Efficient Algorithms for Agent-based Semantic Resource Discovery

António Luís Lopes and Luís Miguel Botelho

We, the Body, and the Mind Research Lab of Adetti-ISCTE,
Av. Forças Armadas, Edifício ISCTE, 1600-082 Lisboa, Portugal
{antonio.luis, luis.botelho}@iscte.pt
http://www.we-b-mind.org

Abstract. A semantic overlay network is a powerful mechanism for collaborative environments where multiple agents, managing several resources, can cooperate in pursuing common and individual goals while achieving good overall performance. However, building such a social structure dynamically from an unstructured peer-to-peer network is a lengthy process if appropriate algorithms and techniques are not used. In this paper, we analyse a set of network evolution techniques that improve the performance of classic approaches, such as the *flooding* search algorithm. We compare the efficiency of these enhanced classic algorithms with our previously proposed search algorithm, which has also been improved through the referred techniques. Evaluation tests show that the improved version of our algorithm outperforms the improved version of the classic search algorithm and efficiently creates a semantic overlay network for agent-based resource coordination.

Keywords: search algorithms, semantic overlay networks, peer-to-peer networks, intelligent agents

1 Introduction

Research on peer-to-peer (*P2P*) computing has delivered promising results, paving the way for developing more robust and scalable applications. Nevertheless, these research efforts have mainly addressed the efficient management of the network, treating each peer as a simple reactive node, with little or no autonomy at all, thus ignoring the potential for developing collaborative environments. In our research, we envision the combination of the distributed capabilities of *P2P* networks with the intelligence of autonomous agents to deploy resource coordination systems allowing seamless access to large-scale distributed resources while maintaining high-availability, fault tolerance and low maintenance application deployment through self-organization.

A resource coordination system is an environment where agents, managing different resources, can cooperate to provide value-added services, which could not be provided if the agents were to operate individually. The heterogeneous and distributed environment of *P2P* networks offers the potential for building powerful applications based on resource coordination. A *semantic overlay network* [7] can enhance the resource sharing process in such distributed networks by establishing semantic links between agents based on similarities or dependencies between the resources that they manage. In our work, the semantic links are built based on the formal description of resources. For example, an executable resource (a web service or program) can be described through its inputs, outputs, pre-conditions (conditions that need to be true prior to the execution of the resource) and effects (conditions that are made true after the execution of the resource). If resource *A*'s effects contribute to resource *B*'s pre-conditions, then we say that *B* is *dependent* on *A* or that *A enables B*. Instead of blindly searching the network, semantically guided search processes direct the search to the agents that are more likely to own the required resources. Unfortunately, building a semantic overlay on top of a network of randomly-connected agents can be a very lengthy process if efficient algorithms are not used.

In previously published work [17], we have proposed the use of a dynamically created semantic overlay network as the basis upon which to build an efficient and effective self-improving agent-based resource coordination system. This included the proposal of a set of search algorithms and network evolution techniques to enhance the resource discovery process in which the agents interact to jointly build a semantic overlay network. This paper focuses on the evaluation of the proposed network evolution techniques by describing the results of applying them to a classic search algorithm (the *flooding* algorithm) and on the evaluation of one of the proposed algorithms, the *Iterative Branching Depth-First Search* (*IBDFS*) algorithm. In section 2, we present related work on search mechanisms and agent-based resource discovery mechanisms. Section 3 briefly describes the network evolution techniques that we have used to enhance classical approaches and the *IBDFS* algorithm and analyses the effects that each of those techniques has on the overall performance of the algorithms. The analysis done in section 3 allows us to conclude that a specific configuration of the *flooding* algorithm (*IF12*) is the best option to dynamically create a semantic overlay network. In section 4, we compare the performance of the enhanced classical algorithm (*IF12*) with the improved version of the *IBDFS* algorithm by analysing the results of several different tests, which show that our algorithm outperforms the *IF12* algorithm (both in result retrieval speed and network coverage) in all variations of the tests. Section 5 concludes the paper and provides some guidelines for future work.

2 Related Work

In distributed networks, where the goal is to build a collaborative environment to facilitate resource sharing, resources need to be easily located so that they can be aggregated and/or executed. Resource coordination research addresses these issues. One of the most important aspects of resource coordination is the discovery process, in which peers[1] perform search requests within the network to find the resources that they need to achieve their goals. A resource can be instantiated as a web service, a

[1] Since the agents on the referred resource coordination system are part of P2P networks (thus playing the role of peers), we sometimes refer to them as *peers*. We do not, however, wish to state that the terms *peer* and *agent* have the same meaning.

file, an intelligent agent capability, storage or processing capabilities or any other computational skill available in a network of interconnected peers.

2.1 Search Algorithms

The nature of the search process in a resource discovery environment depends on the type of network where agents operate. In *unstructured* networks (where all peers are equal in responsibilities and no hierarchy exists) peers cannot rely on any information to optimize the search process. Searching a certain network resource or peer is often carried out by using *blind* algorithms (in the sense that agents randomly query other agents), such as *flooding* or *random walk* algorithms. Both algorithms present some disadvantages: *flooding* increases network load with copies of the query message but may retrieve the results faster, whereas the *random walk* reduces the network load but increases the search latency. In an effort to improve their performance, some search mechanisms based on variations of these two algorithms were created. *Iterative deepening* [27] is an example of an improved *flooding* search mechanism. It initiates multiple *breadth-first* searches, over the iterations of the technique, with successively larger depth limits, until either the query is satisfied, or a maximum depth limit has been reached. However, this only improves the network coverage of the *flooding* algorithm, not its ability for obtaining results faster.

An alternative to these *blind* search algorithms is on the use of *informed* algorithms. In this type of search, peers use additional information about other peers' resources (usually, obtained from previous queries) to select the ones that will be contacted during the search process [5]. For example, *Routing Indices* [6] allow peers to forward queries to a subset of neighbours (which is identified by evaluating an index table that contains the inventory of resources of the neighbouring peers [20]) that are the best candidates to satisfy the query. Similar approaches are exploited in the *Directed Breadth-First Search* [27] and in the *Intelligent Search* mechanism [13] where each peer in the network builds a profile (set of properties) of its peers and uses that profile to determine which ones are more likely to answer each query. Other informed search approaches, which use feedback from previous searches to improve future ones, include *Adaptive Probabilistic Searches* [26] and *Directed Searches* [18], which rely upon statistic information about other peers, including their performance. Unfortunately, these algorithms for unstructured networks use some form of the *Depth-First Search* or *Random Walk* search mechanisms that, even though are able to reduce the network traffic, increase the search latency, thus taking more time to return the queries results.

In *structured* networks, the existence of an organizational structure helps improving search performance by facilitating message routing between the peers. A network structure usually establishes some sort of hierarchy between the peers, such as partitioning the network into a set of communicating clusters of peers that are connected amongst them by a network of *super-peers*. However, hierarchical approaches such as the ones based on these special-purpose peers come at the expense of resilience to semi-catastrophic failures of *super-peers* near the top of the hierarchy [4]. A scalable and yet robust infrastructure for *P2P* networks relies on *Distributed Hash Tables* (*DHT*). *Chord* [25], *Pastry* [22], *Tapestry* [28] and *CAN* [19] are examples of DHT implementations. Although semantic-free approaches, such as *DHT*, provide good performance for point queries (where the search key is known exactly), they are not as effective for approximate, range, or text queries [7] and they do not, on their own, capture the relationships between the resource or peer's name and its content or metadata [21].

2.2 Agent-based Resource Discovery

Early attempts for dealing with resource discovery issues [23] were based on the use of dedicated central servers. However, centralized solutions were deemed unsuitable for large environments and later approaches decided to use the hybrid potential of *P2P* networks, such as dynamic federated environments [24], where *super-peers* share their peers' resources by federating with other content-related *super-peers*; structured networks with resource rating [9]; and distributed multi-registry centres [16][12], where peers register their resources in the appropriate registry centres based on their type or the domain in which they operate. Unfortunately, using properties such as *type* and *domain* is not enough to represent complex connections between resources. An effective way to enhance resource coordination is to establish semantic connections between agents based on the properties of their resources, as these semantic links can be used to improve future searches and collaboration initiatives. The process of establishing meaningful connections in a network of randomly-connected agents may bring, however, some problems regarding efficiency and scalability, especially in large-scale systems.

Some systems rely on structured solutions, such as aggregation of peers [14] in communities [2] or the use of middle layers that have specific coordination capabilities [10][15]. Structured systems contribute to enhance the routing mechanisms in *P2P* computing, however, at the cost of introducing central points of failure such as special-purpose agents (like mediators) and hierarchical dependency relations. To avoid these failure-prone solutions, some approaches are based on pure *P2P* networks. An inference system based on pure *P2P* networks is presented in [1]. In this approach, each peer can answer queries by reasoning from its local (propositional) theory but can also perform queries to some other peers with which it is semantically related by sharing part of its knowledge. In order to create these semantic relations (referred by the authors as *acquaintance networks*), new peers joining the *P2P* system simply declare their *acquaintances* in the network, i.e., the peers they know to be sharing knowledge with, and they declare the corresponding shared variables. However, the authors do not clearly explain how the "*acquaintances declaration*" process is carried out efficiently in the *P2P* network.

The study of ant communities has inspired some research on the development of *P2P* systems based on multi-agent systems. *Anthill* [3] is a *P2P*-based *MAS* which emulates the resource coordination behaviour of ants. In this framework, storage or computational resources (referred to as "*nests*") generate requests (referred to as "*ants*") in response to user requests. These *ants* travel across the network of *nests* in order to be processed and executed. *Ants* do not communicate directly with each other. Instead, they communicate indirectly by leaving information related to the service they are implementing in the appropriate resource manager found in the visited *nests*. This *pheromone*-like approach, also called “*stigmergy*”, allows the network to self-organize and improve its performance over time. The idea of assigning agents to carry on requests (*ants*) avoids a non-scalable flooding search technique, since each *ant* will only travel to a *nest* at a time and will not replicate itself. However, the search performance might not be the best (it is, in fact, equivalent to the *Depth-first Search* algorithm that presents a poor performance compared to other search mechanisms [17]) because each edge of the network (*nests*) is only travelled once at a time for each request. The selection of the next *nest* to be visited by an *ant* can either have a deterministic approach (once the network is organized and appropriate overlay networks are available) or a totally random (*uninformed*) approach. A similar approach to [3] is proposed in [8], where mobile agents use *pheromone*-like behaviour to optimize the trails within a *P2P* network. However, instead of using the update process based on the path where the query originally came (which, as shown in section 3, limits the performance of the search algorithm), as in [3], the mobile agent creates a referral to the query-answering node, thus creating a direct link that will improve future similar searches. Unfortunately, this too uses an equivalent search algorithm to the *Depth-first Search*, which has a poor performance.

3 Network Evolution Techniques

One of the main aspects of our research is the process of dynamically building a semantic overlay on top of a network of randomly connected agents. The semantic overlay network is built through a self-organization process in which the new agents (that are introduced into the network) broadcast, according to a specific protocol, the description of their resources and needs (resources that they depend on) to the remaining agents. This information exchange allows agents to find the resources that they need (and the agents that manage them) and building a set of meaningful connections that will be useful later on, in this collaborative environment. Such a mechanism can, nevertheless, overload the entire network with requests. Thus, it is very important to devise an algorithm that allows the network to self-organise without compromising its efficiency, scalability and robustness. Some of the approaches that we analysed (see section 2) use some interesting mechanisms to improve the performance of the network in the resource discovery process. However, none of the approaches combines all those mechanisms. We have combined these into a set of techniques (including one that is not used by any of the considered approaches) that, by making use of network evolution properties, can improve the performance of any search algorithm in the process of building a semantic overlay network.

3.1 Informed searches, direct replies and referrals

In the beginning of the search process, agents in a *P2P* network do not have enough information about other agents since they are randomly connected to each other. As they go along, the interactions between them become valuable sources of information that can be used to guide future searches. Furthermore, the use of informed search techniques scales a lot better throughout time as agents improve their connections with other agents based on previous interactions [11].

In order for an agent to improve its participation in future searches, it is important that it caches information from previous search processes, as it is done in [6] [13] [18] [26] and [27]. For example, as a query response travels back to the requester agent, all agents in that specific path can either store the response themselves (*Anthill* [3]) or cache a link to the agent which has the response, thus working as a referral for future searches (as in [8]). An alternative approach can be based on the query response being returned directly to the query requester, instead of being carried back through the path originally travelled by the query. For example, if agent *A1* has the response for the query made by agent *A2*, *A1* will directly send the response to *A2*. Even though the agents in the *request* path will not learn the result of the query, the result will reach the requester agent faster and a lot of messages can be spared. Furthermore, the agents that participated in the search, even if just for forwarding or propagating the request, can assume that, after some time, the requester agent has already received the necessary response. Hence, future similar searches (for example, an agent *A3* requesting the same contents as *A2*) can be referred to the previous requester agent (*A2*), which in turn can refer it to the responder agent (*A1*) or provide itself the response directly (to *A3*). None of the approaches cited in the related work section (see section 2) uses this technique.

3.2 Improved Flooding

To determine the effects of these techniques on the process of dynamically building a semantic overlay network, we have tested different configurations of several search mechanisms. The results of our tests show that the *flooding* algorithm has the best performance of classical approaches. Due to space limitations we will only show the results for the *flooding* algorithm. Table 1 presents the different configurations of the algorithm that we have tested.

Table 1. Configurations of the *Flooding* algorithm

Name	Description
Flooding	Agents do not cache previous searches and reply through the path where the request came from.
Improved Flooding 1	Agents do not cache previous searches and reply directly to the requester agent.
Improved Flooding 2	Agents cache previous searches, use referrals and reply through the path where the request came from.
Improved Flooding 12	Agents cache previous searches, use referrals and reply directly to the requester agent.

We have tested these different configurations of the *flooding* algorithm in an environment of 1000 agents, randomly connected to 3 neighbours each and with a *time-to-live* of 3 (each request can only be forwarded 3 times). In these tests (results are shown in Figure 1) all agents start searching for the resources that they depend on at the same time. The expression *network completeness* represents the percentage of a complete semantic overlay network that is created at a specific moment. A semantic overlay network is complete when it contains an arc for each possible dependency between any pair of resources.

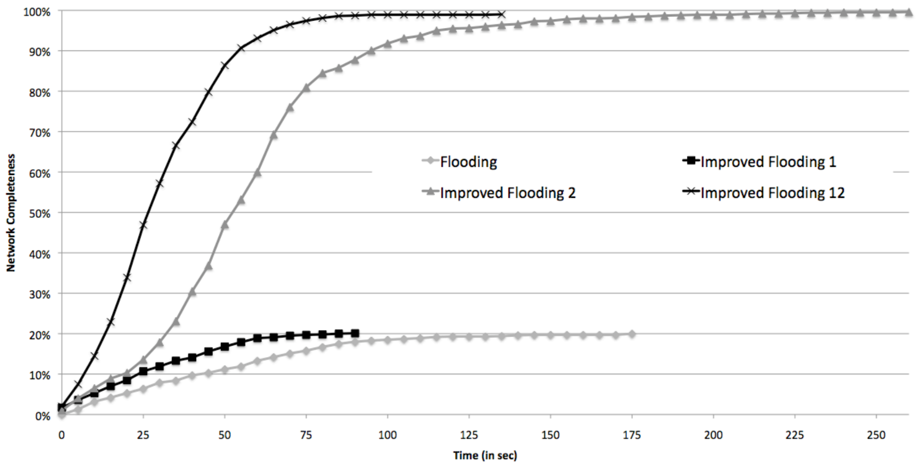

Fig. 1 – Comparison of different configurations of the *Flooding* algorithm

As depicted in Figure 1, the classical configuration of the *flooding* algorithm has the worst performance of all configurations, achieving a network completeness of only 20%. The *improved flooding* 1 configuration, representing the version of the algorithm that allows agents to reply directly to the requester agent (instead of using the *request* path), presents an improvement in time performance whereas the network completeness is maintained at 20%. This allows us to conclude that the reduction in the number of messages (in consequence of the introduced variation) and consequently on the workload of each agent, is a good network evolution technique to be applied to a search algorithm. However, as we stated in section 3.1, this does not allow all contributing agents to learn the response to the request.

The *improved flooding* 2 configuration, representing the version of the algorithm that allows agents to cache information about previous searches and use referrals, has the worst time performance but a considerably better network completeness than the previous two algorithms. This shows that caching has also a positive effect on the search algorithm's overall performance, since it allows agents to take advantage of previously collected information to trigger an evolution process that will improve future searches.

The combination of all the techniques described in section 3.1 (*improved flooding* 12 configuration) presents an excellent performance (comparatively to the other configurations) both in time and network completeness. This allows us to conclude that these network evolution techniques (and especially their combination) have a very positive influence in search algorithms.

4 Efficient Search Algorithms

Even though current algorithms for unstructured networks (such as *flooding*, *depth-first search*, *random walk* and others presented in section 2) can be improved by the use of the referred network evolution techniques (in section 3), we believe that the performance of search mechanisms can be further improved through the use of a more efficient algorithm. In this section we briefly present the *Iterative Branching Depth-First Search* (*IBDFS*) algorithm (proposed in [17]) and compare it with the *improved flooding* 12 (*IF12*) algorithm, by showing the test results of dynamically building a semantic overlay network on top of a totally unstructured network of randomly connected agents. We also analyse how the variations of some of the test parameters influence the performance of the considered algorithms.

The *IBDFS* algorithm introduces the use of an iterative process in the *depth-first* search to increase the coverage of the network without overloading the network. When initiating a search query, an agent will randomly contact one of its neighbours. If the neighbour immediately replies with the answer, then the process ends. If the neighbour does not have the answer, then the agent contacts a second neighbour and so forth[2], while the neighbour applies the same *iterative branching depth-first* search process with its neighbours. This approach increases the branching level iteratively on each hop count, thus increasing the chances of finding the answer faster, comparatively to the *depth-first* search approach [17]. This algorithm also uses the network evolution techniques described in section 3.

4.1 Evaluation Test Results

We ran several tests for building a semantic overlay on top of a network of 1000 randomly connected agents and we compared the performance of the *IBDFS* and the *IF12* algorithms. We have limited the comparison of our proposal to the *IF12* algorithm because it was the best variation in all our tests of classic (and improved) algorithms. To fully understand the differences between the algorithms, we changed several parameters of the test configuration and analysed the effects of those variations on the performance of both algorithms.

One of the parameters that we changed was the *Time To Live* (*TTL*) value. Figure 2 shows the results of the variations of the *TTL* in the test (1000 agents connected to 3 neighbours each). As depicted by Figure 2, the higher the *TTL*, the higher is the difference between the performances of both algorithms. The *IBDFS* is better (in both time and network completeness) for *TTL* values greater than or equal to three. This is due to the overloading factor of the *flooding* algorithm, which gets worse as the *TTL* increases.

Another parameter that influences the performance of search algorithms is the number of neighbours that each agent is connected to at the start of the test. Figure 3 shows the results of different tests using 4 and 5 neighbours (1000 agents and a *TTL* of 3).

[2] See [17] for a formal description of the algorithm.

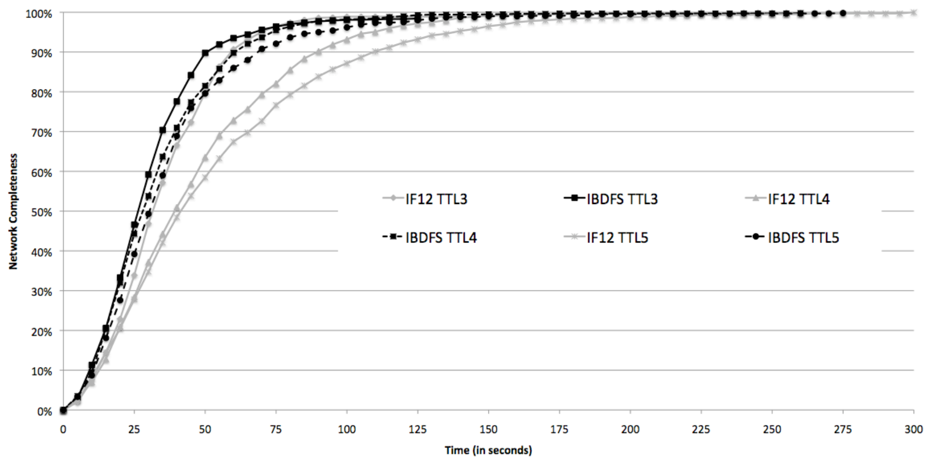

Fig. 2 – Test results for variations of *TTL*

It is visible from Figure 3 that the difference between both algorithms is greater as the number of neighbours (*NN*) that each agent is connected to increases (the darker the line, the lower the number of neighbours). Again the poor performance of the *IF12* algorithm is influenced by the overloaded network (due to the increase in messages sent), which is caused by the increased number of connections from each agent. We can also see in the figure that the *IBDFS* algorithm is not affected by the change in the number of neighbours, which allows us to conclude that the algorithm is well suited for high-load and high-connectivity networks.

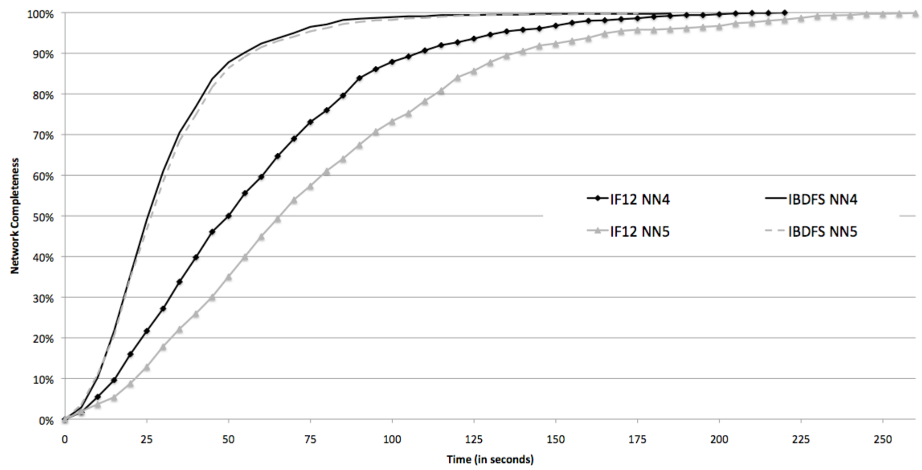

Fig. 3 – Test results for variations of number of neighbours

Up to now, the agents in these tests managed resources that were unique in the network, that is, each agent manages one resource that cannot be found anywhere else in the network. To analyse how resource distribution influences the performance of both algorithms, we decided to perform the tests using different distributions of resources. We use the term *resource distribution factor* to determine the amount of different resources existing in the network (relative to the total number of agents) and consequently their availability. For example, if the resource distribution factor is 100% (which was the case for all previous tests shown in the this paper), then the amount of resources in the network is equal to the number of agents, thus making the resources unique. If the resource distribution factor is 70%, then the amount of different resources in a network of 1000 agents is 700, thus allowing multiple resources of the same type to exist in the network.

Figure 4 shows the test results for different resource distribution factors (1000 agents connected to 3 neighbours each and with a *TTL* of 3). As depicted by Figure 4, the difference between the *IBDFS* and the *IF12* algorithms' efficiency seems to increase as the resource distribution factor (*RDF*) decreases. Once again the excessive propagation and duplication of messages through the network overloads the agents with the unfruitful task of processing useless messages, which limits their capability to perform an efficient search.

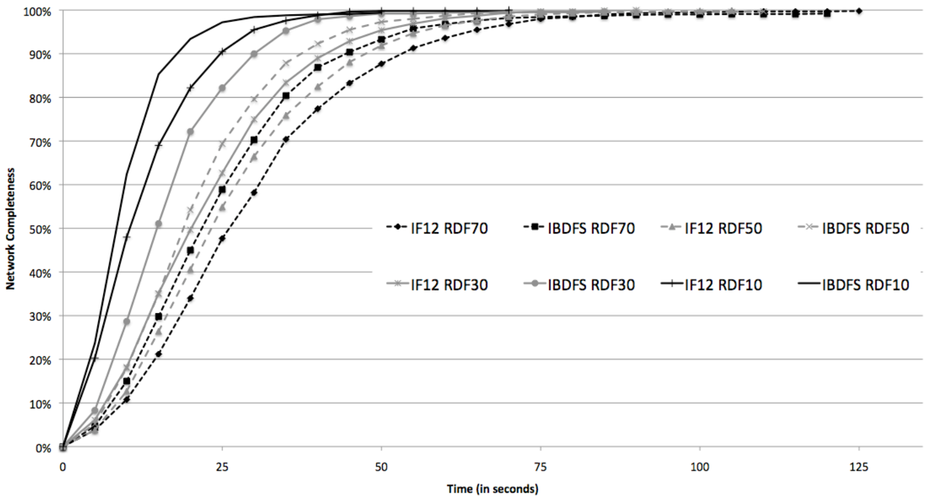

Fig. 4 – Test results for variations of resources distribution

We have also performed tests with combinations of the variations presented in Figures 2-4 (*TTL*=4, *NN*=4, *RDF*=30% and *TTL*=5, *NN*=5, *RDF*=10%) and the obtained results are coherent, showing that the *IBDFS* algorithm outperforms the *IF12* algorithm both in time and network completeness. However, we cannot show the figures related to those tests due to space limitations.

5 Conclusions and Future Work

In this paper, we have shown how the efficiency of dynamically building a semantic overlay network on top of unstructured *P2P* networks can be improved by applying a set of network evolution techniques to classic search algorithms. We have also

presented an efficient algorithm that outperforms the improved version of the *flooding* algorithm (the one that was shown to be the best of the tested classic algorithms), both in time and network completeness. Through an adequate balance between the network load and the learning capabilities of the presented network evolution techniques, the *Iterative Branching Depth-First Search* algorithm is a far more efficient and effective alternative to current *P2P* search algorithms for unstructured networks, especially for high-load and high-connectivity networks. These results were obtained by changing some parameters of the tests, namely the number of neighbours, the value of the *TTL* and the distribution of resources, which allowed us to determine the effects of applying the algorithms to different types of networks.
Our next steps include performing a deep analysis of the application of *IBDFS* (and certain variations of the algorithm) in dynamic networks and in large-scale networks (with hundreds of thousands of agents and resources) to determine if the algorithm still presents the same efficiency.

Acknowledgments. This work has been supported in part by the European Commission under the project grant FP6-IST-511632-CASCOM and by the Portuguese Foundation for Science and Technology under the scholarship grant SFRH/BD/27533/2006 and the funding Adetti-U[605].

References

1. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.-C., Laurent, S.: Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web. Journal of Artificial Intelligence Research, 25, 269-314 (2006)
2. Arpinar, I. B., Aleman-Meza, B., Zhang, R., Maduko, A.: Ontology-Driven Web Services Composition Platform. In: Proc. of the IEEE Int. Conf. on E-Commerce Technology. IEEE Computer Society, Washington D.C., pp. 146-152 (2004)
3. Babaoglu, O., Meling, H., Montresor, A.: Anthill: a Framework for the Development of Agent-Based Peer-to-Peer Systems. In: Proc. IEEE ICDCS'02, pp. 15-22 (2002)
4. Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., Stoica, I.: Looking Up Data in P2P Systems. Communications of the ACM, 46(2), 43-48 (2003)
5. Bianchini, D., De Antonellis, V., Melchiori, M., Salvi, D.: Peer-to-Peer Semantic-Based Web Service Discovery: State of the Art. Technical Report, Dipartimento di Elettronica per l'Automazione Università di Brescia (2006)
6. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-Peer Systems. In: Proc of the 22nd Int. Conf. on Distributed Computing Systems, pp. 23-30 (2002)
7. Crespo, A., Garcia-Molina, H.: Semantic Overlay Networks for P2P Systems. In: AP2PC 2005. LNCS, vol. 3601, pp. 1-13. Springer (2005)
8. Dasgupta, P.: Improving Peer-to-Peer Resource Discovery Using Mobile Agent Based Referrals. In: AP2PC 2004, LNCS, vol. 2872, pp. 186-197, Springer-Verlag (2004)
9. Emekci, F., Sahin, O. D., Agrawal, D., El Abbadi, A.: A Peer-to-Peer Framework for Web Service Discovery With Ranking. In: Proc. of the IEEE Int. Conf. on Web Services, pp. 192- 199 (2004)
10. Ermolayev, V., Keberle, N., Plaksin, S., Terziyan, V., Kononenko, O.: Towards a Framework for Agent-Enabled Semantic Web Service Composition. International Journal of Web Service Research, vol. X (2004)

11. Fletcher, G. H. L., Sheth, H. A., Börner, K.: Unstructured Peer-to-Peer Networks: Topological Properties and Search Performance. In: AP2PC 2004, LNCS, vol. 3601, pp. 14-27, Springer-Verlag (2004)
12. Gagnes, T., Plagemann, T., Munthe-Kaas, E.: A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments. In: Proc. of the 22nd Int. Conf. on Data Engineering Workshops, IEEE Computer Society, Washington, DC, p. 74 (2006)
13. Kalogeraki, V., Gunopulos, D., Zeinalipour-Yazti, D.: A Local Search Mechanism for Peer-to-Peer Networks. In: Proc. of the 11th Int. Conf. on Information and Knowledge Management, pp. 300-307 (2002)
14. Küngas, P., Matskin, M.: Semantic Web Service Composition Through a P2P-Based Multi-Agent Environment. In: LNCS, vol. 4118, pp. 106-119, Springer-Verlag, (2006)
15. Li, T.-Y., Zhao, Z.-G., You, S.-Z.: A-Peer: An Agent Platform Integrating Peer-to-Peer Network. In: Proc. of 3rd IEEE/ACM Int. Symposium on Cluster Computing and the Grid, pp. 614- 617 (2003)
16. Lin, Q., Rao, R., Li, M.: DWSDM: a Web Services Discovery Mechanism Based on a Distributed Hash Table. In: Proc. of the 5th Int. Conf. on Grid and Cooperative Computing Workshops, pp. 176-180 (2006)
17. Lopes, A., Botelho, L.: Improving Multi-Agent Based Resource Coordination in Peer-to-Peer Networks. J. Networks, to appear (2008)
18. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. In: Proc. of the 16th Int. Conf. on Supercomputing, ACM Press, New York, NY, pp. 84-95 (2002)
19. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proc. of ACM SIGCOMM'01 Conference (2001)
20. Ratsimor, O., Chakraborty, D., Joshi, A., Finin, T., Yesha, Y.: Service Discovery in Agent-Based Pervasive Computing Environments. J. Mobile Network Applications, 9(6), 679-692 (2004)
21. Risson, J., Moors, T.: Survey of Research Towards Robust Peer-to-Peer Networks: Search Methods. Comput. Networks, 50(17), 3485-3521 (2006)
22. Rowstron, A. I., Druschel, P.: Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In: Proc. of the IFIP/ACM Int. Conf. on Distributed Systems Platforms. LNCS, vol. 2218, pp.329-350. Springer, Heidelberg (2001).
23. Sheldon, M. A., Duda, A., Weiss, R., Gifford, D. K.: Discover: a Resource Discovery System Based on Content Routing. In: Proc. of the 3rd Int. World-Wide Web Conference on Technology, Tools and Applications, pp. 953-972 (1995)
24. Sivashanmugam, K., Verma, K., Sheth, V.: Discovery of Web Services in a Federated Registry Environment. In: Proc. of the IEEE Int. Conf. on Web Services, pp. 270-278 (2004)
25. Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. of the ACM SIGCOMM 01 Conference (2001)
26. Tsoumakos, Q., Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks. In: Proc. of the 3rd Int. Conf. on Peer-to-Peer Computing, pp. 102-110 (2003)
27. Yang, B., Garcia-Molina, H.: Efficient Search in Peer-to-Peer Networks. In: Proc. of Int. Conf. on Distributed Computing Systems (2002)
28. Zhao, B. Y., Kubiatowicz, J. D., Joseph, A. D.: Tapestry: an Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Technical Report. UMI Order Number: CSD-01-1141., University of California at Berkeley (2001)