problem relevant.

As indicated in [8] the dynamism of a network can impair severally the quality of the results obtained. The issue centers on the fact that ACO algorithms are not capable of redirecting the established routes immediately; a problem to which we refer as slow re-convergence. In a moderate size P2P network it can take up to a few thousand execution iterations in order to accommodate the changes, in this time an additional step of the system's evolution might occur and make a full convergence nearly impossible. In order to counteract this problem we introduce a diffusion layer in P2P networks, within the scope of ACO's pheromone paradigm, designed to propagate the information quicker and more efficiently than a pure ACO would. We named it the Diffusion Model Framework.

Classically the ants were considered the only *intelligent* element in the system, while the rest was purely passive, only capable of receiving, resending and answering queries. In our case we let the nodes themselves take some of the responsibility, by being able to analyze its own content and take decisions, thus redistributing the focus more in the network.

In this paper we introduce a novelty in the field of P2P semantic search, namely our Diffusion Model Framework as an entire platform that permits defining many diffusion algorithms of diverse nature. The main idea behind it is to improve the re-convergence of the system to a degree that allows an efficient P2P search, regardless of the level of the system's dynamism. In this respect, we propose two diffusion strategies: in-width and in-depth diffusion, both built within the mentioned framework and we perform an experimental study of these strategies to evaluate their effectiveness. We chose the Ant Colony System (ACS) [36] as our ACO implementation, seeing that it is considered the most classical of all, with one crucial modification called the Routing Concept (RC), which was proposed, in different aspects, by several authors [43] [8]. The results obtained by the above

studies show that RC can significantly improve query routing in ACO-based P2P systems [15].

The experimental data obtained demonstrate that, in terms of re-convergence speed, the in-width strategy has proven to be a feasible solution in systems that have a downtime period. Moreover, in the case of the in-depth diffusion, the overall strain on the system is minimal and the effectiveness of the search process is improved up to 30% with respect to a non-diffusion-based search process.

The paper is structured as follows. In section 4.2 we summarize the related work in the field of P2P and ACO with respect to information diffusion. In section III we introduce some of the ACS basis and expand on the subject of the Routing Concept. In section IV we describe in detail the Diffusion Model Framework and the concrete implementations we chose to experiment with. In section 0 we present our experimental setup, formalize all the concepts and propose a quality measure; section VI shows the results obtained and in section 0 we conclude and summarize our main findings.

## 4.2 Related Work

To the best of our knowledge the subject, as we define it, has not been explored. There are several works that focus on the diffusion of the resources in the system, but in all the cases it is solely a study of how it occurs naturally in various systems [74] [75], regardless whether they include ACO concepts or not. Never is it an induced diffusion, with the objective of improving a specific quality measure nor is it an extension of the ACO paradigm.

The most similar subject is the publish/feed paradigm where a particular node is the generator of content while a certain subset of the network consumes the resources [76]. However the main focus in those works is the efficiency and scalability, not an ACO-based method nor a resource-query mechanism. We might qualify it as a deterministic and complete point-to-point resource propagation, in contrast to our dynamic resource Diffusion Model Framework.

## 4.3 Formal Basis

### 4.3.1 ACS

Ant Colony System [29] is one of the most popular implementations of the ACO metaheuristic. It is an extension and improvement over the Ant System (AS) [33] that we chose as our main strategy for P2P search. The reasons for this choice are twofold: firstly, we wanted to experiment with a fairly well known and

pure ACO implementation, secondly, as shown in [8], many of the common ACO implementations are not suitable for neither the query-resource paradigm, the P2P environment nor the dynamic system setup. ACS performed well in all those aspects.

We choose precisely the same mathematical basis of ACS as were presented in [8] [29]. Having this ant strategy in place, we will formalize the semantic search in P2P in a simple manner. Firstly we establish a network of homogeneous nodes. Each of them may be in possession of a certain set of resources of any nature, but also each is capable of generating a query – i.e. require resources of a given taxonomical kind. An ant corresponding to the query is created and routed according to the ACO algorithm of our choosing collecting resources labeled with the indicated taxonomical entity from the nodes it visits. Once the algorithm is complete the ant evaluates its findings and returns to its emitting node, performing pheromone-related tasks along the way.

### 4.3.2 Routing Concept

In most ACO algorithms there is just a single value of pheromone per outgoing link in each node. The Routing Concept, however, introduces an additional dimension (a layer), giving each link a full table of pheromones, each corresponding to a class of queries that might appear in a given node. Therefore a single ant, at its creation, is given a Routing Concept value and only manages pheromone that has been deposited in that particular layer of the system. Hence, it is closely related to the Taxonomy Based Routing, presented in [15] [77].

The technical basis is as follows (see [8] for a detailed formal description). Every node $N$ keeps a 2–dimensional matrix $\Omega_N : L_N \times C_N$, with real, positive values, where $L_N$ is the space of outgoing links from the node $N$ and $C_N$ is the space of Routing Concepts maintained by this particular node $N$. This matrix is referred to as routing table, or routing matrix. The l–th, c–th element of $\Omega_N$ corresponds to the pheromone value of the l–th outgoing link for the c–th routing concept, which can be written as $\Omega_N(l, c) = \tau$.

## 4.4 Diffusion Model Framework

The diffusion is an additional source of pheromone trail creation in the system, which is managed by the nodes themselves and can be performed at will, usually as a response to a node's internal event, such as resource addition. It is achieved through the introduction of a new type of ant that exists alongside the classical Forward Ants (FA) and Backward Ants (BA), namely a Diffusion Ant (DA). Every node of the system is given the ability to analyze its own content's evolution and generate a DA at any moment it sees fit. It can also be externally forced to do so.

**Table 4.1:** Global Diffusion Parameters

| Name (symbol) | Comment | Constraint |
|---|---|---|
| Diffusion Ant Max TTL ($TTL_{max}$) | The maximum distance from the origin node the DA can reach | $TTL_{max} > 0$ |
| Diffusion Delta ($D_\delta$) | The effect of a DA on the node's pheromone | $D_\delta \in [0, 1]$ |
| Diffusion Random Spread Chance ($D_\rho$) | Chance to continue spreading pheromone to uninitialized regions of the system | $D_\delta \in [0, 1]$ |
| Diffusion Minimum Link Spread ($MIN_\rho$) | The minimum amount of links to spread to, at each step | $MIN_\rho \in \{-1, 0, 1, \dots\}$, -1 means all available |
| Diffusion Maximum Link Spread ($MAX_\rho$) | The maximum amount of links to spread to, at each step | $MAX_\rho \in \{-1, 0, 1, \dots\}$, -1 means all available |

Each DA carries two values: Diffusion Power and Diffusion Routing Concept. See Table 4.2 for details on both of these values; they are established by the emitting node at DA's creation.

The DA behavior is governed by the below algorithm.

**Algorithm 6:** DA behavior

1: **assume** DA arrives in node N, DA's Time To Live is $DA_{TTL}$
2: **if** $\Omega_N$ matrix contains pheromone for the Diffusion Routing Concept carried by DA **then**
3: go to 10

**Table 4.2:** Local Diffusion Parameters

| Name (symbol) | Comment | Constraint |
|---|---|---|
| Diffusion Ant Time To Live ($DA_{TTL}$) | The current time to live of a given DA, decreases as an ant continues | $DA_{TTL} \in [0, TTL_{max}]$ |
| Diffusion Power ($DA_p$) | The scaling parameter of a given DA, affects the pheromone deposition | $DA_p > 0$ |
| Diffusion Routing Concept ($DA_c$) | The Diffusion objective of a given DA, affects the pheromone deposition | One of the Routing Concepts present in the system |

4: **else**
5: generate a random real value $s \in [0, 1]$
6: **if** $s < D_p$ **then** For $D_p$ see Table 4.1
7: go to 10
8: **else**
9: end algorithm
10: **end if**
11: **end if**
12: Perform Diffusion Pheromone Update (see next algorithm) in $N$, unless $N$ is the emitting node of DA
13: Let $aLinks$ be the set of all the outgoing links from $N$, to nodes not visited previously by the DA nor its clones; let $gLinks$ be a random subset of the $aLinks$ set, of size $m$, where $m$ is a random integer value, and $m \in [MIN_\rho, MAX_\rho]$
14: Decrement $DA_{TTL}$ and send a clone of DA to every link in $gLinks$.

**Algorithm 7:** Diffusion Pheromone Update in Node N

1: **let**

$$\delta\tau(N, N_{in}) = DA_\rho \times \frac{DA_{TTL} + 1}{TTL_{max} + 1} \tag{4.1}$$

where $N$ is the current node, $N_{in}$ is the node from which DA arrived to $N$ and $DA_p$ is the diffusion power.
2: **if** $\Omega_N$ matrix contains pheromone for the Diffusion Routing Concept carried by DA **then**
3: go to 10
4: **else**
5: Add the $DA_c$ to $\Omega_N$
6: **end if**
7: Set the $DA_c$ pheromone value, for the link from which DA arrived:

$$\tau(N, N_{in}) \leftarrow \delta\tau(N, N_{in}) \tag{4.2}$$

8: In $N$ set the pheromone value corresponding to $DA_c$ for the link from which DA arrived in $N$:

$$\tau(N, N_{in}) \leftarrow D_\delta \times (\delta\tau(N, N_{in}) + p_{init}) + (1 - D_\delta) \times \tau(N, N_{in}) \tag{4.3}$$

where $p_{init}$ is the default value of the pheromone and $D_\delta$ is the Diffusion Delta

In Table 4.2 we summarize the parameters that define the character of the diffusion in question; they remain unchanged during the entire execution of the algorithm. In Table 4.1 we summarize parameters attached to a particular act of diffusion, all of which may be variable during a single execution. This choice of parameters allows us to create an abundant amount of types of diffusion. In order to make the analysis more focused we decided to extract two diffusion families with a manageable set of independent variables to examine.

**Table 4.3:** In-width diffusion parameters

| Parameter Name (symbol) | Value |
|---|---|
| Diffusion Ant Max TTL ($TTL_{max}$) | Subject to test: 1, 2, 3, 4 |
| Diffusion Delta ($D_\delta$) | 0.1 |
| Diffusion Random Spread Chance ($D_\rho$) | 1 |
| Diffusion Minimum Link Spread ($MIN_\rho$) | -1 |
| Diffusion Maximum Link Spread ($MAX_\rho$) | -1 |

**Table 4.4:** In-depth diffusion parameters

| Parameter Name (symbol) | Value |
|---|---|
| Diffusion Ant Max TTL ($TTL_{max}$) | Subject to test: 5, 10, 15, 20 |
| Diffusion Delta ($D_\delta$) | 0.1 |
| Diffusion Random Spread Chance ($D_\rho$) | 1 |
| Diffusion Minimum Link Spread ($MIN_\rho$) | 1 |
| Diffusion Maximum Link Spread ($MAX_\rho$) | 1 |

### 4.4.1 In-width diffusion

The most straightforward diffusion is a k-flood diffusion. In simple words it can be understood as unconditionally notifying all the nodes up to the distance k from the initiating node [41]. The k is simply the $TTL_{max}$ in our setup, and therefore the, so called, diffusion depth, would be $TTL_{max}$ as well. Table 4.3 presents the values of the diffusion parameters that are necessary to achieve a proper k-flood. See Figure 4.1 for a graphical representation of the in-width diffusion and how it affects its neighbors. It creates an area in the P2P network with increasingly attractive force to the emitter, somewhat similar to a gravity field.

### 4.4.2 In-depth diffusion

The opposite approach is the in-depth diffusion. Here we do not permit the cloning of the DA, by establishing $MIN_\rho = MAX_\rho = 1$, which, in combination with the prohibition of the reentry, creates long chains of diffused pheromone, rather than areas. The effect DAs have on the system is much lower and more distributed in time, but it is not as strong as the previous method, due to the fact that there might be an ant that misses the chain. As it can be seen in Figure 4.2 Ant B behaves identically, regardless of the state of the diffusion. In spite of the given counter-example we suspect that this lightweight mode of diffusion might prove better in terms of the ratio: nodes affected by diffusion to ants attracted; especially so if one chooses to release several independent DA instead of one. Consult Table 4.4 for parameter values necessary to achieve in-depth diffusion in our model. Again, the $TTL_{max}$ will be referred to as the Diffusion Depth later on.

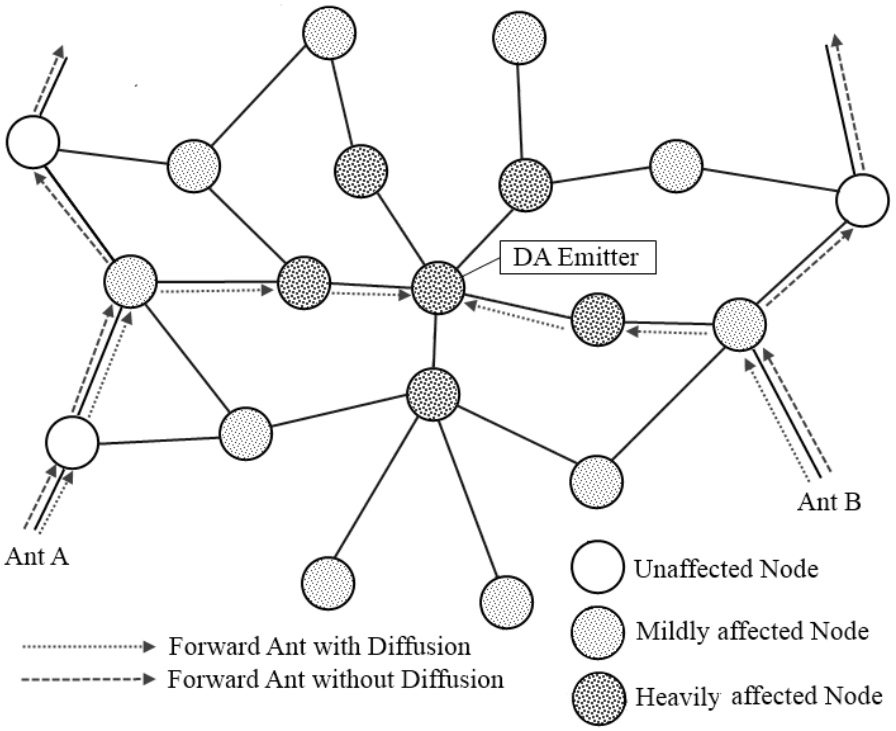



**Figure 4.1:** In-width diffusion, $DA_{TTL} = 2$

## 4.5 Experimental Setup

In this section we provide an exhaustive overview of all the decisions and assumptions we took in order to formalize the experiments.

### 4.5.1 Resource distribution and labeling

As in [15], the ACM Computing Classification System [53] will be the taxonomical vocabulary used. Every resource in the network 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 depicted 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 $t$.

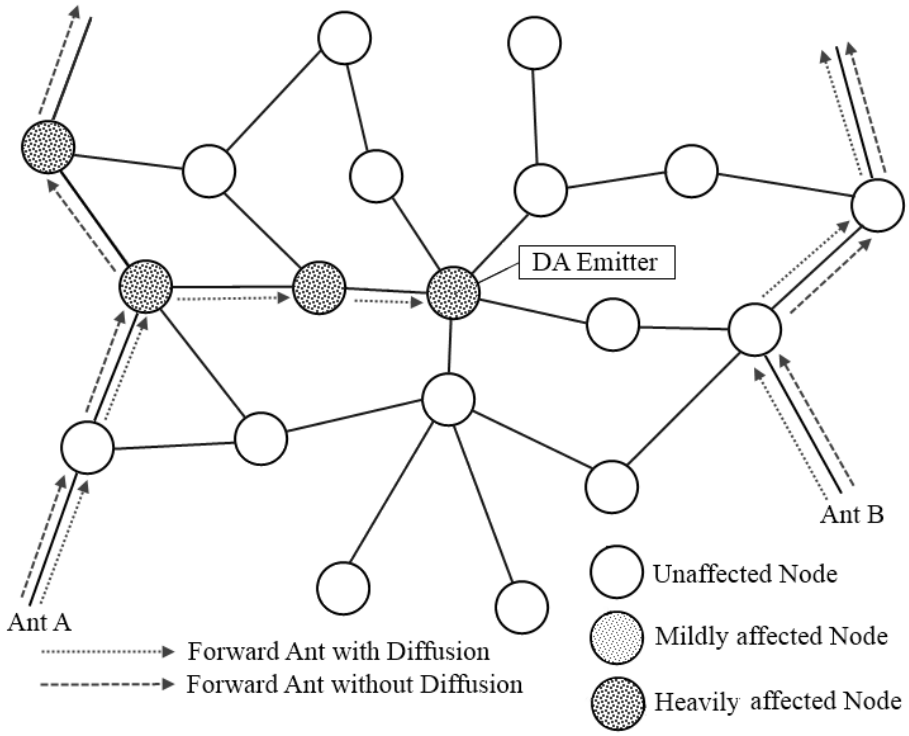



**Figure 4.2:** In-depth diffusion, $DA_{TTL} > 3$

The distribution of resources within the network follows strictly the approach by the test setup [15]. 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 resource 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, reflecting the fact that people have specific interests and hobbies [54].

**Table 4.5:** P2p worlds present in the experiments

| World edge size | World total size | Initial resources per node | Nodes affected |
|---|---|---|---|
| $N$ | $N \times N$ | - | $M$ |
| 32 | 1024 | 30 | 48 |
| 40 | 1600 | 30 | 75 |
| 48 | 2304 | 30 | 108 |
| 56 | 3136 | 30 | 147 |
| 64 | 4096 | 30 | 192 |

### 4.5.2 Query and query resolution

Every non-diffusion ant is connected to a query q and 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. The resolution of a query $q(t)$ in a node $N_i$ consists of finding the resources that have been labeled with $t$, that is, all the resources $\{r|\exists r(N_i, t)\}$.

### 4.5.3 P2P network setup

Every P2P network in our tests is a fully connected toroidal world of $N \times N$ nodes, with the resource distribution that obeys the rules explained in previous sections. See Table 4.5 for all the world variants. In order to follow a coherent approach with our previous works we chose to present a minor alteration to the pure toroid topology. Namely: every node will additionally have one long distance connection between itself and a node randomly selected of all those not directly neighboring. The probability of linking two nodes is inversely proportional to the distance between them.

### 4.5.4 Quality measure

As many works before [12] [15] [31] we decided to adopt the Hop per Hit (HpH, dimensionless) measure. It can be read as: how many steps a single ant must take to obtain a single resource during a given iteration. The graphical analysis of many plotted evolutions of HpH in function of iterations led us to confirm that it gets fairly stable after a certain point, called the convergence point [8]. Therefore we chose our final measure to be the arithmetical average of exactly 1000 last values of HpH in a given experiment; we expect it to be characterized by low standard deviation.

Additionally we decided to distinguish between two separate interpretations of the data. It is obvious that every act of diffusion consumes a portion of system resources, whether it is bandwidth, processing time or any other. One might argue, however, that the entire diffusion process can be executed when the system is in its downtime, when there is no or very little real usage. A good example is forcing all the diffusion at night, reducing the probability of a conflict with a user-released ant to a minimum. Hence, in that case the diffusion would not interfere with system's functionality and it could be omitted in the calculations. Therefore the two interpretations are:

**non-strained results** where the resources consumed by diffusion ants are not taken in consideration, and are treated as neutral to the system. The non-strained average of HpH Diffusion Depth $d$ will be symbolized by $\tilde{\mu_d}$

**strained results** where every diffusion ant is treated as inseparable from the system and included in the results. The strained average of HpH of the Diffusion Depth $d$ will be symbolized by $\mu_d$

A way to include the diffusion strain in the calculations is based around the fact that it does not evolve with the system. We can infer from this that the density of resources consumed by diffusion in the function of iterations remains constant as well. Consequently we sum the total diffusion strain over all the iterations, divide it over the number of iterations and add the resulting value to all the HpH results obtained the classical way, might it be a single iteration or an average of a given period.

### 4.5.5 Experimental methodology

In its most general form the execution of every experiment will be as follows:

**Algorithm 8:** Single Experiment in world $N \times N$

1: **for** $i_{max}$ **do**
2: Choose a random node $N_1$ of all available in $N \times N$ and select a random resource $r$ of its repository.
3: Choose a random node $N_2$ of all available in $N \times N$ and generate in it an Ant whose objective is the discovery of the resource $r$.
4: After its completion the query will report the amount of resources obtained and the amount of steps taken. Data will be used to calculate the HpH measure for that particular iteration.
5: Every $nMod$ iterations initiate the Resource Distribution procedure (see Algorithm 9).
6: **end for**

**Algorithm 9:** Resource Distribution in world $N \times N$

1: Choose a set $M$ of random nodes of available in $N \times N$.
2: **for all** $m \in M$ **do**
3: select one random resource from all the ones it possesses and generate a set of 10 resources within its category.
4: Add all generated resources to the node $m$; the process referred to as node enrichment.
5: Force the node $m$ to release a diffusion ant into the system that informs about its newly acquired content.
6: **end for**

See Table 4.5 for sizes of $M$; it was adjusted to the scale linearly with the total size of the world. The amount of iterations per execution was chosen to be $i_{max} = 10^5$. This particular value permits us to see the full evolution of the system while still presenting us with a manageable amount of data. The period between consecutive resource distributions is set at $nMod = 10^3$, therefore the resource distribution will be executed 99 times within a single experiment. The two above statements are concluded from [12].

For each combination of any set of the parameters chosen to test, we will execute the Single Experiment algorithm three times and, after confirming consistency of the results, proceed to work with the average of the three for further analysis.

## 4.6 Experimental Study

We decided to formulate the statistical analysis in the following way. Our null hypothesis $H_0$ is such that the averages of HpH, grouped by a particular Diffusion Depth, in all experiments are equal: $H_0$:$\mu_1^* = \mu_2^* = \cdots = \mu_k^*$, where $\mu_d^*$ is the normalized average of HpH values within the Diffusion Depth $d$. The normalization is with regard to the Diffusion Depth $d = 0$; a way to express the relative improvement. The alternative hypothesis $H_1$ assumes differences between the means: $H_1$:$\mu_1^* \neq \mu_2^* \neq \cdots \neq \mu_k^*$. The significance levels we aim for are at $\alpha_1 = 0.1$ and $\alpha_2 = 0.05$. Before each test we apply the Shapiro-Wilk test to establish the fact of data normality; it is the most adequate test for such an amount of data [78].

A good omnibus test, when the normality of the data cannot be guaranteed, is the Kruskal-Wallis test [79] which even in cases of near-normality can prove to be more sensitive than ANOVA. In the event of detecting significant differences between diffusion levels we will perform the Dunn's Pairwise Comparison method [80], the results of which should provide us with knowledge on the relations between different Diffusion Depths d. We also must take into account the amount of data groups present in the test by applying the necessary Bonferroni correction. To achieve the agreed significance of $\alpha_1 = 0.1$ and $\alpha_2 = 0.05$ we must confirm results

only at the significance $\alpha_{b1} = 0.1$ and $\alpha_{b2} = 0.005$ respectively, which is a very strong requirement.

All the experiments will be run on our proprietary software, written by us in the Java programming language. It was designed specifically for the simulation of the accelerated evolution of ACO algorithms in various settings. The machine used in testing is a PC Intel Pentium 4 630 at 3.00GHz with 4 GB of RAM on a 32bit Windows 7. Typically a single experiment might take up to 15 minutes of processing time.

### 4.6.1 In-width diffusion experiment

With the configuration presented in Table 4.3 we attempted the recreation of the most basic, flood-like behavior in our system. The independent variable d is the TTL of Diffusion Ants, taking values *d / 0*, *d / 1*, *d / 2*, *d / 3* and *d / 4*. Each test is repeated three times and the data will be presented as strained and non-strained.

The full dataset of the results is presented in Table 4.6. The normality was not achieved here due to a priori selection of the depth values; the Shapiro-Wilk test does not allow the assumption of normality with high certainty, see Table 4.7.

Kruskal-Wallis test (Table 4.8) has proven beyond any doubt that groups formed by different diffusion settings are significantly different and are disjointed populations of data points with respect to the HpH measure. The most interesting conclusions stem out of Table 4.9 and Table 4.10. For instance, in Table 4.9 we see that the difference between non-str. *d / 2* and *d / 0* is of 12.100 ranks; from the corresponding field in the Table 4.10 we conclude that it is significant at $\alpha = 0.01$. In case of the non-strain values we confirm that even an algorithm as crude as k–flood is significantly better than the lack of diffusion for depths of diffusion equal or larger than *d / 2*, which are *d / 2*, *d / 3* and *d / 4*. The strained data are less promising as there is no statistically significant benefit from the diffusion and therefore more elaborate algorithms are in place, if one cannot pinpoint system's downtimes to perform the diffusion and treat it as non-straining. We feel we need to point out however that the diffusion of size *d / 2* was very nearly accepted as better than no diffusion, with $p - value = 0.036$. A simple look at the graphical representation of the strained results (Fig. 4.3a) suggests that around the diffusion *d / 2* there is a local minimum of net benefit at about 15% HpH. In case of the non-strained results (Fig. 4.3b) no such minimum is observed and we assume that it is located outside the examined scope; the best value is 23% lower than the diffusion-less system.

**Table 4.6:** In-width experiment, Full results

| **N** | **d** | **Size** | **HpH strain** | **HpH strain** | **HpH non-strain** | **HpH non-strain** |
|---|---|---|---|---|---|---|
| - | - | - | $\mu_d$ | $\mu_d^*$ | $\tilde{\mu_d}$ | $\tilde{\mu_d}^*$ |
| 1024 | 0 | 0 | 20.52 | 1.00 | 20.52 | 1.00 |
| 1024 | 1 | 4 | 17.74 | 0.87 | 17.58 | 0.86 |
| 1024 | 2 | 12 | 17.98 | 0.88 | 17.61 | 0.86 |
| 1024 | 3 | 24 | 19.07 | 0.93 | 17.30 | 0.84 |
| 1024 | 4 | 40 | 21.19 | 1.04 | 16.34 | 0.80 |
| 1600 | 0 | 0 | 21.78 | 1.00 | 21.78 | 1.00 |
| 1600 | 1 | 4 | 19.63 | 0.90 | 19.34 | 0.89 |
| 1600 | 2 | 12 | 18.81 | 0.86 | 17.89 | 0.82 |
| 1600 | 3 | 24 | 19.68 | 0.90 | 16.97 | 0.78 |
| 1600 | 4 | 40 | 26.64 | 1.22 | 18.22 | 0.84 |
| 2304 | 0 | 0 | 21.53 | 1.00 | 21.53 | 1.00 |
| 2304 | 1 | 4 | 20.38 | 0.95 | 19.96 | 0.93 |
| 2304 | 2 | 12 | 19.75 | 0.92 | 18.38 | 0.85 |
| 2304 | 3 | 24 | 22.53 | 1.05 | 18.32 | 0.85 |
| 2304 | 4 | 40 | 27.92 | 1.30 | 16.77 | 0.78 |
| 3136 | 0 | 0 | 21.56 | 1.00 | 21.56 | 1.00 |
| 3136 | 1 | 4 | 21.10 | 0.98 | 20.49 | 0.95 |
| 3136 | 2 | 12 | 19.37 | 0.90 | 17.59 | 0.82 |
| 3136 | 3 | 24 | 22.92 | 1.06 | 17.43 | 0.81 |
| 3136 | 4 | 40 | 33.05 | 1.53 | 17.31 | 0.80 |
| 4096 | 0 | 0 | 22.41 | 1.00 | 22.41 | 1.00 |
| 4096 | 1 | 4 | 21.30 | 0.95 | 20.51 | 0.92 |
| 4096 | 2 | 12 | 20.19 | 0.90 | 17.84 | 0.80 |
| 4096 | 3 | 24 | 25.95 | 1.16 | 18.38 | 0.82 |
| 4096 | 4 | 40 | 37.52 | 1.67 | 17.21 | 0.77 |

**Table 4.7:** In-width experiment, Normality tests

| **Shapiro-Wilk** | **Statistic** | **df** | **p-value** |
|---|---|---|---|
| In-Width strain | 0.749 | 25 | 0.000 |
| In-Width non-strain | 0.874 | 25 | 0.005 |

**Table 4.8:** In-width experiment, Kruskal Wallis Test

| Kruskal-Wallis test | In-Width non-strain | In-Width strain |
|---|---|---|
| K (Observed value) | 18.604 | 16.896 |
| K (Critical value) | 7.779 | 7.779 |
| Df | 4 | 4 |
| p-value (Two-tailed) | 0.001 | 0.002 |
| Alpha | 0.01 | 0.01 |

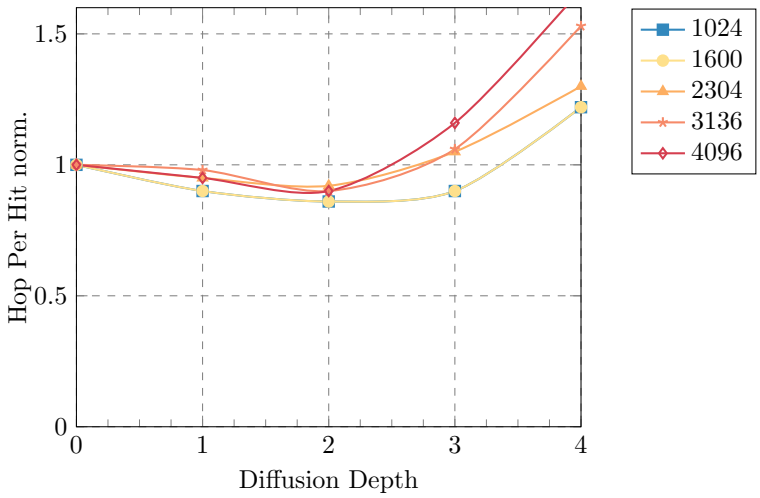
**Table 4.9:** In-width experiment, Dunn's comparison, Ranks

| | | d \| 0 | d \| 1 | d \| 2 | d \| 3 | d \| 4 |
|---|---|---|---|---|---|---|
| non strain | d \| 0 | 0 | 4.600 | 12.100 | 13.800 | 17.500 |
| | d \| 1 | -4.600 | 0 | 7.500 | 9.200 | 12.900 |
| | d \| 2 | -12.100 | -7.500 | 0 | 1.700 | 5.400 |
| | d \| 3 | -13.800 | -9.200 | -1.700 | 0 | 3.700 |
| | d \| 4 | -17.500 | -12.900 | -5.400 | -3.700 | 0 |
| strain | d \| 0 | 0 | 6.400 | 9.900 | 0.200 | -6.900 |
| | d \| 1 | -6.400 | 0 | 3.500 | -6.200 | -13.300 |
| | d \| 2 | -9.900 | -3.500 | 0 | -9.700 | -16.800 |
| | d \| 3 | -0.200 | 6.200 | 9.700 | 0 | -7.100 |
| | d \| 4 | 6.900 | 13.300 | 16.800 | 7.100 | 0 |

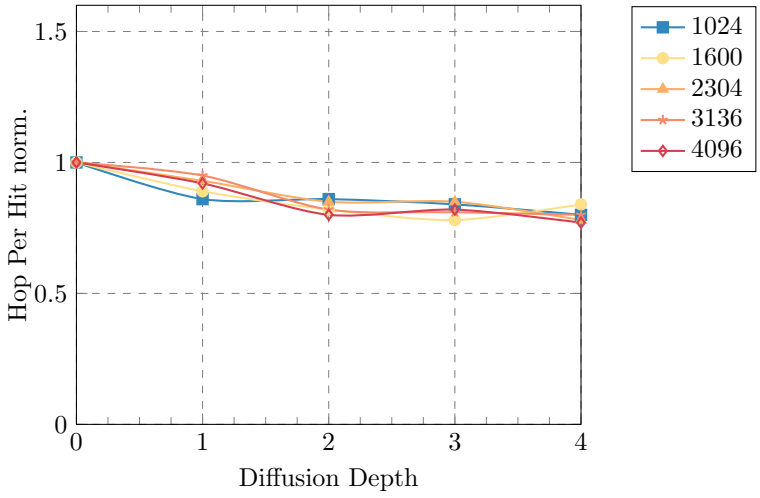
**Table 4.10:** In-width experiment, Dunn's comparison, p-values

| | | d \| 0 | d \| 1 | d \| 2 | d \| 3 | d \| 4 |
|---|---|---|---|---|---|---|
| non strain | d \| 0 | 1 | 0.330 | **0.010** [a] | **0.003** [ab] | **0.000** [ab] |
| | d \| 1 | 0.330 | 1 | 0.092 | 0.039 | **0.004** [ab] |
| | d \| 2 | **0.010** [a] | 0.092 | 1 | 0.702 | 0.225 |
| | d \| 3 | **0.003** [ab] | 0.039 | 0.702 | 1 | 0.406 |
| | d \| 4 | **0.000** [ab] | **0.004** [ab] | 0.225 | 0.406 | 1 |
| strain | d \| 0 | 1 | 0.175 | 0.036 | 0.966 | 0.144 |
| | d \| 1 | 0.175 | 1 | 0.432 | 0.164 | **0.003** [ab] |
| | d \| 2 | 0.036 | 0.432 | 1 | 0.029 | **0.000** [ab] |
| | d \| 3 | 0.966 | 0.164 | 0.029 | 1 | 0.111 |
| | d \| 4 | 0.144 | **0.003** [ab] | **0.000** [ab] | 0.111 | 1 |

Bonferroni corrected significance levels: [a] $\alpha_{b1} = 0.01$ [b] $\alpha_{b2} = 0.005$




**(a)** strain




**(b)** non-strain

**Figure 4.3:** In-width diffusion

### 4.6.2 In-depth diffusion experiment

The in-depth diffusion experiment was executed according to the parameters established in Table 4.4. We manipulated the degree of diffusion by changing the TTL of Diffusion Ants over 5 values: *d | 0*, *d | 5*, *d | 10*, *d | 15* and *d | 20*, it will be our independent variable *d*. Each test is repeated three times and the data will be presented as strained and non-strained.

Similarly to the previous experiment the data cannot be considered normal, as it is shown in TABLE 4.12. We therefore must again run the statistical analysis with the Kruskal-Wallis test. With near perfect certainty (>99.99%, see Table 4.13) it was confirmed that there are different populations of points within our results in terms of HpH measure. In case of In-Depth diffusion the strain put on the results is marginally low, from Table 4.11 we conclude that even at the *d | 20* Diffusion Depth the difference between strain and non-strain results is below 0.8 HpH. It comes as no surprise that strained and non-strained results are very much alike, see Table 4.14 and Table 4.15 for Dunn's pairwise comparison. We can conclude that the Diffusion Depths of *d | 15* and higher are significantly better than no diffusion at all. It is also worth pointing out that the minute Diffusion Depth of *d | 10* may be nearly considered appropriate as well. In these cases the improvement reaches quite an impressive value of 32% in the non-strained variant and up to 29% in the strained one. Both of these can be seen on Fig. 4.4b and Fig. 4.4a respectively. No clear local minimum was detected so pushing the Diffusion Depth to even higher values may be justified.

## 4.7 Conclusions/Future Work

In this work we have shown clearly that the introduction of our idea of the Diffusion Model Framework, with the objective of improving the re-convergence speed in a P2P environment managed by ACO algorithm, is beneficial. Even the crudest variant provided a certain net improvement to the system, in terms of HpH, and was helpful in combating the slow re-convergence.

The in-width version has proven to be a feasible solution in systems that have a downtime period. As mentioned before, those including day/night cycles, weekday/weekend cycles, etc. In this situation one might disregard the strain that the Diffusion Ant has on the system and successfully apply the in-width diffusion depth *d | 3* or *d | 4*. To some extent the in-width diffusion depth *d | 2* is nearly applicable to the strained system as well.

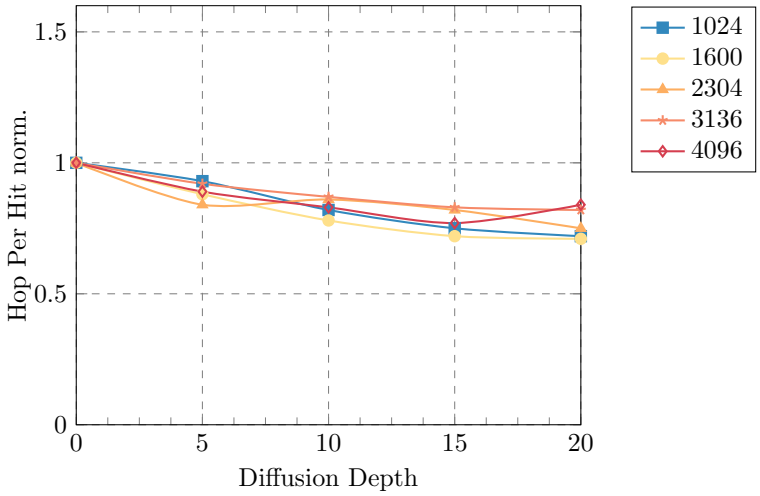
In the case of our in-depth strategy the results were much stronger. Not only the overall strain on the system was minimal, but also the improvement reached values as high as 30%. Regardless of whether we used strained or non-strained results we come to the conclusion that in-depth diffusion depths *d | 15* and *d |*

**Table 4.11:** In-depth experiment, Full results

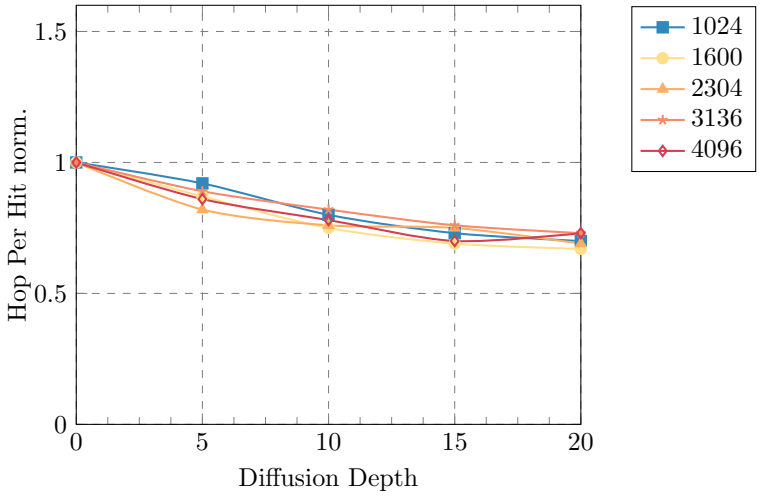
| **N** | **d** | **Size** | **HpH strain** | **HpH strain** | **HpH non-strain** | **HpH non-strain** |
|---|---|---|---|---|---|---|
| - | - | - | $\mu_d$ | $\mu_d^*$ | $\tilde{\mu_d}$ | $\tilde{\mu_d}^*$ |
| 1024 | 0 | 0 | 19.46 | 1.00 | 19.46 | 1.00 |
| 1024 | 1 | 4 | 18.02 | 0.93 | 17.85 | 0.92 |
| 1024 | 2 | 12 | 15.89 | 0.82 | 15.59 | 0.80 |
| 1024 | 3 | 24 | 14.57 | 0.75 | 14.16 | 0.73 |
| 1024 | 4 | 40 | 14.04 | 0.72 | 13.52 | 0.70 |
| 1600 | 0 | 0 | 22.05 | 1.00 | 22.05 | 1.00 |
| 1600 | 1 | 4 | 19.42 | 0.88 | 19.13 | 0.87 |
| 1600 | 2 | 12 | 17.21 | 0.78 | 16.71 | 0.75 |
| 1600 | 3 | 24 | 15.82 | 0.72 | 15.14 | 0.69 |
| 1600 | 4 | 40 | 15.55 | 0.71 | 14.68 | 0.67 |
| 2304 | 0 | 0 | 21.99 | 1.00 | 21.99 | 1.00 |
| 2304 | 1 | 4 | 18.38 | 0.84 | 17.99 | 0.82 |
| 2304 | 2 | 12 | 18.91 | 0.86 | 16.69 | 0.76 |
| 2304 | 3 | 24 | 18.12 | 0.82 | 16.33 | 0.75 |
| 2304 | 4 | 40 | 16.50 | 0.75 | 15.20 | 0.69 |
| 3136 | 0 | 0 | 22.02 | 1.00 | 22.02 | 1.00 |
| 3136 | 1 | 4 | 20.18 | 0.92 | 19.60 | 0.89 |
| 3136 | 2 | 12 | 19.12 | 0.87 | 18.05 | 0.82 |
| 3136 | 3 | 24 | 18.29 | 0.83 | 16.80 | 0.76 |
| 3136 | 4 | 40 | 17.99 | 0.82 | 16.13 | 0.73 |
| 4096 | 0 | 0 | 21.91 | 1.00 | 21.91 | 1.00 |
| 4096 | 1 | 4 | 19.45 | 0.89 | 18.73 | 0.86 |
| 4096 | 2 | 12 | 18.21 | 0.83 | 16.90 | 0.78 |
| 4096 | 3 | 24 | 16.92 | 0.77 | 15.17 | 0.70 |
| 4096 | 4 | 40 | 18.27 | 0.84 | 15.86 | 0.73 |

**Table 4.12:** In-depth experiment, Normality tests

| **Shapiro-Wilk** | **Statistic** | **df** | **p-value** |
|---|---|---|---|
| In-depth strain | 0.926 | 25 | 0.069 |
| In-depth non-strain | 0.885 | 25 | 0.009 |




**(a)** strain




**(b)** non-strain

**Figure 4.4:** In-depth diffusion

**Table 4.13:** In-depth experiment, Kruskal Wallis Test

| **Kruskal-Wallis test** | **In-depth non-strain** | **In-depth strain** |
|---|---|---|
| K (Observed value) | 20.677 | 18.16 |
| K (Critical value) | 7.779 | 7.779 |
| Df | 4 | 4 |
| p-value (Two-tailed) | 0.000 | 0.001 |
| Alpha | 0.01 | 0.01 |

**Table 4.14:** In-depth experiment, Dunn's comparison, Ranks

| | | **d \| 0** | **d \| 1** | **d \| 2** | **d \| 3** | **d \| 4** |
|---|---|---|---|---|---|---|
| **non strain** | **d \| 0** | 0 | 4.600 | 9.800 | 15.500 | 18.100 |
| | **d \| 1** | -4.600 | 0 | 5.200 | 10.900 | 13.500 |
| | **d \| 2** | -9.800 | -5.200 | 0 | 5.700 | 8.300 |
| | **d \| 3** | -15.500 | -10.900 | -5.700 | 0 | 2.600 |
| | **d \| 4** | -18.100 | -13.500 | -8.300 | -2.600 | 0 |
| **strain** | **d \| 0** | 0 | 5.000 | 10.800 | 15.800 | 16.400 |
| | **d \| 1** | -5.000 | 0 | 5.800 | 10.800 | 11.400 |
| | **d \| 2** | -10.800 | -5.800 | 0 | 5.000 | 5.600 |
| | **d \| 3** | -15.800 | -10.800 | -5.000 | 0 | 0.600 |
| | **d \| 4** | -16.400 | -11.400 | -5.600 | -0.600 | 0 |

**Table 4.15:** In-depth experiment, Dunn's comparison, p-values

| | | **d \| 0** | **d \| 1** | **d \| 2** | **d \| 3** | **d \| 4** |
|---|---|---|---|---|---|---|
| **non strain** | **d \| 0** | 1 | 0.330 | 0.038 | **0.001**$^{ab}$ | **0.000**$^{ab}$ |
| | **d \| 1** | 0.330 | 1 | 0.243 | 0.014 | 0.002$^{ab}$ |
| | **d \| 2** | 0.038 | 0.243 | 1 | 0.201 | 0.062 |
| | **d \| 3** | **0.001** $^{ab}$ | 0.014 | 0.201 | 1 | 0.559 |
| | **d \| 4** | **0.000** $^{ab}$ | **0.002** $^{ab}$ | 0.062 | 0.559 | 1 |
| **strain** | **d \| 0** | 1 | 0.290 | 0.022 | **0.001** $^{ab}$ | **0.001** $^{ab}$ |
| | **d \| 1** | 0.290 | 1 | 0.193 | 0.015 | **0.010** $^{a}$ |
| | **d \| 2** | 0.022 | 0.193 | 1 | 0.262 | 0.209 |
| | **d \| 3** | **0.001** $^{ab}$ | 0.015 | 0.262 | 1 | 0.893 |
| | **d \| 4** | **0.001** $^{ab}$ | **0.010** $^{a}$ | 0.209 | 0.893 | 1 |

Bonferroni corrected significance levels: $^{a}$ $\alpha_{b1} = 0.01$ $^{b}$ $\alpha_{b2} = 0.005$

*20* are significantly better than no diffusion at all. This permits us to say that any system, that follows our prerequisites even loosely, would benefit from such an extension.

This paper focused on some basic ideas of the applicability of the diffusion. We want to take the subject further and test it in a much more general environment, where resources appear, disappear and modify. Our next step will also consist of applying the concept of diffusion to real-life networks, with where the dynamism is not limited to the nodes' content, but also to the nodes themselves which attach and detach from the system, creating breaks in the pheromone trail's continuity. We also intent to work on more elaborate and sophisticated diffusion algorithms, which revolve around the concept of the directed diffusion achieved through exploiting the topology of the network. This will lead to the creation of a lightweight diffusion algorithm that complements efficiently the classical ACO paradigm in P2P environments.

## 4.8 Acknowledgments

Kamil Krynicki is supported by a FPI fellowship from Universitat Politècnica de València, with reference number 3117. This work received financial support from the Spanish Ministry of Education under the National Strategic Program of Research and Project TSI2010-20488.

# Chapter 5

# An ACO-based personalized learning technique in support of people with ABI

Kamil Krynicki[†] Javier Jaen[†] Elena Navarro[*]

[†] ISSI Research Group, Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

[*] LoUISE Research Group, Computing Systems Department
University of Castilla-La Mancha, Avda España s/n, 02071 Albacete, Spain

*Under Review, as of December 4, 2015*

## Abstract

The ever-increasing cases of acquired brain injury (ABI), especially among young people, have prompted a rapid progress in research involving neurological disorders. One important path is the concept of relearning, which attempts to help people regain basic motor and cognitive skills lost due to illness or accident. The goals of relearning are twofold. First, there must exist a way to properly assess the necessities of an affected person, leading to a diagnosis, followed by a recommendation regarding the exercises, tests and tasks to perform; and second, there must be a way to confirm the results obtained from these recommendations in or-

der to fine-tune and personalize the relearning process. This presents a challenge, as there is a deeply-rooted duality between the personalized and the generalized approach. In this work we propose a personalization algorithm based on the Ant Colony Optimization (ACO), which is a bio-inspired meta-heuristic. As we show, the stochastic nature of ants has certain similarities to the human learning process. We combine the adaptive and exploratory capabilities of ACO systems to respond to rapidly changing environments and the ubiquitous human factor. Finally, we test the proposed solution extensively in various scenarios, achieving high quality results.

## 5.1 Introduction

Computer-aided recommendation, which is now extensively used, ranges from content recommendation for ad-related issues to music and video suggestions. A particular niche of the subject is the learning unit recommendation, which usually consists of suggesting to a student the next learning unit to complete with e-learning software. The computation techniques behind these problems are plentiful, ranging from fuzzy logic, neural networks to ant colony optimization (ACO). Due to its nature, ACO is very well suited for establishing paths in a discrete space, which can be seen as a metaphor for progressing through a given set of study material. An interesting case is one where the search over the space of learning units is not limited to a sequence, but rather centers on finding the most suitable unit for a given user at a given time.

One particular use of the learning schema is the *relearning process*, which is, as the name suggests, the process of learning abilities that have been lost. Relearning is especially important in the context of people with Acquired Brain Injury (ABI). The World Health Organization states that people with ABI have suffered "Damage to the brain, which occurs after birth and is not related to a congenital or a degenerative disease. These impairments may be temporary or permanent and cause partial or functional disability or psychosocial maladjustment." [81]. The causes of ABI vary; they include, but are not limited to skull-brain trauma, degeneration of the blood vessels, meningitis and brain tumors. ABI affects many people every year; in the United States alone 1.7 million people suffer brain injury each year [82]. The impact of ABI is wide-ranging, from a multitude of physical effects, such as muscle spasticity, paralysis or weakness, to cognitive abilities, such as memory, thinking skills or concentration, and organization or planning abilities.

There is a general consensus that an automatized relearning process has a high impact on the individuals affected by ABI. However, this process must be carefully customized to individual abilities/disabilities, since no two people can expect to have the same outcome or problem, even after a seemingly similar brain injury. Proposing a relearning model is more complex than the straightforward learning

unit recommendation or progressing through a sequence of tasks. For people with ABI there is no recognized path through the space of relearning activities to success. Instead the most appropriate activities must be found for each new situation. In addition, the software must take into account the user's cognitive and motor skills to an extent that is simply unnecessary for classical learning software.

In this work we present an ACO-based recommendation technique that incorporates the above assumptions and concerns. This technique suggests activities to individuals with ABI, starting with those that have proven to be the most generally successful, and then progressing towards more personalized recommendations, at all times taking into account the user's mental and physical state.

The remainder of this paper is structured as follows. Section 5.2 discusses related works and highlights the novelty of the present work. Section 5.3 presents the problem domain in more detail. Section 5.4 presents the mathematical definition of our ACO algorithm for personalized learning. Section 5.5 describes the experiments performed with their results and a discussion. Finally, Section 5.6 presents our main conclusions and areas of future work.

## 5.2 Related Work

E-learning systems are powerful tools that complement or even replace regular teaching/learning activities. As mentioned, the relearning process for individuals with ABI is also a learning/teaching process that they must go through to recover their lost abilities. The reasons for the adoption of e-learning systems are many, but the fact that it is available at all times and places is one of its most attractive aspects. However, despite undeniable advantages, their use can be frustrating for educators, as they can find it difficult and time consuming to define the paths each learner has to follow [83]. For this reason, the introduction of *recommendation systems* [84] meant a meaningful step forward for both, educators and learners. These systems have taken advantage of different techniques, such as Collaborative Filtering [85], Ant Colony Optimization [86], Data mining [87], particle swarm optimization [88] or combinations of different techniques [89].

Unfortunately, most of the recommendation systems simply exploit learner preferences, interests and browsing behaviors as input to recommend learning activities. As stated in [90], these approaches miss an important point, which is taking into account the abilities of the learner, in order to truly equip e-learning systems with *personalized learning mechanisms*. De Maio et al. [91] encourage the exploitation of personalized e-learning as the way to become "more effective and efficient when a great number of educational content requires to be dynamically filtered and assembled with respect to learners' preferences and cognitive states." The user's

cognitive state thus becomes the cornerstone of the recommendation systems in the personalization process.

The development of personalized e-learning systems has received a great deal of attention from the Artificial Intelligence community. Some studies have promoted the use of artificial neural networks to implement the aforementioned aspects of intelligence. For instance, Baylari and Montazer [92] developed a multi-agent system able to estimate the learners' abilities in order to provide them with tests personalized and adapted to those abilities. One of the main strengths of this work is the exploitation of an artificial neural network to discover users' learning problems when using the system and then recommend them appropriate learning material. Another interesting proposal is called MASACAD [93], a multi-agent e-learning system able to provide students with academic assistance by using an artificial neural network based on their preferences to infer such advice.

Fuzzy systems have also been used in the development of personalized e-learning systems. Lu et al. [89] applied this approach in a system that provides learners with learning objects by analyzing both, the course difficulty of the material and the abilities of the learner. They employ a fuzzy set clustering algorithm to create clusters of similar learners and assign them learning objects.

The introduction of evolutionary algorithms has also been exploited in the context of personalized e-learning systems. For instance, Maio et al. [91] developed a system, named Intelligent Web Teacher (IWT). IWT builds the user's learning context (current needs, cognitive abilities and preferences) in an automatized way, personalizing the learning experience through ad-hoc educational paths. One of the main foundations of IWT is that the generation of learning paths for a specific learner, or a class of learners, is transformed into an optimization problem. Authors combine global and local searches to perform the exploration of the set of learning objects to find the most appropriate learning path that includes the target concepts. Another interesting work is that presented by Acampora et al. [94], who propose the definition of sequences of learning objects in terms of competencies and transform the problem of finding the proper sequence of learning objects into a classical Constraint Satisfaction Problem (CSP). They chose, however, to use a particle swarm optimization algorithm in their solution.

Other interesting proposals have also been developed by employing ACO algorithms. Sharma et al. present in [95] an ACO based algorithm, named Adaptive Content Sequencing in eLearning (ACSeL), that uses ant colonies to evaluate the learning paths and the learners' profiles to recommend suitable content. It is worth noting the dynamic nature of the proposal, as it takes into account the varying (most likely increasing) knowledge levels of the learners in order to tune the strategy to produce a recommendation. Wang [96] [26] also used an ACO model to analyze past learning experiences to discover new learning paths. Although this proposal also uses previous learning experiences to update the pheromone trails

in a similar way to that presented by ACSeL, it only considers the best solution to avoid long convergence times. An interesting aspect here is the introduction of a training strategy that promotes the division of learning goals into sub-goals to fit the time available for the learning session.

Similarly to some of the aforementioned authors, in our proposal we opted for an evolutionary algorithm (ACO). A desirable feature of our solution is that it takes into account the state of the individuals with ABI to drive the optimization search. It is also dynamic and able to adapt the recommendation according to the recent behavior of not only the user in question, but a cluster of users of similar characteristics or even the system as a whole. However, its most remarkable feature is that it does not focus on actually finding a learning path, as in all the cited proposals, but on the selection of a ranked set of learning activities. It also takes into consideration the cognitive state of the user as the set of deficits (deductive reasoning, sustained attention, short-term memory, etc.) that is treated as a characteristic unrelated to the knowledge levels, which is a unique property. To the best of the authors' knowledge no similar proposals have previously been defined for ABI sufferers.

## 5.3 Problem Domain

ABI, as defined in the Introduction, may result from a number of different causes, either internal or external. Internal causes are the most frequent in the elderly, usually due to vascular disorders, such as strokes or hemorrhages. External causes, generally known as traumatic brain injury (TBI), are usually due to traffic accidents, falls, etc. As it can be seen from these examples, every one of us is exposed to this problem at any point in our lives. This explains why the number of people with ABI is growing every year and is currently one of the most frequent health problems. For instance, according to the Brain Injury Center [97] TBI is more common than breast cancer, spinal cord injury, HIV/AIDS, and multiple sclerosis (MS) combined. In the United Kingdom alone it is estimated that at least 1 million people live with the long-term effects of brain injury [98]. In fact, ABI is recognized throughout the world as a problem of epidemic proportions, known as "the Silent Epidemic".

Brain damage can result in different long-term deficits, depending on the area injured and the level of damage. These deficits can be classified into four categories: i) physical deficits that limit the control of a part of the body, such as paralysis or motor coordination; ii) cognitive deficits that impair intellectual performance, such as memory problems; iii) emotional problems that limit or change the control of the feelings, such as depression or anxiety; iv) behavioral deficits that negatively affect the interaction with the environment, such as irritability and restlessness. Although physical deficits are difficult for people to adapt to [99], the cognitive

ones are highly disabling, as they interfere with the rehabilitation process and have the most negative effect on the quality of life [93].

The following impairment types are usually related to cognitive deficits [100]:

- Executive function impairments: problems in controlling and regulating activities or behaviors that include abstraction, categorization (the ability to recognize objects and actions), cognitive flexibility (the ability to adapt cognitive processing to new and unexpected situations), deductive reasoning, planning and problem solving.
- Attention impairments. These can be detected when a person with ABI exhibits problems with abilities [101] such as sustained attention (to direct and focus a cognitive activity, given a specific series of stimuli), divided attention (to be able to simultaneously respond to multiple stimuli), or selective attention (to be able to identify the relevant stimulus while several distracting stimuli are generated).
- Memory impairments: including short-term memory, semantic memory, related to the ability to collect information and knowledge about the world without considering previous experiences, episodic memory of personal events, such as places and emotions, which can be explicitly stated and procedural memory, based on implicit learning, mainly for motor skills.
- Language deficits: detected when a person has difficulties in understanding or communicating, reading or understanding a document.
- Spatial perception deficits: people with these deficits exhibit difficulties with construction activities that require spatial abilities.

There is increasing evidence that individuals with ABI should be provided with proper treatment as soon as possible [102]. This treatment is often carried out in a center where they perform a number of supervised activities, usually employing a board game or learning cards. However, this alternative has several drawbacks, especially in terms of the time available for the relearning process, since it is highly dependent on the number of specialists available. In addition, Christiansen et al. [103] confirmed that the use of computers in the relearning process helps to encourage and stimulate cognitive behavior and allows the disabled to reinstate damaged functions.

The concept of using e-learning systems for their treatment thus emerges in a natural way. One of these systems is called HABITAT [104, 105] and has been developed by the authors of this work in collaboration with the ABI Association of Castilla-La Mancha (ADACE). For its development, researchers from the University of Castilla-La Mancha carried out continuous tracking of the relearning

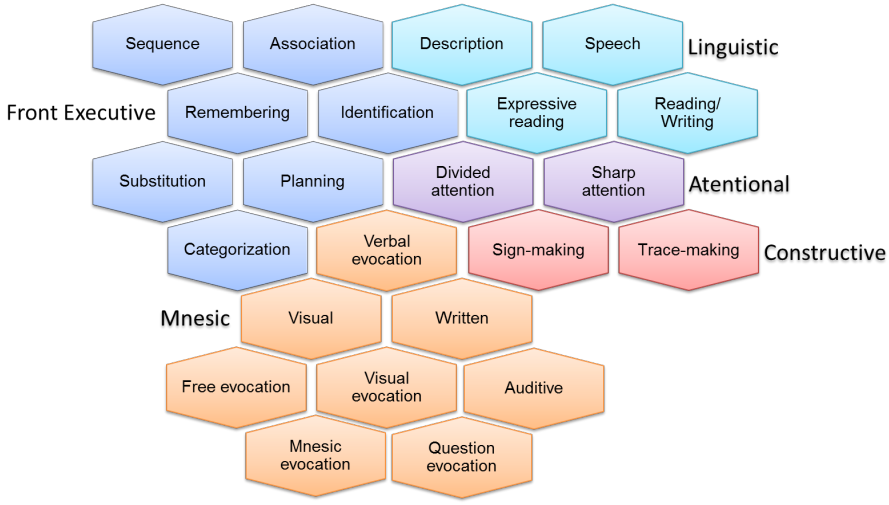



**Figure 5.1:** ReAP catalog with 23 cognitive activity patterns.

process of the handicapped over a period of two years. This study was designed to determine how the relearning process could be supported by software, what different kinds of relearning activities should be provided by an e-learning system and how they could be created and parameterized by specialists.

All this experience was documented as a catalog of relearning activity patterns (ReAP) [106]. Twenty-three cognitive activity patterns were identified and their names are shown graphically in Fig. 5.1. It is worth noting that these patterns are classified according to the main cognitive area they are designed to treat. Within the scope of the activity patterns specialists create new relearning activities, such as the one illustrated in Fig. 5.2, customized according to the specific needs of the individual subjects.

One of the most demanding tasks that specialists have to carry out is the design of the relearning process of each individual. They have to select learning activities taking into account a range of different characteristics, such as age, damage level, stress, etc. of the person being treated. Due to the success of the e-learning software, we can claim that this step can be largely simplified or even eliminated by the proper use of dynamic e-learning models. The definition and development of such a recommendation technique would be a valuable asset for both specialists and people affected by ABI alike. The former could devote more time to the treatment and the latter would benefit from a relearning process specially adapted to their needs.

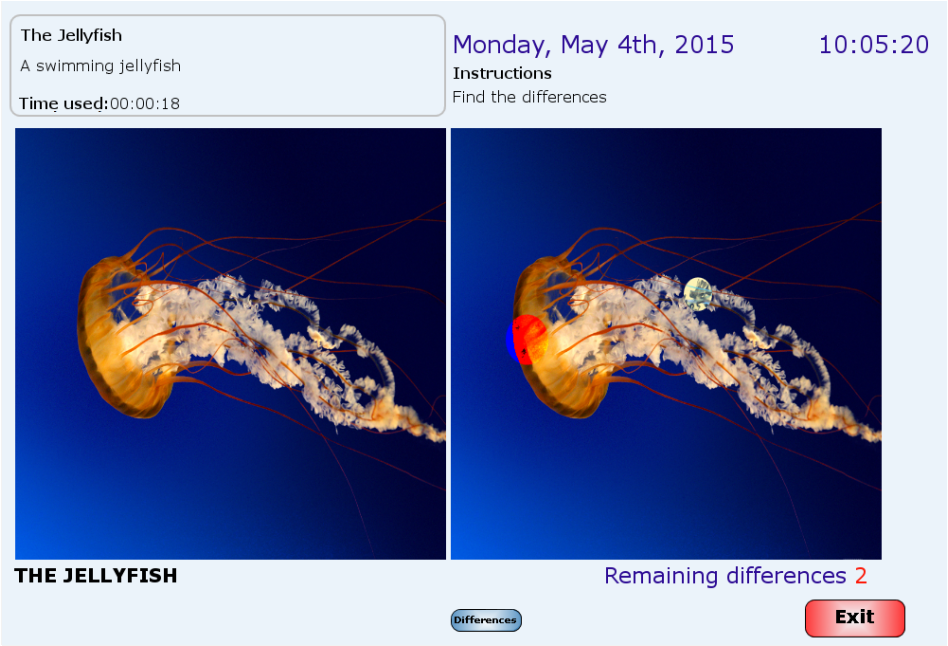



**Figure 5.2:** Divided Attention Relearning Activity.

## 5.4 An ACO Algorithm for ABI Rehabilitation Tests Recommendation

### 5.4.1 Ant Colony Optimization

ACO [29] is a non-deterministic evolutionary algorithm based on the use of simple and stochastic automata to perform complex optimization tasks. The automata in question are like ants in a colony searching for food and resources. The mathematical model behind ACO is simple, yet powerful; each ant may find itself in two states - either searching for food randomly, the so-called exploration phase, or following the established paths, the exploitation phase. Without the loss of generality, the worlds in which the ants live are limited to bidirectional graphs rather than continuous open spaces.

Every search for resources must begin and conclude in one of the nodes of the system. Once an ant is brought to life, it is given an objective and end conditions; as soon as they are fulfilled the ant returns to the emitting node and reports its findings. In each node it visits, it collects resources corresponding to the goal associated. Pheromones are the essence of inter-ant communication; ants either

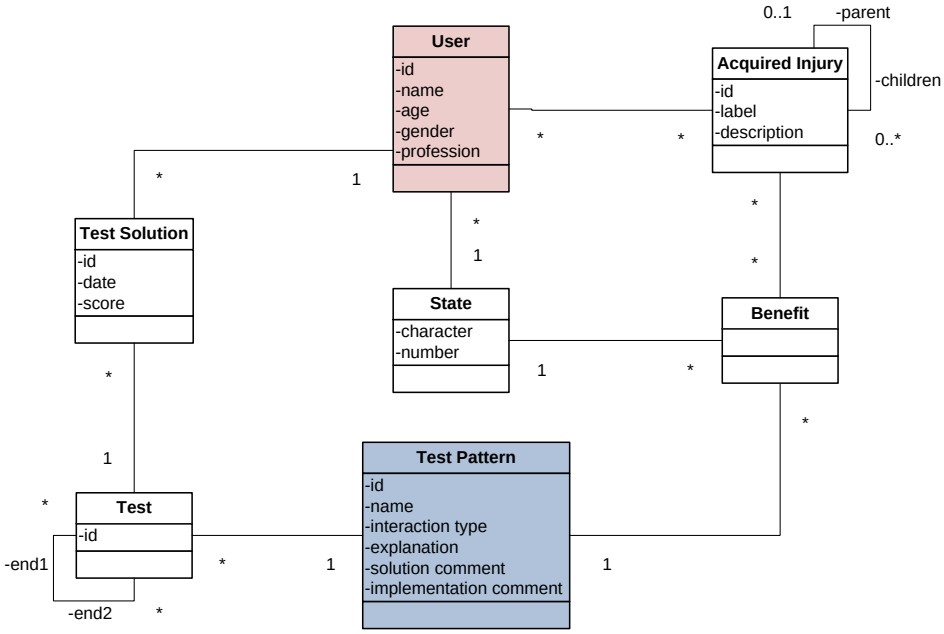



**Figure 5.3:** Conceptual Model.

follow or deposit the pheromone according to the quality of their findings and the mode of behavior they choose.

In short, the basic components of the ACO model are: i) graphs, composed of nodes; ii) links between nodes, with assigned values of pheromone and cost; iii) the resources in each node, representing goods of various types; iv) ants, the search automata. Each of these elements finds its corresponding metaphor in our conceptual model, as explained in Subsection 5.4.2.

### 5.4.2 Problem space conceptual model

In Fig. 5.3 we present the conceptual model of the problem space. The two main components are: i) *user*, which represents a person, with its basic data and ii) *test pattern*, or a *relearning activity* pattern $p$ in the ABI domain (see Section 5.3), which is an abstraction of a group of tests $t$. An example of a test pattern would be the *Association* test pattern, under which specialists create tests containing a particularized set of images and texts to match, depending on the specific abilities to treat.

Each user $u$ is described by i) an impairment state $e_u$ which is denoted A, B, C, D or E. From A to D the degree of cognitive impairment increases, and E indicates

a person in coma who does not receive any treatment as explained in [106]; ii) the list of acquired injuries $dc_u = [dc_1, dc_2, dc_3, \dots]$, in order of importance. The first acquired injury is referred to as main acquired injury; iii) additional metadata that are taken into account, such as the user's age $w_u$ (integer value) and gender $g_u$ (boolean value, 1 for female, 0 for male), as well as the profession $j_u$, drawn from the hierarchical structure provided by [107]. The full description of a user is written as: $u = (e_u, dc_u, j_u, w_u, g_u)$.

On the other hand, each test pattern $p$ provides a set of benefits $dc_p$ for users in given states $e_i$ and acquired injuries $dc_i$, written as: $dc_p = \{(e_1, \{dc_{11}, dc_{12}, \dots\}), (e_2, \{dc_{21}, dc_{22}, \dots\}), (e_3, \{\dots\}), \dots\}$. The $dc_p$ notation can be read as: for users in the state $e_1$ the test will be beneficial in case of the acquired injuries $dc_1, dc_2, \dots$, etc. Note that each state corresponds to a set of acquired injuries rather than a list, therefore the order of acquired injuries does not reflect the increase or decrease of efficiency of the test pattern. In addition, not all possible states must be benefited by a given $p$. For instance, state E is never benefited, as it is the comatose state, which means user-system interaction is impossible. All the test patterns available in the system are organized in a tree-like structure that reflects dependencies and relates them to each other.

We argue that this approach, alongside underlying ACO strategy, is sufficient to model the user–system interactions. The key algorithmic problem is to propose a measure that would be capable of reflecting the satisfaction level of a user $u$ with a test pattern $p$ that goes beyond a simple scoring system. Our proposal bases the measure on the performance history of all the users, as well as the potential match between the list of user's acquired injuries and test pattern's benefits. We elaborate on the subject in detail in Subsection 5.4.3.

The modeling of the problem space with the concepts offered by the ACO domain is as follows: i) tests $t$ are represented by nodes $N$; ii) tests $t$ are organized in a graph structure $K$, which is the aggregate of all the tests available; iii) the action of querying tests for a user $u$ is implemented as a release of a number of virtual ACO ants onto $K$; iv) the action of solving tests produces a test solution $Ts$ object, which is related the test in question $t$ and the user involved $u$, as well as the completion date $d$ and score $s$ and written as $Ts = (t, u, d, s)$; v) test solutions $Ts$ are the resources of the model. They are evaluated and collected by ants and stored in nodes $N$ of $K$.

The graph $K$ has a toroidal topology with random distribution. In the work [12] we demonstrate that the topology has only a minor impact on the efficiency. We therefore decided not to take this property as a factor in our study.

As mentioned, each ant models the action of searching for a test pattern suitable for a given user. The ants are routed following a well-known ACO algorithm, the Ant Colony System (ACS) [17]. For each visited test in the node $n_i$ the ant

generates an evaluation response $res_i = (n_i, s_i)$, where $s_i$ is the score of the node for the given query. See Algorithm 15 for details on how the responses $res_i$ are obtained. The algorithmic mechanics behind the process of querying is explained in Subsection 5.4.3.

It is important to point out here our approach to the pheromone concept. Traditionally, there is one layer of pheromone on top of the underlying graph. This means that each ant reads and writes the same values. In some studies, however, various levels of pheromone per graph have been introduced [12, 13]. This matches our situation, as in our problem-space there are many diverse users with unrelated disabilities. Therefore, we use one pheromone level per disability. As a consequence, the ants are introduced only to the level corresponding to the main disability of the user related to the query.

### 5.4.3 Algorithmic Design

Our algorithmic design allows users to query for tests at any moment, requiring as input the minimum and a maximum normalized score of tests to solve, $s_{min}$ and $s_{max}$ respectively. The higher the $s_{min}$ the higher the match quality required to present the test found to the user. The lower the $s_{max}$ the more diverse and unexpected the suggestions become. The narrower the gap between the $s_{min}$ and the $s_{max}$ the smaller the number of results becomes.

High level recommendation querying is governed by Algorithm 10. Once the query is launched, the system releases a series of ants in search of tests. During the querying process, the ants evaluate nodes along the way. The ants are routed according to the traditional ACS rules, as mentioned in Section 5.4.1, with the exception of the pheromone, which has been divided into layers. Each pheromone layer corresponds to an acquired injury $dc$.

**Algorithm 10:** High level query execution

1: **assume**

- user $u$ with main acquired injury $dc_u$
- minimum $s_{min} \in \mathbb{R}_0^1$ and maximum $s_{max} \in \mathbb{R}_0^1$ desired score
- a subset of the nodes of the graph $K$, called injection points $I_p$

2: **for each** injection point $i_p \in I_p$ produce an ant $A_{ip}$ and release it into the pheromone layer corresponding to $dc_u$

3: **while** all $A_{ip}$ not finished **do in parallel** $A_{ip}$ builds a proposal of a solution $aRes_{ip}$ (ant response set): a response $res_i = (n_i, s_i)$ for every node $n_i$ visited

4: **end while**

5: **let** $pRes$ (partial response set) be the aggregate of the $aRes_{ip}$. $pRes$ contains repetitions with respect to the $n_i$ value, as various ants might have evaluated the same node.

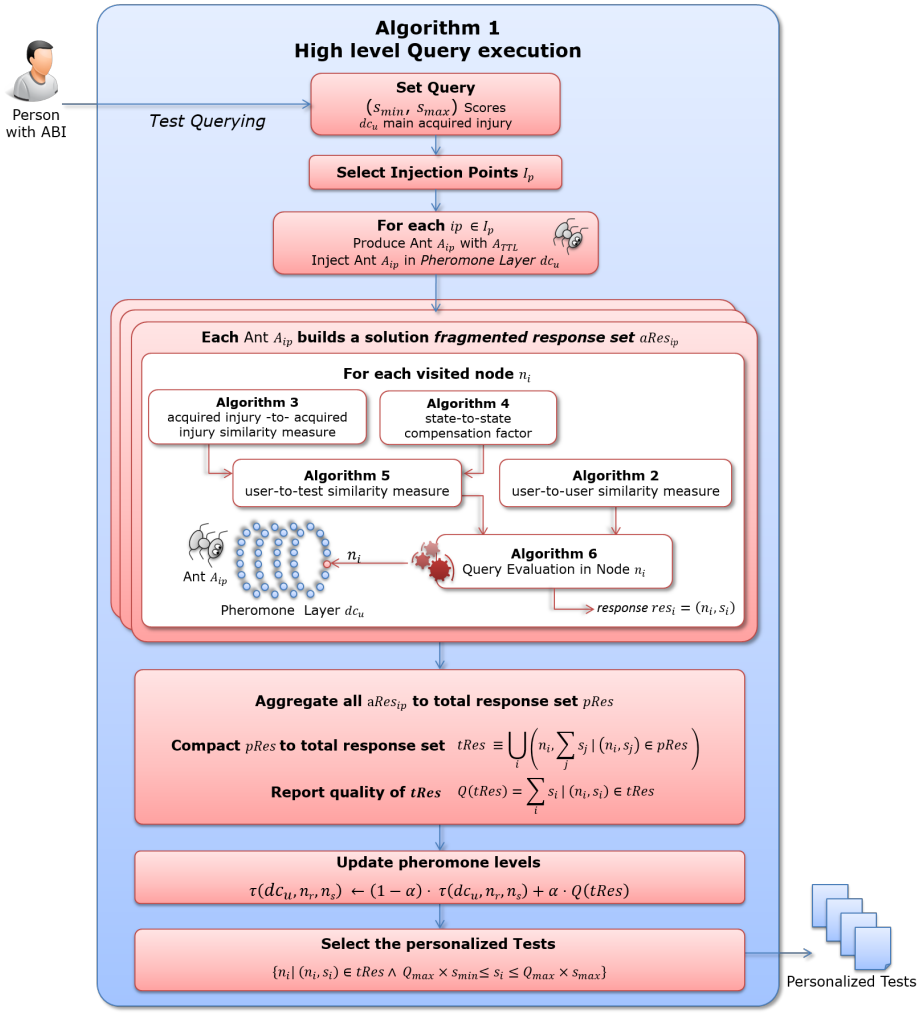



**Figure 5.4:** Graphical description of Algorithm 10.

6: **obtain** total response set $tRes$, by removing the repetitions from the partial response set as shown in (5.1)

$$tRes = \bigcup_i \left( n_i, \sum_j s_j | (n_i, s_j) \in pRes \right) \tag{5.1}$$

7: **obtain** the total (5.2) and top (5.3) quality of $tRes$:

$$Q(tRes) = \sum_i s_i | (n_i, s_i) \in tRes \tag{5.2}$$

$$Q_{max}(tRes) = \arg\max_{s}(n_i, s_i) \in tRes \tag{5.3}$$

8: Update pheromone values on links that participated in the solution:

$$\tau(dc_u, n_r, n_s) \leftarrow (1-\alpha) \cdot \tau(dc_u, n_r, n_s) + \alpha \cdot Q(tRes) \tag{5.4}$$

where:

- $\tau(dc_u, n_r, n_s)$ is the pheromone value, in the pheromone layer $dc_u$, between the tests (nodes) $n_r$ and $n_s$
- $\alpha \in \mathbb{R}_0^1$ regulates the influence of the new pheromone

9: The final result consists of tests $t$ associated with nodes $n_i$ that fulfill (5.5)

$$\{n_i | (n_i, s_i) \in tRes \wedge Q_{max} \cdot s_{min} \leq s_i \leq Q_{max} \cdot s_{max}\} \tag{5.5}$$

Our algorithmic design, as described, is based on the definition of similarity functions between users (Algorithm 11), between acquired injuries (Algorithm 12) and between user states (Algorithm 13). Fig. 5.4 offers a graphical description of Algorithm 10 to illustrate how the different subsequent algorithms are used.

The ability to compare different users is essential. It enables us to extrapolate the behavior of one user to recommend tests to users without performance history. In (5.6) we obtain a user-to-user distance $d(u_1, u_2)$ as a weighted linear combination of several subdistances that have been deemed the most relevant. Apart from the natural distances of disability and physical state we chose to distinguish users of different professional backgrounds. The gender- and age- distribution is based on the fact that the same disabilities acquired at different moments and situations in life can have different sources and, therefore, must not be considered completely identical. With (5.7), we obtain a smooth and normalized similarity function $sim_{u-u}$ with the softest and most gradual curve around the mean of the population. The function used for the transformation of the unbound distance value into a bound similarity level must have the following properties: i) domain of $\mathbb{R}$; ii) upper- and lower-bound codomain, limits approached asymptotically; iii) be antisymmetric and monotonically growing. We chose arctan, as it fulfills the above properties. We transformed arctan to center on the $(0, 0)$ point and normalized it with $\pi^{-1}$ factor.

**Algorithm 11:** $sim_{u-u}$, user-to-user similarity measure

1: **assume**

- users $u_1 = (e_1, dc_1, j_1, w_1, g_1)$ and $u_2 = (e_2, dc_2, j_2, w_2, g_2)$, where: $e_i$ is the physical state, $dc_i$ is the main acquired injury, $j_i$ is the profession, $w_i$ is the age and $g_i$ is the gender of the $i$-th user, for $i = 1$ and $i = 2$.

2: **obtain**

$$d(u_1, u_2) = 1 + \alpha_{dc} \times d(dc_1, dc_2) + \alpha_j \times d(j_1, j_2) + \alpha_w |w_1 - w_2| + \alpha_g |g_1 - g_2| \tag{5.6}$$

where

- $d(dc_1, dc_2)$ and $d(w_1, w_2)$ are the shortest distances in the corresponding hierarchical structure between the values in question
- the parameter values: $\alpha_{dc} = 0.7$, $\alpha_j = 0.1$, $\alpha_w = 0.1$, $\alpha_g = 0.1$, and serve as weights in the equation. The most important component was given a dominating value, the rest is evenly distributed among the remaining factors.

3: **obtain**

$$sim_{u-u}(u_1, u_2) = \frac{1}{2} - \frac{1}{\pi} \times \arctan\left(\frac{d(u_1, u_2) - \mu_d}{\sigma_d}\right) \tag{5.7}$$

where

- $\mu_d$ and $\sigma_d$ are the mean and the standard deviation of the user population

Algorithm 12 calculates the similarity between acquired injuries, $sim_{d-d}$. As the acquired injuries are provided in a hierarchical structure [106] we derive the similarity from the in-structure distance $d(dc_1, dc_2)$. Note that here we obtain a bound and normalized value, based on an unbound distance measure. The *Type Score Penalty* parameter in (5.8) is used to express how the absolute distance between two concepts translates into a normalized value.

**Algorithm 12:** $sim_{d-d}$, acquired injury-to-acquired injury similarity measure

1: **assume**

- two acquired injuries $dc_1$ and $dc_2$

2: **if** $d(dc_1, dc_2) = \infty$ **then** $sim_{d-d}(dc_1, dc_2) = 0$.

3: **else**

$$sim_{d-d}(dc_1, dc_2) = \delta^{d(dc_1, dc_2)} \tag{5.8}$$

where:

- $d(dc_1, dc_2)$ is defined in Algorithm 11
- $\delta \in \mathbb{R}_0^1$ is a parameter called *Type Score Penalty*

4: **end if**

Algorithm 13 produces a compensation factor $c_{s-s}$ between users in different states. One must expect that users in identical states behave similarly. However, it is also possible to draw some limited conclusions from similar users in different states. We designed the compensation factor with diminishing values according to the distance between the states to highlight this property.

When calculating the compensation factor between states $e_1$ and $e_2$ we distinguish the first state (the input state) and the second (the base state). The naming convention was established as we tend to iterate the input state, which is bound to the current user, over all the possible states in order to obtain a list of factors, which

later serve as weights in the final similarity calculation. Due to this distinction, the compensation factor is not symmetric:

$$c_{s-s}(e_1, e_2) \neq c_{s-s}(e_2, e_1) \tag{5.9}$$

The complexity of the Algorithm 13 is a consequence of the most crucial requirement imposed on $c_{s-s}$. Namely, the sum of all the factors must be constant when iterating one input value over all the possible base values. This condition allows us to use the calculated factors as weights in (5.12). In addition, the coefficients must decrease in the function of state-to-state distance. Factors linearly dependent on the distance would cause central states to receive higher quality measures than extreme states, simply due to lower average distance. For instance, in the case of three states, the middle one has an average distance of 1 to all the others, while the extreme ones are exactly at a distance of 1.5. Therefore, the use of linear compensation as weights is not recommended, as it favors the central state. The same can be said for all polynomially-dependent coefficients. With our approach we obtain an unbiased coefficient matrix of an exponential nature (see Table 5.1 for an example) that permits us to take into account diminishing impacts of some states over others. Algorithm 13 takes 2 as the base of the exponential dependency, but it can be easily reformulated for other values.

**Algorithm 13:** $c_{s-s}$, state-to-state compensation factor

**assume**

- states $e_i$ (input state) and $e_b$ (base state) with numerical values of $|e_i|$ and $|e_b|$
- $E$ the number of possible states

**if** $|e_i| = |e_b|$ **then** $c_{s-s}(e_i, e_b) = 1$
**else**
  **if** $|e_b| > \frac{1}{2}(E-1)$ **then** $x_1 = E - |e_b|$
  **else** $x_1 = |e_b|$
  **end if**
  $x_2 = E - x_1$
  $C = (2 - 2^{-x_1} - 2^{-x_2})^{-1}$
  **obtain**

$$c_{s-s}(e_i, e_b) = C \times 2^{-||e_i|-|e_b||} \tag{5.10}$$

**end if**

With the above algorithms in place we may now define a measure for the user-to-test similarity (Algorithm 14) that summarizes all the properties of a test pattern and a user. The value obtained from the algorithm is used as the static component in the posterior quality analysis.

**Table 5.1:** Example of Algorithm 13.

| | | | Base | | | | | Sum |
|---|---|---|---|---|---|---|---|---|
| | State | | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | |
| | | Value | 0 | 1 | 2 | 3 | 4 | |
| Input | $e_1$ | 0 | 1.00 | 0.53 | 0.27 | 0.13 | 0.07 | 2.00 |
| | $e_2$ | 1 | 0.36 | 1.00 | 0.36 | 0.18 | 0.09 | 2.00 |
| | $e_3$ | 2 | 0.17 | 0.33 | 1.00 | 0.33 | 0.17 | 2.00 |
| | $e_4$ | 3 | 0.09 | 0.18 | 0.36 | 1.00 | 0.36 | 2.00 |
| | $e_5$ | 4 | 0.07 | 0.13 | 0.27 | 0.53 | 1.00 | 2.00 |

**Algorithm 14:** $sim_{u-t}$, user-to-test similarity measure

1: **assume**

- user $u$ in the state $e_u$ with the acquired injuries: $dc_u = [dc_{u1}, dc_{u2}, \dots]$
- test $t$, created under the scope of the test pattern $p$, with sets of benefits: $dc_p = \{(e_1, \{dc_{11}, dc_{12}, \dots\}), (e_2, \{dc_{21}, dc_{22}, \dots\}), (e_3, \{\dots\}), \dots\}$.

2: **for each** $(e_k, D_k) \in dc_p$, **obtain** the *partial similarity component* $psim_{u-t}$:

$$psim_{u-t}(dc_u, D_k) = \sum_{dc_i \in dc_u} \sum_{dc_j \in D_k} \gamma^{i-1} \times sim_{d-d}(dc_i, dc_j) \tag{5.11}$$

where:

- $\gamma \in \mathbb{R}_0^1$ is a parameter named *Benefit Score Penalty*
- $i$ iterates over all the acquired injuries of the user $u$ in the order provided
- $j$ iterates over all the benefits the test pattern $p$ provides for users in the state $e_k$
- $sim_{d-d}$ is the acquired injury-to-acquired injury similarity measure (see Algorithm 12)

3: **obtain**

$$sim_{u-t}(u, t) = \sum_{(e_k, D_k) \in dc_p} psim_{u-t}(dc_u, D_k) \times c_{s-s}(e_k, e_u) \tag{5.12}$$

Equation 5.12 should be understood as a weighted sum of all the partial similarity components $psim_{u-t}$, where the state-to-state compensation factor $c_{s-s}$ serves as a weight. The partial similarity components (5.11) are the total effect of all the possible cross-combinations of the user's acquired injuries and the test pattern's benefits, with adequate weights in form of $\gamma$. We propose the following example for clarification. Assume i) user $u$ in the state $e_1$ and acquired brain injuries $dc_u = [dc_1, dc_2, dc_3, dc_4, dc_5]$; ii) test pattern $p$, with benefits $dc_p = \{(e_1, D_1), (e_2, D_2)\} = \{(e_1, \{dc_1, dc_3, dc_4\}), (e_2, \{dc_2, dc_4\})\}$; iii) $sim_{s-s}(e_1, e_2) = 0.85$

Based on these assumptions, in Table 5.2 we perform a step-by-step calculation of the two possible partial similarity components. With the $psim_{u-t}$ values ob-

**Table 5.2:** Example of Algorithm 14.

| | | $dc_u$ | | | | |
|---|---|---|---|---|---|---|
| | | $dc_1$ | $dc_2$ | $dc_3$ | $dc_4$ | $dc_5$ |
| $D_1$ | $dc_1$ | 1 | 0* | 0* | 0* | 0* |
| | $dc_3$ | 0* | 0* | 1 | 0.1* | 0.3* |
| | $dc_4$ | 0* | 0* | 0.1* | 1 | 0.3* |
| Sum | | 1 | 0 | 1.1 | 1.1 | 0.6 |
| Weight | | $\gamma^0$ | $\gamma^1$ | $\gamma^2$ | $\gamma^3$ | $\gamma^4$ |
| Weighted Sum | | 1 | 0 | $1.1\gamma^2$ | $1.1\gamma^3$ | $0.6\gamma^4$ |
| $psim_{u-t}(dc_u, D_1)$ | | $0.6\gamma^4 + .1\gamma^3 + 1.1\gamma^2 + 1$ | | | | |
| $D_2$ | $dc_2$ | 0* | 1 | 0* | 0.1* | 0.2* |
| | $dc_4$ | 0* | 0* | 0.1* | 1 | 0.3* |
| Sum | | 0 | 1 | 0.1 | 1.1 | 0.5 |
| Weight | | $\gamma^0$ | $\gamma^1$ | $\gamma^2$ | $\gamma^3$ | $\gamma^4$ |
| Weighted Sum | | 0 | $\gamma$ | $0.1\gamma^2$ | $1.1\gamma^3$ | $0.5\gamma^4$ |
| $psim_{u-t}(dc_u, D_2)$ | | $0.5\gamma^4 + 1.1\gamma^3 + 0.1\gamma^2 + \gamma$ | | | | |

* example value

tained there, and taking $\gamma = 0.5$, we can calculate $sim_{u-t}(u,t)$ as: $sim_{u-t}(u,t) = (0.6\gamma^4 + 1.1\gamma^3 + 1.1\gamma^2 + 1) \times sim_{s-s}(e_1, e_1) + (0.5\gamma^4 + 1.1\gamma^3 + 0.1\gamma^2 + \gamma) \times sim_{s-s}(e_2, e_1) = 1.45 \times 1 + 0.675 \times 0.85 = 2.05$, which is considered a close similarity.

The final algorithm (Algorithm 15) is the evaluation that ants perform in each node $n_i$. First note that we divide the components of the evaluation into two groups: the static components $s_{stat}(u,t)$ and the variable components $s_{var}(u, Ts)$. As the name suggests the *static* components hardly ever evolve in time, they are based on the invariants of the users and the test patterns. The age of the user, their profession or other parameters are not absolutely fixed, and, therefore, even this component may change. The variable component, however, is highly dynamic. It is based on the test solutions $Ts$ present at a given moment in the node $n_i$.

The equations (5.14) and, especially, (5.15) require additional comment. First note that (5.14) is the static score component multiplied by a geometric average of all the variable components. We opted for the geometric average rather than arithmetic average due to the fact that the data in this case is of multiplicative, not additive nature. In (5.15) we single out two clauses. The clause $(a)$ increases the impact of users similar to the user $u$. The second component of this equation is $(b)$ which acts as a score normalizer. It smoothens and softens the linear score dependency and emphasizes higher scores. The $2\pi^{-1}$ factors convert the value range of the subcomponents to $(-1, 1)$ from $(-\pi/2, \pi/2)$. The final $+1$ brings the

value to the $(0, 2)$ range. As we are not dealing with probabilities, there is no need to normalize the value to the range $(0, 1)$.

**Algorithm 15:** Query Evaluation in Node $n_i$

1: **assume**

- ant $A$ searches for tests for the user $u$
- node $n_i$ corresponds to the test $t_i$, within the test pattern $p$
- set of test solutions $Ts = \{(t_i, u_1, d_1, s_1), (t_i, u_2, d_2, s_2), \ldots, (t_i, u_j, d_j, s_j), ...\}$, of size $|Ts|$ which encompasses the performance history of all the past solving of test $t_i$.

2: **let**

$$s_{stat}(u, t_i) = sim_{u-t}(u, t_i) \tag{5.13}$$

where $sim_{u-t}(u, t_i)$ is the static score component, defined in Algorithm 14

3: **obtain** score $s_i(u, t_i)$ of the test $t_i$ for the user $u$:

$$s_i(u, t_i) = s_{stat}(u, t_i) \times \sqrt[|Ts|]{\prod_{Ts_j \in Ts} s_{var}(u, Ts_j)} \tag{5.14}$$

where $s_{var}(u, Ts_j)$ is the variable score component, calculated for every test solution $Ts_j = (t_i, u_j, d_j, s_j)$ available for the test $t_i$:

$$s_{var}(u, Ts_j) = \underbrace{\left(2\pi^{-1} sim_{u-u}(u, u_j)\right)^3}_{\text{(a)}} \times \underbrace{\left(2\pi^{-1} \tan\left(2s_j - 1\right)\right)}_{\text{(b)}} + 1 \tag{5.15}$$

4: **let** the evaluation of the node $n_i$ be the tuple $res_i = (n_i, s_i(u, t_i))$

## 5.5 Experimental Study

### 5.5.1 Experimental Procedure

In order to make mass experiments economically feasible we decided to generate the sets of tests and sets of users in a predetermined and probabilistic manner. The set of users is created according to the strictest rules provided by [108]. Both age-wise, gender-wise and injury-wise distributions are drawn from the mentioned source, which is the most complete we were able to locate. The professions are drawn from the classification [107]. In this way, we argue, one might obtain a statically justifiable set of users. The test patterns are taken directly from our previous work in [106]. Each time we require a set of tests of a given size we generate them maintaining even distribution among the test patterns, i.e. we must expect approximately similar amounts of tests in each test pattern. For

**Table 5.3:** ACS Execution parameters.

| ACO | | |
|---|---|---|
| Parameter | Interpretation | Value |
| $q_0$ | Weight of exploiting vs. exploring strategy | 0.80 |
| $\alpha$ | Pheromone deposition parameter | 0.07 |
| $\rho$ | Pheromone evaporation parameter | 0.10 |
| $\beta$ | Weight of link costs | 1.00 |
| $\gamma$ | Weight of evaporation | 0.02 |
| $\tau_{min}$ | Minimum pheromone level | 0.001 |
| $\tau_{max}$ | Maximum pheromone level | 1.000 |
| $\tau_0$ | Initial pheromone level | 0.009 |

instance, in a test graph of $10^4$ tests, with 23 test patterns, we have about 430 tests per test pattern, which fulfills the requirements of small-world network.

In Table 5.3 we summarize the ACO-related execution parameters. All of them have been taken from the literature and are at their standard values.

### 5.5.2 Experiment 1: Zero-knowledge correctness

The first experiment was designed to confirm the correctness of the model and its implementation. In its uninitialized and uninformed state (zero-knowledge state), without pheromone trails or stored test solutions, the system must behave in a fairly straightforward manner, namely, it should be able to find and positively evaluate the theoretically optimal test patterns in a large number of queries. Any deviations from this are only allowed once the system has stored enough information to incorporate additional factors.

This experiment was designed in the following way: 1) generate a test world of $n$ tests and 40 users; 2) select a random user along with his 3 theoretically most adequate test patterns; 3) perform 100 queries; in each recommendation list obtained, save the best position of any of the 3 most adequate test pattern. No test solving takes place.

The steps $1-3$ were repeated 200 times, 100 for $n = 5 \times 10^3$ and 100 for $n = 10^4$. The experiment-wide average of test recommendations is presented in Fig. 5.5. The theoretically best test patterns was nearly always placed on top 10 ($\sim 97\%$ of the cases) and, in a majority of cases, it was evaluated as the best one ($83\%-89\%$). From these results we can conclude that it is highly improbable that a new user will fail to receive optimal suggestions. The suggestions are, naturally, subject to change according to the performance history of the user and the behavior of others. All the aforementioned phenomena are examined in subsequent experiments.

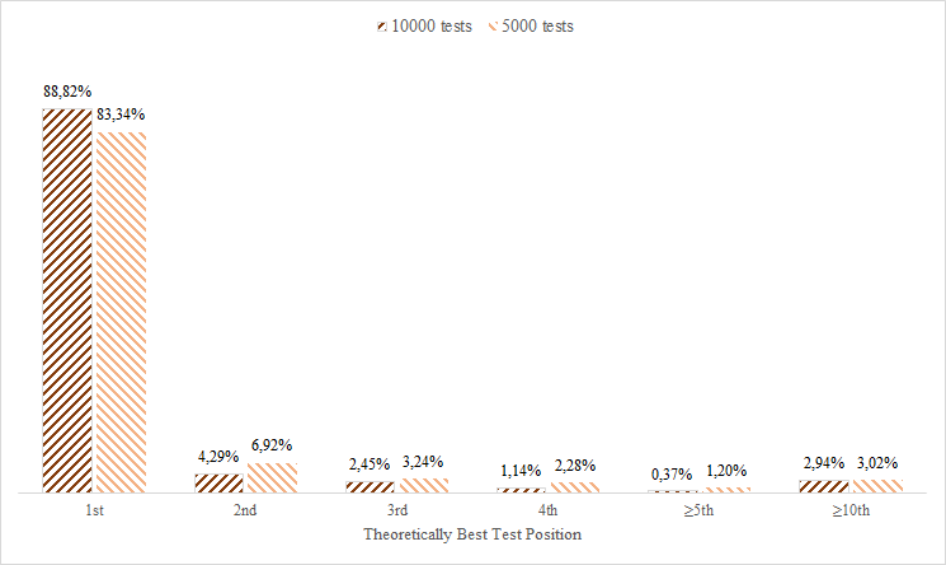



**Figure 5.5:** Zero knowledge state correctness.

### 5.5.3 DNA Graphs

From the user's perspective each query is a simple input-output operation. As input we take the querying characteristics of the user and as output a list of test patterns in descending order of quality. An adequate representation of the system's evolution presented us with a challenge, seeing how the results are multidimensional and change over time. Traditional line graphs were illegible due to the amount of dimensions in question, which in our case surpasses 40. Faced with these challenges we opted for a novel way of representing the evolution that capitalizes on the two dimensions available, in combination with gray-scale intensity. This way we created a visual tool to represent any d-dimensional value evolving over $n_{max}$ discrete time units.

We refer to this type of graphs as *dna-graphs* (see Fig. 5.6)), as they resemble quite closely dna test results. Reading the dna-graphs is straightforward. First, the time axis runs, traditionally, left to right; $n = 0$ is the leftmost edge, while the rightmost one is the end of the experiment. Each vertical cut is the set of recommendations in the n-th time unit, also referred to as the n-th iteration. Each horizontal cut represents the evolution of a single test pattern. Finally, the gray-scale intensity represents the position of the test pattern in the recommendation list - the brighter it is, the higher on the list. Black areas represent test patterns absent from the list. In order to facilitate the graph reading we transferred the idea of rolling average into the visual representation as motion blur, which runs left to right, along the

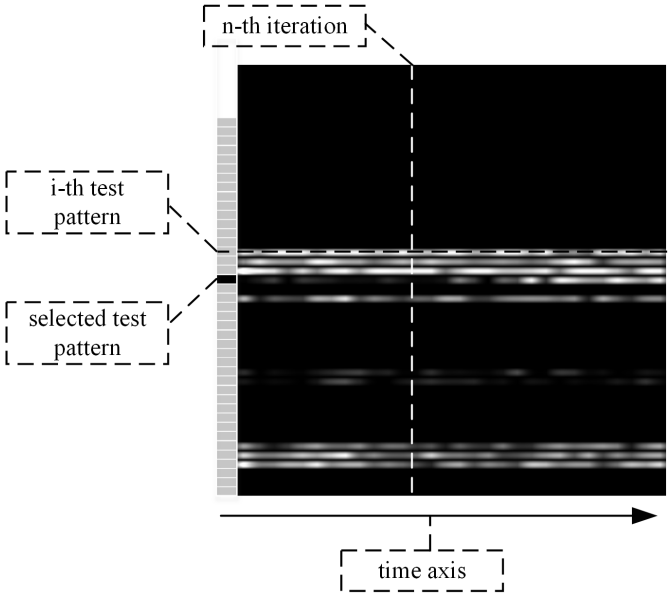



**Figure 5.6:** Dna-Graph example. The parallel evolution of the rank (gray-scale intensity) of 46 test patterns (vertical axis) in function of time (horizontal axis).

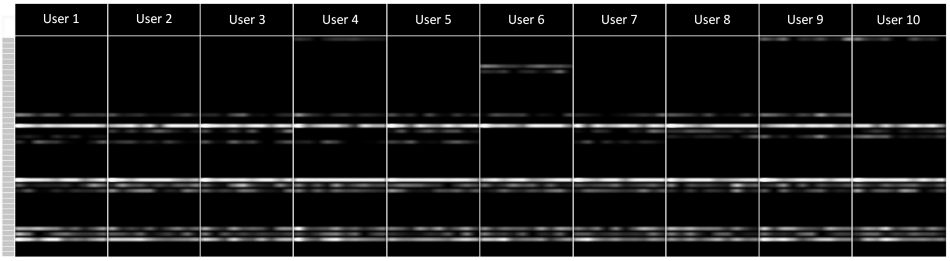
time axis. Note the black rectangle on the left side. On some occasions we use it to mark a test pattern of special significance, the rest will be left in gray.

### 5.5.4 Experiment 2: Inter-user Similarity

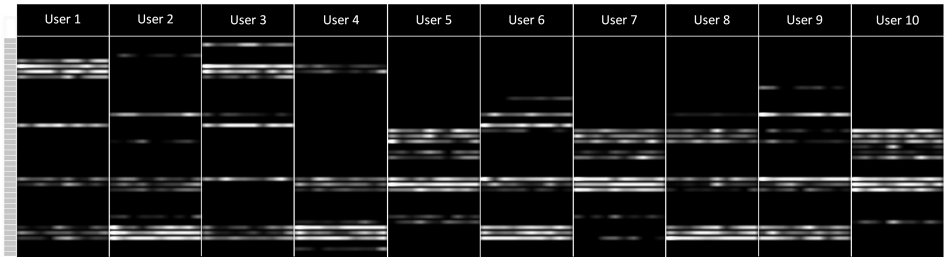
In this experiment we analyze the correctness of the user-to-user ($sim_{u-u}$) similarity measure. Two similar users must receive approximately the same response from an uninitialized system, i.e. before any tests have been solved or pheromone spread and additional profiling could take place.

We generate the test world with $10^3$ tests and 10 users, with the prerequisite that the similarity between each pair of users must not be inferior to 0.85, and we have each user performs 100 queries. There is no test solving involved, similarly to Experiment 1, we only observe the responses of the system.

As can be clearly seen in the series of 10 dna-grahps in Fig. 5.7a, users have received a very similar set of test patterns, with only minor deviations. For instance, User 6 receives some additional test patterns, whereas the rest do not. These differences confirm that the system is capable of distinguishing between even very similar users, yet maintaining the cohesion of results to a high degree. On the




**(a)** Similar Users




**(b)** Random Users

**Figure 5.7:** User Groups. System outputs for different groups of users.

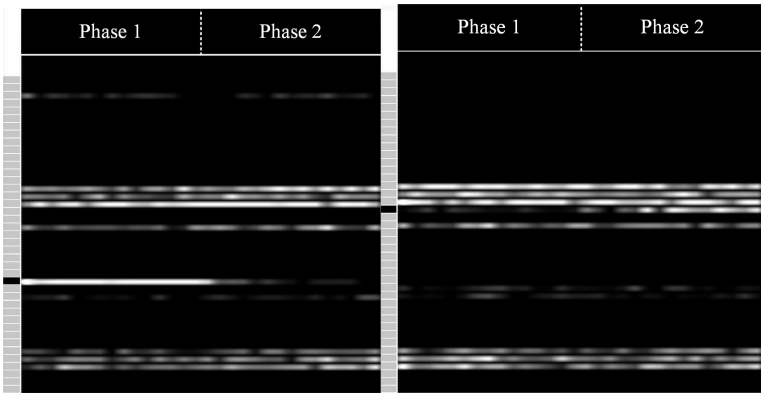
contrary, Fig. 5.7b is an example of a group of random users with their different test pattern recommendation.

### *Statistical Analysis*

**Goal.** Demonstrate that the User-to-User similarity corresponds to an actual similarity of the recommendations. We chose the Pearson Correlation, which is commonly used in such cases.

**Procedure.** First we generate a pool of 100 users with random pairwise similarities, next we let each user perform 100 queries in order to obtain for each one a stable dna-graph of responses, as in Fig. 5.7. Then we process our data crosswise obtaining $10^5$ pairs of (User-to-User similarity, Result-to-Result similarity), where the Result-to-Result similarity is the pixel-by-pixel overlap measures of two dna-graphs. Note that it is an inverse scale, which means that the perfect similarity between graphs is evaluated as 0.

**Results.** Our results show that there is a strong Pearson Correlation ($Pc = -0.423$, $Sig < 0.01$) between the two types of similarity, which in turn tells us that the user-to-user similarity measure is correct; i.e. similar users will




**(a)** Unexpected bad **(b)** Unexpected Good

**Figure 5.8:** Unexpected Behavior. Reinforcement or removal of test patterns.

receive similar recommendations, while different users will receive increasingly different recommendations.

### 5.5.5 Experiment 3: Unexpected Good and Unexpected Bad

In Experiments 1 and 2 we have shown that users are initially given a reasonable set of tests and similar users receive similar recommendations. In this experiment we chose to see how the system reacts to unorthodox behavior, i.e. one that contradicts the base profile of a user. The design of the experiment was the following:

1. Take a single user.
2. Perform 50 queries, no test solving (Fig. 5.8, Phase 1).
3. (Unexpected Bad variant) Select one of the top evaluated test patterns and solve it very badly 50 times. Score 10% of the maximum score. (Fig. 5.8a, Phase 2).
4. (Unexpected Good variant) Select one of the bottom evaluated test patterns and solve it very well 50 times. Score 95% of the maximum score. (Fig. 5.8b, Phase 2).

What we observe is that the system quickly corrects the recommendation lists, incorporating the newly gained, user-related knowledge.

In the Fig. 5.8a we can see that, starting from the halfway point, the test pattern in question immediately drops from the top position to one of the last, eventually

disappearing from the results completely. The opposite behavior is witnessed in Fig. 5.8b where one of the worst of the top ten test patterns is suddenly elevated, after being solved well several times. Note the black rectangle on the left side of both subfigures in Fig. 5.8 that indicates the test pattern in question.

***Statistical Analysis***

**Goal.** Demonstrate that the unexpected behavior results in significant changes in the recommendation lists. We chose the Mann Whitney U test, as it is the most commonly used in non-parametric tests with two populations.

**Procedure.** We decided to construct the test focusing on the recommended test pattern relative rank shift. This rank shift was calculated as a difference between the mean rank of the test pattern during the first 50 iterations and the last 50 iterations, i.e. if a test pattern was 2nd on the recommendation list and it changed with time to the 6th position, then we would consider $RankShift = -4.0$; if it changed from the 5th position to the 3rd position, then $RankShift = 2.0$, and so on. In both variants, we tracked the evolution of 10 test pattern recommendations, 90% of which were left unsolved (Mode = Unaffected) and 10% solved in an unexpected manner (Mode = Unexpected), all repeated 100 times.

**Hypothesis.** Mann-Whitney U test.

**Dependent Variable.** *RankShift* is defined as $RankShift = \overline{Rank_{51-100}} - \overline{Rank_{1-50}}$, where $\overline{Rank_{j-i}}$ is the average rank of the given test pattern between iterations $i$ and $j$.

**Independent Variables.** *Mode* is defined as *Unaffected* for unaffected test patterns and *Unexpected* for tests solved in either unexpectedly well or unexpectedly badly.

**Null Hypotheses.** $H_0$ the Mode variable does not influence the $RankShift$, $H_1$ the Mode variable does influence the $RankShift$.

**Results.** In the Unexpected Good variant we observed the mean $RankShift$ of 0.27 for unaffected tests and $-4.1$ for tests solved well, with test statistics $U = 5.5$, $Sig < 0.001$. Similarly, in the Unexpected Bad variant we observed the $RankShift$ of $-0.22$ and 4.25 for unaffected tests and test solved badly, respectively, with test statistics $U = 1.0$, $Sig < 0.001$. In both variants we must reject the null hypothesis and assume that the fact of solving tests in an unexpected manner affects the test ordering in a significant way and, therefore, that the system is capable of adjusting itself to the user's progress.

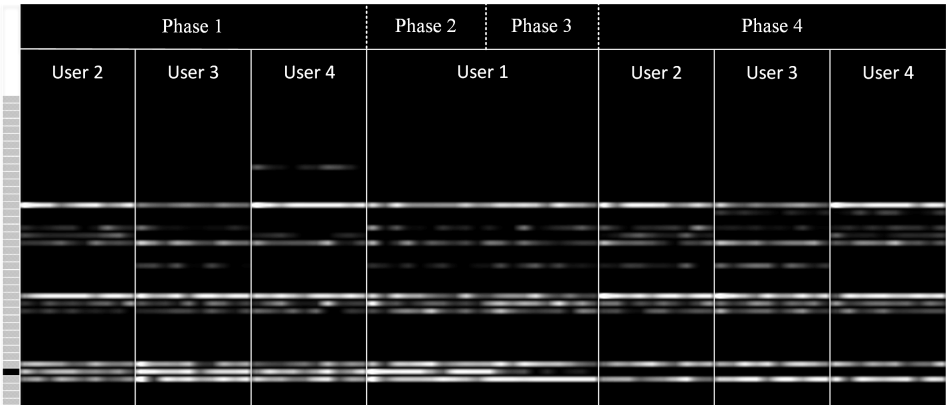



**Figure 5.9:** Inter-user influence. Indirect influence of User 1 on Users 2 - 4.

### 5.5.6 Experiment 4: Indirect Inter-user Influence

In this experiment we intended to show how the behavior of one user can affect a whole cluster of similar users in an indirect manner. A test world with $10^3$ tests and 4 users, with high degree of pairwise similarity ($> 0.85$) was generated. We choose a central user, with the highest average similarity to other users and label him user 1; others become user 2, user 3 and user 4. The experiment consists of the following steps:

1. Users 2 - 4 perform 100 queries, without solving tests (Fig. 5.9, Phase 1).
2. User 1 performs 100 queries without solving tests (Fig. 5.9, Phase 2).
3. The best common test pattern for all the Users is chosen.
4. User 1 performs 100 queries, each time solving badly the previously chosen pattern (Fig. 5.9, Phase 3).
5. Users 2 - 4 repeat the 100 queries, without solving tests (Fig. 5.9, Phase 4).

Note that during the run of this experiment only user 1 solves tests.

Fig. 5.9 shows the responses of the system. We can see how the test pattern marked with a black rectangle on the left side was completely removed from the suggested list of all the users after only one member of the user group solved it badly.

***Statistical Analysis***

**Goal.** Demonstrate that unexpected behavior of a user significantly affects the recommendation lists of closely similar users. We chose the Mann Whitney U test, as in Experiment 3.

**Procedure.** The statistical analysis in this experiment was similar to the one in Experiment 3. We generated 20 groups of 4 users and performed the full experiment 20 times. To every test pattern for every user we assigned the $RankShift$ value, which is the difference between the average rank of the test pattern in Phase 1 and Phase 4 and the Mode value. The Mode takes two values: *Affected* in case of the test pattern in question and *Unaffected* in every other case.

**Hypothesis.** Mann-Whitney U test.

**Dependent Variable.** *RankShift* is defined as $RankShift = \overline{Rank_{phase4}} - \overline{Rank_{phase1}}$, where $\overline{Rank_{phaseN}}$ is the average rank of the given test pattern in the phase $n$.

**Independent Variables.** *Mode* is defined as *Unaffected* for unaffected test patterns and *Affected* for tests solved by the User 1.

**Null Hypotheses.** $H_0$ the Mode variable does not influence the $RankShift$, $H_1$ the Mode variable does influence the $RankShift$.

**Results.** Using the Mann-Whitney U test we concluded that there was a statistically significant difference in rank changes between the Mode of the two groups, $U = 8.0$, $Sig < 0.01$. The average RankShift was $-0.08$ and 3.91 for the Unaffected and Affected group, respectively. We therefore must reject the null hypothesis and assume that the recommendation for test patterns solved by the central user was affected significantly for all the users involved in the experiment.

It is now obvious that similar users affect each other. However, this could cause unwanted results, if the behavior of the user *constantly* becomes dominated by the behavior of others. In order to examine this question we proposed Experiment 5.

### 5.5.7 Experiment 5: Fine-scale user clustering

The previous experiment answers the question about what happens if a subset of a group of similar users acts in an unexpected manner. Here we try to show a more difficult case when similar users act in contradictory ways. In other words, we explore to what degree the system can distinguish between users if their calculated similarity measure does not coincide with their actual behavior.

We assume that 10 similar users exist (pairwise similarity $> 0.80$), but behave in two distinct ways. First, there is a group of *Good Users* that solve the queried tests as expected, i.e. theoretically suitable tests are solved well and the unsuitable ones badly; second, there is a group of *Bad Users* that do the exact opposite. These are the phases in the experiment:

1. All users query test patterns, without solving, for the first half of the experiment.
2. The best common test pattern is chosen.
3. Good Users start solving well the test within the chosen test pattern, while Bad Users start solving them badly.

Fig. 5.10 is a visualization of the responses for: A) 1 Good User and 9 Bad Users; B) 3 Good Users and 7 Bad Users. As we can see, in spite of the high level of similarity, the system was capable of modifying its behavior in two opposite ways. All Good Users still receive the test pattern in question, while all Bad Users have it pushed off the first place on the list to a point at which it almost disappears. In both figures the test patterns affected are marked with a black rectangle on the left. This fine level split proves that the system can detect subgroups of users based on their behavior as well as static characteristics.

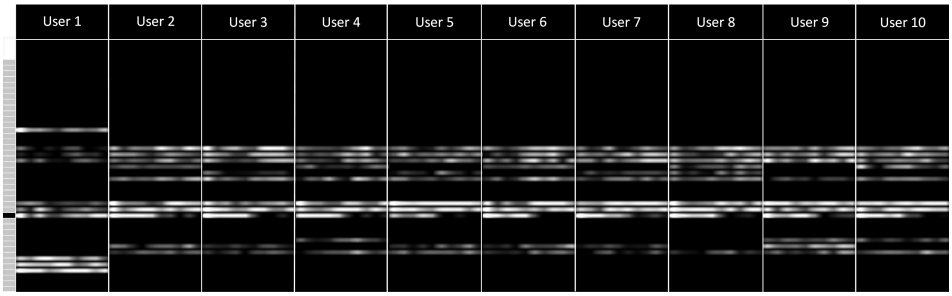
### 5.5.8 Experiment 6: Global Experiment

After analyzing the system in a series of isolated situations, in which the outputs were foreseeable and the inputs had direct consequences, we proceed to a global experiment. In this final experiment we would like to examine the performance of the system under a more realistic situation of a mixture of the aforementioned behaviors. We also allow users to evolve, change their behavior modes and attitudes, and we incorporate several physiological components such as boredom and curiosity.

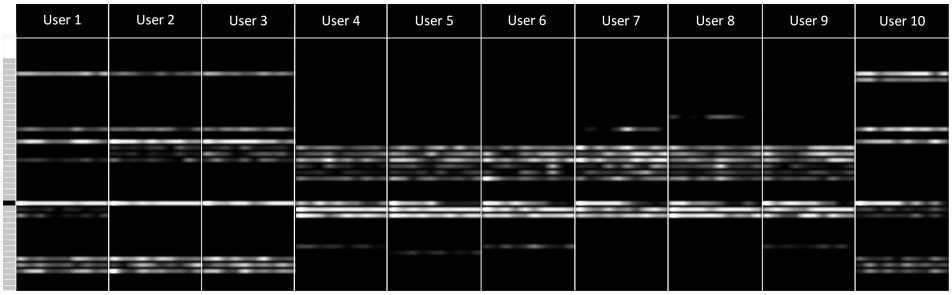
There is an inherent difficulty in performing a global, realistically simulated experiment. The more effort that is put into modeling natural user behavior, the more difficult it is to predict the outputs of the system, which, in turn, makes the statistical analysis less straightforward.

In order to accurately model the users, we need to introduce a new concept, *User Mentality*, which in our model has the following values: Passive, Neutral, UnexpectedGood, UnexpectedBad and Random. They can be understood as follows:

**Passive.** Users with this mentality are only querying for tests, but never solve them. It should be perceived as initial curiosity in the system, combined with a certain degree of timidity. Every user starts with Passive mentality.




**(a)** 1 Good User, 9 Bad Users




**(b)** 3 Good Users, 7 Bad Users

**Figure 5.10:** Fine-scale clustering. Subclustering of similar users under the condition of contradictory behavior

**Neutral.** These users solve all the tests according to the quality provided by the model. This means that they should have a neutral impact on the evolution of the system. Having said that, as we show in the Experiment 4, they can be influenced by other users. When a User chooses to evolve from the Passive mentality, it always changes first to Neutral.

**UnexpectedGood and UnexpectedBad.** These two mentalities are identical to those in Experiment 3. With a certain probability users with Neutral mentality can choose one Test Pattern as the objective of their unexpected behavior. A test from the recommendation rank 5 - 10 can be selected as the objective of UnexpectedGood mentality, and one from rank 1 - 3 can become the objective of an UnexpectedBad mentality. Each time a test is solved in an unexpected manner we increase or decrease the accumulated *Impact* the users had on it (+1 for UnexpectedGood solution, −1 for UnexpectedBad solution).

**Random.** The last mentality is random behavior. Neutral, UnexpectedGood and UnexpectedBad users can start and behave irrationally, solving tests in a random manner. This sort of mentality corresponds to bored or annoyed

users, as well as input errors. It serves as a way of establishing how well the system copes with input noise. Random users can return to behaving in a Neutral way.

The transitions between states are probabilistic and expressed in the evolution matrix $E_M$, (5.16). The probability of the transition from mentality $m_1$ to $m_2$ is in the the $m_1$-st row and $m_2$-nd column, $E_M[m_1][m_2]$. For instance: the probability for Neutral (2) to convert to UGood (3) is $E_M[2][3] = 0.01$.

$$E_M = \begin{array}{c} \\ Passive \\ Neutral \\ UGood \\ UBad \\ Random \end{array} \begin{array}{c} \begin{array}{ccccc} Passive & Neutral & UGood & UBad & Random \end{array} \\ \begin{pmatrix} - & 0.03 & 0 & 0 & 0.002 \\ 0 & - & 0.01 & 0.01 & 0.002 \\ 0 & 0 & - & 0 & 0.0001 \\ 0 & 0 & 0 & - & 0.0001 \\ 0 & 0.008 & 0 & 0 & - \end{pmatrix} \end{array} \tag{5.16}$$

***Statistical Analysis***

As the results of this experiment are too complex to visualize by the dna-graphs, we therefore only perform a statistical analysis.

**Goal.** Demonstrate that in a non-isolated, global experiment the recommendation list ranks are correlated with the behavior of users. We chose the Pearson Correlation $Pc$, as in Experiment 3.

**Procedure.** For each experiment run we generate a pool of $10^3$ tests under 46 tests patterns and $U$ users grouped in $C$ clusters. Users from within a cluster have pairwise similarity $> 0.8$, users from different clusters have pairwise similarity of $< 0.6$. In addition, we take the evolution matrix as $E_M/R$, where $R$ is referred to as *Randomness decrease*, i.e. the higher the $R$, the less probable the evolution, the less users behave in an unexpected manner and the lower the Impact value for each test pattern.

We have split this experiment into two variants. Variant i) *uniform*, in which the evolution of the entire population is uniform and determined by $R$; ii) *per-cluster* in which the evolution of each cluster of the population is determined by a different $R_C$. One experiment run consists of $10^3$ iterations. A full iteration is composed of: i) selecting a random user from $U$; ii) performing a query for tests for the selected user; iii) handling the query results according to the selected user's mentality; iv) evolving all users in $U$ with probabilities given by $E_M/R$ (*uniform* variant) or $E_M/R_C$ (*per-cluster* variant) v) saving the ranks of all the test patterns queries and updating the impacts, if the solution was performed in an unexpected manner.

Each experiment run is repeated five times for the same $U$, $C$ and $R$. In our experiment we took $U = \{10, 25, 50, 100\}$, $C = \{1, 2, 3, 5\}$, $R = \{1, 10, 50, 100\}$ and $R_C \in \{1, 10, 50, 100\}$, which resulted in 64 combinations, $5 \times 64 = 320$ experimental runs for each variant and 640 experimental runs total. The statistical analysis is reported for all the aggregated results $Pc$, as well as, grouped for each $R$: $Pc[R = 1]$, $Pc[R = 10]$, $Pc[R = 50]$, $Pc[R = 100]$ for both variants.

**Hypothesis.** Pearson Correlation $Pc$ test.

**Dependent Variable.** *RankShift* is defined as $RankShift = \overline{Rank_{0-50}} - \overline{Rank_{950-1000}}$. It is the difference in average rank of a given test pattern in the first 5% and the last 5% iterations.

**Independent Variables.** *Impact* is calculated as an absolute sum of all the unexpected solutions, tracked independently for each test pattern.

**Null Hypotheses.** $H_0$ the Impact variable does not influence the RankShift, $H_1$ the Impact variable does influence the RankShift.

**Results.** In the *uniform* variant the aggregated results, as well as each of the results grouped by Randomness decrease, have been shown significant, see Table 5.4. We must therefore conclude that Impact variable influences RankShift in an expected manner in the *uniform* execution variant, confirming Experiments 3 and 4 in the global setup. In addition, we see that the Impact is inversely proportional to RankShift and decreasing with $R$.

If the evolution of the system is not uniform (*per-cluster*) we also obtain statistically significant results, see Table 5.4. This means that even if the evolution, and consequently the behavior of the users of the system, varies from cluster to cluster we still obtain statistically significant recommendation list changes, confirming Experiments 3 and 5 in the global setup. The Impact variable is again inversely proportional to RankShift and decreasing with $R$

We therefore conclude that our recommendation system works as expected under all conditions.

Aside from the main statistical study we performed an analysis of the distribution of the Impact variable, grouped by $R$ (Fig. 5.11). We chose $R$ as the grouping variable, as it is the main factor in the generation of User Mentalities. We can clearly see the mean in all the cases remains very close to 0, which is desirable and expected. The user pool must behave unexpectedly in a small portion of the iterations, rather than a majority. The User population generated with $R = 1$ (Fig. 5.11a) is the most quickly evolving and, in consequence, generates the most impacts. The standard deviation of the Impact variable distribution is 33.126. As we proceed towards slowly evolving populations, the standard deviation decreases,

**Table 5.4:** Experiment 6. Pearson Correlation results.

| | Uniform | | Per-cluster | |
|---|---|---|---|---|
| Parameter | $Pc$ | $Sig$ | $Pc$ | $Sig$ |
| $Pc$ | $-0.469^{**}$ | $< 0.00$ | $-0.352^{**}$ | $< 0.00$ |
| $Pc[R = 1]$ | $-0.652^{**}$ | $< 0.00$ | $-0.503^{**}$ | $< 0.00$ |
| $Pc[R = 10]$ | $-0.451^{**}$ | $< 0.00$ | $-0.290^{**}$ | $< 0.00$ |
| $Pc[R = 50]$ | $-0.146^{**}$ | $< 0.00$ | $-0.168^{**}$ | $< 0.00$ |
| $Pc[R = 100]$ | $-0.094^{**}$ | $< 0.00$ | $-0.143^{**}$ | $< 0.00$ |

$^{**}$ Correlation is significant at the 0.01 level (2-tailed).

$^{*}$ Correlation is significant at the 0.05 level (2-tailed).

reaching 2.014 in the Fig. 5.11d. This suggests that our model of user and user mentality generation is reasonable and realistic.

## 5.6 Conclusions and Future Work

In this work we have shown that our proposal proved to be appropriate for the problem in question. ACO-based suggestion building is efficient starting from the zero-knowledge stage, achieving precision of 80% - 90%. This indicates that any user approaching the system would immediately receive quality suggestions, as demonstrated in Experiment 5.5.2. In later experiments we have shown conclusively that the precision only improves from that point.

The inter-user indirect influence is akin to ant communication in ACO models and is, therefore, deemed the most suitable metaphor. An important property of ACO is the ability to reconverge if the desired conditions arise. Naturally, the users' behavior can evolve gradually, but it can also change abruptly at any point in time. Such an event causes a quick response in the form of a reevaluation and reconvergence with system-wide consequences. Experiment 3 demonstrates precisely this situation. In addition, Experiment 4 shows the wider consequences of such an event. Here a group of users starts receiving significantly different suggestions based purely on the actions of others.

Apart from its adaptability the suggested model has the potential to distinguish fine differences between apparently similar users. This means that any groups that may have formed over the evolution of the system can be very quickly split into sub-groups if the conditions so dictate. Our proposed distinction between the static and dynamic score components facilitates this process. This fine scale similarity reevaluation was shown in Experiment 5.

Finally, we performed a global experiment (Experiment 6), the largest in terms of iterations and user-pools, which was designed to integrate all the partial views

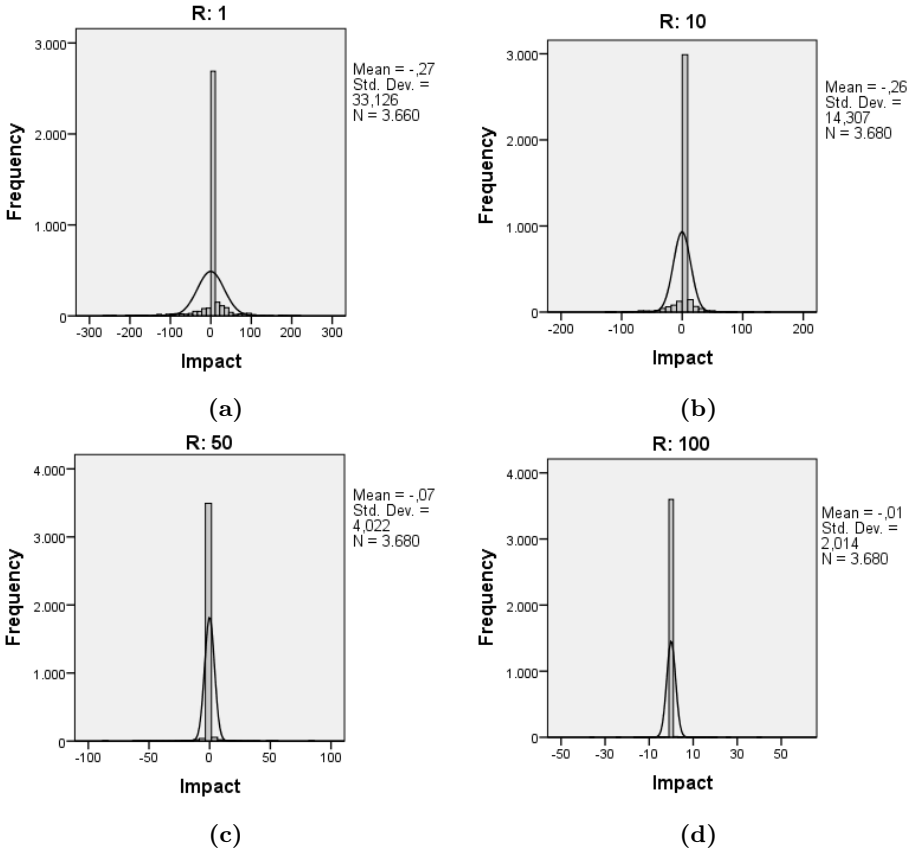



**Figure 5.11:** The distribution of the Impact. Grouped by $R$

provided by Experiments 1 - 5. In this experiment we maintain a dynamic, evolving user population with different behaviors, ranging from Passive to Random, with the full range of disabilities, and in all possible physical states. We consider that in this way we achieve an acceptably accurate representation of a set of real-life users. From Experiment 6 we conclude that our model reacts as expected to user actions and generates statistically relevant recommendation lists.

Several challenges constitute our future work. Currently, we are working on the integration of the solution here presented into HABITAT, the application developed for the treatment of individuals with ABI in order to enable them to take full advantage of our findings. This would involve a step forward in their treatment and reduce the need for specialist attention. This integration must be carefully planned, not because of technological issues, but to control its impact on the evo-

lution of system users. Therefore, a careful deployment plan must be designed to maximize the acceptability of the proposal in clinical terms.

In addition, from a purely technological point of view, several issues must be resolved to enable interaction with these individuals. For instance, it would be necessary to design proper facilities to guide them in the query processes, so that they know the real meaning of the minimum and maximum scores. It would also be interesting to model additional factors, such as daily progress and fatigue levels, so that the system can encourage the users to either continue with additional tests or to rest.

Finally, in relation to the algorithm presented here, we are evaluating new metaheuristics that take into account the stress of the person with ABI while they are using the system. These heuristics are oriented towards providing full support for the personal part of the relearning process. They would also consider additional inputs, not only about the users' cognitive state, but also about their physical conditions while solving the tests.

## Acknowledgments

This research has been funded by the Spanish Ministry of Economy and Competitiveness and by the FEDER funds of the EU under the project Grant CreateWorlds (TIN2010-20488) and insPIre (TIN2012-34003). Kamil Krynicki is supported by the FPI fellowship from Universitat Politècnica de València,.

# Chapter 6

# A non-Hybrid Ant Colony Optimization Heuristic for Convergence Quality

Kamil Krynicki[†] Michael E. Houle[*] Javier Jaen[†]

[†] ISSI Research Group, Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

[*] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

(2015), In 2015 IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC 2015

## Abstract

Ant Colony Optimization has proven to be an important optimization technique. It has provided a solid base for solving classical computational problems, networks routing problems and many others. Nonetheless, algorithms within the Ant Colony metaheuristic have been shown to struggle to reach the global optimum of the search space, with only a few select ones guaranteed to reach it at all. On the other hand, Ant Colony-based hybrid solutions that address this issue suffer from either severely decreased efficiency or low scalability and are usually static

and custom-made, with only one particular use. In this paper we present a generic and robust solution to this problem, restricted rigorously to the Ant Colony Optimization paradigm, named Angry Ant Framework. It adds a new dimension — a dynamic, biologically-inspired pheromone stratification, which we hope can become the objective of further state-of-the-art research. We present a series of experiments to enable a discussion on the benefits provided by this new framework. In particular, we show that Angry Ant Framework increases the efficiency, while at the same time improving the flexibility, the adaptability and the scalability with a very low computational investment.

## 6.1 Introduction

The mimicking of nature has often led to important discoveries in science and technology. This is especially true in the field of computation, where an entire branch of research is focused on translating animal behavior into mathematical models, which later serve as basis for computational frameworks. One fruitful approach employs these concepts is the notion of a *swarm* of animals and consequently, the emerging *swarm intelligence* [109]. It shifts the complexity of the model from the behavior of a single animal, which in this case is treated very simplistically, sometimes even trivially, to the intra-animal interaction. An example of swarm intelligence is the *Ant Colony Optimization* (ACO) metaheuristic.

ACO is a computational technique for finding shortest paths in graphs, that is inspired by the behavior of ants [6] [7]. It has allowed some very efficient solutions to classical problems, such as the Traveling Salesman Problem or the Quadratic Assignment Problem, as well as other that can be represented as graphs with edge traveling cost and resources in nodes.

Although much effort has been dedicated to the topic of empirical and experimental studies of ACO, the theoretical analysis of convergence and optimality is yet lacking [110] [111]. Of the many ACO algorithms, only two (*Ant Colony System*, ACS [17] and *Max-Min Ant System*, MMAS [112]) have been proven to converge on a global optimum [113] [114], although even in these cases the convergence property is weak [111]. In practice, even though ACS and MMAS are guaranteed to eventually reach a global optimum, their convergence period may be impractically long.

Attempts at fine-tuning ACO have not provided any solution because of the excessive parametric sensitivity of ACO. An overly strong convergence process leads to stagnation in suboptimal solutions, while a weak one might not conclude, as the random component of the algorithm dominates. The survey [110] discusses in high detail the aforementioned problem. A step towards a possible solution has

been the hybridization of ACO with other methods, [115], resulting in what is collectively known as *Hybrid Ant Systems* or *HAS*.

*HAS* algorithms tend to produce high quality results, oftentimes outperforming their non-hybrid counterparts. However, they entail strict limitations, on which we elaborate in section 6.2, and tend to be custom-made for only one predetermined use. Even more importantly, within the spirit of the No Free Lunch Theorem [116] [117], they almost always sacrifice the strongest advantage of ACO metaheuristic — the decentralized nature and, in consequence, the scalability and the potential to work in distributed environments, such as P2P networks.

For these reasons we attempt to improve the convergence quality of ACO with a non-hybrid approach, strictly within the frames of the ACO paradigm. We opted for a non-hybrid approach to assure a dual benefit. First and foremost, we preserve all the qualities of ACO. We guarantee that our model can be applied to any problem ACO is applicable to. Second, if the restrictions on scalability or centralization are relaxed or waived our non–hybrid ACO can be hybridized, stacking the benefits and, possibly, improving the overall quality further.

Of the two known optimal ACO algorithms ACS is arguably the more popular. It has been used in countless derived hybrid systems and it has nearly become synonymous with the ACO metaheuristic. Thus, we choose it as the conceptual base algorithm for our algorithm. We claim that our model, which we refer to as *Angry Ant Framework* (AAF), helps with the convergence quality of ACS unconditionally. Moreover, AAF provides a new dimension to the ACO metaheuristic by extending the classical, single-value pheromone model with a dynamic multi-pheromone capability. We enable the pheromone model to adapt to a given problem space with more precision and flexibility and to reconverge out of local minima at any point in the evolution of the system. Even though our solution is based on ACS, AAF is designed with generality in mind. It can be transfered onto any non-hybrid ACO algorithm on one hand, and take part in any hybrid setup, on the other. Therefore our framework should be perceived as a complementary rather than competing approach.

The remainder of the paper is structured as follows. In section 6.2 we elaborate on *HAS* algorithms, their uses and drawbacks. In section 6.3 we present the mathematical formalization of the Angry Ant Framework. In section 6.4 we provide the results of the empirical research phase. We conclude with section 6.5, where we summarize the results obtained and outline the possible hybridizations of AAF.

## 6.2 Hybrid Ant Systems

ACS, alongside MMAS, have remained a de facto state-of-the-art of ACO. ACS, since it was proposed by Dorigo [17], served as a backbone for many hybrid models, all of which could be positively influenced by AAF. As we discussed in the previous section, a more efficient non-hybrid ACO algorithm is more widely beneficial than a custom-made hybrid algorithm.

There exist numerous hybridization techniques [118]. However, in the domain of *HAS* the spectrum of approaches narrows down to three families:

1. ACO with selective evolution
2. ACO with local search
3. ACO coupled with another heuristic

The most lightweight in terms of affecting the underlying ACO metaheristic is the idea of selective evolution. The overall algorithm is two-phase. In the first phase ACO is used to generate a number of independent solutions, while in the second phase another mechanism analyses the solutions and chooses a subset of them for the pheromone reinforcement. There is always a global overseer capable of radically switching the evolution of the system as soon as it encounters a better solution than the current minimum. Careful implementation of this idea has such a low impact on ACO that it could be argued that these techniques are, in fact, not an example of hybridization, but rather are at the core of ACO. Examples include k-Elitist Ant System [119], in which only the k-top solutions are reinforced, and the Rank Based Ant System [120], in which the reinforcement is weighted by the rank of the quality of the solution.

ACO has the ability to quickly find and explore promising regions in the search spaces, yet it tends to slow down the nearer it finds itself to the optimum. To aid this, a more disruptive variant of *HAS* was proposed [115]: ACO coupled with local search. The results produced by ACO are not only filtered but also postprocessed with a local search algorithm, thus improving the convergence rate. The drawback to this method is a great loss of generality and increased centralization. It is the most numerous family of *HAS* algorithms, used for problems such as Sequential Ordering Problem [121] and Quadratic Assignment Problem [122].

ACO can also be coupled with another heuristic, which can be done a multitude of ways. The most elaborate ones include parallel execution of two or more algorithms of distinct metaheuristics over the same search space, later proceeding to compare and contrast the results and perform result crossovers. These algorithms always require global daemons and are incompatible with the principal paradigm of ACO. However some simple preprocessing-postprocessing setups, such as *inward processing* are more relaxed. In the work [123] ACO selects the region of

the search space and another algorithm continues with fine-tuning. An interesting algorithm of this kind is Recursive ACO [124], in which different instances of ACO algorithms are executed recursively over smaller and smaller search spaces.

On the other hand our solution (AAF) is a non-hybrid one. We postulate that it surpasses the non-hybrid ACO algorithms in terms of and convergence quality at a cost of increased memory usage. It could help to advance the reported results of all the aforementioned works. Additionally, it perseveres the full scope of the applicability of ACO.

## 6.3 Angry Ant Framework

### 6.3.1 Overview

Traditional ACS uses a single pheromone value for trail marking. However, it has been shown that using multiple pheromone types benefits ACO [125] [44] [126] [77] and it also reflects more closely the actual behavior of real-life ants [127]. In the common vocabulary pheromone types are often referred to as *pheromone levels*.

The roles of multiple pheromone levels differ among algorithms. Two predominating approaches are: using two types of pheromone (attractive and repulsive, [125]) or pheromone-per-problem-class [44] [126] [77]. Such multi-pheromone solutions impose limitations: the number of pheromone levels must be established a priori, they require custom-made pheromone deposition and evaporation strategies as well as the state transition technique. In our approach we enable AAF to grow the number of pheromone levels $L$ as needed, starting at $L = 1$ in each node. We argue that allowing this form of dynamism preserves versatility of the algorithm, as there is no more need to externally establish the number of pheromone levels.

Rather than have preassigned roles, the pheromone levels are created with the sole purpose of improving efficiency and preventing ACO stagnation in local minima [110]. They enable independent ant explorations to take place, including cases in which ants in newly created level travel against the pheromone indications of the lower levels. The non-zero chance of generating a new pheromone levels is preserved during the execution of the algorithm, thus, the stagnation period is always finite. To the best of the authors' knowledge this form of dynamism in ACO has remained unexplored.

The dynamic level creation is limited to situations where the *entropy* of the pheromone information provided by existing pheromone is too high. The decision to tie the level creation with an entropic measure is based on biological and psychological grounds. If we think about one single node $r$ it is expected that, after a period of time, one edge dominates others in terms of pheromone value. However, in

sensitive regions of the search space or when the algorithm approaches a minimum there might be no strongly dominating edge, or the strongest pheromone might change back and forth. The lack of a clear–cut path is a state of high entropy, or chaos, which causes irritation in living beings. Accordingly, our model has been named *Angry Ant Framework*.

AAF builds upon the model provided by ACS. In place of the one $\tau_{rs}$ pheromone value per directed edge AAF uses an array of $\tau_{rs}^{(\ell)}$ values, of size $L$. The levels are contiguous, from $\ell = 1$ to $\ell = L$. For a node $r$, the $\tau_{rs}^{(\ell)}$ must be defined for every outgoing edge $s$ in its neighborhood, $N_G(r)$ and for every level $\ell$ present in $r$. The size of the pheromone array can change as needed. The equations that govern the pheromone arrays are presented in subsection 6.3.2.

At any given time, an ant is assigned a unique pheromone level. Ant $a^{(\ell)}$, that is being routed within the level $\ell$, uses its corresponding level pheromone values only (Fig 6.1, step $D_*$). The pairing between ant and level happens with the help of the *level assignment matrix* $\mathcal{L}(r)$ (Fig 6.1, step $A_*$) at the ant generation. $\mathcal{L}(r)$ is stored and maintained in every node $r$ as explained in the subsection 6.3.3. Ants are permitted to change their level and to create new levels under certain conditions, explained in subsection 6.3.4 (Fig 6.1, step $B_*$).

### 6.3.2 State Transition and Pheromone Evolution

The state transition and pheromone evolution models are almost identical to those in the ACS model. As in ACS the initial pheromone value is $\tau_0 \in \mathbb{R}_0^+$, the minimum is $\tau_{min} \in \mathbb{R}_0^+$ and the maximum is $\tau_{max} \in \mathbb{R}_0^+$. With the inclusion of the pheromone level concept the classical equations must be rewritten with $\tau^{(\ell)}$ in place of $\tau$. The use of various levels in the state transition is reflected in the step $(D_*)$ of Fig 6.1. Note that the cost $\eta$ is level-independent, as it is tied to the physical properties of the edge.

$$s = \text{argmax}_{u \in N_G(r)} \{\tau_{ru}^{(\ell)} \times \eta_{ru}^{\beta}\} \tag{6.1}$$

$$p_{rs} = \begin{cases} \frac{\tau_{rs}^{(\ell)} \times \eta_{rs}^{\beta}}{\sum_{z \in N_G(r)} \tau_{rz}^{(\ell)} \times \eta_{rz}^{\beta}} & \text{if } s \in N_G(r) \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

Similarly, the pheromone deposition and evaporation are limited to the pheromone level used by the ant $a_q^{(\ell)}$, as below:

$$\tau_{ru}^{(\ell)} \leftarrow (1 - \alpha) \cdot \tau_{ru}^{(\ell)} + \alpha \cdot \delta\tau \tag{6.3}$$

**Figure 6.1:** Angry Ant Framework high level pseudocode

```
1: INITIALIZE()
2: while i ≤ agent_max do
3:     i_node ← SELECT_NODE()
4:     level ← SELECT_LEVEL(L(i_node))                          ▷ (A*)
5:     LAUNCH_AGENT(i_node, level)
6:     PHEROMONE_EVAPORATE()
7:     i ← i + 1
8: end while
9: for all agent do
10:    solution ←GET_SOLUTION(i_node, query, level)
11:    goodness ←EVALUATE_SOLUTION(solution)
12:    PHEROMONE_DEPOSIT(solution, goodness)
13: end for

14: function GET_SOLUTION(i_node, query, level)
15:    res ← empty_set
16:    node ← i_node
17:    path ← empty_list
18:    while not stop_condition do
19:        res ∪ QUERY_RESOURCES(node, query)
20:        if agent_is_irritated then                           ▷ (B*)
21:            level ← level + 1                                ▷ with prob. ι (C*)
22:        end if
23:        node ←STATE_TRANSITION(node, level)                  ▷ (D*)
24:        path ∪ node
25:    end while
26:    return solution(path, res)
27: end function
```

$$\tau_{ru}^{(\ell)} \leftarrow (1-\rho)\cdot\tau_{ru}^{(\ell)} + \rho\cdot\gamma\cdot\max_{z\in N_G(r)}\tau_{rz}^{(\ell)} \tag{6.4}$$

The number of ants depositing pheromone is always identical to ACS. As long as there is just one pheromone level, $L = 1$, the behavior of this model is indistinguishable from ACS.

### 6.3.3 Level Assignment Matrix

The assigning of ants to pheromone levels is performed at the ant generation, with the help of *level assignment matrix* $\mathcal{L}(r)$. Every node $r$ stores a matrix $\mathcal{L}_{L\times Q}(r)$ of dimensions $L \times Q$. The rows of $\mathcal{L}(r)$ correspond to the levels of pheromone present in $r$, while the columns of $\mathcal{L}$ (written as $\mathcal{L}_q$) correspond to all the queries $q$ that originated at the node $r$. The matrix is initialized as $1 \times 1$: $\mathcal{L}_{1\times 1}(r) = I_0$, and can grow dynamically as needed.

When a query $q$ is launched from a node $r$ one generic ant $a_q$ is created. Based on the information present in $\mathcal{L}(r)$ the node decides to which level should the ant be assigned (Fig 6.1, step $(A_*)$). The probability of assigning the ant $a_q$ to any level $\ell$ is given by:

$$p[a_q \to a_q^{(\ell)}] = \frac{\mathcal{L}_{\ell q}(r)}{\sum_{\ell' \in \{1..L\}} \mathcal{L}_{\ell' q}(r)} \tag{6.5}$$

Equation 6.5 is evaluated for every level present in $r$. It is possible to choose more than one level or to choose none. In the first case multiple ants are sent to the chosen levels independently and the final response to the query is the sum of all the partial responses. This approach causes the number of ants per query to fluctuate.

The probability of not selecting any level is equal to:

$$\prod_{\ell \in \{1..L\}} (1 - p[a_q \to a_q^{(\ell)}]) \tag{6.6}$$

In such a case the best possible level is chosen:

$$\ell = \text{argmax}_{\ell' \in \{1..L\}} \{\mathcal{L}_{\ell' q}(r)\} \tag{6.7}$$

The information stored in $\mathcal{L}(r)$ is updated by returning ants. When an ant $a_q^{(\ell)}$ returns to its emitting node $r$, yielding results evaluated at goodness $\delta\tau$, it updates $\mathcal{L}(r)$ by first applying (6.8), followed by (6.9):

$$\mathcal{L}_{\ell q}(r) \leftarrow (1 - \alpha_*) \cdot \mathcal{L}_{\ell q}(r) + \alpha_* \cdot \delta\tau \tag{6.8}$$

$$\mathcal{L}_q(r) \leftarrow (1 - \rho_*) \cdot \mathcal{L}_q(r) \tag{6.9}$$

where:

$\alpha_* \in \mathbb{R}_0^1$ is the deposition parameter, analogous to the $\alpha$ parameter of the classical equation.

$\rho_* \in \mathbb{R}_0^1$ is the evaporation parameter, analogous to the $\rho$ parameter of the classical equation.

If at any point either $\ell > L$ (the returning ant has raised a level during its life span) or $q > Q$ (the first time a query is launched in the node), then $\mathcal{L}$ is expanded with rows and columns of zeros to accommodate the required new concepts:

$$\mathcal{L}_{\ell \times q} = (I_{\ell \times L} \times \mathcal{L}_{L \times Q}) \times I_{Q \times q} \tag{6.10}$$

where:

$I_{m \times n}$ is a binary matrix of dimensions $m \times n$, with ones on its main diagonal and zeros elsewhere

### 6.3.4 Ant irritation

An ant $a_q^{(\ell)}$ in the node $r$ will become irritated with the probability of $p_{irr}^{(\ell)}(r)$ (Fig 6.1, step $(B_*)$). The value $p_{irr}^{(\ell)}(r)$ is the central element of the pheromone-level dynamism, it allows the creation of new levels. If $p_{irr}^{(\ell)}(r) = 0$ is assumed AAF is reduced to ACS.

In order to easily explain how $p_{irr}^{(\ell)}(r)$ is calculated we first define a preliminary concept, namely the *pheromone sum*, the sum of all the pheromone values of all the outgoing edges on the level $\ell$ from the node $r$. It is written as $\tau_{sum}^{(\ell)}(r)$:

$$\tau_{sum}^{(\ell)}(r) = \sum_{s \in N_G(r)} \tau_{rs}^{(\ell)} \tag{6.11}$$

With that:

$$p_{irr}^{(\ell)}(r) = \iota \times \underbrace{\frac{1}{\tau_{max}}}_{\text{(a)}} \times \underbrace{\frac{\tau_{sum}^{(\ell)}(r)}{deg(r)}}_{\text{(b)}} \times \underbrace{\times \sum_{z \in N_G(r)} \frac{\tau_{rz}^{(\ell)}}{\tau_{sum}^{(\ell)}(r)} \ln \frac{\tau_{sum}^{(\ell)}(r)}{\tau_{rz}^{(\ell)}}}_{\text{(c)}} \tag{6.12}$$

where:

$\iota \in \mathbb{R}_0^1$ is a parameter that can be used to regulate the strength of the irritation. If $\iota = 0$ then $p_{irr}^{(\ell)}(r) = 0$ (Fig. 6.1, step $(C_*)$). We can express explicitly the value of $\iota$ by naming the algorithm $\iota-$AAF.

The components of the (6.12) are the following:

**(a)** is the normalization factor based on the maximum pheromone value $\tau_{max}$; to guarantee $p_{irr}^{(\ell)}(r) \in \mathbb{R}_0^1$.

**(b)** is the average pheromone per link in the node $r$. The higher the average amount of pheromone, the more *mature* the node should be considered. This reduces greatly the ants' irritation with *fresh nodes* that have very high entropy, due to identical pheromone value, $\tau_0$ on all the edges, but very low maturity.

**(c)** is the entropy of the pheromone distribution in the node $r$ on the level $\ell$

We include two versions of the same equation. (6.12) is the clearest form, which serves for explanations, while (6.13) is its simplified version. We use in our implementation for experimental purposes. Note the symmetry between the $\tau_{sum}^{(\ell)}(r)$ and $\tau_{rz}^{(\ell)}$.

$$\begin{aligned} p_{irr}^{(\ell)}(r) = & \frac{\iota}{\tau_{max} \times deg(r)} \times \tau_{sum}^{(\ell)}(r) \times \ln(\tau_{sum}^{(\ell)}(r) - \\ & - \frac{\iota}{\tau_{max} \times deg(r)} \times \sum_{z \in N_G(r)} \tau_{rz}^{(\ell)} \times \ln(\tau_{rz}^{(\ell)}) \end{aligned} \tag{6.13}$$

In the search phase of the ant, the probability $p_{irr}^{(\ell)}(r)$ is checked against a uniform random variable $P$ upon transitioning into a node. If $P > p_{irr}^{(\ell)}(r)$ the ant ascends one pheromone level:

$$a_q^{(\ell)} \to \begin{cases} a_q^{(\ell+1)} & \text{if } P > p_{irr}^{(\ell)}(r) \\ a_q^{(\ell)} & \text{otherwise} \end{cases} \tag{6.14}$$

If at any point of the routing process the ant $a_q^{(\ell)}$ finds itself in a node that does not contain the pheromone level $\ell$, the required level will be created and all the outgoing links within it will be initialized with the value $\tau_0$. This will be referred to as *pheromone split*. At the same time the level assignment matrix is appropriately extended, as in (6.10).

## 6.4 Experimental results

In our experiments we attempt to demonstrate three main points. First, we analyze the convergence rate of the each respective algorithm. Second, we show that our algorithm performs better than ACS in terms of the ability to find minima and finally, we take a look at the computational cost of As long as there is just one pheromone level, $L = 1$, the behavior of this model is indistinguishable from ACS. . In order to preserve maximum generality and validity of our experiments we focus in study on the generic problem of resource querying in graphs. Good performance in resource querying is an indication that the algorithm should have the same effect on different classical problems, as many of them can be modeled as graphs with resources.

### 6.4.1 Experimental procedure

In the following experiments we generate a number of resources, distribute them in a graph $G$ (a 2-dimensional toroid) of $N = 32 \times 32 = 1024$ nodes and perform $4 \times 10^6$ resource queries ($1 \times 10^6$ in the computational cost experiments).

The quality measure will be Hop per Hit $HpH$: the ratio of steps taken to resources obtained. It should be seen as the average cost in terms of transitions between nodes of obtaining one resource. In case of not finding any resources, $Hit = 0$, we take $HpH = 2 \times TTL$. The values of parameters used in the execution and experimentation are summarized in table 6.1. ACO parameters are at standard values. The parameters $\alpha_*$ and $\rho_*$ have been set to match the analogous values of $\alpha$ and $\rho$. The initial state of the Level Assignment Matrix $I_0$ is by definition a $1 \times 1$ matrix, we chose a small value to easily avoid division by zero in (6.5).

The parameters of the experiments reflect our hardware and time limitations. The workstation used in the experiment execution is a $64bit$, Intel Core $i5$ $540m$, $8Gb$ Ram. A single execution generates about $70MB$ of raw data. Each plot is an average of 10 independent executions over different datasets, three of which ($kry1024c1a$, $kry1024c1b$ and $kry1024c1c$) we make available at [128], for anyone to use. The remaining seven are named $rand$: from $rand\ a*$ to $rand\ g*$. We plot 5 graphs for each figure.

**Table 6.1:** Execution parameters

| ACO | | |
|---|---|---|
| Parameter | Interpretation | Value |
| $q_0$ | Weight of exploiting vs. exploring strategy | 0.80 |
| $\alpha$ | Pheromone deposition parameter | 0.07 |
| $\rho$ | Pheromone evaporation parameter | 0.10 |
| $\beta$ | Weight of link costs | 1.00 |
| $\gamma$ | Weight of evaporation | 0.02 |
| $\tau_{min}$ | Minimum pheromone level | 0.001 |
| $\tau_{max}$ | Maximum pheromone level | 1.000 |
| $\tau_0$ | Initial pheromone level | 0.009 |
| AAF | | |
| Parameter | Interpretation | Value |
| $I_0$ | $\mathcal{L}(r)$ initial value | [0.01] |
| $\alpha_*$ | $\mathcal{L}(r)$ deposition parameter | 0.07 |
| $\rho_*$ | $\mathcal{L}(r)$ evaporation parameter | 0.10 |
| Experiments | | |
| Parameter | Interpretation | Value |
| $TTL$ | Hop-based stop condition, Time to Live | 10 |
| $R_{max}$ | Hit-based stop condition | 10 |
| $R_{min}$ | Minimum Hit to consider a query successful | 5 |
| $N$ | Amount of the nodes | 1024 |

### 6.4.2 Resource distribution

The distribution of the resource queries is uniform: the probability that the node $n$ in launches a resource query is $1/N$.

Every node $n$ can hold $n_r$ resources. The exact distribution of resources is not crucial. It affects such outcomes as the final convergence value, but the relative position of the algorithms remains unaffected, as we conclude in [12]. We create a setup in which, following the suggestions of [54], there are relatively rare resource-rich nodes ($n_r = 10$), somewhat common normal nodes ($n_r = 1$) and a great majority of nodes without resources ($n_r = 0$). The distribution of $n_r$, follows a probabilistic model:

$$p[n_r = 0] = \frac{44}{50},\ p[n_r = 1] = \frac{5}{50},\ p[n_r = 10] = \frac{1}{50}$$

We execute the algorithm with the $\iota$ parameter ranging from $\iota = 0.00$ (equivalent to ACS) to $\iota = 1.00$, over three intermediate values: $\iota = 0.25$, $\iota = 0.50$ and $\iota = 0.75$. Our goal is to show that increasing values of $\iota$ provide increasing benefits, that is:

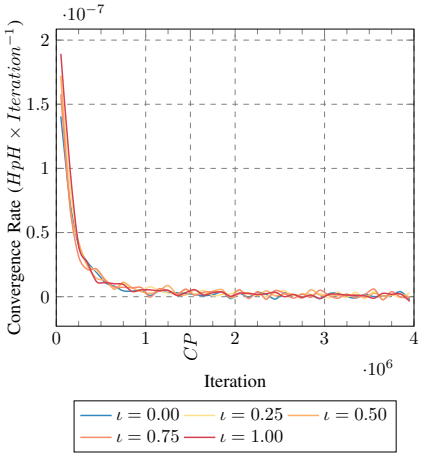



**Figure 6.2:** Convergence Rate. Average from datasets $kry1024c1a$, $kry1024c1b$, $kry1024c1c$ and $rand\ a*$ - $rand\ g*$. More is better.

$\iota_1 > \iota_2 \iff \iota_1-$AAF is better, in terms of $HpH$, than $\iota_2-$AAF. Most crucially, that $1.00-$AAF is better that $0.00-$AAF (ACS).

### 6.4.3 Results

First we take a look at the evolution of the algorithms' convergence rate, expressed as the ratio of the change of the chosen quality measure ($HpH$) and the iteration span in which the change occurred, written as $HpH \times Iteration^{-1}$. In Fig. 6.2 we present the full evolution of the convergence rate. There is low variability among the execution variants, yet $\iota = 0.00$ tends to be slower $\iota = 1.00$. We can also conclude that regardless of the variant the convergence rate starts oscillating around 0 at the iteration $1.5 \times 10^6$. This is the definition of arriving at the convergence point ($CP$). We can, therefore, conclude that AAF and ACS do not differ in terms of convergence rate.

The most important measure for any graph search is its effectiveness, or convergence quality, which is the value of the chosen quality metric ($HpH$ in our case) after the convergence point ($CP$). The full results are presented in Tables 6.2 and 6.3. The graphical representation Fig 6.3 we can observe the evolution of the $HpH$ value for the five variants of the execution of the algorithm. From there we conclude that the higher the irritation value ($\iota$) the lower the overall $HpH$. Moreover, having demonstrated that the all algorithms converged successfully, we claim that $0.00-$AAF (ACS) variant finds local minima worse than $1.00-$AAF ($0.65\ HpH$ to $0.59\ HpH$ respectively). From this analysis we cannot assume that the global minimum was reached, yet the AAF with $\iota = 1.00$ is on average 11%

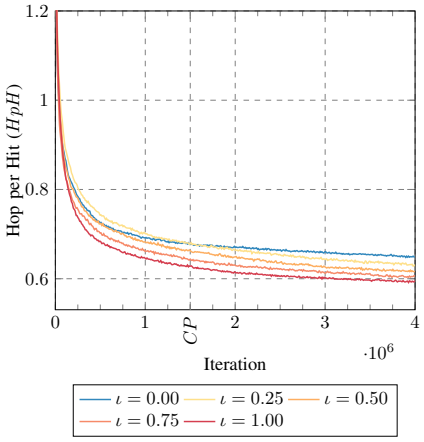



**Figure 6.3:** Convergence Quality. Average from datasets *kry*1024*c*1*a*, *kry*1024*c*1*b*, *kry*1024*c*1*c* and *rand a∗* - *rand g∗*. Less is better.

better. In the three standard datasets (*kry*1024*c*1*a*, *kry*1024*c*1*b* and *kry*1024*c*1*c*) AAF surpasses ACS by 5.3%, 14.5% and 22.3% respectively. The $\iota = 1.00$ (AAF) variant achieves the best average convergence in terms of $HpH$ (0.59), the shortest paths in terms of $Hop$ (5.8) as well as the best relative improvement in both of the aforementioned metrics. The $\iota = 0.75$ (AAF) variant scores the highest in terms of $Hit$ (11.3). On the other hand, $\iota = 0.00$ (ACS) scores, on average, the worst in terms of $HpH$ and $Hit$ as well as it improves the least throughout the experiment.

It is interesting to note AAF continues improving throughout the totality of the experiment, even though the convergence rate drops to a near zero. This is caused by the fact that it retains both: a non-zero chance of performing a random level creation and the capacity to reconverge quickly if a newly found path yield better results. This suggests that AAF is much more efficient in avoiding stagnation in local minima.

Having shown that AAF surpasses ACS in terms of convergence quality, we take a look at the computation- and memory-related costs of this benefit.

One way of comparing the computational cost is by means of the Ant per Query ($ApQ$) factor. ACS always uses a fixed number of ants per query, typically $ApQ = 1$. In case of AAF this value fluctuates, as shown in Fig 6.4. The irritation value $\iota$ is proportional to the $ApQ$ of each respective AAF variant. $1.00-$AAF reaches $ApQ = 1.16$ and then slowly continues towards $ApQ = 1.14$. This suggests that we experienced from 14% to 16% increase in ant traffic with respect to ACS ($ApQ = 1.00$), which should sublineally impact the computational cost, depending heavily on the hardware and implementation details. In our experiments: $1\% - 2\%$ CPU time increase.

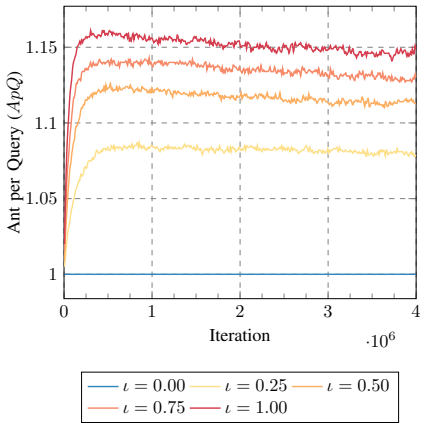



**Figure 6.4:** Ant per Query. Average from datasets *kry*1024*c*1*a*, *kry*1024*c*1*b*, *kry*1024*c*1*c* and *rand a∗* - *rand g∗*.

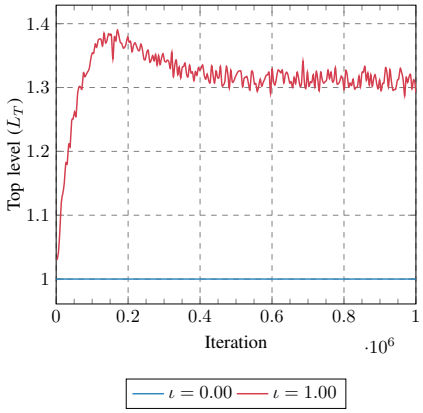



**Figure 6.5:** Top level used by AAF. Graph-wide average. Average from datasets *kry*1024*c*1*a*, *kry*1024*c*1*b*, *kry*1024*c*1*c* and *rand a∗* - *rand g∗*. Less is better.

Finally, in order to conveniently estimate the memory usage overhead we employ the concept of average *Top Level* ($L_T$) used. $L_T$ is the graph-wide average of the maximum level used by any ant in a given iteration. From Fig. 6.5 ($\iota = 0.00$ and $\iota = 1.00$ only, for clarity) we conclude that in the initial, exploration-heavy phase of AAF the average level reaches $L_T = 1.38$. It then begins to decrease and oscillates above $L_T = 1.3$, which reflects the pruning of unused paths. This translates to a slightly increased memory usage. In our *Java 7* implementation: 250.04*Mb* for ACS and 254.03*Mb* for AAF on average, measured with *VisualVM 1.3.6*.

## 6.5 Discussion and future works

In this paper we have shown that our *Angry Ant Framework*, with its dynamic multilevel approach, is an effective and novel solution in the field Ant Colony Optimization. We have demonstrated in section 6.4 that we surpass ACS in the metric of $HpH$ without the need for any hybrid approach. Worth mentioning is the fact that AAF with $\iota = 1.00$ achieves the shortest paths, which is an interesting property to explore. The overall cost of the AAF algorithms is always sublinealy proportional to the results obtained.

Moreover, we have enabled a new dimension of the ACO-based algorithms, namely the *irritation probability function* $p_{irr}^{(\ell)}(r)$. In our AAF implementation we attempted to optimize the use of pheromone levels versus the quality of the results, however there is a trade–off possibility: to sacrifice a percentage of quality to achieve better scalability or, on the contrary, increase memory consumption to push the quality even higher, depending on the current needs. We expect further developments with AAF to include the usage of different irritation functions and other ACO algorithms as a base.

In section 6.2 we presented three feasible hybridization techniques involving ACO (*HAS*): ACO with selective evolution, ACO with local search and ACO coupled with another heuristic. AAF is fully compatible with the first two *HAS* methods. In the third case the substitution of ACS with AAF should be subject to a careful study, yet intuitively the improvement is to be expected.

AAF, because of its dynamic multilevel pheromone mechanism will also be useful for the resolution of problems involving multiclass resource querying. The application of AAF to this category of problems will be the focus of our future work.

## 6.6 Acknowledgments

Kamil Krynicki is supported by a FPI fellowship from the Universitat Politècnica de València, Spain, reference number 3117. This work received financial support from the Spanish Ministry of Education under the National Strategic Program of Research and Project TIN2010-20488 and from the National Institute of Informatics, Tokyo, Japan.

**Table 6.2:** Numerical results (1)

| Dataset | Hit $t_0$ | Hit $t_m$ | Hit | Hop $t_0$ | Hop $t_m$ | Hop | HpH $t_0$ | HpH $t_m$ | HpH |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $\iota = 0.00$ | | | | | |
| kry1024c100a | 7.2 | 9.9 | 37% | 6.1 | 5.6 | -7% | 0.84 | 0.57 | -32% |
| kry1024c100b | 6.6 | 9.8 | 48% | 6.4 | 6.2 | -4% | 0.97 | 0.63 | -35% |
| kry1024c100c | 5.2 | 8.5 | 62% | 6.4 | 6.1 | -5% | 1.23 | 0.72 | -41% |
| rand c100a* | 5.7 | 8.8 | 54% | 6.5 | 6.2 | -4% | 1.14 | 0.71 | -38% |
| rand c100b* | 6.5 | 9.1 | 39% | 6.1 | 5.7 | -6% | 0.93 | 0.63 | -33% |
| rand c100c* | 6.3 | 9.3 | 48% | 6.3 | 6.0 | -5% | 1.01 | 0.65 | -36% |
| rand c100d* | 7.4 | 9.5 | 28% | 6.1 | 6.0 | -3% | 0.83 | 0.63 | -24% |
| rand c100e* | 6.1 | 9.7 | 59% | 6.6 | 6.2 | -6% | 1.08 | 0.64 | -41% |
| rand c100f* | 5.8 | 9.3 | 61% | 6.3 | 6.1 | -4% | 1.10 | 0.66 | -40% |
| rand c100g* | 6.4 | 9.1 | 43% | 6.4 | 6.2 | -2% | 1.00 | 0.68 | -32% |
| Average | 6.3 | 9.3 | 48% | 6.3 | 6.0 | -5% | 1.01 | 0.65 | -35% |
| | | | | $\iota = 0.25$ | | | | | |
| kry1024c100a | 8.0 | 10.8 | 34% | 6.8 | 6.1 | -10% | 0.85 | 0.57 | -33% |
| kry1024c100b | 7.6 | 10.7 | 42% | 6.8 | 6.1 | -9% | 0.90 | 0.57 | -36% |
| kry1024c100c | 6.1 | 10.6 | 73% | 6.9 | 6.5 | -6% | 1.13 | 0.62 | -45% |
| rand c100a* | 5.6 | 9.4 | 67% | 7.0 | 6.7 | -4% | 1.24 | 0.71 | -43% |
| rand c100b* | 6.2 | 10.0 | 61% | 6.8 | 6.7 | -1% | 1.10 | 0.67 | -39% |
| rand c100c* | 6.5 | 10.5 | 60% | 6.7 | 6.5 | -3% | 1.03 | 0.62 | -39% |
| rand c100d* | 6.1 | 10.2 | 66% | 6.4 | 6.4 | -1% | 1.05 | 0.63 | -40% |
| rand c100e* | 6.7 | 10.4 | 56% | 7.2 | 6.8 | -6% | 1.08 | 0.65 | -40% |
| rand c100f* | 6.4 | 10.5 | 64% | 6.7 | 6.7 | 0% | 1.06 | 0.64 | -39% |
| rand c100g* | 5.8 | 10.6 | 81% | 7.1 | 6.7 | -5% | 1.21 | 0.64 | -47% |
| Average | 6.5 | 10.4 | 60% | 6.8 | 6.5 | -5% | 1.06 | 0.63 | -40% |
| | | | | $\iota = 0.50$ | | | | | |
| kry1024c100a | 8.3 | 11.5 | 39% | 6.6 | 6.3 | -4% | 0.80 | 0.55 | -31% |
| kry1024c100b | 7.3 | 11.3 | 55% | 6.5 | 6.4 | -1% | 0.89 | 0.57 | -36% |
| kry1024c100c | 6.9 | 11.1 | 62% | 6.9 | 6.9 | -1% | 1.01 | 0.62 | -38% |
| rand c100a* | 6.4 | 11.0 | 72% | 7.1 | 6.8 | -4% | 1.12 | 0.62 | -44% |
| rand c100b* | 6.9 | 11.1 | 61% | 7.3 | 7.1 | -3% | 1.06 | 0.64 | -40% |
| rand c100c* | 7.6 | 11.4 | 49% | 7.4 | 7.2 | -3% | 0.97 | 0.63 | -35% |
| rand c100d* | 5.7 | 11.0 | 91% | 7.2 | 7.3 | 1% | 1.27 | 0.67 | -47% |
| rand c100e* | 8.1 | 11.2 | 38% | 6.9 | 6.5 | -6% | 0.85 | 0.58 | -32% |
| rand c100f* | 6.3 | 10.8 | 73% | 7.2 | 7.2 | 0% | 1.16 | 0.67 | -42% |
| rand c100g* | 6.1 | 10.5 | 72% | 6.8 | 6.6 | -3% | 1.11 | 0.62 | -44% |
| Average | 7.0 | 11.1 | 61% | 7.0 | 6.8 | -2% | 1.02 | 0.62 | -39% |

$t_0$: average of the initial $10^4$ iterations
$t_m$: average of the final $10^4$ iterations

**Table 6.3:** Numerical results (2)

| Dataset | Hit $t_0$ | Hit $t_m$ | Hit | Hop $t_0$ | Hop $t_m$ | Hop | HpH $t_0$ | HpH $t_m$ | HpH |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $\iota = 0.75$ | | | | | |
| kry1024c100a | 7.8 | 11.5 | 47% | 6.9 | 6.5 | -7% | 0.89 | 0.56 | -37% |
| kry1024c100b | 8.1 | 11.4 | 42% | 7.2 | 6.6 | -9% | 0.89 | 0.58 | -36% |
| kry1024c100c | 7.5 | 11.1 | 47% | 6.5 | 6.5 | 0% | 0.87 | 0.59 | -32% |
| rand c100a* | 7.5 | 11.3 | 51% | 7.2 | 6.7 | -6% | 0.96 | 0.60 | -38% |
| rand c100b* | 7.9 | 11.5 | 46% | 7.1 | 6.7 | -6% | 0.91 | 0.59 | -35% |
| rand c100c* | 6.8 | 10.8 | 59% | 7.0 | 6.9 | -1% | 1.02 | 0.64 | -38% |
| rand c100d* | 6.4 | 11.2 | 73% | 7.5 | 7.3 | -2% | 1.16 | 0.65 | -44% |
| rand c100e* | 6.8 | 11.5 | 70% | 7.4 | 7.2 | -2% | 1.10 | 0.63 | -42% |
| rand c100f* | 7.0 | 11.2 | 58% | 6.9 | 6.8 | -2% | 0.99 | 0.62 | -38% |
| rand c100g* | 7.7 | 11.5 | 48% | 7.2 | 6.8 | -6% | 0.93 | 0.60 | -36% |
| Average | 7.3 | 11.3 | 54% | 7.1 | 6.8 | -4% | 0.97 | 0.61 | -38% |
| | | | | $\iota = 1.00$ | | | | | |
| kry1024c100a | 7.3 | 10.0 | 36% | 6.5 | 5.4 | -16% | 0.89 | 0.54 | -39% |
| kry1024c100b | 7.5 | 9.9 | 32% | 6.3 | 5.4 | -14% | 0.84 | 0.55 | -35% |
| kry1024c100c | 6.9 | 10.1 | 46% | 6.4 | 5.6 | -12% | 0.94 | 0.56 | -40% |
| rand c100a* | 6.4 | 10.1 | 56% | 6.9 | 5.9 | -14% | 1.08 | 0.59 | -45% |
| rand c100b* | 5.7 | 9.5 | 64% | 7.2 | 6.4 | -10% | 1.25 | 0.68 | -45% |
| rand c100c* | 5.8 | 9.6 | 65% | 6.9 | 6.0 | -13% | 1.20 | 0.63 | -47% |
| rand c100d* | 6.5 | 9.6 | 47% | 7.0 | 6.0 | -13% | 1.07 | 0.63 | -41% |
| rand c100e* | 6.9 | 10.0 | 46% | 6.7 | 5.8 | -13% | 0.98 | 0.58 | -40% |
| rand c100f* | 6.7 | 10.0 | 48% | 6.5 | 5.7 | -12% | 0.96 | 0.57 | -41% |
| rand c100g* | 6.5 | 9.8 | 50% | 6.7 | 5.9 | -12% | 1.03 | 0.60 | -41% |
| Average | 6.6 | 9.8 | 49% | 6.7 | 5.8 | -13% | 1.02 | 0.59 | -41% |

$t_0$: average of the initial $10^4$ iterations
$t_m$: average of the final $10^4$ iterations

# Chapter 7

# An Efficient ACO Strategy for the Resolution of Multi-Class Queries

Kamil Krynicki[†] Michael E. Houle[*] Javier Jaen[†]

[†] ISSI Research Group, Departamento de Sistemas Informáticos y Computación Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

[*] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

*Under Review, as of December 4, 2015*

## Abstract

Ant Colony Optimization is a bio-inspired computational technique for establishing optimal paths in graphs. It has been successfully adapted to solve many classical computational problems, with considerable results. Nevertheless, the attempts to apply ACO to the question of multidimensional problems and multi-class resource querying have been somewhat limited. They suffer from either severely decreased efficiency or low scalability, and are usually static, custom-made solutions with only one particular use. In this paper we employ Angry Ant Framework, a multi-pheromone variant of Ant Colony System that surpasses its predecessor in terms of convergence quality, to the question of multi-class resource queries. To the best of the authors knowledge it is the only algorithm capable of dynamically creating and pruning pheromone levels, which we refer to as *dynamic pheromone strati-*

*fication.* In a series of experiments we verify that, due to this pheromone level flexibility, Angry Ant Framework, as well as our improvement of it called Entropic Angry Ant Framework, have significantly more potential for handling multi-class resource queries than their single pheromone counterpart. Most notably, the tight coupling between pheromone and resource classes enables convergence that is both better in quality and more stable, while maintaining a sublinear cost.

## 7.1 Introduction

The efficient resolution of resource queries in networks is an abstract mathematical problem with a multitude of real-life interpretations and uses: from task dispatching in computer grids to document search in peer-to-peer networks and routing of a parcel drop-off vans through an urban neighborhood. In mathematical terms a resource query $q$ in a graph is defined as an act of establishing a path linking a network node $r_0$, lacking resources, with a node or a chain of $i$ nodes $(r_1, r_2, .., r_i)$ that provide them. Resources are transported back to the query emitting node over this newly established path. The path can be reutilized in the future, when a query of a similar nature is launched in the vicinity of $r_0$. The collecting of resources can be either destructive or not, meaning that it can consume the stock or just simply clone it.

If the resources are indistinguishable between each other (in other words, they belong to one class) we speak of single resource class queries, or *single class queries.* Such queries consist of simply gathering a number of resources at the highest possible goodness. A generalization of a query in a single resource class environment is a query in a multiple resource class environment; *multi-class query* for short. If we assume a number of $|C|$ different resource classes, distinguishable between each other, a multi-class query is defined as a resource query limited to a selected resource class $c$, written $q(c)$. In this case, the objective is to collect, as efficiently as possible, resources belonging to $c$ only.

Given the underlying graph structure of this problem, there have been attempts in the past to solve it by using Ant Colony Optimization (ACO) [129]. ACO is a computational technique for a broad class of algorithms that perform path searches in graphs. This technique is based on observations in the field of entomology revealing that a group of relatively simple organisms is often capable of completing complex tasks in a surprisingly efficient manner. A broad, generic term for this phenomenon is *emergent* or *swarm intelligence* [109] and the most prominent and well-known examples of it include the social behavior of ants and bees.

Despite the unquestionable success of ACO, with its numerous and diverse applications [130] [131] [132], it is well-known that resource querying involving multi-class resources [12], dynamic resource redistribution [11] and evolving graphs [8] have

a less natural representation within its metaheurisitc, which manifests itself in reduced efficiency. The main reason behind the difficulties of modeling with ACO is that it entails a strict limitation, which reflects the constrained nature of real-life ants' communication capabilities [33]. Ants leave trails of pheromone, connecting the colony to locations of interest. The pheromone level can be seen only as a general *goodness* of what is at the end of the trail, without specifying directly what it is, nor the type of resource it might be. Therefore, ACO search agents possess only a rough indication of a good direction to follow. A problem arises if the model in question includes various types of objectives. Consider the example of two ants in search for two different types of food. Each one aware of its task, but limited in its search to following high pheromone trails. The trail is essentially unable to guarantee that it leads towards precisely what the ant searches for. Ants searching for the food type $A$ might end up following trails that lead to good type $B$-nodes (and vice versa); yielding a long search without results. A single pheromone value is not sufficient to guarantee efficient routing in models with ontology- or taxonomy-based resource classifications.

In a multi-class query environment the basic $ACO$ single pheromone model proves insufficient. If there exist $|C| > 1$ resource classes, the probability that a generic pheromone trail leads to a high quality node for all the possible classes decreases rapidly. In such cases the pheromone values are likely to finish the convergence in a suboptimal state (*suboptimal convergence*). The higher the value of $|C|$ the more probable it is that the full pheromone trail convergence is never achieved. The system enters a state called *pheromone thrashing*, a situation in which agents with different queries continually change the pheromone trails back and forth. No coherent information can be concluded from the pheromone state and the model falls back to a near-random walks.

To illustrate this problem, in Fig 7.1 we show the efficiency decline of multi-class queries with the use of *Ant Colony System* (ACS) [17], a well-known and highly acclaimed ACO implementation. If only one class of resources exists ($|C| = 1$) the graph is smooth and the convergence is quick and stable. Already at $|C| = 10$ the algorithm performs substantially worse. Even though convergence is achieved, it is suboptimal: about 145% worse than with $|C| = 1$. With $|C| = 100$ the result is only marginally better than the one obtained by the Random Walks algorithm, which we include for comparison. At $|C| = 100$ the Hop-per-Hit ($HpH$) values are 280% worse than the $|C| = 1$ baseline. In addition, the $|C| = 100$ is much less smooth than the two other due to strong pheromone thrashing. The topic of quality decline of the convergence of ACO algorithms in various scenarios is explored in depth in [12] [8].

In order to efficiently solve the distributed multi-class resource querying problem, we propose the application of a dynamic multipheromone approach *Angry Ant Framework* (AAF). AAF algorithm has already proven its efficiency in the context of single-resource class queries [20], surpassing ACS, one of the most popular ACO

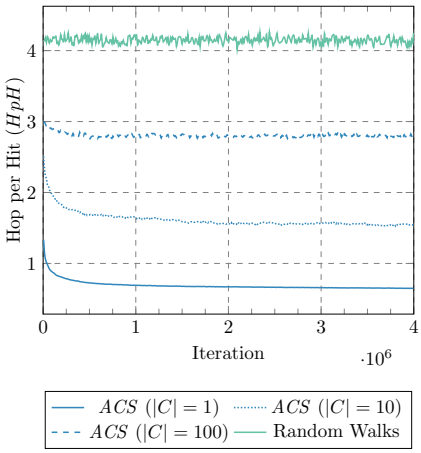



**Figure 7.1:** $ACS$ quality loss in function of $|C|$.

implementations. AAF improves the path quality of ACS by a significant margin, while at the same time decreasing the chance that the algorithm converges to a local minimum. This previous work is extended here confirming that the AAF is beneficial in a more general way by applying it to the resolution of multi-class resource queries. Moreover, we propose an enhancement of AAF's algorithmic strategy based on entropic ant reassignments, which consists in finding the best pheromone level for a given ant at each moment of the computation process. The strongest suite of AAF is that it allows dynamic pheromone stratification, that is, creating ad-hoc multipheromone models, which adapt themselves to the resource ontology, ant traffic and query distribution. The results obtained open a new area of research in which a computational model based on entropy levels of the underlying pheromone state can be applied to the efficient resolution of multi-class resource queries.

The remainder of this paper is organized as follows. In section 7.2 we expand on the concepts of single class and multi-class queries in the context of ACO-based queries in graphs and summarize the related works. Section 7.3 contains the outline of the mathematical model of AAF. The experimental results are presented in 7.4. We conclude with a discussion on our findings in section 7.5.

## 7.2 Related Works

The problem of multi-class queries has only received a moderate attention. In [31] Santillán et al. resolve a problem of a similar nature, which they refer to as Semantic Query Routing problem (SQRP). As in multi-class queries, in SQRP the class of the query impacts the routing of the search agent, which has the task of determining the shortest path from the query issuing node to the node with appropriate resources. Their single-pheromone algorithmic solution to SQRP, called Neighboring-Ant Search (NAS), is reported as a order-of-magnitude improvement over Random Walks, however no comparison with more elaborate techniques is presented. Moreover, NAS uses lookahead techniques that are outside the well-known ACO paradigm and, in consequence, not fully compatible with distributed networks.

Various works, such as [133], [60] and [134] ignore the problem and focus on the efficiency increase in this scope of single class queries exclusively. Oftentimes authors contemplate the aforementioned multipheromone approach, i.e. modeling the pheromone with (depending on the naming convention) *scents*, *flavors* or simply *levels*, however, they still apply their extended algorithms to single resource queries only [110]. Alternatively, they label the classes in a prearranged, rather than a dynamic, manner [126] [13]. In addition, in the majority of the cases the additional pheromone levels are fixed, have predetermined semantical interpretations and are statically incorporated in the classical equations [77].

Multi-class queries have a very natural representation in the multipheromone model. The most trivial multipheromone approach to the multi-class query problem is to decompose it into $|C|$ single class queries sub-problems. A solution of this kind is presented in [44], where each resource class $c \in C$ is statically assigned an individual pheromone level $\ell \in L$, alongside agents that operate exclusively within that level. This type of solution is sufficient under certain circumstances and for small $|C|$. Unsurprisingly however, a static setup, in which one pheromone level is dedicated to ants handling just one resource class is suboptimal, redundant and suffers from several serious drawbacks, especially as $|C|$ gets large. Note that here each pheromone level spans the entire graph, even nodes where there are no resources corresponding to it. Resources hardly present in the system also require a level of pheromone spanning the entire graph. We shall refer to this approach as static multilevel.

There are other critical issues with the static multilevel model. First, the entire taxonomy of resources must be known in advance and the knowledge of it must be transferred onto ants. Second, it fails if the resource classes are added or removed during the algorithm's execution. Third, the evolution of the system is slowed down by a factor of the number of levels $|L|$, due to the fact that, on average, only $1/|L|$ of all the ants contribute to each pheromone level. In addition, there is no

intra-level knowledge transfer, even if certain types of resources always coincide. From a purely computational point of view, maintaining data structures for all the pheromone values is memory-demanding.

These disadvantages motivated the use of an alternative strategy, a middle ground solution between the single pheromone level and the static multilevel: the *dynamic multilevel* of AAF [20]. AAF was established as the first dynamic multilevel implementation of *ACS* that had the initial intention, like many of the aforementioned works, of improving the efficiency of single class queries. In AAF multiple pheromone levels are allowed, but the number of levels $|L|$ is not fixed at a one-to-one ratio with the number of resource classes, instead it can grow as needed. The levels can also be present locally and span a portion of the graph, where the choice of path is crucial. In this work we demonstrate that AAF, with its dynamic multilevel mechanism, is well suited for the multi-class queries problem as well.

## 7.3 Angry Ant Framework

### 7.3.1 Overview

*Angry Ant Framework* (AAF) is formally introduced in [20], with the full mathematical explanation and the formal motivation. Here we present some of the main ideas of AAF with our recent improvements of the original model.

AAF is an ACO strategy that builds upon the model provided by *ACS*, one of the most popular ACO implementations. The high-level ACO paradigm has been inspired by the behavior of real-life ants in search for food. The agents in ACO are simple automata, they lack intelligence and they coordinate their efforts exclusively via a *stigmergic* mechanism [135] - one that is indirect and relies on intermediate means, such as updates to the environment [136]. In ACO this is achieved by having agents interact with the stigmergic medium, referred to as *the pheromone*, in two modalities: the *search mode* and the *deposition mode*. In search mode agents attempt to establish a solution path, by building upon solutions marked by past agents. In *deposition mode* they themselves mark the trace of the solution path they obtained. Even though each individual in an ant nest makes simple, stochastic decisions, based on the current pheromone state, the overall ant swarm behavior can be perceived as intelligent, which is a phenomenon present in all of the algorithms of the ACO metaheurisitcs.

Rather than operating in a contiguous world, like real-life ants, the agents of ACO algorithms operate in environments that are modeled, without loss of generality, as a symmetric directed graph $G = (V, E)$. Nodes $V$ of the graph represent the possible locations of agents, while directed edges $E$ define adjacency of nodes. The set of nodes adjacent to a given node $r$ is called the neighborhood of $r$, $N_G(r)$; of

size $\deg(r)$. A directed edge between the pair of nodes $r$ and $u$ is written as $ru$. An agent is permitted to travel between the nodes $r$ and $u$ if $ru \in E$. The edge $ru$ has a fixed transition cost $\eta_{ru} \in \mathbb{R}^+$ and a pheromone value $\tau_{ru} \in \mathbb{R}^+$ that serves as the stigmergic medium for agents. The initial value of $\tau$ on each edge is $\tau_0 \in \mathbb{R}^+$ and the minimum and the maximum permitted are $\tau_{min} \in \mathbb{R}^+$ and $\tau_{max} \in \mathbb{R}^+$, respectively.

Multipheromone approaches, including AAF, use an array of $\tau_{ru}^{(\ell)}$ values, of varying sizes, instead of just one $\tau_{ru}$ pheromone value per directed edge. The levels are contiguous, from $\ell = 1$ to $\ell = |L|$. For node $r$, the $\tau_{rs}^{(\ell)}$ must be defined for all its adjacent nodes $N_G(r)$, and for every level $\ell$ present in $r$; that is, every level has fully defined pheromone values for all the outgoing links. The size of the pheromone array can change as requested.

Ants are level-dependent, which means that, being routed within the level $\ell$, ant $a^{(\ell)}$ will only use its corresponding level pheromone values. The equations that govern the multipheromone update, evaporation and state transition are presented in subsection 7.3.2 (Fig 7.2, step $D_*$). The two main questions to address are: when are new pheromone levels created and how are ants assigned a pheromone level to resolve their query.

Pheromone level creation reflects the nature of the resources present in the system. An additional level will be created as needed if the existing pheromone levels prove ineffective, that is, when the *entropy* of the pheromone information provided by them is high. This incremental process takes place without any prior knowledge about the ontology of the resources or their distribution. In biological terms the idea could be seen as measuring the *ant irritation*, explained in detail in subsection 7.3.4. Depending on the chaos or clarity of the pheromone state, ants may become increasingly irritated with not having a clear-cut way of achieving the required goal. When a particular ant reaches a certain irritation threshold (Fig 7.2, step $B_*$), it abandons the work and resumes anew in another level, possibly creating additional pheromone levels (Fig 7.2, step $C_*$).

The initial assigning of ants to pheromone levels is performed at the query generation, with the help of the *level assignment matrix* $\mathcal{L}(r)$ (Fig 7.2, step $A_*$), which is stored and maintained in every node $r$, as explained in the subsection 7.3.3.

**Figure 7.2:** Entropic Angry Ant Framework high level pseudocode

```
1: INITIALIZE()
2: while i ≤ agent_max do
3:     i_node ← SELECT_NODE()
4:     level ← ASSIGN_LEVEL(L(i_node))                          ▷ (A*)
5:     LAUNCH_AGENT(i_node, level)
6:     PHEROMONE_EVAPORATE()
7:     i ← i + 1
8: end while
9: for all agent do
10:    solution ←GET_SOLUTION(i_node, query, level)
11:    goodness ←EVALUATE_SOLUTION(solution)
12:    PHEROMONE_DEPOSIT(solution, goodness)
13: end for

14: function GET_SOLUTION(i_node, query, level)
15:    res ← empty_set
16:    node ← i_node
17:    path ← empty_list
18:    while not stop_condition do
19:        res ∪ QUERY_RESOURCES(node, query)
20:        if agent_is_irritated then                            ▷ (B*)
21:            reassign_level()          ▷ /not in original AAF/ (C*)
22:        end if
23:        node ←STATE_TRANSITION(node, level)                   ▷ (D*)
24:        path ∪ node
25:    end while
26:    return solution(path, res)
27: end function
```

### 7.3.2 State Transition and Pheromone Evolution

With the inclusion of the pheromone level concept the classical state transition equations of *ACS* need to be rewritten with $\tau^{(\ell)}$ in place of $\tau$. As in *ACS* the initial pheromone value is $\tau_0 \in \mathbb{R}^+$, the minimum is $\tau_{min} \in \mathbb{R}^+$ and the maximum is $\tau_{max} \in \mathbb{R}^+$. The use of various levels in the state transition is indicated in Fig 7.2, step $D_*$. Note that the cost $\eta$ is level-independent.

$$s = \text{argmax}_{u \in N_G(r)} \{\tau_{ru}^{(\ell)} \times \eta_{ru}^{\beta}\} \tag{7.1}$$

$$p_{rs} = \begin{cases} \frac{\tau_{rs}^{(\ell)} \times \eta_{rs}^{\beta}}{\sum_{z \in N_G(r)} \tau_{rz}^{(\ell)} \times \eta_{rz}^{\beta}} & \text{if } s \in N_G(r) \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

Similarly, the pheromone deposition and evaporation are limited to the pheromone level used by the ant $a_q^{(\ell)}$, as below:

$$\tau_{ru}^{(\ell)} \leftarrow (1 - \alpha) \cdot \tau_{ru}^{(\ell)} + \alpha \cdot \delta\tau \tag{7.3}$$

$$\tau_{ru}^{(\ell)} \leftarrow (1 - \rho) \cdot \tau_{ru}^{(\ell)} + \rho \cdot \gamma \cdot \max_{z \in N_G(r)} \tau_{rz}^{(\ell)} \tag{7.4}$$

where:

$\alpha \in \mathbb{R}_0^1$ expresses the weight of the laid pheromone when depositing

$\rho \in \mathbb{R}_0^1$, $\gamma \in \mathbb{R}_0^1$ express the weight of the maximum pheromone when evaporating

$\delta\tau \in \mathbb{R}^1$ is the deposited pheromone value, related to the quality of the solution

As long as there is just one pheromone level, $L = \{\ell_0\}$, $|L| = 1$ the behavior of this model is indistinguishable from *ACS*.

### 7.3.3 Level Assignment Matrix

The aforementioned question of the assignment of ants to pheromone levels is resolved via the *level assignment matrix* $\mathcal{L}(r)$. Every node $r$ stores a matrix $\mathcal{L}(r)$ of dimensions $|L| \times |Q|$. The rows of $\mathcal{L}(r)$ correspond to the levels of pheromone present in $r$, while the columns of $\mathcal{L}$ (written as $\mathcal{L}_q$) correspond to all the query types $q(c)$ that originated at the node $r$. The matrix is initialized as $1 \times 1$: $\mathcal{L}(r) = I_0$, and can grow dynamically as needed.

When a query $q(c)$ is launched from a node $r$ a generic ant $a_q$ is created. Based on the information present in $\mathcal{L}(r)$ the node decides to which level should the ant be assigned (Fig 7.2, step $(A_*)$). The probability of assigning the ant $a_q$ to any level $\ell$ is given by:

$$p[a_q \to a_q^{(\ell)}] = \frac{\mathcal{L}_{\ell q}(r)}{\sum_{\ell' \in \{1..L\}} \mathcal{L}_{\ell' q}(r)} \tag{7.5}$$

Equation (7.5) is evaluated for every level $\ell$ present in $r$. It is possible to select more than one level or to select none. If multiple levels are selected, multiple ants are sent out independently, one to each pheromone level and the final response to the query is the sum of all the partial responses. If no level is selected, the ant is assigned the *best possible level* ($\hat{\ell}$):

$$\hat{\ell} = \text{argmax}_{\ell' \in \{1..L\}} \{\mathcal{L}_{\ell' q}(r)\} \tag{7.6}$$

The information stored in $\mathcal{L}(r)$ is updated by returning ants. When an ant $a_q^{(\ell)}$ returns to its emitting node $r$, yielding results evaluated at goodness $\delta\tau$, it updates $\mathcal{L}(r)$ by first applying (7.7), followed by (7.8):

$$\mathcal{L}_{\ell q}(r) \leftarrow (1 - \alpha_*) \cdot \mathcal{L}_{\ell q}(r) + \alpha_* \cdot \delta\tau \tag{7.7}$$

$$\mathcal{L}_q(r) \leftarrow (1 - \rho_*) \cdot \mathcal{L}_q(r) \tag{7.8}$$

where:

$\alpha_* \in \mathbb{R}_0^1$ is the $\mathcal{L}(r)$ deposition parameter, analogous to the $\alpha$ parameter of the classical equation.

$\rho_* \in \mathbb{R}_0^1$ is the $\mathcal{L}(r)$ evaporation parameter, analogous to the $\rho$ parameter of the classical equation.

If at any point either $\ell \notin L$ (the returning ant has changed the level during its life span) or $q \notin Q$ (the first time a query is launched in the node), then $\mathcal{L}$ is expanded with rows and columns of zeros to accommodate the required new concepts:

$$\mathcal{L}_{\ell \times q} = (I_{\ell \times L} \times \mathcal{L}_{L \times Q}) \times I_{Q \times q} \tag{7.9}$$

where:

$I_{m \times n}$ is a binary matrix of dimensions $m \times n$, with ones on its main diagonal and zeros elsewhere

### 7.3.4 Ant irritation

An ant $a_q^{(\ell)}$ in the level $\ell$ of the node $r$ can become irritated with probability $p_{irr}^{(\ell)}(r)$ (Fig 7.2, step $(B_*)$). In order to easily explain how $p_{irr}^{(\ell)}(r)$ is calculated, we first define a preliminary concept, namely the *pheromone sum*, the sum of all the pheromone values of all the outgoing edges on the level $\ell$ of the node $r$, written as $\tau_{sum}^{(\ell)}(r)$:

$$\tau_{sum}^{(\ell)}(r) = \sum_{s \in N_G(r)} \tau_{rs}^{(\ell)} \tag{7.10}$$

With (7.10), the ant irritation in node $r$ on the level $\ell$ is:

$$p_{irr}^{(\ell)}(r) = \iota \times \underbrace{\frac{1}{\tau_{max}}}_{\text{(a)}} \times \underbrace{\frac{\tau_{sum}^{(\ell)}(r)}{deg(r)}}_{\text{(b)}} \times \underbrace{\sum_{z \in N_G(r)} \frac{\tau_{rz}^{(\ell)}}{\tau_{sum}^{(\ell)}(r)} \ln \frac{\tau_{sum}^{(\ell)}(r)}{\tau_{rz}^{(\ell)}}}_{\text{(c)}} \tag{7.11}$$

where:

$\iota \in \mathbb{R}_0^1$ is a parameter that can be used to regulate the strength of the irritation. If $\iota = 0$ then $p_{irr}^{(\ell)}(r) = 0$, while if $\iota = 1$ the system uses the irritation model to its full potential. (Alg 7.2, step $(C_*)$)

**(a)** is the normalization factor based on the maximum pheromone value $\tau_{max}$; to guarantee $p_{irr}^{(\ell)}(r) \in \mathbb{R}_0^1$.

**(b)** is the average pheromone per link in the node $r$ on the level $\ell$. The higher the average amount of pheromone, the more *mature* the level should be considered. This reduces greatly the ants' irritation with *fresh levels* that have very high entropy, due to identical pheromone value, $\tau_0$ on all the edges, but very low maturity.

**(c)** is the entropy of the pheromone distribution in the node $r$ on the level $\ell$

In our original proposal [20] of AAF we employ the irritation value as a probabilistic threshold. The probability $p_{irr}^{(\ell)}(r)$ is checked against a uniform random variable $P$ upon an ant transitioning into a node $r$. If $P > p_{irr}^{(\ell)}(r)$, the ant ascends one pheromone level (7.12) and creates a new one, if needed:

$$a_q^{(\ell)} \rightarrow \begin{cases} a_q^{(\ell+1)} & \text{if } P > p_{irr}^{(\ell)}(r), \\ a_q^{(\ell)} & \text{otherwise,} \end{cases} \tag{7.12}$$

However, this simple $\ell \rightarrow \ell + 1$ level ascension is an uninformed action, one that underutilized the knowledge accumulated in level assignment matrix $\mathcal{L}$. Since the appearance of the original AAF algorithm we developed a more efficient approach, called *entropic ant reassignment*.

Entropic ant reassignment is a complementary approach to the simple level ascension. The agents, rather than always increasing their level by 1 when the irritation is positively evaluated, can either: i) ascend one level; or ii) be reassigned to the best possible level among the existing ones. The reassigning of the agent $a_q$ to the best possible level occurs only if the query $q$ is present in the level reassignment matrix $\mathcal{L}(r)$ of the current node ($q \in Q$), otherwise the simple level ascension is always performed. A factor in choosing between the two modes of reassignment is the quality of the information present in the column vector $\mathcal{L}_q$ that represents the query $q$. The entropic ant reassignment rules are written as:

$$a_q^{(\ell)} \rightarrow \begin{cases} a_q^{(\hat{\ell})} & \text{if } P > p_{irr}^{(\ell)}(r) \text{ and } R \geq \frac{H(\mathcal{L}_q)}{H_{max}(|Q|)} \text{ and } q \in Q, \\ a_q^{(\ell+1)} & \text{if } P > p_{irr}^{(\ell)}(r) \text{ and } (R < \frac{H(\mathcal{L}_q)}{H_{max}(|Q|)} \text{ or } q \notin Q), \\ a_q^{(\ell)} & \text{otherwise,} \end{cases} \tag{7.13}$$

where:

$\hat{\ell}$ is the best possible level, calculated in (7.6).

$H(\mathcal{L}_q)$ is the Shannon entropy of the column vector $\mathcal{L}_q$, calculated in (7.14), which should be understood as the amount of information to conclude from contained in the vector.

$H_{max}(|Q|)$ is the maximum Shannon entropy of a column vector $\mathcal{L}_q$ of size $|Q|$, calculated in (7.15). It is the maximum possible amount of information that could be contained in $\mathcal{L}_q$.

$P$, $R$ are uniform random variables in the $[0, 1]$ range.

$$H(\mathcal{L}_q) = \frac{ln(\sum_{\ell' \in \mathcal{L}_q} \mathcal{L}_{\ell' q})}{\sum_{\ell' \in \mathcal{L}_q} \mathcal{L}_{\ell' q}} \times \sum_{\ell' \in \mathcal{L}_q} \mathcal{L}_{\ell' q} ln(\mathcal{L}_{\ell' q}) \tag{7.14}$$

$$H_{max}(|Q|) = ln(|Q|) \tag{7.15}$$

Logically, this process should be understood as the following: if the agent is irritated with its current level it tries to i) find a better level in an informed way, by selecting the best available one; or ii) find a better level in an uninformed way, by

selecting the next one. We argue that this allows the algorithm to use the information stored in $\mathcal{L}$ in a more efficient manner and to reduce the random exploration slightly.

In the remainder of the paper we will refer to the version of AAF with entropic ant reassignment as *EntropicAAF*.

## 7.4 Experimental results

### 7.4.1 Experimental procedure

In our experiments we attempt to demonstrate four main points. First, we perform a brief comparative of AAF and EntropicAAF for single class queries ($|C| = 1$). Second, having shown in our previous work [20] that AAF surpassed ACS in single class queries ($|C| = 1$), we want to verify that it does so, as well as EntropicAAF, in multi-class queries ($|C| > 1$). Third, we analyze the behavior of our algorithms and ACS in a non-random distribution of resources to evaluate their effectiveness in dealing with clustered resources. Finally, we examine the computational cost of the efficiency increase in function of $|C|$.

For each experiment we generate $R$ resources of $|C| = 10$ or $|C| = 100$ resource classes, distribute them in a graph $G$ (a 2-dimensional toroid) of $N = 32 \times 32 = 1024$ nodes and perform $4 \times 10^6$ resource queries $q(c)$ ($1 \times 10^6$ in the computational cost experiments).

The workstation used in the experiment execution is a 64bit, Intel Xeon X3430 (8MB Cache, 2.40 GHz), limited to 2 of the 4 physical cores and 2GB Ram. A single execution takes about 45 - 60 seconds to complete and generates about $70Mb$ of raw data. Each plot is an average of 10 independent executions over different graphs, three of which are available for download at [128]. They are $kry1024c10a$, $kry1024c10b$ and $kry1024c10c$ for the $|C| = 10$ experiment and $kry1024c100a$, $kry1024c100b$ and $kry1024c100c$ for $|C| = 100$ experiment. The remaining seven are randomly generated. We trace 3 plots for each figure: ACS ($\iota = 0.00$), AAF ($\iota = 1.00$) and EntropicAAF ($\iota = 1.00$).

The parameters used in the execution and experimentation are summarized in Table 7.1. *ACS* parameters are at their standard values, taken from literature. The parameters $\alpha_*$ and $\rho_*$ have been set to match the analogous values of $\alpha$ and $\rho$. The initial state of the Level Assignment Matrix $I_0$ is, by definition, a $1 \times 1$ matrix, we chose a small value to easily avoid division by zero in (7.5). The parameters of the experiments reflect the hardware and time constraints.

The convergence quality metrics will be: i) the length of paths found ($Hop$); ii) the amount of resources found ($Hit$); iii) the efficiency in terms of Hop per Hit

**Table 7.1:** Execution parameters

| Parameter | Interpretation | Value |
|---|---|---|
| | ACS | |
| $q_0$ | Weight of strategy selection | 0.80 |
| $\alpha$ | Pheromone deposition parameter | 0.07 |
| $\rho$ | Pheromone evaporation parameter | 0.10 |
| $\beta$ | Weight of link costs | 1.00 |
| $\gamma$ | Weight of evaporation | 0.02 |
| $\tau_{min}$ | Minimum pheromone level | 0.001 |
| $\tau_{max}$ | Maximum pheromone level | 1.000 |
| $\tau_0$ | Initial pheromone level | 0.009 |
| | AAF | |
| $I_0$ | $\mathcal{L}(r)$ initial value | [0.01] |
| $\alpha_*$ | $\mathcal{L}(r)$ deposition parameter | 0.07 |
| $\rho_*$ | $\mathcal{L}(r)$ evaporation parameter | 0.10 |
| | Experiments | |
| $TTL$ | Time to Live | 10 |
| $R_{max}$ | Maximum resources to fetch | 10 |
| $R_{min}$ | Minimum resources to fetch | 5 |
| $N$ | Amount of the nodes | 1024 |
| $\|C\|$ | Number of resource classes | 10 – 100 |

($HpH$); iv) the relative improvement of $HpH$ in the experiment. If a query fails ($Hit = 0$) we take $HpH = 2 \times TTL$ in order to heavily penalize failing to provide results.

### 7.4.2 Random multi-class queries

The distribution of the resource queries $q(c)$ is random. The probability that the node $n$ launches a resource query at a given iteration is $1/N$, while the probability of selecting the resource class $c$ is $1/|C|$. Every node can hold $r(c)$ resources of class $c$. As shown in [12] the exact distribution of resources is not crucial, as it affects the convergence quality equally, maintaining the relative position of the algorithms. We create a setup in which, following the suggestions of [54], there are relatively rare resource-rich nodes ($r(c) = 10$), somewhat common normal nodes ($r(c) = 1$) and a great majority of nodes without resources ($r(c) = 0$) for each resource class $c$. The distribution of $r(c)$, follows a probabilistic model:

$$p[r(c) = 0] = \frac{44}{50},\ p[r(c) = 1] = \frac{5}{50},\ p[r(c) = 10] = \frac{1}{50}$$

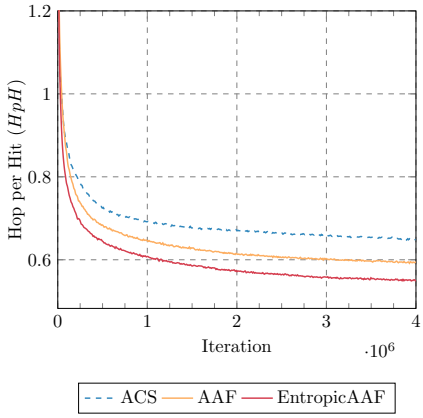


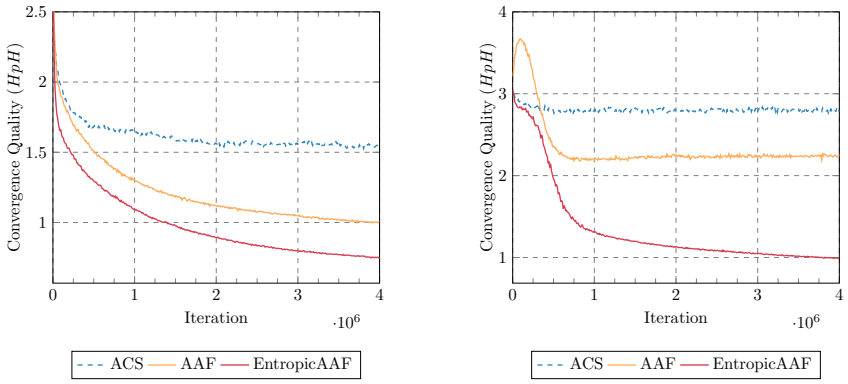

**Figure 7.3:** Convergence Quality. $|C| = 1$. Average of datasets kry1024c1a - rand c1g*. Less is better.

Note that the higher $|C|$ the more total resources are to be distributed among the nodes. In a graph of $N$ nodes, with $|C|$ resource classes the total amount of resources $R$ in the system is given by:

$$R = N \times \ |C| \times \sum_{i \in \{0,1,10\}} (i \times p[r(c) = i]) \tag{7.16}$$

First we would like to show the impact on convergence quality of AAF caused by the substitution of the simple level ascension with the entropic ant reassignment. In Fig. 7.3 we include the side-by-side comparison of $\iota = 0.00$ (here as ACS) and $\iota = 1.00$ (here as AAF) taken from Fig. 3 of our previous work [20] with the new $|C| = 1$ EntropicAFF. All the execution details, under which the EntropicAFF plot was obtained, are identical to those from our previous work. We can see that there is a substantial, 5.1% improvement of EntropicAAF over the older version, which was already shown to outperform ACS. In the remainder of this section we proceed to examine multi-class queries, where the difference becomes more pronounced.

The first multi-class experiment is performed with $|C| = 10$ resource classes. In Fig 7.4a we can see that, both AAF and EntropicAAF, are an important improvement over single-pheromone ACS in this setup. In addition, EntropicAAF retains a steady, positive gap over the older AAF. In numerical terms (Table 7.2): EntropicAAF ($\iota = 1.00$) offers the highest amount of resources found (best 8.82 $Hit$, average 8.63 $Hit$), the best overall improvement (best $-73\%$ $HpH$, average $-70\%$ $HpH$), the shortest paths (best 6.32 $Hop$, average 6.46 $Hop$) and the best Convergence Quality (best 0.72 $HpH$, average 0.75 $HpH$). On average we show that




**(a)** $|C| = 10$ **(b)** $|C| = 100$

**Figure 7.4:** Convergence Quality for multi-class queries. $|C|$ classes. Less is better. Average of datasets kry1024c10a - rand c10g* (7.4a) and kry1024c100a - rand c100g* (7.4b)

EntropicAAF ($\iota = 1.00$) is 52% better than ACS ($\iota = 0.00$) and 17% better than AAF. It produces the shortest, most resource-yielding and most efficient paths.

In $|C| = 100$ variant (Fig 7.4b) ACS makes very little progress in terms of $HpH$ ($-6\%$) and behaves nearly as Random Walks throughout the entirety of the experiment. The random oscillations affect somewhat the AAF as well, but it still reaches better convergence quality in terms of $HpH$. There is a characteristic *explosion* of $HpH$ in the initial stages of the experiment, due to increased exploration of the search space, but it quickly returns to sub-ACS values. This tells us that $|C| = 100$ is a rather demanding experiment and it is well beyond the applicability range of ACS, as well as close to the limit of the AAF. The relative difference between the algorithms is 18.9% in favor of AAF. Both AAF and ACS demonstrate a stagnation phase in which no positive system evolution takes place.

These results motivated us to formulate the new algorithmic strategy of EntropicAAF. EntropicAAF remains unaffected by pheromone trashing and quality oscillations, it is completely free from the HpH explosion in the initial stages of the exploration and improves throughout the totality of the experiment.

The numerical results (Table 7.3) show that EntropicAAF ($\iota = 1.00$) surpasses ACS in terms of absolute Convergence Quality (best 0.97 $HpH$, average 0.99 $HpH$, compared to 2.72 $HpH$ and 2.79 $HpH$ respectively), relative improvement (best $-66\%$ $HpH$, average $-67\%$ $HpH$) and resources found (best 4.39 $Hit$, average 4.24 $Hit$). It obtains the best results in all of the tracked metrics. Based on these observations we claim that EntropicAAF is an unconditional improvement over, both AAF and ACS, and its usability limit lies well beyond the $|C| = 100$ point.

### 7.4.3 Non-random multi-class queries

In this experiment we assume only two resource classes: $C = \{c_1, c_2\}$ ($|C| = 2$). Next we divide the graph $G$ two-fold. First, we select two continuous subgraphs of nodes $G_{c1}$ and $G_{c2}$, each of which contains only resources of class $c_1$ or $c_2$ respectively. Independently, we select two other continuous subgraphs: $G_{q1}$ and $G_{q2}$ of nodes that will be allowed to query for either $c_1$ or $c_2$ only. The divisions are not exclusive; a node can find itself belonging to both subgraphs of each division.

The division into $G_{c1}$ and $G_{c2}$ is constant. They are both the precise halves of the initial 2-dimensional toroid of graph $G$, spanning 50% of $G$. For simplicity we can assume that $G_{c1}$ is the left side of the toroid and $G_{c2}$ is the right side of the toroid.

The division into $G_{q1}$ and $G_{q2}$ comes in three variants:

$v0.5$ $G_{q1}$ and $G_{q2}$ are identical to $G_{c1}$ and $G_{c2}$. There must be no query traffic between $G_{c1}$ and $G_{c2}$ as all the queries should be solved in their corresponding section of the graph.

$v0.7$ $G_{q1}$ and $G_{q2}$ are of 70% of $G$, which means that there is a 40% overlap between $G_{q1}$ and $G_{q2}$ in the center of the toroid. There should be moderate query traffic between $G_{c1}$ and $G_{c2}$.

$v1.0$ $G_{q1} = G_{q2} = G$. All the nodes can query for both types of resources.

Intuitively this experiment can be understood as two toroidal domes of query-generating nodes progressively overlapping each other, with complete disjoint at $v0.5$ and full overlap at $v1.0$. Naturally, the bigger the overlap section the more need there is for the ants to distinguish between both types of resources $c_1$ and $c_2$, and therefore, the more necessary the irritation model becomes.

In Fig 7.5 we compare AAF ($\iota = 1.00$) and ACS ($\iota = 0.00$) in the three aforementioned execution variants. In the $v0.5$ variant there is no mismatch between the distribution of resources and queries and, in consequence, no need for pheromone splits to take place. We observe a very similar evolution for both versions of the algorithm. However as soon as the overlap is introduced the AAF starts showing its superiority, achieving a 13% $HpH$ difference in the $v0.7$ case, and 22% $HpH$ difference in the $v1.0$ case.

This is an important observation that allows us to reaffirm the conclusions from [20]. If there is no need for the irritation model it does not impair the convergence quality. As soon as the search agents find themselves in a situation where distinguishing between resource classes becomes necessary, the irritation model benefits the convergence quality.

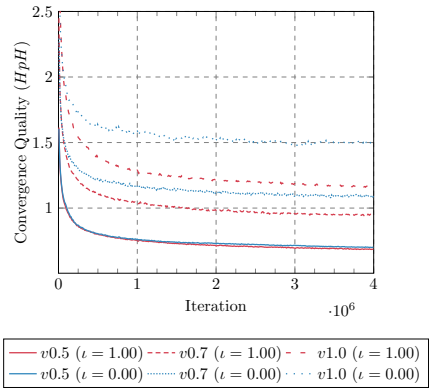


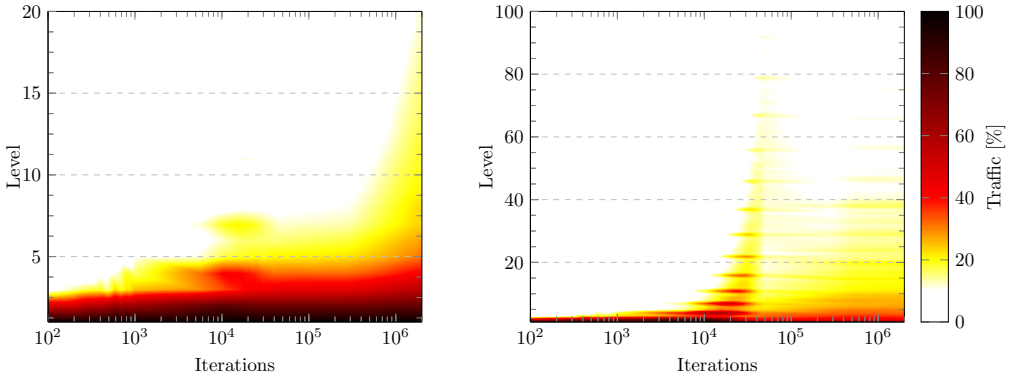

**Figure 7.5:** Convergence Quality. Non-random multi-class queries. Random datasets. Less is better.

### 7.4.4 Computational cost

Seeing that the direct memory and CPU load depend highly on the implementation details and the workstation used, we opted for two objective measures that express the overhead added by the introduction of the multilevel pheromone.

In order to estimate the memory increase we analyze the graph-wide pheromone level usage. As described EntropicAAF starts with just one pheromone level and incorporates additional levels as needed. Naturally, the pheromone values of all the pheromone levels are the dominating factor in the memory consumption of the algorithm, as there is little more information stored. Therefore, we consider this measure a strong indicator of the memory overhead multipheromone ACO-algorithms introduce over the single pheromone ACO strategies.

An interesting discussion can be build around the Fig. 7.6a and Fig. 7.6b, where we display the pheromone level utilization in the $|C| = 10$ and $|C| = 100$ variants respectively (EntropicAAF only). The first conclusion is that the algorithm tends to utilize significantly less pheromone levels than the number of classes of resources present. A great majority of the traffic is routed in less than $|L| = 0.5 \times |C|$ levels for $|C| = 10$ and $|L| = 0.2 \times |C|$ levels for $|C| = 100$. For the variant $|C| = 100$ a secondary process becomes visible, namely the *level pruning*. At a midpoint of the execution the level utilization reaches $|L| = 0.7 \times |C|$, it, nonetheless, only decreases from that moment on. This observation suggests that the more resource classes the more efficient each level becomes. In addition, taking in consideration that the levels span only small portions of the overall graph $G$, rather than the totality, we reported only a 10% - 19% memory consumption increase overall. Comparatively, the classical $|L| = |C|$ approach with each level spanning the entire graph requires $|C|$ times more memory.

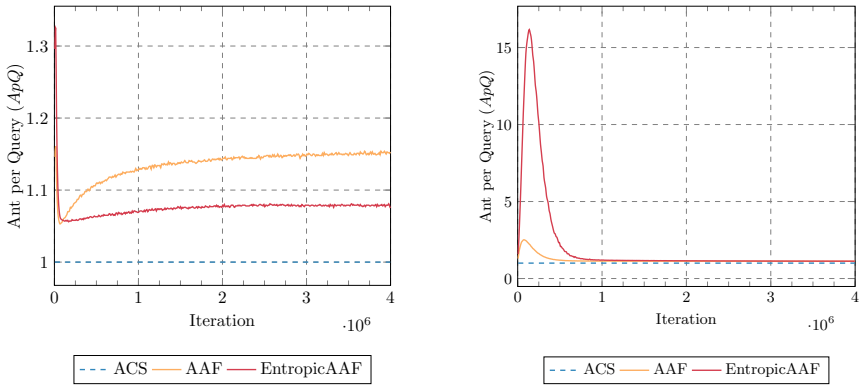



**(a)** $|C| = 10$ **(b)** $|C| = 100$

**Figure 7.6:** Traffic distribution among pheromone levels. EntropicAAF for $|C|$ classes. Average of datasets kry1024c10a - rand c10g* (7.6a) and kry1024c100a - rand c100g* (7.6b)

The increase in terms of CPU load is rather accurately depicted by the *Ant per Query* ($ApQ$) measure. Multipheromone ACO implementations oftentimes use more than one ant per query. Assuming that each ant requires approximately the same computational effort we can view $ApQ$ as a multiplicative CPU cost increase with respect to ACS ($\iota = 0.00$), seeing that, for ACS, $ApQ \equiv 1$.

In Fig. 7.7a and Fig. 7.7b we can see that AAF reaches roughly $ApQ = 1.15$ steadily and quickly in both cases. For EntropicAAF, however, Fig 7.7a and 7.7b reveal the cost of the efficiency improvements reported in previous sections. In the early stages of the exploration there is a very intense stage of ant creation. At its low-points EntropicAAF uses $ApQ = 1.3$ and $ApQ = 15.9$ for the variants $|C| = 10$ and $|C| = 100$ respectively. Both values, especially the latter, are high. Nevertheless, the $ApQ$ value decreases rapidly from that point on, reaching the final $ApQ$ value only 1.078 in both cases, which is less than the corresponding value of AAF (1.15), and only 0.078 (or 7.8%) more than ACS. This might suggest that the algorithm requires about 8% more CPU time. However, in our carefully designed, multithreaded implementation we experienced no such increase.




**(a)** $|C| = 10$ **(b)** $|C| = 100$

**Figure 7.7:** Ant per Query. $|C|$ classes. Average of datasets kry1024c10a - rand c10g* (7.7a) and kry1024c100a - rand c100g* (7.7b)

## 7.5 Conclusions and future works

Angry Ant Framework was shown be a significant improvement in the field of non-hybrid ACO for single class queries [20]. In this paper we have presented an improved version of AAF (Entropic AAF) and we have shown that it surpasses *ACS* in a more general manner by also efficiently solving multi-class queries.

We determined that, in terms of *HpH* measure in multi-class resource querying, AAF and EntropicAAF achieved markedly better results than the static single level ACO, represented by ACS. The improvement was of up to 68% in the random resource distribution experiments (subsection 7.4.2) and 22% in the non-random ones (subsection 7.4.3).

On the other hand we validate that both EntropicAAF and AAF require a similar amount of computational effort as ACS. The growth of memory consumption in function of $|C|$ is strongly sublinear, possibly logarithmic, while at the same time static multilevel solutions, such as SemAnt [13] offer a superlinear growth, unsuitable for $|C| > 100$. The scalability of our solution is, therefore, superior. The same can be said about the computational complexity increase, which, with the correct approach to the parallelization of tasks, we were able to neutralize completely, rendering it negligible.

We have concluded a study centered on the query routing. A similar study should follow with AAF in the packet-routing context. Therefore, as a future work, we would like to examine how Angry Ant Framework compares to AntNet [38] and AntHocNet [39], which are considered state-of-the-art in the field of ACO-based packet routing.

## 7.6 Acknowledgments

Kamil Krynicki is a FPI fellow of Universitat Politècnica de València, number 3117. This work received support from the Spanish Ministry of Education under the National Strategic Program of Research, Project TIN2014-60077-R and the National Institute of Informatics, Tokyo, Japan.

**Table 7.2:** Multi-class queries for $|C| = 10$ classes. Numerical results

| Dataset | Hit $t_0$ | Hit $t_m$ | Hit | Hop $t_0$ | Hop $t_m$ | Hop | HpH $t_0$ | HpH $t_m$ | HpH |
|---|---|---|---|---|---|---|---|---|---|
| ACS ($\iota = 0.00$) | | | | | | | | | |
| kry1024c100a | 2.90 | 4.60 | +59% | 8.15 | 7.52 | -8% | 2.81 | 1.63 | -42% |
| kry1024c100b | 3.26 | 4.83 | +48% | 7.81 | 7.29 | -7% | 2.40 | 1.51 | -37% |
| kry1024c100c | 3.12 | 4.72 | +51% | 7.93 | 7.42 | -6% | 2.54 | 1.57 | -38% |
| rand c100a* | 3.48 | 5.14 | +48% | 7.78 | 7.22 | -7% | 2.24 | 1.40 | -37% |
| rand c100b* | 3.01 | 4.70 | +56% | 7.99 | 7.54 | -6% | 2.65 | 1.60 | -39% |
| rand c100c* | 3.17 | 4.87 | +54% | 8.10 | 7.49 | -8% | 2.55 | 1.54 | -40% |
| rand c100d* | 3.08 | 4.63 | +50% | 8.05 | 7.52 | -7% | 2.62 | 1.63 | -38% |
| rand c100e* | 3.11 | 4.71 | +51% | 7.98 | 7.49 | -6% | 2.56 | 1.59 | -38% |
| rand c100f* | 3.30 | 4.58 | +39% | 7.93 | 7.47 | -6% | 2.40 | 1.63 | -32% |
| rand c100g* | 3.25 | 5.36 | +65% | 7.83 | 7.13 | -9% | 2.41 | 1.33 | -45% |
| Average | 3.17 | 4.81 | +52% | 7.96 | 7.41 | -7% | 2.52 | 1.54 | -39% |
| AAF ($\iota = 1.00$) | | | | | | | | | |
| kry1024c100a | 3.98 | 8.29 | +108% | 10.42 | 8.39 | -20% | 2.62 | 1.01 | -61% |
| kry1024c100b | 3.68 | 8.43 | +129% | 10.50 | 8.30 | -21% | 2.86 | 0.98 | -66% |
| kry1024c100c | 3.70 | 8.04 | +117% | 10.41 | 8.42 | -19% | 2.82 | 1.05 | -63% |
| rand c100a* | 4.08 | 8.11 | +99% | 10.64 | 8.36 | -21% | 2.61 | 1.03 | -60% |
| rand c100b* | 4.08 | 8.34 | +104% | 10.68 | 8.31 | -22% | 2.62 | 1.00 | -62% |
| rand c100c* | 3.90 | 8.58 | +120% | 10.69 | 8.31 | -22% | 2.74 | 0.97 | -65% |
| rand c100d* | 3.80 | 8.26 | +118% | 10.06 | 8.17 | -19% | 2.65 | 0.99 | -63% |
| rand c100e* | 4.06 | 8.12 | +100% | 10.71 | 8.39 | -22% | 2.64 | 1.03 | -61% |
| rand c100f* | 4.04 | 8.54 | +111% | 10.15 | 7.95 | -22% | 2.51 | 0.93 | -63% |
| rand c100g* | 4.07 | 8.28 | +103% | 10.46 | 8.34 | -20% | 2.57 | 1.01 | -61% |
| Average | 3.94 | 8.30 | +111% | 10.47 | 8.29 | -21% | 2.66 | 1.00 | -62% |
| EntropicAAF ($\iota = 1.00$) | | | | | | | | | |
| kry1024c100a | 3.93 | 8.78 | +124% | 10.04 | 6.38 | -36% | 2.56 | 0.73 | -72% |
| kry1024c100b | 4.14 | 8.56 | +107% | 10.25 | 6.32 | -38% | 2.48 | 0.74 | -70% |
| kry1024c100c | 4.30 | 8.59 | +100% | 10.38 | 6.36 | -39% | 2.41 | 0.74 | -69% |
| rand c100a* | 4.31 | 8.82 | +105% | 10.50 | 6.37 | -39% | 2.43 | 0.72 | -70% |
| rand c100b* | 4.02 | 8.54 | +112% | 10.60 | 6.54 | -38% | 2.64 | 0.77 | -71% |
| rand c100c* | 4.26 | 8.78 | +106% | 10.33 | 6.48 | -37% | 2.43 | 0.74 | -70% |
| rand c100d* | 3.97 | 8.77 | +121% | 10.54 | 6.53 | -38% | 2.65 | 0.74 | -72% |
| rand c100e* | 3.69 | 8.58 | +132% | 10.39 | 6.52 | -37% | 2.81 | 0.76 | -73% |
| rand c100f* | 3.93 | 8.34 | +112% | 10.45 | 6.58 | -37% | 2.66 | 0.79 | -70% |
| rand c100g* | 4.22 | 8.55 | +103% | 9.95 | 6.50 | -35% | 2.36 | 0.76 | -68% |
| Average | 4.08 | 8.63 | +112% | 10.34 | 6.46 | -38% | 2.54 | 0.75 | -70% |

$t_0$: average of the initial $10^4$ iterations
$t_m$: average of the final $10^4$ iterations

**Table 7.3:** Multi-class queries for $|C| = 100$ classes. Numerical results

| Dataset | Hit $t_0$ | Hit $t_m$ | Hit | Hop $t_0$ | Hop $t_m$ | Hop | HpH $t_0$ | HpH $t_m$ | HpH |
|---|---|---|---|---|---|---|---|---|---|
| ACS ($\iota = 0.00$) | | | | | | | | | |
| kry1024c100a | 2.80 | 2.94 | +5% | 8.18 | 8.05 | -2% | 2.93 | 2.74 | -7% |
| kry1024c100b | 2.76 | 2.88 | +4% | 8.14 | 8.13 | 0% | 2.95 | 2.82 | -4% |
| kry1024c100c | 2.79 | 2.97 | +7% | 8.20 | 8.07 | -2% | 2.94 | 2.72 | -8% |
| rand c100a* | 2.68 | 2.89 | +8% | 8.21 | 8.14 | -1% | 3.07 | 2.82 | -8% |
| rand c100b* | 2.68 | 2.91 | +9% | 8.20 | 8.12 | -1% | 3.06 | 2.79 | -9% |
| rand c100c* | 2.77 | 2.97 | +7% | 8.14 | 8.09 | -1% | 2.94 | 2.72 | -7% |
| rand c100d* | 2.67 | 2.83 | +6% | 8.18 | 8.13 | -1% | 3.07 | 2.88 | -6% |
| rand c100e* | 2.70 | 2.78 | +3% | 8.21 | 8.16 | -1% | 3.05 | 2.93 | -4% |
| rand c100f* | 2.81 | 2.93 | +4% | 8.18 | 8.08 | -1% | 2.92 | 2.76 | -5% |
| rand c100g* | 2.78 | 2.95 | +6% | 8.20 | 8.13 | -1% | 2.95 | 2.75 | -7% |
| Average | 2.74 | 2.91 | +6% | 8.19 | 8.11 | -1% | 2.99 | 2.79 | -6% |
| AAF ($\iota = 1.00$) | | | | | | | | | |
| kry1024c100a | 4.11 | 4.24 | +3% | 13.81 | 9.62 | -30% | 3.36 | 2.27 | -32% |
| kry1024c100b | 3.92 | 4.07 | +4% | 12.45 | 9.57 | -23% | 3.17 | 2.35 | -26% |
| kry1024c100c | 4.20 | 4.16 | -1% | 13.29 | 9.64 | -27% | 3.17 | 2.32 | -27% |
| rand c100a* | 4.22 | 4.33 | +3% | 13.35 | 9.62 | -28% | 3.16 | 2.22 | -30% |
| rand c100b* | 4.01 | 4.35 | +8% | 13.27 | 9.50 | -28% | 3.31 | 2.19 | -34% |
| rand c100c* | 4.21 | 4.30 | +2% | 13.65 | 9.66 | -29% | 3.25 | 2.24 | -31% |
| rand c100d* | 4.27 | 4.23 | -1% | 13.67 | 9.55 | -30% | 3.20 | 2.26 | -29% |
| rand c100e* | 3.98 | 4.22 | +6% | 13.21 | 9.53 | -28% | 3.32 | 2.26 | -32% |
| rand c100f* | 4.14 | 4.40 | +6% | 13.22 | 9.51 | -28% | 3.19 | 2.16 | -32% |
| rand c100g* | 4.96 | 4.39 | -12% | 15.32 | 9.39 | -39% | 3.09 | 2.14 | -31% |
| Average | 4.20 | 4.27 | +2% | 13.52 | 9.56 | -29% | 3.22 | 2.24 | -30% |
| EntropicAAF ($\iota = 1.00$) | | | | | | | | | |
| kry1024c100a | 4.24 | 7.28 | +72% | 13.01 | 7.34 | -44% | 3.07 | 1.01 | -67% |
| kry1024c100b | 4.15 | 7.41 | +79% | 12.55 | 7.33 | -42% | 3.03 | 0.99 | -67% |
| kry1024c100c | 4.06 | 7.25 | +79% | 13.04 | 7.27 | -44% | 3.21 | 1.00 | -69% |
| rand c100a* | 4.27 | 7.34 | +72% | 12.94 | 7.34 | -43% | 3.03 | 1.00 | -67% |
| rand c100b* | 4.16 | 7.22 | +74% | 12.78 | 7.37 | -42% | 3.07 | 1.02 | -67% |
| rand c100c* | 4.47 | 7.40 | +65% | 13.26 | 7.23 | -45% | 2.97 | 0.98 | -67% |
| rand c100d* | 4.26 | 7.49 | +76% | 13.00 | 7.34 | -44% | 3.05 | 0.98 | -68% |
| rand c100e* | 4.29 | 7.45 | +73% | 12.83 | 7.25 | -43% | 2.99 | 0.97 | -67% |
| rand c100f* | 4.14 | 7.36 | +78% | 12.82 | 7.27 | -43% | 3.10 | 0.99 | -68% |
| rand c100g* | 4.39 | 7.30 | +66% | 12.74 | 7.22 | -43% | 2.90 | 0.99 | -66% |
| Average | 4.24 | 7.35 | +73% | 12.90 | 7.30 | -43% | 3.04 | 0.99 | -67% |

$t_0$: average of the initial $10^4$ iterations
$t_m$: average of the final $10^4$ iterations

# Chapter 8

# AntElements: An Extensible and Scalable Ant Colony Optimization Middleware

Kamil Krynicki Javier Jaen

ISSI Research Group, Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

(2015), In Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference, GECCO Companion '15 (pp. 1109-1116). ACM. doi:10.1145/2739482.2768464

**Abstract**

Ant Colony Optimization (ACO) has become a popular metaheuristic approach for solving hard combinatorial optimization problems. However, most existing ACO software systems are domain-specific, dedicated to concrete problems or non-extensible, non-portable and non-scalable solutions that have been evaluated for problem spaces of limited size. In this context, we present AntElements (AntE), a portable Java-based ACO middleware, designed and implemented with the highest consideration for versatility. The extensibility of the proposed middleware allows its use in virtually any ACO deployment, ranging from experimental to commercial. In this work, the overall object-oriented architecture and the soft-

ware design patterns of AntE are explained, alongside the main concepts behind them. Furthermore, AntE is analyzed with respect to the computational efficiency, parallelization capacities and memory consumption, which allows to establish its usability and scalability range. In its current implementation, on an average to mid-high workstation, our middleware is capable of processing upwards of $10^5$ agents per second, in graphs of the order of $10^5$ nodes and sustain a stable, fully logged experiment for over 12 hours. The proposed middleware has already been deployed in several research projects that are outlined in this paper, illustrating the range of possibilities it offers.

## 8.1 Introduction

Evolutionary computation is an area of intensive research that requires, due to the internal complexity of the algorithms and the data structures involved, advanced software solutions. This is especially true in the case of multiagent, swarm-related or bio-inspired metaheurisitcs, such as Ant Colony Optimization ($ACO$) [129]. In this specific case, non-trivial software is generally indispensable in order to validate the soundness and effectiveness of the designed ACO strategies. Such implementations must incorporate, to name a few: advanced multithreading handling, software design patterns, software modularity and low level efficiency optimization. Many of these advanced computational techniques often need to be designed and coded by software engineers, but writing a specialized piece of software anew for each iteration of a problem entails considerable workload.

Software Engineering addresses this matter by introducing the concept of middleware. A middleware is a piece of software that provides a set of black-boxed complex algorithms and encapsulated low-level concepts. It enables researchers to work with higher-level concepts or even entire algorithms as atomic entities, significantly shortening the preparation for the experimental phase and breeching the gap between the theoretical computer science and the experimentation. Standardized middlewares also help to make the experimental results more easily comparable, eliminating the potential for an implementation-bound bias.

In this work we present AntElements ($AntE$), an extensible and highly customizable middleware that facilitates an Ant Colony Optimization (ACO) testbed. The main rationale behind the development of AntE was the creation of a software infrastructure that could be used in three ways. First, for educational purposes, by enabling easy interaction. Second, for experiment execution, due to the extensive logging and configuration abilities. Finally, for software deployment, by achieving very strong scalability, portability and compatibility with mobile devices.

The remainder of this paper is organized as follows. In Section 8.2 we provide an overview of existing ACO middlewares and simulation environments. In Section

8.3 we present the AntE middleware. Section 8.4 studies the efficiency and scalability of the proposed software infrastructure and Section 8.5 discusses the specific deployments that have been realized. We summarize our findings in Section 8.6.

## 8.2 Related Works

ACO has received only a moderate attention with respect to simulation environments and middlewares. The existing solutions tend to be devoted to one particular ACO implementation or even just one classical problem. An example of this domain-specific implementation trend is *ACOTSP* [137], a high performing software with C and Java versions, dedicated exclusively to the Traveling Salesman Problem. The configuration of ACOTSP is restricted to the command line interface and experimental data are supplied in external files. On the upside, ACOTSP incorporates a number of ACO algorithms, while most middlewares are based on a single ACO strategy. This is the case of another domain-specific package, *hc-mmas-ubqp* [138] which is restricted to MAX-MIN Ant System (MMAS) for Unconstrained Binary Quadratic Programming (UBQP) in Hypercubes. Similar in nature is *AntClique* [139], which was written for maximum clique problems. The three aforementioned software packages are reasonably efficient implementations of ACO, but offer limited flexibility, as they focus on one specific problem. In addition, the C code developed under Linux provides no guarantee to work in different environments, such as Windows, OSX or Android. No distributed versions of the packages exist and adaptation is done mainly on the code-level.

*Gui Ant-Miner* [140] and *Myra* [141], are two portable, Java-based implementations of ACO. According to the authors, their implementation is reasonably well performing in terms of CPU time and memory consumption, although only experimental and educational uses should be considered. Again, these two packages work exclusively with two concrete ACO algorithms - Ant Colony-based Data Miners, called Ant-Miner and cAnt-Miner, used for extracting classification rules.

A more general-purpose middleware is *JACSF* [142]. The author presents a centralized, bare-bones ACO middleware and performs no study of efficiency or scalability. Many details of the implementation, such as the results logging, are treated simplistically or are completely absent. On the other hand JACSF is easily extensible and very versatile, it can be reprogrammed for any classical problem and it supports any arbitrary ACO algorithm. The author advises to use his software for experimental purposes only.

Another generic solution, AntLib v1.0 [143] is a promising C++ approach to hybrid ACO middlewares. The authors focus on speed, efficiency and scalability, as well as extensibility, with emphasis on the use of templates and object-orientation.

Their solution is not bound to any algorithm nor a specific problem. However, the code is for local and experimental use only. It is reported as a work in progress.

Finally, *AntHill* [144], is a very advanced ACO middleware, written in Java, that provides full support for parallel and distributed execution. It operates with JXTA P2P technology and is suited for both: experimental and deployment applications. Even though the authors fail to comment on execution times, an experiment of $5 \times 10^5$ iterations is reported which indicates a moderately high scalability. This promising work, however, has not been in development since 2001 and, as a consequence, the structures and technologies it uses have become obsolete. In addition, the customization of AntHill is questionable, as it seems to be strongly bound to the topic of P2P query propagation.

**Table 8.1:** Middleware Comparison

| | | | | | | Scalability | | Problems | | Purpose | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Language | Year | Algorithms | Open | Custom. | Ants | Nodes | Iterations | Open | Dynamic | Edu. | Exp. | Deploy. | Distr. | Multi-thread |
| ACOTSP | C/Java | 2002/4 | 7 | ✓ | ✓* | $> 1500$ | $> 2932$ | N/S | | | ✓ | ✓ | | | |
| hc-mmas-ubqp | C++ | 2004 | 1 | ✓ | | N/S | $> 500$ | N/S | | | ✓ | ✓ | | | |
| AntClique | C | 2006 | 1 | ✓ | | N/S | $> 500$ | N/S | | | ✓ | ✓ | | | |
| Gui Ant-Miner | Java | 2002 | 1 | ✓ | N/S | $> 3000$ | $< 5000$ | N/S | | | ✓ | ✓ | | | N/S |
| Myra | Java | 2008 | 1 | ✓ | N/S | $> 3000$ | $< 5000$ | N/S | | | ✓ | ✓ | | | N/S |
| JACSF | Java | 2009 | N/A | ✓ | ✓ | $> 30$ | $> 50$ | $> 2500$ | ✓ | | ✓ | ✓ | | | |
| AntLib v1.0 | C++ | 2008 | N/A | ✓ | ✓ | N/S | N/S | N/S | ✓ | | ✓ | ✓ | | | |
| AntHill | Java | 2001 | N/A | ✓ | ✓ | N/S | $> 2000$ | $> 5 \times 10^5$ | ✓ | | | ✓ | ✓ | ✓ | ✓ |
| AntElements | Java | 2015 | N/A | ✓ | ✓ | $> 5 \times 10^6$ | $> 65536$ | $> 10^7$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

N/A not applicable, N/S not specified, * command line only

In Table 8.1 we summarize the results stemming from this brief discussion. Note that in most cases it is not straightforward to establish the scalability and the maximum working ranges of a given middleware. If the ranges are not explicitly specified by the authors we estimate them using the largest documented and published execution of the software we were able to find. Some of the presented middlewares have a fixed number of algorithms incorporated, in such cases a value is provided in the column *Algorithms*. In the *Problems* column group, *Open* indicates that a given middleware can be applied to any problem, rather than just a preestablished one. Finally, *Dynamic* is a unique feature of our middleware. It denotes middlewares that support the modification of the problem mid-execution.

In general, the middlewares are often not modular nor extensible and the dominating programming language is C under Linux, which limits significantly the deployment possibilities. Thus, the focus tends to be experimental or educational. The authors hardly ever elaborate on the efficiency or the scalability of their software and almost always neglect portability. The efficiency is usually obtained at the cost of flexibility.

Most importantly, the execution environment is always static, i.e, the parameters of the execution have to be provided before it commences and once the algorithm is set in motion no parameters can be changed. Moreover, the evolution of the problem space is universally not permitted.

## 8.3 AntE Overview

Our solution, AntElements, permits easy creation of complex and compound testing facilities even with limited knowledge of software engineering. Our primary concern, with respect to the software design, was to provide a modular architecture in which each element could be extended and improved upon. It allows to model virtually all ACO-based algorithms and to apply them to an arbitrary problem. In this section we will discuss the high-level architecture of AntE, explain the range of data-logging possibilities, demonstrate snippets of configuration, as well as reveal some interesting low-level optimization techniques.

### 8.3.1 Architecture

A very common, practical and justifiable constraint of ACO mathematical models is to represent the contiguous world, in which the real-life ants operate, with a finite graph. To model the problem space in terms of graphs and nodes varies in difficulty. For instance, in the case of the Traveling Salesman Problem the matching between the domain of the problem and the graph is straightforward. The nodes would model cities, while the edges - the possible transitions between

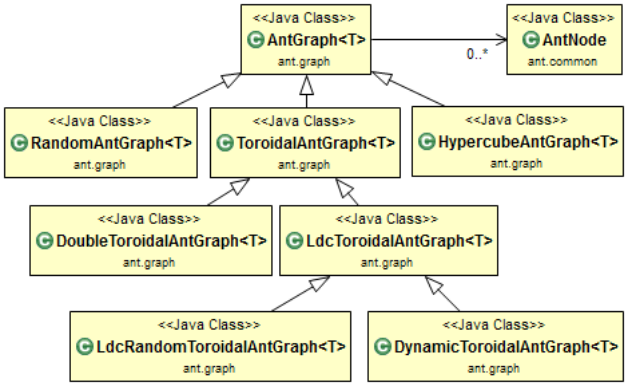



**Figure 8.1:** AntE graph family

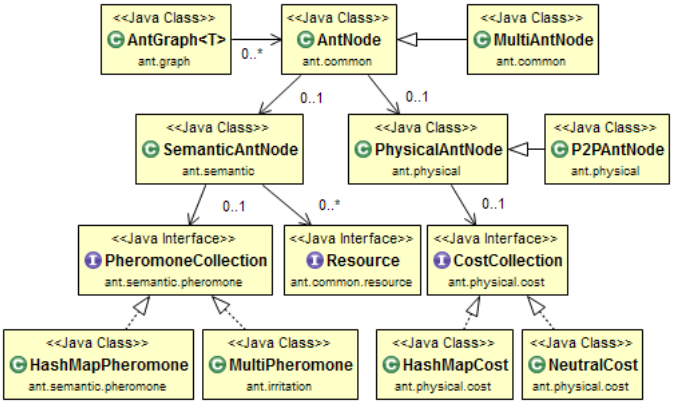



**Figure 8.2:** AntNode class diagram

them. Regardless of the complexity of the modeling process, this approach is considered not to affect the generality of the solutions obtained.

The main component of our architecture is the problem graph (problem-space, Figure 8.1) which is defined in our model by the *AntGraph* class. *AntGraph* provides a basic interface for manipulating a graph, the graph creation and a set of useful additional methods. We produced a number of *AntGraph* implementations, such as *ToroidalAntGraph*, *HyperCubeAntGraph*, *RandomAntGraph* and more. Furthermore, any custom graph is possible, as well as an extension of any of the preexisting ones.

As shown in Figure 8.1 *AntGraph* is composed of problem nodes, represented by the *AntNode* class (Figure 8.2), which are the backbone of the system. The *AntNode*

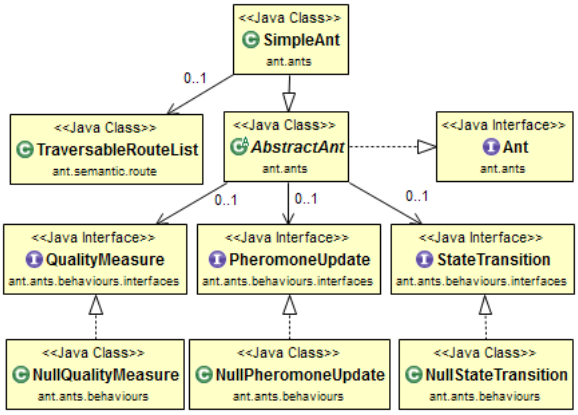



**Figure 8.3:** Ant architecture

is divided into the *SemanticAntNode* and *PhysicalAntNode*, which correspond to the semantic and the physical levels of the system.

The semantic level is in charge of controlling the pheromone values, as well as maintaining local resources and resolving resource queries. The pheromone manipulation is performed via the *PheromoneCollection* interface. We provide two implementations of the *PheromoneCollection*: *HashMapPheromone* and *MultiPheromone*, which differ in efficiency, scalability and versatility. The choice of the correct implementation of the *PheromoneCollection* is crucial for the performance of the whole model.

The physical level is responsible for maintaining intra-node communication and the cost thereof. AntE comes with two existing implementations: *PhysicalAntNode*, which is optimized for local execution and *P2PAntNode*, which enables deployment in a P2P network. We based our P2P implementation on the PastryRing middleware. If the cost of the communication is either not a factor or an invariant the use of the *NeutralCost* class is recommended, in other cases the *HashMapCost* is made available. The physical level must obligatorily enable access to methods for linking and unlinking nodes and offer the possibility to obtain or modify the costs of all the links present.

The central piece of any ACO-related middleware is the concept of an ant. In AntE the abstraction of the ACO-ant is the *Ant* interface (see Figure 8.3), which contains, according to the *strategy* software design pattern, three encapsulated and self-explanatory behaviors: *PheromoneUpdate*, *StateTransition* and *QualityMeasure*. This approach allows the code to be hot-pluggable and the behavior to be changed during the algorithm's execution. All three come with a base *null implementation*, as recommended by the *nullobject* design pattern.

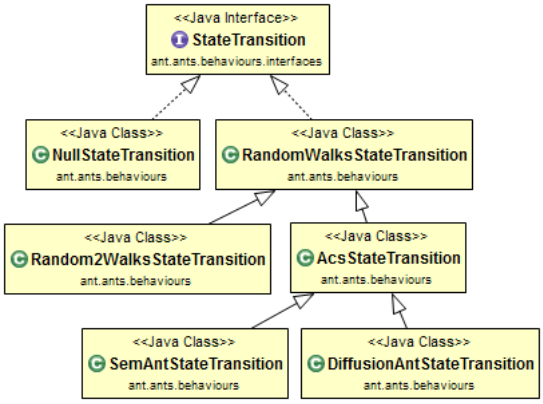



**Figure 8.4:** State transition implementations

```
1 public interface StateTransition
2 {
3   public SemanticId[] performStateTransition(SemanticId[] links, float[]
        ph, float[] cs, RouteList visited);
4 }
```

**Code Snippet 8.1:** State Transition interface

```
1 public SemanticId[] performStateTransition(SemanticId[] links, float[] ph,
      float[] cs, RouteList visited)
2 {
3   float[] weights = new float[links.length];
4   float  sum = 0;
5
6   float weight;
7   for(int i = 0; i < links.length; i++)
8   {
9     weight = calculateWeight(ph[i], cs[i], beta);
10    sum += weight;
11    weights[i] = weight;
12  }
13
14  double f = this.sum * (new Random().nextFloat());
15
16  int i = -1;
17
18  do
19  {
20     f -= this.weights[++i];
```

```
21   }
22   while (f > 0.00001d);
23
24   return links[i];
25 }
```

**Code Snippet 8.2:** Example State Transition (ACS)

The state transition interface (Code Snippet 8.1) takes in four parameters: *links*, an array of links to choose from; *ph* and *cs*, arrays of pheromones and costs corresponding to the links provided; and an optional *visited* parameter: the list of visited nodes, which may or may not influence the state transition. The method returns an array of links to travel to. We chose the return value to be an array, rather than a single value to preserve maximum generality of the code. Some algorithms, such as SemAnt [44], permit ant cloning and splitting, which requires multiple results from a single state transition step. A family of existing state transition rules is provided (Figure 8.4). See Code Snippet 8.2 for a simple example of a state transition rule.

```
1 public interface PheromoneUpdate
2 {
3   float[] performLocalUpdate(float[] ph);
4   float[] performGlobalUpdate(float[] ph, int rewardedLink, float
        solutionQuality);
5 }
```

**Code Snippet 8.3:** Pheromone Update interface

The pheromone update interface (Code Snippet 8.3) defines two methods: *performLocalUpdate* and *performGlobalUpdate*, which represent pheromone evaporation and deposition respectively. Both methods take a pheromone array as input, however the deposition requires an additional specification of the link that participates in the deposition process, as well as the quality of the solution obtained.

```
1 public interface QualityMeasure
2 {
3   public float getQuality(Ant a);
4 }
```

**Code Snippet 8.4:** Quality Measure interface

The quality measure interface (Code Snippet 8.4) is trivial. It is designed to enable a simplified access to the quality of the solution the ant created. The quality must be returned as a float value and should be positive.

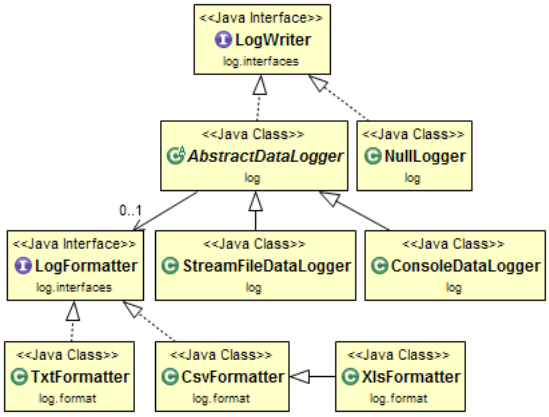



**Figure 8.5:** Logger structure

### 8.3.2 Output and data logging

One of the challenges of every middleware is an efficient data logging module. A badly designed one can slow down the system beyond usability, even when it is not in operation. Due to this reason we paid special attention to it, making sure that it would not render our middleware non-responsive under any circumstances.

The most adequate approach for handling a detached process, such as the logger, is the *short-circuit* design pattern. The short circuit pattern is a multithread software design pattern, which consists of splitting the threads into two groups: worker threads $wt$ and handler threads $ht$. Worker threads produce results, which are later processed by handler threads. Depending on the computation load, the ratio of $wt/ht$ must be adjusted. In our case the handler thread (the logger) is far less CPU-consuming. Thus, we have only one $ht$ and an adjustable amount of $wt$. As we empirically established, in order to maximize CPU usage it is recommended to have $ht = 1$ and $wt = 2 \times cores - 1$, where $cores$ is the amount of physical CPU cores available.

The *LogWriter* interface (Figure 8.5), alongside its implementation of *AbstractDataLogger*, are the base elements of the logging module. They provide methods for logging all the primitive values, common Java collections, char strings and numerals. In the current version of the middleware three implementations of the LogWriter interface are provided: *NullLogger* (nullobject), *ConsoleDataLogger* (on-screen data display) and *StreamFileDataLogger* (saving data via FileStream).

*LogWriter* enables to select what data is to be logged, while the *LogFormatter* establishes the format of the data. Three classes of LogFormatter have been made available by default: *TxtFormatter*, *CsvFormatter* and *XlsFormatter*, which gen-

erate data compatible with *.txt* files, *.csv* files and *.xls* files respectively. This part of the system is, again, fully customizable.

Another important aspect of the logging module is the data post-processing. Raw data from long executions can attain very large sizes, measured in hundreds of gigabytes. AntE offers an extensive spectrum of post-processing tools that include, but are not limited to: data sorting, data compacting by rolling average, merging independent executions, standard deviation extraction as well as lineal, non-lineal and arbitrary data bucketing.

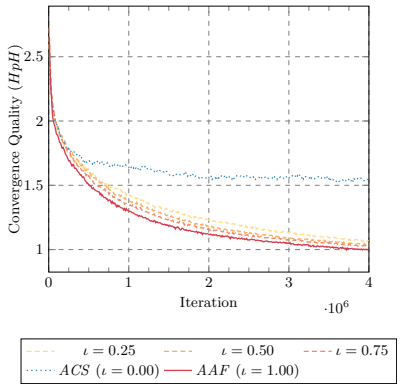
In Figure 8.6 we present some of the possible outputs our middleware can produce. The linear plots (Figure 8.6a) are the most basic form of output. Upon indicating the source of the x-axis from the raw data file we can plot the evolution of any of the logged values or a combination thereof. In this case it is the classic hop per hit ($HpH$) measure in function of algorithm's iteration.

If data of higher dimensionality need to be visualized, the use of heatmaps is recommended (Figure 8.6b and Figure 8.6c). Our middleware can prepare the data for a visualization with the Ti*k*Z LaTeX package for traditional heatmaps (in the shown case - ant traffic density per pheromone level in a multi-pheromone implementation as a function of algorithm's iteration), or it can produce a custom *evolving* heatmap, which we called *DNA-heatmaps*. DNA-heatmaps are a gray-scale 2D variant of 3D plots, useful for plotting data comprised of several independent values that evolve in time. Figure 8.6c shows an evolution of the response quality (color intensity) per problem (y-axis) in time (x-axis). The scripts that produce DNA-heatmaps are bundled-in with our post-processing module.

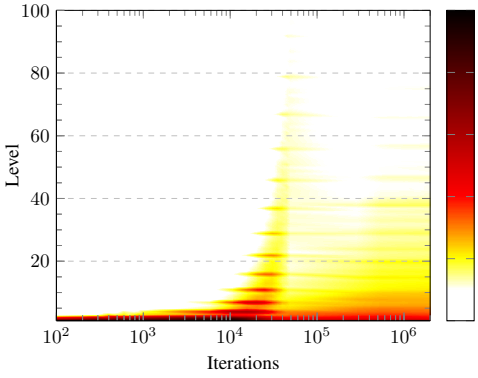
### 8.3.3 Extensibility and Configuration

AntE achieves a high degree of customizability due to the widespread use of the strategy design pattern. In Code Snippet 8.5 we present a fragment of a configuration of the algorithm. Note lines (1), (2) and (5), where a class-encapsulated behavior is passed as parameter. In line (4) we establish a numerical parameter and in line (7) a boolean parameter.

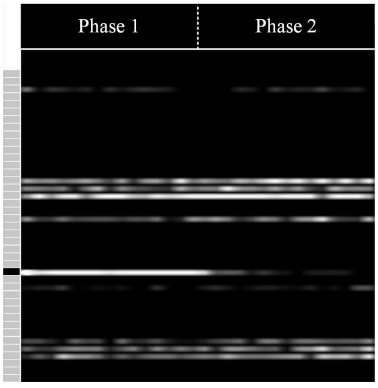
```
1 antConfig.setParameter(AntBehaviourConfig.ANT_ST,
      AcsStateTransition.class);
2 antConfig.setParameter(AntBehaviourConfig.SOLUTION_QM,
      AcsQualityMeasure.class);
3 antConfig.setParameter(AntBehaviourConfig.TTL_MAX, 8);
4
5 phConfig.setParameter(PheromoneConfig.PHEROMONE_COLLECTION,
      HashMapPheromone.class);
6
7 logConfig.setParameter(Logger.LOG_ENABLED, true);
```




**(a)** Linear plot (with Ti*k*Z LaTeX)




**(b)** Heat-map plot (with Ti*k*Z LaTeX)




**(c)** DNA-like heat-map (proprietary)

**Figure 8.6:** Examples of AntE logging outputs

**Code Snippet 8.5:** An example of configuration (1)

This configuration technique, coupled with the modular approach allows defining arbitrary behaviors. Consider the following example (Code Snippet 8.6). Our objective is to create a quality measure based on the square on the well-known hop per hit (HpH) metric. First we create a behavior class (line 1) as an extension of the predefined HpHQuality class, which encapsulates the calculation of the quality measure. Next, we introduce it into the configuration as the chosen behavior of the algorithm (line 12).

```
1  class SquareRootHpHQuality extends HpHQuality implements QualityMeasure
2  {
3    @Override
4    public float getQuality(Ant a)
5    {
6      float hphQuality = super.getQuality(a);
7      float squareRootHphQuality = Math.sqrt(hphQuality);
8      return squareRootHphQuality;
9    }
10 }
11
12 antConfig.setParameter(AntBehaviourConfig.SOLUTION_QM,
       SquareRootHpHQuality.class);
```

**Code Snippet 8.6:** An example of configuration (2) - behavior class definition

Note that the change of configuration can be done during the execution of the algorithm, allowing interesting and unusual observations and experiments.

### 8.3.4 Low Level Optimization Techniques

The choice between the programming languages in which to write the middleware is a trade-off. The main benefits of Java are obvious: portability and hardware and software abstraction. Java, however, produces slower code than the corresponding solutions in C++. To counteract this we were forced to design and apply a series of low level optimization techniques. Here we list some of the most notable ones.

The most basic step is not to permit Java to perform, so called, boxing and unboxing of numerical values. In Java, numerical values are both object-based and primitives. Boxing and unboxing are the names given to the conversions between the two types. The conversions are, typically, transparent from the programmer's

point of view and unnoticeably quick. However, in our case they were very frequent and consumed an important portion of the CPU time.

This prompted us to eliminate completely one of the two types of the numerals. We decided to exclude the object-based numerical values from our code, which benefited us thrice. First, objects are larger in terms of memory than their primitive counterparts (64 bytes versus 2 - 8 bytes). Second, as mentioned, the unboxing is avoided completely. And third, the creation and destruction of objects and, in consequence, the garbage collector usage are reduced significantly.

As a result, we were forced to rewrite the generic Java collections, such as *ArrayList* and *HashSet* (which use object-based numerals) into their corresponding, primitive-based counterparts. This was an opportunity to incorporate in the newly created classes (*PrimitiveIntArrayList* and *PrimitiveFloatArrayList*) low-level methods, which accelerated common operations, such as *max*, *min*, *sum* etc. This change alone helped to reduce the memory consumption by a margin of 75% and the CPU consumption by 90%.

The biggest challenge in ACO related middlewares is the pheromone container. It is very often accessed and it is read and updated with similar frequency, which excludes a write- or read-focus optimization. Our solution, alongside the aforementioned primitive-based computation, was to handle the pheromone writes and reads in batches, via the low level method *System.arraycopy* (Code Snippet 8.7, from *HashMapPheromone* class). In order to benefit from this technique, the code must be redesigned with batch operations in mind. The difference of the execution time of the batch-approach with respect to the value-by-value-approach is above 71% in favor of the former.

```
1  public void setAll(float[] pheromone)
2  {
3    System.arraycopy(pheromone, 0, elementData, 0, pheromone.length);
4    size = pheromone.length;
5  }
6
7  public float[] getAll()
8  {
9    float[] pheromone = new float[size];
10   System.arraycopy(elementData, 0, pheromone, 0, size);
11
12   return pheromone;
13 }
```

**Code Snippet 8.7:** Batch operations

Another common efficiency bottleneck is the method that establishes if a given element is present or absent from a data collection, typically called *.contains*. It

is, at best, of O(log(n)) complexity. Quite often however, as in the case of unsorted lists, it is O(n). We designed a technique which guarantees a numerical complexity of O(1). Our method is possible if the elements stored in the collection are labeled with an index, and it is especially effective if the range of the values of the indices is known, limited and relatively small.

First we instantiate a bit array of size *max_ index*. Each time an element is inserted into the collection, the field in the bit array that corresponds to the index of the inserted element is set to *true*. The opposite happens when the last element of a given index is removed. This way the *.contains* check is reduced to reading the bit corresponding to the examined element. The memory increase is modest. For max_index of 32768 the bit array only uses $4kB$ of memory. See Code Snippet 8.8 for a simplified example. This approach has reduced the overall execution time by 16.7%. We believe that this technique has a wide range of uses, well beyond our middleware.

```
1  class MonitoredCollection<T> extends Collection<T>
2  {
3  private BitSet cMonitor = new BitSet(max_index);
4
5  (...)
6
7  @Override
8  public void add(int index, T data)
9  {
10   super.add(index, data);
11   cMonitor.set(index);
12 }
13
14 @Override
15 public void remove(int index)
16 {
17   super.remove(index);
18   cMonitor.clear(index);
19 }
20
21 @Override
22 public boolean contains(int index)
23 {
24   return cMonitor.get(index);
25 }
26
27 (...)
```

**Code Snippet 8.8:** *contains* operation

**Table 8.2:** Execution Times, Intel Xeon X3430

| Worker Threads | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Logger Threads | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Physical Cores | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Time [s] | 8.4 | 6.7 | 6.4 | 6.9 | 7.0 | 7.3 | 8.0 |
| Ants$\times s^{-1}$ [$\times 10^4$] | 11.9 | 14.9 | 15.6 | 14.3 | 14.3 | 13.6 | 12.5 |
| Ant-steps$\times s^{-1}$ [$\times 10^6$] | 2.4 | 2.9 | 3.1 | 2.8 | 2.8 | 2.7 | 2.4 |

**Table 8.3:** Memory Usage, Intel Core i5 540m

| Graph size | 256 | 1024 | 4096 | 16384 | 65536 |
|---|---|---|---|---|---|
| Total memory [MB] | 2.6 | 9.5 | 37.1 | 147.4 | 588.8 |
| Memory per node [kB] | 10.2 | 9.3 | 9.0 | 8.9 | 8.9 |

## 8.4 Efficiency and Scalability

Having described some of the most notable elements of our implementation we proceed to the analysis of the efficiency and scalability. We start with, arguably, the most crucial parameter, which is the overall execution time. We executed our middleware under the following conditions: a graph with 1024 nodes with $10^6$ ants released simultaneously onto it. Each ant performs a full search of $TTL = 20$ steps and saves the search results in a file. The hardware configuration is: Intel Xeon X3430 (8MB Cache, 2.40 GHz), limited to 2 of the 4 physical cores and 2GB Ram. In Table 8.2 we present the execution times in function of processing threads used. We conclude that in the best case of 3 worker threads, the task is terminated in under 6.5 seconds. At the peak, the efficiency of ant processing was about $15.6 \times 10^4$ ants per second and $3.1 \times 10^6$ *ant-steps* per second. By ant-step we understand a full processing of one ant in one node. It includes: the resource query, the pheromone read and write as well as the state transition.

The same experiment on a far slower Intel Core i5 540m takes roughly 43 seconds. In both cases the CPU resources consumption reached 98% per core on average, which demonstrates an above-average effectiveness of our short-circuit approach. This suggests that the CPU-related scalability of AntE is high, allowing to benefit from multicore processors in a satisfying degree.

Another common limitation of other existing middlewares is the size of the graph they can produce and process. Classical graph-related problems tend to be rather small, mostly under $10^4$ nodes. This means that a trivial (non-memory efficient) implementation would possibly suffice. Still, we attempted to reduce the memory consumption to a relative minimum, keeping in mind that the shorter CPU time, within reasonable bounds, is favored. In Table 8.3 we summarize the memory consumption in function of graph size. It can be observed that the property is

very weakly sublineal, and therefore, scalable. On average one node occupies $8.9kB$ of memory. Under these conditions, with a 32-bit version of Java the upper limit of the graph size is situated at around $10^5$ nodes, on a 64-bit version it is virtually unlimited.

The stability of the middleware is also quite important. In our extensive experiments we were able to sustain a constant, uninterrupted flow of ants during 12 hours, generating roughly $7GB$ of log files. We claim that, with AntE, experiments of the order of magnitude of $10^{10}$ ants are feasible.

## 8.5 Existing Implementations

We have used our middleware in several experimental setups and one possible commercial deployment.

The early versions of the software have been used to examine ACO algorithms under the condition of the dynamism of the problem space [12]. This is a largely unexplored area of ACO, however, it was essential in our study of the applicability of ACO to P2P networks. The unique property of supporting modifications of the graph while the algorithm is under execution allowed us to observe insufficiencies of ACO in this regard [8].

After having defined and demonstrated the problem, a proposal of a P2P-compatible ACO was deployed as an experimental module, formulated completely within AntE [18]. We opted for counteracting the problems associated to the dynamism of the graph by correcting the pheromone paths with the help of a new type of ants, the *graph structure diffusion* ants. The coding effort involved in the aforementioned line of research, albeit seemingly complex, was in fact minimal and could have been completed by researchers with limited knowledge of programming. In this line of research the graph sizes ranged between 1024 and 32768 and the experiment lengths were of $10^5$ iterations.

Our middleware has also been used to implement a deployable piece of software that served as a recommendation service for rehabilitation tasks for people with Acquired Brain Injuries [19]. This extensive software can be used as both: an experimental module and a deployable application. In its essence it was an example of a practical application of ACO algorithms. Here, we treated the graph nodes as rehabilitation tasks, the pheromone as a similarity measure between them, and the ants as a representation of a query issued for a patient. All of the changes were possible strictly within what the configuration of our middleware permits.

Our most recent research is centered on the question of resolving multi-class queries in P2P networks with ACO. We extended our software with multi-pheromone capacities in order to experiment with the effects of multiple types of ants and

how they impact the overall efficiency of ACO. We also increased the length of the experiments to $4 \times 10^6$ iterations. Working with our middleware enabled us to put the formulated ideas to a test quickly and obtain a very early experimental validation, which, in turn, was a way to avoid or abandon unpromising concepts.

## 8.6 Conclusions and Future Works

In this paper we presented a customizable ACO middleware, AntE. We elaborated on the novelty of our software, as well as outlined some of its more interesting features. We discussed its performance and scalability, and described its architectural design alongside some selected optimization techniques, which, we hope, will benefit the ACO community.

In the direct future we will focus on the release of our software as open source and we will bundle it with a documentation allowing users to use it to its full potential. In addition, we continually rework and update our source code, improving safety, stability, as well as the key aspects: scalability, speed and customizability. In this respect, we are currently considering new and promising ACO-hibridization techniques that will be included as optional components of the next release of the AntE software infrastructure.

We would also like to compare the effectiveness of our approach with general purpose environments, such as *HeuristicLab* [145]. HeuristicsLab has been in development since 2002 and it has become a very extensive and configurable environment for broadly understood algorithmic experimentation, with a modern GUI, visual algorithm and experiment designers, as well as analytical tools. Due the scope of the software, HeuristicsLab bares no characteristic of a middleware, but is, in fact, a meta-level environment. Therefore, we argue that AntE should not be perceived as a competing, but rather complementary approach.

A possible continuation of our work could focus on the conversion of AntE into a plug-in of HeuristicsLab. This would enable us to benefit from the powerful framework HeuristicsLab is. Such a plug-in, supporting multiple pheromone levels, graph structure diffusion ants and resolution of multi-class queries in P2P networks with ACO, has, to our best knowledge, not been implemented.

## 8.7 Acknowledgments

Kamil Krynicki is a FPI fellow of Universitat Politècnica de València, number 3117. This work received support from the Ministry of Science and Innovation under the National Strategic Program of Research, Project TIN2010-20488 and TIN2014-60077-R, as well as the National Institute of Informatics, Tokyo, Japan.

# Chapter 9

# Discussion

The results stemming from our research are multiple. In order to coherently analyze them we must recall the two global challenges we put forward in the initial statement of this thesis.

1. Equip ACO metaheuristic with capability to efficiently carry out P2P resource queries, withstanding dynamism of the underlying network topology

2. Equip ACO metaheuristic with semantic/multiclass routing capabilities in P2P networks in a non-trivial manner

Bearing in mind that any solution to the above challenges must not tread beyond what the ACO metaheuristic offers in terms of decentralization, locality and applicability scope. These high-level objectives can be split and expressed as a list of more easily manageable tasks:

1. Extract the desired and the undesired properties of algorithms for search in P2P networks.

2. Experimentally analyze existing strategies against the extracted properties and establish the most promising ones for further development

3. Address the challenge 1 (dynamic graphs). Incorporate new strategies into the algorithms in question and perform an efficiency analysis.

4. Locate a real-life problem that could benefit from the newly created dynamic graph ACO strategy and demonstrate its usefulness.

5. Address the challenge 2 (semantic/multiclass routing). Incorporate new strategies into the algorithms in question and perform an efficiency analysis.

6. Identify a real-life problem that could benefit from the newly created semantic routing ACO strategy and demonstrate its usefulness.

A major part of our early work revolved around the tasks 1 and 2. In our paper *On the performance of ACO–based methods in P2P resource discovery* we formulate the *query-resource* (*q-r*) requirements for an effective P2P ACO-based query. In addition, we present the first approximation of the semantic capacities of an ACO algorithm - the Routing Concept, based on the work by Michlmayr [15]. We also conclude that the topology of the underlying graph, unless exploited in a predetermined way, is not of crucial importance. To demonstrate this fact we design a topology-aware ACO extension, called *TRO*, and perform experiments in various graph topologies with and without it active. The main contribution stemming out from this research was a more formal look at some intuitive problems with the use of ACO as a query routing algorithm in P2P environments.

Our following article (*Ant Colony Optimization for resource querying in dynamic peer-to-peer grids*) is a direct continuation of the first one. Here we took on the dynamic aspects of P2P networks: the variating number and location of nodes as well as the amount of resources, put in the context of computer grids. The discussion focused on the comparison of two techniques: single agent against multi agent, and the impact the network dynamism has on the convergence quality and speed of ACO algorithms. The results informed us that the idea of multi-agent techniques, if used at all, must be very carefully designed and as lightweight as possible; a conclusion, that has be taken into account in the global solution. In this research we used, what had been determined the best candidate: the aptly named Routing Concept Ant Colony System (*RC-ACS*), which is a crossover of classical Ant Colony System with the Routing Concept idea. We also coin a new term: pheromone *reconvergence*. Reconvergence is used when the pheromone convergence state of a network has been perturbed and invalidated, which causes the convergence to occur anew. Reconvergence is a more difficult task, because rather than starting from an uninitialized state, it must start from an incorrectly initialized one.

Afterwards we continued with task 3 and designed a model that tackles the effects of the dynamism of the network. The study *A Diffusion-Based ACO Resource Discovery Framework for Dynamic P2P Networks* focuses on one of the dynamism types extracted in the previous work: the growth and, more generally, the evolution of the content and quality of resource repositories. We propose a novel solution by extending the classical ACS and our RC-ACS with, what we refer to as, *information diffusion* ability. The information diffusion is a simple network events notification system that allows nodes to correct their pheromone paths. It was designed not to violate the q-r prerequisite through adding centralized knowledge about the state of the network. We present two information diffusion strategies and examine their impacts on the convergence and reconvergence of the system. We determine that the model equipped with information diffusion is statistically

superior to a one without it. We, therefore, provide an approximation of the solution to the problem of the resource dynamism and the dynamism in general in P2P networks.

Task 4 consisted of utilizing the to-date research in a real-life setting. We applied RC-ACS with information diffusion the problem of learning unit recommendation in the context of ability learning due to acquired brain injuries. The problem of selecting optimal learning resources from a set of given ones was a good example of one that can benefit from our improvements: routing concepts and the information diffusion. In the paper *An ACO-based personalized learning technique in support of people with Acquired Brain Injury* we present a fully functional learning unit recommendation software centered exclusively on class-based routing achieved through routing concepts. Due to the time limitations we were unable to examine the impact of the network dynamism on our solution. Nevertheless, we showed, with high statical significance, that our ACO-based model produces quality recommendations.

Next, in order to execute task 5, we combined partial solutions and transitioned to a global solution to the problem of semantic routing. The thus far used solution, RC-ACS, is largely sufficient for simple cases, but it fails when the number of routing concepts $RC$, or resource classes $C$ grows beyond a certain threshold, which was established to be located between $C = 10$ and $C = 100$. The authors of the multipheromone, pheromone-per-concept approach [15], which was the motivation for the RC idea, conclude their work with similar statements. The key notion that guided the research from this point on was to allow natural emergence of pheromone types in the system, rather than to establish the direct one pheromone for one routing concept. This task was resolved in two substeps. First we proposed a new mathematical model, fully in-tune with the q-r principles, and demonstrated that it did not impair the efficiency of the algorithm when semantic search is not used. Once this was complete we presented and analyzed it in the context of semantic search.

Our global solution is called *Angry Ant Framework* ($AAF$). In AAF the search agents can create ad-hoc multipheromone models (*stratify* the pheromone in layers) depending on the satisfaction with the efficiency of the queries in the existing layers. It is an important improvement over the static, preassigned pheromone-per-concept idea. Surprisingly, the inclusion of the dynamic pheromone stratification into the classical ACS improved its efficiency even when not theoretically necessary, when maintaining efficiency was sufficient and expected. Such was our conclusion from the article *A non–Hybrid Ant Colony Optimization Heuristic for Convergence Quality*. This convinced us that AAF had a more profound benefit on the search process in P2P networks, but we still cannot conclusively account for the positive impact seen here. Having shown this we examined AAF with semantic search, represented by the simple idea of multi-class queries (*An Efficient ACO Strategy for the Resolution of Multi-Class Queries*). Between the two afore-

mentioned publications we improved the original AAF mathematical design with a process we named entropic ant reassignment. The entropic ant reassignment allows the search agents, not only to dynamically create new pheromone types as needed, but also to reassign themselves to other, existing ones, provided they consider it beneficial. We called the resulting algorithm *EntropicAAF*.

Throughout our research we kept on evaluating ACO algorithms in increasingly uncommon and difficult instances of P2P problems. Being unable to locate an existing middleware of sufficient capabilities we created our own, called AntElements. It combined all the required properties, such as dynamism, routing concepts and a high degree of flexibility, with an unprecedented stability and execution speed. It allowed us to perform some of the longest (order of magnitude of $10^9$ of agents), most extensive (order of magnitude of $10^5$ network nodes) and most precise (pheromone analysis after each agent release) ACO-related P2P experiments to date. Upon the research competition our software solution was published as open-source for the community to use.

# Chapter 10

# Conclusions and Future Works

In this thesis we have shown approaches to efficient semantic searches in P2P networks. We propose a number of solutions, that cover a major portion of the possibilities:

- If the environment is structured, we propose RC-ACS algorithm with the TRO extension (a static multipheromone model)
- If the environment is dynamic, but the semantic classification of the resources is small, we propose the use of RC-ACS with information diffusion (a static multipheromone model)
- If the resource classification is large, variable or unknown, we propose the use of EntropicAAF (a dynamic multipheromone model)

See Table 10.1 for a short summary of our contributions.

Thus, the two challenges raised in chapter 1 have been achieved throughout the course of the investigation under the scope of this thesis. However, of the six tasks mentioned in chapter 9 the final one: *Identify a real-life problem that could benefit from the newly created semantic routing ACO strategy and demonstrate its usefulness* has not been addressed fully due to time limitations. As of today we were able to locate a problem that could benefit immensely from the efficiency offered by EntropicAAF: large-scale vehicle routing. This topic is our current focus, due to the fact that, as it was noticed, large scale traffic can be modeled as a graph with agents that carry class-based objectives.

With respect to the pure field of semantic searches in P2P, we would like to apply our algorithms to existing problems. One example of such a problem is the widely distributed multimedia content. Our multi-class algorithms find a very natural

**Table 10.1:** Summary of algorithmic contributions

| Contribution | Description | Source |
|---|---|---|
| RC-ACS | A static multipheromone variant of ACS, usable in semantic queries in static P2P networks | Chapter 2 |
| TRO | A path post-processing algorithm for any ACO algorithm for P2P networks with hypercube topologies | Chapter 2 |
| ACO Diffusion | A P2P- and ACO-compatible algorithm that mitigates the effects of the network dynamism on ACO | Chapter 4 |
| AAF | A dynamic multipheromone variant of ACS with ad-hoc pheromone stratification | Chapter 6 |
| EntropicAAF | An improvement over AAF, with more efficient pheromone utilization | Chapter 7 |

representation in this problem, as the metadata of multimedia files can be viewed as a semantical classification, which can be easily exploited with EntropicAAF.

As future works we would like to clearly explain the query efficiency increase in single class queries with EntropicAAF. As mentioned in the chapter 9 this improvement is not fully accounted for or understood. In addition, we would like to perform a study on EntropicAAF with the information diffusion enabled. Therefore allowing EntropicAAF to be executed in dynamic graphs.

To the best of our knowledge EntropicAAF is the only available dynamic multipheromone ACO model and a substantial improvement over the static multipheromone ACO models. Furthermore, EntropicAAF is in all likelihood the most efficient ACO algorithm for semantic searches in P2P networks in general, and therefore, the possible future state-of-the-art. We would like to promptly perform a statistical study in order to check and prove the accuracy of this statement.

To conclude we would like to mention that the dynamic multipheromone approach of AAF is, thus far, one of its kind and it would be of high importance to examine its broader implications and perquisites, well outside the field of semantic searches in P2P. It is, on its own, a novel idea that has the potential to benefit algorithmic approaches different to ACO.

# Bibliography


[1] *Internet Protocol, Version 6 (IPv6) Standard*. 1998. URL: https://tools.ietf.org/html/rfc2460 (cit. on p. 1).

[2] *Hypertext Transfer Protocol Version 2 (HTTP/2) Proposed Standard*. 2015. URL: https://tools.ietf.org/html/rfc7540 (cit. on p. 1).

[3] D. Tsoumakos and N. Roussopoulos. "Analysis and comparison of P2P search methods". In: *Proceedings of the 1st international conference on Scalable information systems - InfoScale '06* (2006), 25–es. DOI: 10.1145/1146847.1146872 (cit. on p. 2).

[4] T. Klingberg. *Gnutella 0.6*. 2002 (cit. on pp. 2, 10, 19).

[5] M. Dorigo. "Optimization, Learning and Natural Algorithms". PhD thesis. Politecnico di Milano, 1992 (cit. on p. 2).

[6] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. "The self-organizing exploratory pattern of the argentine ant". In: *Journal of Insect Behavior* 3.2 (03/1990), pp. 159–168. ISSN: 0892-7553. DOI: 10.1007/BF01417909 (cit. on pp. 3, 126).

[7] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. "Self-organized shortcuts in the Argentine ant". In: *Naturwissenschaften* 76 (1989), pp. 579–581. ISSN: 00281042. DOI: 10.1007/BF00462870 (cit. on pp. 3, 11, 52, 126).

[8] K. Krynicki, J. Jaen, and J. A. Mocholí. "Ant colony optimisation for resource searching in dynamic peer-to-peer grids". In: *International Journal of Bio-Inspired Computation* 6.3 (2014), pp. 153–165. ISSN: 1758-0366. DOI: 10.1504/IJBIC.2014.062634 (cit. on pp. 4, 6, 70, 72, 144, 145, 184).

[9] M. Mavrovouniotis and S. Yang. "An Immigrants Scheme Based on Environmental Information for Ant Colony Optimization for the Dynamic Travelling Salesman Problem". In: *EA-2011: Proceedings of the 10th International Conference on Artificial Evolution*. 2011. ISBN: 9781450334723. DOI: 10.1145/2739480.2754651 (cit. on p. 4).

[10] M. Mavrovouniotis and S. Yang. "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors". In: *Applied Soft Computing Journal* 13.10 (2013), pp. 4023–4037. ISSN: 15684946. DOI: 10.1016/j.asoc.2013.05.022. URL: http://dx.doi.org/10.1016/j.asoc.2013.05.022 (cit. on p. 4).

[11] M. Mavrovouniotis, F. M. Müller, and S. Yang. "An Ant Colony Optimization Based Memetic Algorithm for the Dynamic Travelling Salesman Problem". In: *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*. GECCO '15. Madrid, Spain: ACM, 2015, pp. 49–56. ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754651 (cit. on pp. 4, 144).

[12] K. Krynicki, J. Jaen, and J. A. Mocholí. "On the performance of ACO-based methods in p2p resource discovery". In: *Applied Soft Computing Journal* 13 (2013), pp. 4813–4831. ISSN: 15684946. DOI: 10.1016/j.asoc.2013.07.022 (cit. on pp. 4, 6, 70, 78, 80, 100, 101, 136, 144, 145, 156, 184).

[13] E. Michlmayr. "Self-organization for search in peer-to-peer networks". In: *Studies in Computational Intelligence* 69 (2007), pp. 247–266. ISSN: 1860949X. DOI: 10.1007/978-3-540-72693-7_13 (cit. on pp. 4, 11, 101, 147, 162).

[14] H. T. Shen, Y. Shu, and B. Yu. "Efficient Semantic-Based Content Search in P2P Network". In: *IEEE Transactions on Knowledge and Data Engineering* 16.07 (07/2004), pp. 813–826. ISSN: 1041-4347. DOI: 10.1109/TKDE.2004.1318564 (cit. on p. 4).

[15] E. Michlmayr. *Ant Algorithms for Self-Organization in Social Networks*. Vienna, Austria: Vienna University of Technology, 2006 (cit. on pp. 4, 14, 17, 23, 24, 26, 29, 46, 53, 55, 57, 70–72, 76–78, 188, 189).

[16] K. Krynicki, M. E. Houle, and J. Jaen. "An Efficient ACO Strategy for the Resolution of Multi-Class Queries". In: Under Review. 2015 (cit. on pp. 4, 7).

[17] M. Dorigo and L. M. Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions*

*on Evolutionary Computation* 1 (1997), pp. 53–66. ISSN: 1089778X. DOI: 10.1109/4235.585892 (cit. on pp. 5, 11, 13, 24, 50, 51, 53, 100, 126, 128, 145).

[18] K. Krynicki, J. Jaen, and A. Catala. "A diffusion-based ACO resource discovery framework for dynamic p2p networks". In: *2013 IEEE Congress on Evolutionary Computation*. IEEE, 06/2013, pp. 860–867. ISBN: 978-1-4799-0454-9. DOI: 10.1109/CEC.2013.6557658 (cit. on pp. 6, 55, 65, 184).

[19] K. Krynicki, J. Jaen, and E. Navarro. "An ACO-based personalized learning technique in support of people with Acquired Brain Injury". In: *Applied Soft Computing Journal* (). Under Review (cit. on pp. 7, 184).

[20] K. Krynicki, M. E. Houle, and J. Jaen. "A non-Hybrid Ant Colony Optimization Heuristic for Convergence Quality". In: *IEEE International Conference On Systems, Man, and Cybernetics*. Accepted for presentation. 2015 (cit. on pp. 7, 145, 148, 153, 155, 157, 159, 162).

[21] K. Krynicki and J. Jaen. "AntElements: An Extensible and Scalable Ant Colony Optimization Middleware". In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. Madrid, Spain: ACM, 2015, pp. 1109–1116. ISBN: 978-1-4503-3488-4. DOI: 10.1145/2739482.2768464 (cit. on p. 7).

[22] A. S. Tanenbaum and M. Van Steen. *Distributed Systems: Principles and Paradigms, 2/E*. 2007, p. 686. ISBN: 9780132392273. DOI: 10.1002/1521-3773(20010316)40:6<9823::AID-ANIE9823>3.3.CO;2-C (cit. on p. 10).

[23] University of California. *Seti@Home. University of California* (cit. on pp. 10, 50).

[24] *Collatz Conjecture*. 2008 (cit. on p. 10).

[25] *Top500*. 2012 (cit. on p. 10).

[26] L. Wang. "SoFA: An expert-driven, self-organization peer-to-peer semantic communities for network resource management". In: *Expert Systems with Applications* 38 (2011), pp. 94–105. ISSN: 09574174. DOI: 10.1016/j.eswa.2010.06.020 (cit. on pp. 10, 94).

[27] G. Beydoun, G. Low, N. Tran, and P. Bogg. "Development of a peer-to-peer information sharing system using ontologies". In: *Expert Systems with*

*Applications* 38 (2011), pp. 9352–9364. ISSN: 09574174. DOI: 10.1016/j.eswa.2011.01.104 (cit. on p. 10).

[28] Teliqo. *Voice over Internet Protocol (VoIP) Definition and Overview*. 2012 (cit. on p. 10).

[29] M. Dorigo and G. D. Caro. "Ant colony optimization: a new meta-heuristic". In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* 2 (1999). DOI: 10.1109/CEC.1999.782657 (cit. on pp. 11, 23, 70–72, 98).

[30] T. Stützle and H. H. Hoos. "Improvements on Ant System: Introducing MAX-MIN Ant System". In: *Proceedings of the International Conference in Norwich, U.K.* 1996, pp. 245–249. ISBN: 3211830871. DOI: 10.1007/978-3-7091-6492-1_54 (cit. on p. 11).

[31] C. G. Santillán, L. C. Reyes, E. Meza Conde, E. Schaeffer, and G. Castilla Valdez. "A Self-Adaptive Ant Colony System for Semantic Query Routing Problem in P2P Networks". In: *Computación y Sistemas* 13.4 (2010), pp. 433–448 (cit. on pp. 11, 16, 17, 23, 46, 65, 78, 147).

[32] B. Ghit, F. Pop, and V. Cristea. "Epidemic-style global load monitoring in large-scale overlay networks". In: *Proceedings - International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2010*. 2010, pp. 393–398. ISBN: 9780769542379. DOI: 10.1109/3PGCIC.2010.62 (cit. on p. 11).

[33] A. Colorni, M. Dorigo, and V. Maniezzo. "Distributed Optimization by Ant Colonies". In: *Proceedings of the European Conference on Artificial Life*. 1991, pp. 134–142. ISBN: 8522105251 (cit. on pp. 11, 52, 71, 145).

[34] J. Jaén, J. A. Mocholí, A. Catalá, and E. Navarro. "Digital ants as the best cicerones for museum visitors". In: *Applied Soft Computing Journal* 11 (2011), pp. 111–119. ISSN: 15684946. DOI: 10.1016/j.asoc.2009.11.002 (cit. on pp. 11, 52, 70).

[35] J. A. Mocholi, J. Jaen, A. Catala, and E. Navarro. "An emotionally biased ant colony algorithm for pathfinding in games". In: *Expert Systems with Applications* 37 (2010), pp. 4921–4927. ISSN: 09574174. DOI: 10.1016/j.eswa.2009.12.023 (cit. on pp. 11, 70).

[36] M. Dorigo, V. Maniezzo, and A. Colorni. "Ant system: Optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man,*

*and Cybernetics, Part B: Cybernetics* 26 (1996), pp. 29–41. ISSN: 10834419. DOI: 10.1109/3477.484436 (cit. on pp. 13, 70).

[37] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. *Search and replication in unstructured peer-to-peer networks*. 2002. DOI: 10.1145/511399.511369 (cit. on p. 17).

[38] G. D. Caro and M. Dorigo. "AntNet: Distributed stigmergetic control for communications networks". In: *Journal of Arti cial Intelligence Research* 9 (1998), pp. 317–365 (cit. on pp. 18, 20, 162).

[39] G. D. Caro, F. Ducatelle, and L. M. Gambardella. "AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks". In: *European Transactions on Telecommunications* 16 (2005), pp. 443–455. ISSN: 1124318X. DOI: 10.1002/ett.1062 (cit. on pp. 18, 162).

[40] R. Schoonderwoerd, O. E. Holland, J. L. Bruten, and L. J. M. Rothkrantz. "Ant-Based Load Balancing in Telecommunications Networks". In: *Adaptive Behavior* 5.2 (01/1997), pp. 169–207. ISSN: 1059-7123. DOI: 10.1177/105971239700500203 (cit. on p. 18).

[41] M. T. Prinkey. *An Efficient Scheme for Query Processing on Peer-to-Peer Networks*. Tech. rep. 2001 (cit. on pp. 19, 75).

[42] J.-q. Li, Q.-k. Pan, and S.-x. Xie. "Research on Peer Selection in Peer-to-Peer Networks using Ant Colony Optimization". In: *Natural Computation, 2008. ICNC '08. Fourth International Conference on*. Vol. 7. 2008, pp. 516–520. DOI: 10.1109/ICNC.2008.264 (cit. on p. 19).

[43] A. Crespo and H. Garcia-Molina. "Semantic overlay networks for P2P systems". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 3601 LNAI. 2005, pp. 1–13. ISBN: 3540297553. DOI: 10.1007/11574781_1 (cit. on pp. 20, 58, 70).

[44] E. Michlmayr, A. Pany, and G. Kappel. "Using taxonomies for content-based routing with ants". In: *Computer Networks* 51 (2007), pp. 4514–4528. ISSN: 13891286. DOI: 10.1016/j.comnet.2007.06.015 (cit. on pp. 21, 129, 147, 176).

[45] K. A. Publishers and G. Talbi. "A Taxonomy of Hybrid Metaheuristics". In: (2002), pp. 541–564 (cit. on p. 21).

[46] R. Raidl. "A Unified View on Hybrid Metaheuristics". In: (2006), pp. 1–12 (cit. on p. 21).

[47] Q. Duan, T. W. Liao, and H. Z. Yi. "A comparative study of different local search application strategies in hybrid metaheuristics". In: *Applied Soft Computing Journal* 13 (2013), pp. 1464–1477. ISSN: 15684946. DOI: 10.1016/j.asoc.2012.05.016 (cit. on p. 22).

[48] F. Glover. "Tabu Search - Part I". In: *ORSA Journal on Computing* 1 (1989), pp. 190–206. ISSN: 0899-1499. DOI: 10.1287/ijoc.1.3.190 (cit. on p. 22).

[49] 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 (cit. on p. 22).

[50] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup protocol for Internet applications". In: *IEEE/ACM Transactions on Networking* 11 (2003), pp. 17–32. ISSN: 10636692. DOI: 10.1109/TNET.2002.808407 (cit. on p. 23).

[51] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A Scalable Content Addressable Network*. Tech. rep. 2000, pp. 161–172. DOI: 10.1145/964723.383072 (cit. on p. 23).

[52] C. Blum and M. Dorigo. "The Hyper-Cube Framework for Ant Colony Optimization". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34 (2004), pp. 1161–1172. ISSN: 10834419. DOI: 10.1109/TSMCB.2003.821450 (cit. on p. 23).

[53] *ACM CSS*. Tech. rep. Association for Computing Machinery, 1998 (cit. on pp. 26, 57, 76).

[54] D. Stutzbach, S. Zhao, and R. Rejaie. In: *Multimedia Systems*. Vol. 13. 2007, pp. 35–50. ISBN: 0819461113. DOI: 10.1007/s00530-007-0079-8 (cit. on pp. 26, 57, 77, 136, 156).

[55] 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". In: *Swarm and Evolutionary*

*Computation* 1 (2011), pp. 3–18. ISSN: 22106502. DOI: 10.1016/j.swevo.2011.02.002 (cit. on p. 28).

[56] I. Foster. "What is the Grid ? A Three Point Checklist". In: *GRID today* 1 (2002), pp. 32–36. DOI: 10.1103/PhysRevE.66.031916 (cit. on pp. 50, 51).

[57] W. Hoschek, J. Jaen-martinez, A. Samar, H. Stockinger, and K. Stockinger. "Data Management in an International Data Grid Project". In: *Data Management* (2000), pp. 77–90. ISSN: 0302-9743 (Print) 1611-3349 (Online). DOI: 10.1007/3-540-44444-0 (cit. on p. 50).

[58] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. "Peer-to-Peer resource discovery in Grids: Models and systems". In: *Future Generation Computer Systems* 23 (2007), pp. 864–878. ISSN: 0167739X. DOI: 10.1016/j.future.2006.12.003 (cit. on p. 50).

[59] D. Talia and P. Trunfio. "Toward a synergy between P2P and grids". In: *IEEE Internet Computing* 7 (2003), p. 96. ISSN: 10897801. DOI: 10.1109/MIC.2003.1215667 (cit. on p. 50).

[60] Y. Deng, F. Wang, and A. Ciura. "Ant colony optimization inspired resource discovery in P2P Grid systems". In: *Journal of Supercomputing* 49 (2009), pp. 4–21. ISSN: 09208542. DOI: 10.1007/s11227-008-0214-0 (cit. on pp. 50, 51, 147).

[61] Y. Han and C. H. Youn. "A new grid resource management mechanism with resource-aware policy administrator for SLA-constrained applications". In: *Future Generation Computer Systems* 25 (2009), pp. 768–778. ISSN: 0167739X. DOI: 10.1016/j.future.2008.11.005 (cit. on p. 51).

[62] A. Brocco, A. Malatras, and B. Hirsbrunner. "Enabling efficient information discovery in a self-structured grid". In: *Future Generation Computer Systems* 26 (2010), pp. 838–846. ISSN: 0167739X. DOI: 10.1016/j.future.2010.02.007 (cit. on p. 51).

[63] A. Di Stefano, G. Morana, and D. Zito. "A P2P strategy for QoS discovery and SLA negotiation in Grid environment". In: *Future Generation Computer Systems* 25 (2009), pp. 862–875. ISSN: 0167739X. DOI: 10.1016/j.future.2009.03.001 (cit. on p. 51).

[64] S. S. Dhillon and P. Van Mieghem. "Performance analysis of the AntNet algorithm". In: *Computer Networks* 51 (2007), pp. 2104–2125. ISSN: 13891286. DOI: 10.1016/j.comnet.2006.11.002 (cit. on p. 51).

[65] A. Forestiero, C. Mastroianni, and G. Spezzano. "QoS-based dissemination of content in Grids". In: *Future Generation Computer Systems* 24 (2008), pp. 235–244. ISSN: 0167739X. DOI: 10.1016/j.future.2007.05.003 (cit. on p. 51).

[66] J. A. Mocholí, J. Jaen, K. Krynicki, and A. Catala. "TSACO: Extending a context-aware recommendation system with allen temporal operators". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7656 LNCS. 2012, pp. 253–260. ISBN: 9783642353765. DOI: 10.1007/978-3-642-35377-2_35 (cit. on pp. 52, 70).

[67] B. M. Mahmoudi Aicha. "Two hybrid ant algorithms for the general T-colouring problem". In: *Int. J. of Bio-Inspired Computation* 2.5 (2010), pp. 11–32 (cit. on p. 52).

[68] K. Sripanidkulchai, B. Maggs, and H. Zhang. "Efficient content location using interest-based locality in peer-to-peer systems". In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)* 3 (2003). ISSN: 0743-166X. DOI: 10.1109/INFCOM.2003.1209237 (cit. on p. 58).

[69] V. Cholvi, P. Felber, and E. Biersack. "Efficient Search in Unstructured Peer-to-Peer Networks ". In: *European Transactions on Telecomunications: Special Issue on p2p Networking and p2p Service*. Vol. 15. 2004 (cit. on p. 58).

[70] *Global IP Traffic Forecast and Methodology, 2006–2011*. Tech. rep. Cisco System Inc, 2011 (cit. on p. 58).

[71] E. Michlmayr, S. Graf, W. Siberski, and W. Nejdl. "Query Routing with Ants". In: *In Proceedings of the 1st Workshop on Ontologies in P2P Communities, ESWC2005*. 2005 (cit. on p. 70).

[72] G. Dong, W. W. Guo, and K. Tickle. "Solving the traveling salesman problem using cooperative genetic ant systems". In: *Expert Systems with Applications* 39 (2012), pp. 5006–5011. ISSN: 09574174. DOI: 10.1016/j.eswa.2011.10.012 (cit. on p. 70).

[73] A. O. Bozdogan and M. Efe. "Improved assignment with ant colony optimization for multi-target tracking". In: *Expert Systems with Applications* 38 (2011), pp. 9172–9178. ISSN: 09574174. DOI: 10.1016/j.eswa.2011.01.134 (cit. on p. 70).

[74] S. Hautphenne, K. Leibnitz, and M.-A. Remiche. In: *SIGMETRICS Perform. Eval. Rev.* 34.2 (09/2006), pp. 3–4. ISSN: 0163-5999. DOI: 10.1145/1168134.1168137 (cit. on p. 71).

[75] K. Tutschku, K. Pawlikowski, P. Tran-Gia, R. Pries, T. Hoßfeld, and K. Leibnitz. In: *Proceedings of the ATNAC 2004*. 12/2004, p. 8 (cit. on p. 71).

[76] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. "Content-Based Publish-Subscribe over Structured Overlay Networks". In: *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)* (2005). ISSN: 1063-6927. DOI: 10.1109/ICDCS.2005.19 (cit. on p. 71).

[77] K. M. Salama, A. M. Abdelbar, F. E. B. Otero, and A. A. Freitas. "Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery". In: *Applied Soft Computing Journal* 13 (2013), pp. 667–675. ISSN: 15684946. DOI: 10.1016/j.asoc.2012.07.026 (cit. on pp. 72, 129, 147).

[78] S. S. Shapiro and M. B. Wilk. "An analysis of variance test for normality (complete samples)". In: *Biometrika* 52.3-4 (1965), pp. 591–611 (cit. on p. 80).

[79] W. H. Kruskal and W. A. Wallis. "Use of Ranks in One-Criterion Variance Analysis". In: *Journal of the American Statistical Association* 47 (1952), pp. 583–621. ISSN: 0162-1459. DOI: 10.1080/01621459.1952.10483441 (cit. on p. 80).

[80] O. J. Dunn. "Multiple Comparisons Among Means". In: *Journal of the American Statistical Association* 56 (1961), pp. 52–64. ISSN: 0162-1459. DOI: 10.2307/2282330 (cit. on p. 80).

[81] WHO. *International classification of impairments, activities and participation (ICIDH2): document for field trial purposes*. Tech. rep. World Health Organization, Geneva, Switzerland, 1997 (cit. on p. 92).

[82] M. Faul, L. Xu, M. M. Wald, and V. G. Coronado. *Traumatic Brain Injury in the United States: Emergency Department Visits, Hospitalizations*

*and Deaths 2002–2006*. Tech. rep. U.S. Department of Health and Human Services, 2010 (cit. on p. 92).

[83] O. Zaiane. "Building a recommender agent for e-learning systems". In: *International Conference on Computers in Education, 2002. Proceedings.* (2002). DOI: 10.1109/CIE.2002.1185862 (cit. on p. 93).

[84] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval. *Context-aware recommender systems for learning: A survey and future challenges*. 2012. DOI: 10.1109/TLT.2012.11 (cit. on p. 93).

[85] M. Nilashi, O. B. Ibrahim, and N. Ithnin. "Hybrid recommendation approaches for multi-criteria collaborative filtering". In: *Expert Systems with Applications* 41 (2014), pp. 3879–3900. ISSN: 09574174. DOI: 10.1016/j.eswa.2013.12.023 (cit. on p. 93).

[86] Y. J. Yang and C. Wu. "An attribute-based ant colony system for adaptive learning object recommendation". In: *Expert Systems with Applications* 36 (2009), pp. 3034–3047. ISSN: 09574174. DOI: 10.1016/j.eswa.2008.01.066 (cit. on p. 93).

[87] S. B. Aher and L. M. R. J. Lobo. "Combination of machine learning algorithms for recommendation of courses in E-Learning System based on historical data". In: *Knowledge-Based Systems* 51 (2013), pp. 1–14. ISSN: 09507051. DOI: 10.1016/j.knosys.2013.04.015 (cit. on p. 93).

[88] S. Bakshi, A. K. Jagadev, S. Dehuri, and G.-N. Wang. "Enhancing scalability and accuracy of recommendation systems using unsupervised learning and particle swarm optimization". In: *Applied Soft Computing* 15 (2014), pp. 21–29. ISSN: 15684946. DOI: 10.1016/j.asoc.2013.10.018 (cit. on p. 93).

[89] S. G. Li and L. Shi. "The recommender system for virtual items in MMORPGs based on a novel collaborative filtering approach". In: *International Journal of Systems Science* (2013), pp. 1–16. ISSN: 0020-7721. DOI: 10.1080/00207721.2012.762560 (cit. on pp. 93, 94).

[90] C. M. Chen, H. M. Lee, and Y. H. Chen. "Personalized e-learning system using Item Response Theory". In: *Computers and Education* 44 (2005), pp. 237–255. ISSN: 03601315. DOI: 10.1016/j.compedu.2004.01.006 (cit. on p. 93).

[91] C. De Maio, G. Fenza, M. Gaeta, V. Loia, F. Orciuoli, and S. Senatore. "RSS-based e-learning recommendations exploiting fuzzy FCA for Knowledge Modeling". In: *Applied Soft Computing Journal* 12 (2012), pp. 113–124. ISSN: 15684946. DOI: 10.1016/j.asoc.2011.09.004 (cit. on pp. 93, 94).

[92] A. Baylari and G. A. Montazer. "Design a personalized e-learning system based on item response theory and artificial neural network approach". In: *Expert Systems with Applications* 36 (2009), pp. 8013–8021. ISSN: 09574174. DOI: 10.1016/j.eswa.2008.10.080 (cit. on p. 94).

[93] P. J. Eslinger. *Neuropsychological Interventions: Clinical Research and Practice*. New York, New York, USA: The Guilford Press, 2005, p. 360. ISBN: 978-1593851637 (cit. on pp. 94, 96).

[94] L. De-Marcos, C. Pages, J. J. Martínez, and J. A. Gutiérrez. "Competency-based learning object sequencing using particle swarms". In: *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*. Vol. 2. 2007, pp. 111–116. ISBN: 076953015X. DOI: 10.1109/ICTAI.2007.14 (cit. on p. 94).

[95] R. Sharma, H. Banati, and P. Bedi. "Adaptive content sequencing for e-learning courses using ant colony optimization". In: *Advances in Intelligent and Soft Computing*. Vol. 131 AISC. 2012, pp. 579–590. ISBN: 9788132204909. DOI: 10.1007/978-81-322-0491-6_53 (cit. on p. 94).

[96] F. H. Wang. "On extracting recommendation knowledge for personalized web-based learning based on ant colony optimization with segmented-goal and meta-control strategies". In: *Expert Systems with Applications* 39 (2012), pp. 6446–6453. ISSN: 09574174. DOI: 10.1016/j.eswa.2011.12.063 (cit. on p. 94).

[97] *Brain Injury Statistics*. Last visited: 2015-06-03 (cit. on p. 95).

[98] *Key facts and statistics*. Last visited: 2015-04-17 (cit. on p. 95).

[99] M. Chamberlain, V. Neumann, and A. Tennant. "Traumatic brain injury rehabilitation: Services, Treatements and Outcomes". In: *Principles and practice of treatment*. Ed. by V. Neumann. London: Chapman & Hall Medical, 1995, pp. 101–118 (cit. on p. 95).

[100] K. D. Cicerone, C. Dahlberg, J. F. Malec, D. M. Langenbahn, T. Felicetti, S. Kneipp, W. Ellmo, K. Kalmar, J. T. Giacino, J. P. Harley, L. Laatsch, P. A.

Morse, and J. Catanese. *Evidence-based cognitive rehabilitation: Updated review of the literature from 1998 through 2002*. 2005. DOI: 10.1016/j.apmr.2005.03.024 (cit. on p. 96).

[101] M. S. McKay and C. A. Mateer. *Introduction to cognitive rehabilitation: Theory and practice*. The Guilford Press, 1989, p. 414. ISBN: 978-0898627381 (cit. on p. 96).

[102] K. D. Cicerone, C. Dahlberg, K. Kalmar, D. M. Langenbahn, J. F. Malec, T. F. Bergquist, T. Felicetti, J. T. Giacino, J. P. Harley, D. E. Harrington, J. Herzog, S. Kneipp, L. Laatsch, and P. A. Morse. *Evidence-based cognitive rehabilitation: Recommendations for clinical practice*. 2000. DOI: 10.1053/apmr.2000.19240 (cit. on p. 96).

[103] C. Christiansen, B. Abreu, K. Ottenbacher, K. Huffman, B. Masel, and R. Culpepper. "Task performance in virtual environments used for cognitive rehabilitation after traumatic brain injury." In: *Archives of physical medicine and rehabilitation* 79 (1998), pp. 888–892. ISSN: 0003-9993. DOI: Doi10.1016/S0003-9993(98)90083-1 (cit. on p. 96).

[104] E. Navarro, V. López-Jaquero, and F. Montero. "HABITAT: a web supported treatment for Acquired Brain Injured". In: *Proceedings - The 8th IEEE International Conference on Advanced Learning Technologies, ICALT 2008*. 2008, pp. 464–466. ISBN: 9780769531670. DOI: 10.1109/ICALT.2008.151 (cit. on p. 96).

[105] F. J. Navarro, E. Navarro, and F. Montero. "HABITAT: A tool for interactive activities in the treatment of Acquired Brain Injury". In: *13th International Conference on Interacción Persona-Ordenador*. 2012, pp. 1–2. ISBN: 9781450313148. DOI: 10.1145/2379636.2379641 (cit. on p. 96).

[106] F. Montero, V. López-Jaquero, E. Navarro, and E. Sánchez. "Computer-aided relearning activity patterns for people with acquired brain injury". In: *Computers and Education* 57 (2011), pp. 1149–1159. ISSN: 03601315. DOI: 10.1016/j.compedu.2010.12.008 (cit. on pp. 97, 100, 104, 108).

[107] *Clasificación Nacional de Ocupaciones (in Spanish)*. Last visited: 2015-06-03 (cit. on pp. 100, 108).

[108] L. O'Rance and N. Fortune. *Disability in Australia: Acquired Brain Injury*. Australian Institute of Health and Welfare, 2007, p. 27. ISBN: 978-1740247412 (cit. on p. 108).

[109] G. Beni and J. Wang. "Swarm Intelligence in Cellular Robotic Systems". In: *Robots and Biological Systems: Towards a New Bionics? NATO ASI Series Volume 102*. 1993, pp. 703–712. ISBN: 3642634613. DOI: 10.1007/978-3-642-58069-7_38 (cit. on pp. 126, 144).

[110] Kwang Mong Sim and Weng Hong Sun. "Ant colony optimization for routing and load-balancing: survey and new directions". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 33 (2003), pp. 560–572. ISSN: 1083-4427. DOI: 10.1109/TSMCA.2003.817391 (cit. on pp. 126, 129, 147).

[111] W. J. Gutjahr. "ACO algorithms with guaranteed convergence to the optimal solution". In: *Information Processing Letters* 82.3 (05/2002), pp. 145–153. ISSN: 00200190. DOI: 10.1016/S0020-0190(01)00258-7 (cit. on p. 126).

[112] T. Stützle and H. H. Hoos. "MAX-MIN Ant System". In: *Future Generation Computer Systems* 16 (2000), pp. 889–914. ISSN: 0167739X. DOI: 10.1016/S0167-739X(00)00043-1 (cit. on p. 126).

[113] T. Stützle and M. Dorigo. "A short convergence proof for a class of ant colony optimization algorithms". In: *IEEE Transactions on Evolutionary Computation* 6 (2002), pp. 358–365. ISSN: 1089778X. DOI: 10.1109/TEVC.2002.802444 (cit. on p. 126).

[114] J. Nong and L. Jin. "A convergence proof for ant colony algorithm". In: *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, CSO 2009*. Vol. 2. 2009, pp. 974–977. ISBN: 9780769536057. DOI: 10.1109/CSO.2009.305 (cit. on p. 126).

[115] T. Stützle and H. H. Hoos. "The Max-Min ANT System and Local Search for Combinatorial Optimization Problems". In: *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Ed. by S. Voß, S. Martello, I. H. Osman, and C. Roucairol. Boston, MA: Springer US, 1998, pp. 313–329. ISBN: 978-0-7923-8369-7. DOI: 10.1007/978-1-4615-5775-3 (cit. on pp. 127, 128).

[116] D. Wolpert. *No free lunch theorems for search*. Tech. rep. Technical Report SFI-TR-95-02-010 (Santa Fe Institute), 1995, pp. 1–38. DOI: 10.1145/1389095.1389254 (cit. on p. 127).

[117] D. H. Wolpert and W. G. Macready. "No free lunch theorems for optimization". In: *IEEE Transactions on Evolutionary Computation* 1 (1997), pp. 67–82. ISSN: 1089778X. DOI: 10.1109/4235.585893 (cit. on p. 127).

[118] C. Grosan and A. Abraham. *Hybrid evolutionary algorithms: Methodologies, architectures, and reviews*. 2007. DOI: 10.1007/978-3-540-73297-6_1 (cit. on p. 128).

[119] A. M. Abdelbar and M. Mokhtar. "A k-elitist max-min ant system approach to cost-based abduction". In: *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*. Vol. 4. 2003, pp. 2635–2641. ISBN: 0-7803-7804-0. DOI: 10.1109/CEC.2003.1299420 (cit. on p. 128).

[120] B. Bullnheimer, R. Hartl, and C. Strauss. "A new rank based version of the Ant System. A computational study." In: 1 (1997) (cit. on p. 128).

[121] M. Gambardella, Luca Maria; Dorigo. *HAS-SOP: hybrid ant system for the sequential ordering problem*. Tech. rep. 1997, pp. 1–22 (cit. on p. 128).

[122] S. Sahu and M. Pandey. "Hybrid Ant System Algorithm for Solving Quadratic Assignment Problems". In: *International Journal of Computer Science and Information Technologies* 5.4 (2014), pp. 5950–5956 (cit. on p. 128).

[123] J. Wang, E. Osagie, P. Thulasiraman, and R. K. Thulasiram. "HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network". In: *Ad Hoc Networks* 7 (2009), pp. 690–705. ISSN: 15708705. DOI: 10.1016/j.adhoc.2008.06.001 (cit. on p. 128).

[124] D. K. Gupta, Y. Arora, U. K. Singh, and J. P. Gupta. "Recursive Ant Colony Optimization for estimation of parameters of a function". In: *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 03/2012, pp. 448–454. ISBN: 978-1-4577-0697-4. DOI: 10.1109/RAIT.2012.6194620 (cit. on p. 129).

[125] S.-h. Chen. *Multi-Agent Applications with Evolutionary Computation and Biologically Inspired Technologies : Intelligent Techniques for Ubiquity and Optimization*. Hershey, Pa. : IGI Global (701 E. Chocolate Avenue, Hershey, Pennsylvania, 17033, USA), 2011, p. 353. ISBN: 9781605668987. DOI: 10.4018/978-1-60566-898-7 (cit. on p. 129).

[126] K. M. Salama, A. M. Abdelbar, and A. A. Freitas. "Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery

algorithm". In: *Swarm Intelligence* 5 (2011), pp. 149–182. ISSN: 19353812. DOI: 10.1007/s11721-011-0057-9 (cit. on pp. 129, 147).

[127] B. Hölldobler and E. O. Wilson. *The Ants*. Vol. N1. 1990, p. 732. ISBN: 0674040759 (cit. on p. 129).

[128] K. Krynicki. *Angry Ant Datasets*. 2014 (cit. on pp. 135, 155).

[129] M. Dorigo, M. Birattari, and T. Stützle. "Ant colony optimization". In: *IEEE Computational Intelligence Magazine* 1 (2006). ISSN: 1556-603X. DOI: 10.1109/MCI.2006.329691 (cit. on pp. 144, 168).

[130] Q. Cai, D. Zhang, W. Zheng, and S. C. Leung. "A new fuzzy time series forecasting model combined with ant colony optimization and auto-regression". In: *Knowledge-Based Systems* 74 (01/2015), pp. 61–68. ISSN: 09507051. DOI: 10.1016/j.knosys.2014.11.003 (cit. on p. 144).

[131] J. Kozak and U. Boryczka. "Multiple Boosting in the Ant Colony Decision Forest meta-classifier". In: *Knowledge-Based Systems* 75 (02/2015), pp. 141–151. ISSN: 09507051. DOI: 10.1016/j.knosys.2014.11.027 (cit. on p. 144).

[132] P. Moradi and M. Rostami. "Integration of graph clustering with ant colony optimization for feature selection". In: *Knowledge-Based Systems* 84 (08/2015), pp. 144–161. ISSN: 09507051. DOI: 10.1016/j.knosys.2015.04.007 (cit. on p. 144).

[133] D. Tang, X. Lu, and L. Yang. "ACO-based search algorithm in unstructured P2P Network". In: *Information Technology, Computer ...* (2011) (cit. on p. 147).

[134] Z. Bin. "A resource search method based on ACO in P2P". In: *2011 International Conference on Computer Science and Service System, CSSS 2011 - Proceedings*. 2011, pp. 99–102. ISBN: 9781424497638. DOI: 10.1109/CSSS.2011.5975084 (cit. on p. 147).

[135] M. Dorigo, E. Bonabeau, and G. Theraulaz. "Ant algorithms and stigmergy". In: *Future Generation Computer Systems* 16 (2000), pp. 851–871. ISSN: 0167739X. DOI: 10.1016/S0167-739X(00)00042-X (cit. on p. 148).

[136] G. Theraulaz and E. Bonabeau. "A brief history of stigmergy." In: *Artificial life* 5 (1999), pp. 97–116. ISSN: 1064-5462. DOI: 10.1162/106454699568700 (cit. on p. 148).

[137] T. Stützle and A. Wilke. *ACOTSP*. 2004 (cit. on p. 169).

[138] C. Blum. *hc-mmas-ubqp*. 2004 (cit. on p. 169).

[139] C. Solnon. *AntClique*. 2006 (cit. on p. 169).

[140] F. Meyer and R. Stubs Parpinelli. *GUIAnt-Miner*. 2002 (cit. on p. 169).

[141] F. E. Barril Otero. *Myra*. 2008 (cit. on p. 169).

[142] U. Chirico. "JACSF". In: (2009) (cit. on p. 169).

[143] F. J. Diego Martín, J. Á. González Manteca, R. Carrasco-Gallego, and J. Carrasco Arias. "AntLib v1.0: A generic C++ framework for ant colony optimization". In: *Ant Colony Optimization and Swarm Intelligence*. Vol. 5217 LNCS. 2008, pp. 397–398. ISBN: 3540875263. DOI: 10.1007/978-3-540-87527-7_43 (cit. on p. 169).

[144] O. Babaoglu, H. Meling, and A. Montresor. *Anthill: a Framework for the Development of Agent-Based Peer-to-Peer Systemse*. 2001 (cit. on p. 170).

[145] S. Wagner and G. Kronberger. "Algorithm and experiment design with heuristic lab: an open source optimization environment for research and education". In: *GECCO '12 Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. 2012, pp. 1287–1316. ISBN: 978-1-4503-1178-6. DOI: 10.1145/2330784.2330941 (cit. on p. 185).

# List of Figures

# List of Tables