Heuristic Resource Search in a Self-Organised Distributed Multi Agent System

Muntasir Al-Asfoor, Brendan Neville, and Maria Fasli

School of Computer Science and Electronic Engineering, University of Essex ,
Wivenhoe Park, Colchester, UK, CO4 3SQ
{mjalas,bneville,mfasli}@essex.ac.uk

Abstract. The work presented in this paper has addressed the issue of resource sharing in dynamic heterogeneous Multi Agent Systems as a search problem. When performing a random search, this might lead to traverse the whole network and increase the failure ratio. This paper has introduced heuristic directed search based on the usage of an approximate matching mechanism to overcome this problem. Our implementation of search algorithms differs from traditional algorithms by using semantically guided technique for resource search as well as a dynamically re-organisable network of agents. The experimental results have shown that using directed search techniques is better than random search in terms of number of hops to find the match. Furthermore, network re-organisation has improved the system performance by directing the search based on resources information, especially when high accuracy is required.

Keywords: Multi Agent Systems, Resource Sharing, Self-Organisation.

1 Introduction

The rapid development in the scientific research and business requirements for computational and data resources have increased the need to manage the process of describing and discovering these resources in an efficient way. Furthermore, any suggested solution should take into consideration the key characteristics of pervasive networking which are: heterogeneity, scalability and dynamic behavior.

Resource sharing in a distributed environment has been addressed by many researchers in the industry and academia. Many frameworks have been developed to facilitate the process of sharing a geographically distributed resource. In [13] a dynamic search algorithm in a peer-to-peer system has been proposed, it meant to overcome the traditional flooding and random search algorithms like searching the whole network aggressively and the long search time respectively. The proposed algorithm is a modified version of the traditional algorithms (random walk and flooding) by maintaining a probability function which provides some knowledge to the peer from the previous states so it could use this knowledge to predict the best move.

A dynamic network topology adaptation technique has been proposed by [3] with the aim of improving the search performance of the flooding algorithm by creating semantic communities based on query traffic. In contrast with our work, the authors have focused more on changing the topology based on statistical heuristics by collecting nodes with similar interests together in small communities. They assumed that the file request could be fulfilled by nodes within the same community.

Inspired by the idea of reducing the matching accuracy to discover the resources quickly, we have adapted the conventional random search algorithm

by introducing a directed search technique using the available resources description. The novelty we have introduced is to guide the search through the agents' network based on how semantically close the agents are to the requester in terms of resources. The rest of the paper has been organised as follows: section two has been devoted to the resource search scenario. System evaluation and experimental results are presented in section three. Finally, conclusions and suggestions for future development are discussed in section four.

2 Resource Sharing Scenario

In order to support our investigation into the micro-macro link between the selected search parameters, and their global consequences with respect to our system performance metrics, we have chosen to run multiple simulations of our resource discovery scenario across the available state space as defined by our parameters. A number of agent-based simulation tools exist such as Mason [7], NetLogo [12] and Swarm [6] in addition to specific agent languages [9] and platforms e.g. Jason [2]. From this myriad of tools we have selected to use PreSage [8], a Java based rapid prototyping and simulation tool developed to support the animation of agent societies. Similarly to the platforms and tools mentioned above, PreSage allows us to script a series of simulation runs with varying parameters, and record results from a global perspective. The system has been designed as a distributed multi agent system where each agent could be a provider or requester. Each agent holds resources which are available to share or tasks they need to be executed. Each agent has a unique identifier and knows about a subset of the agents' population and uses them to maintain its own contact database. The contacts database contains a list of agents to which the agent can forward messages. The network of agents starts with one agent and then agents will be added to the network while the simulation is running. When a new agent joins the system it will be given a random peer from the set of peers which are already active in the system and then the agent will start to forward and receive messages.The receiver agents will use the senders' contact information to update their contact databases. The agents are interacting using messages of the form: $Message =< To, From, Resource, Accuracy, TTL >$

where: To: is the sender's ID; $From$: is the potential receiver's ID; $Resource$: is the required resources; $Accuracy$: is the required matching accuracy; TTL: is the maximum time to live (number of hops).

For the purpose of simulation, the contacts database size have been chosen to be finite and each agent can contact k agents directly giving that all the agents are connected in the network layer. Limiting the contacts database size helps to avoid the situation in which each agent is connected directly to all the agents in the network which makes each agent a messages sink; this situation will overload the network and exploit its resources. The simulation's time has been divided into time cycles where a new agent is created each time cycle(t). In our experiments, 300 agents have been created during the first $300t$ of the total $600t$ simulation time.

2.1 Resource Description and Matchmaking

In a distributed computing environment, many standard methods have been developed to describe resources and tasks extensively. For instance, WSDL (Web Service Description Language) [11], OWL (Web Ontology Language) [10] and RDF(Resource Description Framework) [4]. For the purpose of simulation and since the focus of this paper is on improving the network traffic performance, resources and tasks have been described as a three element vector in the form $resource < r_0, r_1, r_2 >$. Where, the vector $resource$ represents the semantic information which then will be used to match resources vs tasks. The resource elements r_i are assigned a random value in the range [0-4] from a uniform distribution which creates 5^3 possible different values. Resource vs. request matchmaking takes place using the Manhattan [5] distance measure between resources and tasks vectors as shown in Equation 1.

$$Dis(R^A, R^B) = \sum_{i=0}^{2} \left| r_i^A - r_i^B \right| \tag{1}$$

where: Dis is the Manhattan distance between resource R^A and resource R^B.

2.2 Resource Search

The resource discovery problem has been considered as a search problem with the aim of finding the required resource across the network. As the network could scale too largely, the aim is to direct the search using the available resource information and by applying the matchmaking technique. Directing the search would decrease the number of hops required to discover the resources and increase the chances of doing that before the request message times out.

As the size of the contacts database is finite, two different methods have been used to deal with the process of adding a new contact to the database. If a new agent has joined and there is a free space in the contacts' database, then it just adds the new contact to the database. Otherwise, if there is no free space in the contacts database, then depending on which adding method has been selected, the agent would either remove the oldest contact and add the new one (First In First Out) in its place or, remove the agent with the lowest similarity and add the new comer (if the new comer is semantically closer than the one to be removed). Technically, the first method (First in First out) helps to maintain the

network connectivity of the newcomer and gives it a chance to communicate with other agents. In contrast, the second method (join based on similarity) helps to re-organise the network by directing the search based on the type of resources.

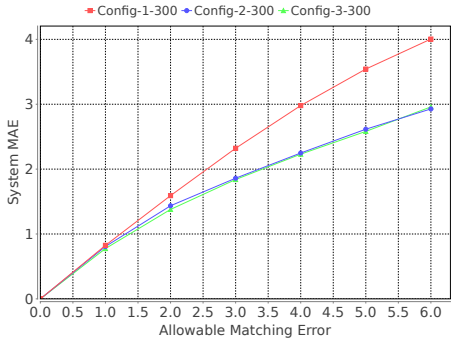
Upon receiving a resource search message (rsm) the receiver agent has to check if the sender is unknown (not available in the contact database), then it adds the sender to the contact database, otherwise, it just updates its record. Furthermore, the receiver will check if the required resource is available locally within the required matching error using Equation 1. Accordingly, if the matching error is less than the maximum error, then the receiver will send a "resource found" message to the initiator and stop. Otherwise, the receiver will have to check if the rsm has timed out or no by checking the time to live parameter of the rsm. After that, if the rsm has timed out, then the search fails otherwise the receiver will have to select the next peer to forward the message to it after updating the path and decreasing the time to live. If there are no more agents to send the rsm to, the agent must send the rsm back to the previous agent in order to enable it to select another branch to send the rsm to.

The next agent(peer) selection is based on two different methods. The first method is to select an agent randomly from the contact database to forward the rsm to excluding the agents which have been already visited before (member of the path). The second method is to select an agent which holds the resource with the smallest matching error with the required resources (given that the error is greater than the maximum allowable error) to forward the rsm to excluding the ones who have already been visited. The modifications we have made during the course of this work are related to the network's connectivity where it changes dynamically when new peers join the system or an existing peer sends an rsm. Even with the conventional random search algorithm, the network's connectivity changes using first In First Out (FIFO) technique to avoid the aging problem we have discussed before. For further details and the Algorithms pseudo code see [1].

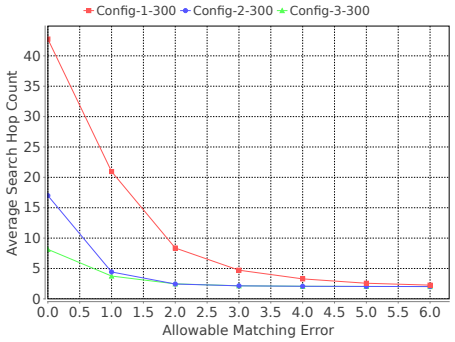
3 System Evaluation and Experimental Results

Three experiments have been designed to measure the fidelity of the three configurations with which each agent decides which agent to forward the resource search message to. These configurations are: Config-1: Random Search with First In First Out (FIFO) contacts; Config-2: Directed Search with First In First Out (FIFO) contacts; Config-3: Directed Search with Self-Organisation. For simulation purposes four parameters have been set as follows: Number of Agents = 300; maximum number of contacts for each agent = 10; maximum number of hops each resource search message can do (ttl) = 100; required accuracy of matching between [0,12]. The aim of these experiments is to evaluate the effects of each configuration on the system performance which has been measured by the following parameters: System Mean Average Error (MAE); Average Hop Count to find the resource successfully; Percentage of Failures : the ratio of failed requests to the total number of requests.

In the experiment we ran with required accuracy of 0, it appeared that the matching error between the acceptable and the achieved accuracy was the same for all configurations. The reason for that is simply because there is no matching error less than 0. Furthermore, as the acceptable error increased, the system MAE increased, with directed search approaches doing better than non-directed search as shown in Figure 1(a). However, the average number of hops to find the resource successfully has increased sharply when the acceptable matching error decreased because with smaller acceptable error the message has to navigate more agents to find the required resource. Directed Search with Self-organisation has shown a rapid decrease in the Average Hop Count to find the resource successfully in comparison with Random Search and a noticeable decrease in comparison with Directed Search especially with small acceptable error values as shown in Figure 1(b).



(a) A comparison between the allowable matching error the found matching error (system MAE).



(b) Average number of hops a resource search message will make before successfully finding the requested resource.

The ability to re-organise the contact list based on the resources information has improved the system performance by enabling the agent to learn from the system about the available resources and direct search accordingly.

4 Conclusions and Future Work

This paper has addressed resource sharing in dynamic heterogeneous Multi Agent Systems as a search problem. Performing random search might lead to traverse the whole network exhaustively and increase the failure ratio; accordingly, we have introduced heuristic directed search to overcome this problem. Our implementation of search algorithms differs from traditional algorithms by using a semantically guided technique for resource search as well as a dynamically re-organisable network of agents. The experimental results have shown that using directed search techniques is better than random search in terms of number of hops to find the match. Furthermore, network re-organisation has improved the system performance by directing the search based on resources information, especially when high accuracy is required.

We are currently working on improving the system performance by adapting the system in a way that enables it to build a virtual organisations based on the resources descriptions by employing semantic matching techniques. Furthermore, each virtual organisation will have to elect an organisation head and forward resource search messages to heads instead of individual agents which decreases the search time by traversing less agents across the network.

References

1. Muntasir Al-Asfoor, Brendan Neville, and Maria Fasli. A Study of the Heuristic Resource Search Algorithms. Technical Report CES-518, University of Essex, Department of Computer Science and Electronic Engineering, 2012.
2. Rafael H. Bordini, Michael Wooldridge, and Jomi Fred Hübner. *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.
3. Vicent Cholvi, Pascal Felber, and Ernst Biersack. Efficient search in unstructured peer-to-peer networks. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '04, pages 271–272, New York, NY, USA, 2004. ACM.
4. Graham Klyne and Jeremy J. Carroll. Resource description framework (rdf): Concepts and abstract syntax, February 2004.
5. Nikola Ljubešić, Damir Boras, Nikola Bakarić, and Jasmina Njavro. Comparing measures of semantic similarity. In Vesna Lužar-Stiffler, Vesna Hljuz Dobrić, and Zoran Bekić, editors, *Proceedings of the 30th International Conference on Information Technology Interfaces*, pages 675–682, Zagreb2, 2008. SRCE University Computing Centre.
6. Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.
7. N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system, a toolkit for building multi-agent simulations, 1996.
8. Brendan Neville and Jeremy Pitt. Presage: A programming environment for the simulation of agent societies. In Koen Hindriks, Alexander Pokahr, and Sebastian Sardina, editors, *Programming Multi-Agent Systems*, volume 5442 of *Lecture Notes in Computer Science*, pages 88–103. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-03278-3_6.
9. Anand S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*, pages 42–55, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.
10. Goutam Kumar Saha. Web ontology language (owl) and semantic web. *Ubiquity*, 2007:1:1–1:1, September 2007.
11. S. Justin Samuel and T. Sasipraba. Article: Trends and issues in integrating enterprises and other associated systems using web services. *International Journal of Computer Applications*, 1(12):17–20, February 2010. Published By Foundation of Computer Science.
12. Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *in International Conference on Complex Systems*, pages 16–21, 2004.
13. Lin Tsungnan, Lin Pochiang, Wang Hsinping, and Chen Chiahung. Dynamic search algorithm in unstructured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 20:654–666, 2009.